Fruit recognition from images using deep learning

Horea Mureşan
Faculty of Mathematics and Computer Science
Mihail Kogălniceanu, 1
Babeş-Bolyai University
Romania
email: horea94@gmail.com

Mihai Oltean
Faculty of Exact Sciences and Engineering
Unirii, 15-17
"1 Decembrie 1918" University of Alba Iulia
Romania
email: mihai.oltean@gmail.com

Abstract.

In this paper we introduce a new, high-quality, dataset of images containing fruits. We also present the results of some numerical experiment for training a neural network to detect fruits. We discuss the reason why we chose to use fruits in this project by proposing a few applications that could use such classifier.

Keywords: *Deep learning*, *Object recognition*, *Computer vision*, *fruits dataset*, *image processing*

1 Introduction

The aim of this paper is to propose a new dataset of images containing popular fruits. The dataset was named Fruits-360 and can be downloaded from the addresses pointed by references [36] and [37]. Currently (as of 2018.08.20) the set contains 50590 images of 75 fruits and it is constantly updated with images of new fruits as soon as the authors have accesses to them. The reader is encouraged to access the latest version of the dataset from the above indicated addresses.

arXiv:1712.00580v5 [cs.CV] 2 Sep 2018

Having a high-quality dataset is essential for obtaining a good classifier. Most of the existing datasets with images (see for instance the popular CIFAR dataset [35]) contain both the object and the noisy background. This could lead to cases where changing the background will lead to the incorrect classification of the object.

As a second objective we have trained a deep neural network that is capable of identifying fruits from images. This is part of a more complex project that has the target of obtaining a classifier that can identify a much wider array of objects from images. This fits the current trend of companies working in the augmented reality field. During its annual I/O conference, Google announced [38] that is working on an application named Google Lens which will tell the user many useful information about the object toward which the phone camera is pointing. First step in creating such application is to correctly identify the objects. The software has been released later in 2017 as a feature of Google Assistant and Google Photos apps. Currently the identification of objects is based on a deep neural network [39].

Such a network would have numerous applications across multiple domains like autonomous navigation, modeling objects, controlling processes or human-robot interactions. The area we are most interested in is creating an autonomous robot that can perform more complex tasks than a regular industrial robot. An example of this is a robot that can perform inspections on the aisles of stores in order to identify out of place items or understocked shelves. Furthermore, this robot could be enhanced to be able to interact with the products so that it can solve the problems on its own. Another area in which this research can provide benefits is autonomous fruit harvesting. While there are several papers on this topic already, from the best of our knowledge, they focus on few species of fruits or vegetables. In this paper we attempt to create a network that can classify a variety of species of fruit, thus making it useful in many more scenarios.

As the start of this project we chose the task of identifying fruits for several reasons. On one side, fruits have certain categories that are hard to differentiate, like the citrus genus, that contains oranges and grapefruits. Thus we want to see how well can an artificial intelligence complete the task of classifying them. Another reason is that fruits are very often found in stores, so they serve as a good starting point for the previously mentioned project.

The paper is structured as follows: in the first part we will shortly discuss a few outstanding achievements obtained using deep learning for fruits

recognition, followed by a presentation of the concept of deep learning. In the second part we describe the Fruits-360 dataset: how it was created and what it contains. In the third part we will present the framework used in this project - TensorFlow[33] and the reasons we chose it. Following the framework presentation, we will detail the structure of the neural network that we used. We also describe the training and testing data used as well as the obtained performance. Finally, we will conclude with a few plans on how to improve the results of this project.

2   Related work

In this section we review several previous attempts to use neural networks and deep learning for fruits recognition.

A method for recognizing and counting fruits from images in cluttered greenhouses is presented in [24]. The targeted plants are peppers with fruits of complex shapes and varying colors similar to the plant canopy. The aim of the application is to locate and count green and red pepper fruits on large, dense pepper plants growing in a greenhouse. The training and validation data used in this paper consists of 28000 images of over 1000 plants and their fruits. The used method to locate and count the peppers is two-step: in the first step, the fruits are located in a single image and in a second step multiple views are combined to increase the detection rate of the fruits. The approach to find the pepper fruits in a single image is based on a combination of (1) finding points of interest, (2) applying a complex high-dimensional feature descriptor of a patch around the point of interest and (3) using a so-called bag-of-words for classifying the patch.

Paper [22] presents a novel approach for detecting fruits from images using deep neural networks. For this purpose the authors adapt a Faster Region-based convolutional network. The objective is to create a neural network that would be used by autonomous robots that can harvest fruits. The network is trained using RGB and NIR (near infra red) images. The combination of the RGB and NIR models is done in 2 separate cases: early and late fusion. Early fusion implies that the input layer has 4 channels: 3 for the RGB image and one for the NIR image. Late fusion uses 2 independently trained models that are merged by obtaining predictions from both models and averaging the results. The result is a multi modal network which obtains much better performance than the existing networks.

On the topic of autonomous robots used for harvesting, paper [2] shows

a network trained to recognize fruits in an orchard. This is a particularly difficult task because in order to optimize operations, images that span many fruit trees must be used. In such images, the amount of fruits can be large, in the case of almonds up to 1500 fruits per image. Also, because the images are taken outside, there is a lot of variance in luminosity, fruit size, clustering and view point. Like in paper [22], this project makes use of the Faster Region-based convolutional network, which is presented in a detailed view in paper [21]. Related to the automatic harvest of fruits, article [19] presents a method of detecting ripe strawberries and apples from orchards. The paper also highlights existing methods and their performance.

In [14] the authors compile a list of the available state of the art methods for harvesting with the aid of robots. They also analyze the method and propose ways to improve them.

In [3] one can see a method of generating synthetic images that are highly similar to empirical images. Specifically, this paper introduces a method for the generation of large-scale semantic segmentation datasets on a plant-part level of realistic agriculture scenes, including automated per-pixel class and depth labeling. One purpose of such synthetic dataset would be to bootstrap or pre-train computer vision models, which are fine-tuned thereafter on a smaller empirical image dataset. Similarly, in paper [20] we can see a network trained on synthetic images that can count the number of fruits in images without actually detecting where they are in the image.

Another paper, [5], uses two back propagation neural networks trained on images with apple "Gala" variety trees in order to predict the yield for the upcoming season. For this task, four features have been extracted from images: total cross-sectional area of fruits, fruit number, total cross-section area of small fruits, and cross-sectional area of foliage.

Paper [12] presents an analysis of fruit detectability in relation to the angle of the camera when the image was taken. Based on this research, it was concluded that the fruit detectability was the highest on front views and looking with a zenith angle of $60°$ upwards.

In papers [1, 29, 30] we can see an approach to detecting fruits based on color, shape and texture. They highlight the difficulty of correctly classifying similar fruits of different species. They propose combining existing methods using the texture, shape and color of fruits to detect regions of interest from images. Similarly, in [18] a method combining shape, size and color, texture of the fruits together with a k nearest neighbor algorithm is used to increase the accuracy of recognition.

One of the most recent works [28] presents an algorithm based on the improved ChanVese level-set model [4] and combined with the level-set idea and M-S mode [17]. The proposed goal was to conduct night-time green grape detection. Combining the principle of the minimum circumscribed rectangle of fruit and the method of Hough straight-line detection, the picking point of the fruit stem was calculated.

3   Deep learning

In the area of image recognition and classification, the most successful results were obtained using artificial neural networks [7, 26]. These networks form the basis for most deep learning models.

Deep learning is a class of machine learning algorithms that use multiple layers that contain nonlinear processing units [23]. Each level learns to transform its input data into a slightly more abstract and composite representation [7]. Deep neural networks have managed to outperform other machine learning algorithms. They also achieved the first superhuman pattern recognition in certain domains [6]. This is further reinforced by the fact that deep learning is considered as an important step towards obtaining Strong AI. Secondly, deep neural networks - specifically convolutional neural networks - have been proved to obtain great results in the field of image recognition.

In the rest of this section we will briefly describe some models of deep artificial neural networks along with some results for some related problems.

3.1   Convolutional neural networks

Convolutional neural networks (CNN) are part of the deep learning models. Such a network can be composed of convolutional layers, pooling layers, ReLU layers, fully connected layers and loss layers [32]. In a typical CNN architecture, each convolutional layer is followed by a Rectified Linear Unit (ReLU) layer, then a Pooling layer then one or more convolutional layer and finally one or more fully connected layer. A characteristic that sets apart the CNN from a regular neural network is taking into account the structure of the images while processing them. Note that a regular neural network converts the input in a one dimensional array which makes the trained classifier less sensitive to positional changes.

Among the best results obtained on the MNIST [34] dataset is done by using multi-column deep neural networks. As described in paper [8], they use multiple maps per layer with many layers of non-linear neurons. Even if the complexity of such networks makes them harder to train, by using graphical processors and special code written for them. The structure of the network uses winner-take-all neurons with max pooling that determine the winner neurons.

Another paper [16] further reinforces the idea that convolutional networks have obtained better accuracy in the domain of computer vision. The paper proposes an improvement to the popular convolutional network in the form of a recurrent convolutional network. Traditionally, recurrent networks have been used to process sequential data, handwriting or speech recognition being the most known examples [13, 10]. By using recurrent convolutional layers with some max pool layers in between them and a final global max pool layer at the end several advantages are obtained. Firstly, within a layer, every unit takes into account the state of units in an increasingly larger area around it. Secondly, by having recurrent layers, the depth of the network is increased without adding more parameters.

In paper [25] an all convolutional network that gains very good performance on CIFAR-10 [35] is described in detail. The paper proposes the replacement of pooling and fully connected layers with equivalent convolutional ones. This may increase the number of parameters and adds inter-feature dependencies however it can be mitigated by using smaller convolutional layers within the network and acts as a form of regularization.

In what follows we will describe each of the layers of a CNN network.

3.1.1 Convolutional layers

Convolutional layers are named after the convolution operation. In mathematics convolution is an operation on two functions that produces a third function that is the modified (convoluted) version of one of the original functions. The resulting function gives in integral of the pointwise multiplication of the two functions as a function of the amount that one of the original functions is translated [31].

A convolutional layer consists of groups of neurons that make up kernels. The kernels have a small size but they always have the same depth as the input. The neurons from a kernel are connected to a small region of

the input, called the receptive field, because it is highly inefficient to link all neurons to all previous outputs in the case of inputs of high dimensions such as images. For example, a 100 x 100 image has 10000 pixels and if the first layer has 100 neurons, it would result in 1000000 parameters. Instead of each neuron having weights for the full dimension of the input, a neuron holds weights for the dimension of the kernel input. The kernels slide across the width and height of the input, extract high level features and produce a 2 dimensional activation map. The stride at which a kernel slides is given as a parameter. The output of a convolutional layer is made by stacking the resulted activation maps which in turned is used to define the input of the next layer.

Applying a convolutional layer over an image of size 32 X 32 results in an activation map of size 28 X 28. If we apply more convolutional layers, the size will be further reduced, and, as a result the image size is drastically reduced which produces loss of information and the vanishing gradient problem. To correct this, we use padding. Padding increases the size of a input data by filling constants around input data. In most of the cases, this constant is zero so the operation is named zero padding. "Same" padding means that the output feature map has the same spatial dimensions as the input feature map. This tries to pad evenly left and right, but if the number of columns to be added is odd, it will add an extra column to the right. "Valid" padding is equivalent to no padding.

The strides causes a kernel to skip over pixels in an image and not include them in the output. The strides determines how a convolution operation works with a kernel when a larger image and more complex kernel are used. As a kernel is sliding the input, it is using the strides parameter to determine how many positions to skip.

ReLU layer, or Rectified Linear Units layer, applies the activation function max(0, x). It does not reduce the size of the network, but it increases its nonlinear properties.

3.1.2 Pooling layers

Pooling layers are used on one hand to reduce the spatial dimensions of the representation and to reduce the amount of computation done in the network. The other use of pooling layers is to control overfitting. The most used pooling layer has filters of size 2 x 2 with a stride 2. This effectively reduces the input to a quarter of its original size.

3.1.3 Fully connected layers

Fully connected layers are layers from a regular neural network. Each neuron from a fully connected layer is linked to each output of the previous layer. The operations behind a convolutional layer are the same as in a fully connected layer. Thus, it is possible to convert between the two.

3.1.4 Loss layers

Loss layers are used to penalize the network for deviating from the expected output. This is normally the last layer of the network. Various loss function exist: softmax is used for predicting a class from multiple disjunct classes, sigmoid cross-entropy is used for predicting multiple independent probabilities (from the [0, 1] interval).

3.2 Recurrent neural network

Another deep learning algorithm is the recursive neural network [16]. In this kind of architecture the same set of weights is recursively applied over some data. Traditionally, recurrent networks have been used to process sequential data, handwriting or speech recognition being the most known examples. By using recurrent convolutional layers with some max pool layers in between them and a final global max pool layer at the end several advantages are obtained. Firstly, within a layer, every unit takes into account the state of units in an increasingly larger area around it. Secondly, by having recurrent layers, the depth of the network is increased without adding more parameters. Recurrent networks have shown good results in natural language processing.

3.3 Deep belief network

Yet another model that is part of the deep learning algorithms is the deep belief network [15]. A deep belief network is a probabilistic model composed by multiple layers of hidden units. The usages of a deep belief network are the same as the other presented networks but can also be used to pre-train a deep neural network in order to improve the initial values of the weights. This process is important because it can improve the quality of the network and can reduce training times. Deep belief networks can be combined with convolutional ones in order to obtain convolutional deep

belief networks which exploit the advantages offered by both types of architectures.

4 Fruits-360 data set

In this section we describe how the data set was created and what it contains.

The images were obtained by filming the fruits while they are rotated by a motor and then extracting frames.

Fruits were planted in the shaft of a low speed motor (3 rpm) and a short movie of 20 seconds was recorded. Behind the fruits we placed a white sheet of paper as background.

Figure 1: Left-side: original image. Notice the background and the motor shaft. Right-side: the fruit after the background removal and after it was scaled down to 100x100 pixels.

However due to the variations in the lighting conditions, the background was not uniform and we wrote a dedicated algorithm which extract the fruit from the background. This algorithm is of flood fill type: we start from each edge of the image and we mark all pixels there, then we mark all pixels found in the neighborhood of the already marked pixels for which

the distance between colors is less than a prescribed value. we repeat the previous step until no more pixels can be marked.

All marked pixels are considered as being background (which is then filled with white) and the rest of pixels are considered as belonging to the object. The maximum value for the distance between 2 neighbor pixels is a parameter of the algorithm and is set (by trial and error) for each movie.

Fruits were scaled to fit a 100x100 pixels image. Other datasets (like MNIST) use 28x28 images, but we feel that small size is detrimental when you have too similar objects (a red cherry looks very similar to a red apple in small images). Our future plan is to work with even larger images, but this will require much more longer training times.

To understand the complexity of background-removal process we have depicted in Figure 1 a fruit with its original background and after the background was removed and the fruit was scaled down to 100 x 100 pixels.

The resulted dataset has 50590 images of fruits spread across 75 labels. The data set is available on GitHub [36] and Kaggle [37]. The labels and the number of images for training are given in Table 1.

Table 1: Number of images for each fruit. There are multiple varieties of apples each of them being considered as a separate object. We did not find the scientific/popular name for each apple so we labeled with digits (e.g. apple red 1, apple red 2 etc).

Label	Number of training images	Number of test images
Avocado	427	143
Avocado ripe	491	166
Banana	490	166
Banana Red	490	166
Cactus fruit	490	166
Cantaloupe 1	492	164
Cantaloupe 2	492	164
Carambula	490	166
Cherry 1	492	164
Cherry 2	738	246
Cherry Rainier	738	246
Cherry Wax Black	492	164
Continued on next page		

Table 1 – continued from previous page

Label	Number of training images	Number of test images
Cherry Wax Red	492	164
Cherry Wax Yellow	492	164
Clementine	490	166
Cocos	490	166
Dates	490	166
Granadilla	490	166
Grape Pink	492	164
Grape White	490	166
Grape White 2	490	166
Grapefruit Pink	490	166
Grapefruit White	492	164
Guava	490	166
Huckleberry	490	166
Kaki	490	166
Kiwi	466	156
Kumquats	490	166
Lemon	492	164
Lemon Meyer	490	166
Limes	490	166
Lychee	490	166
Mandarine	490	166
Mango	490	166
Maracuja	490	166
Melon Piel de Sapo	738	246
Mulberry	492	164
Nectarine	492	164
Orange	479	160
Papaya	492	164
Passion Fruit	490	166
Peach	492	164
Peach Flat	492	164
Pear	492	164
Pear Abate	490	166
Pear Monster	490	166
		Continued on next page

Table 1 – continued from previous page

Label	Number of training images	Number of test images
Pear Williams	490	166
Pepino	490	166
Physalis	492	164
Physalis with Husk	492	164
Pineapple	490	166
Pineapple Mini	493	163
Pitahaya Red	490	166
Plum	447	151
Pomegranate	492	164
Quince	490	166
Rambutan	492	164
Raspberry	490	166
Salak	490	162
Strawberry	492	164
Strawberry Wedge	738	246
Tamarillo	490	166
Tangelo	490	166
Walnut	735	249

5   TensorFlow library

For the purpose of implementing, training and testing the network described in this paper we used the TensorFlow library [33]. This is an open source framework for machine learning created by Google for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays called tensors.

The main components in a TensorFlow system are the client, which uses the Session interface to communicate with the master, and one or more worker processes, with each worker process responsible for arbitrating access to one or more computational devices (such as CPU cores or GPU cards) and for executing graph nodes on those devices as instructed by the master.

TensorFlow offers some powerful features such as: it allows computation mapping to multiple machines, unlike most other similar frameworks; it has built in support for automatic gradient computation; it can partially execute subgraphs of the entire graph and it can add constraints to devices, like placing nodes on devices of a certain type, ensure that two or more objects are placed in the same space etc.

TensorFlow is used in several projects, such as the Inception Image Classification Model [27]. This project introduced a state of the art network for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014. In this project the usage of the computing resources is improved by adjusting the network width and depth while keeping the computational budget constant[27].

Another project that employs the TensorFlow framework is DeepSpeech, developed by Mozilla. It is an open source Speech-To-Text engine based on Baidu's Deep Speech architecture [11]. The architecture is a state of the art recognition system developed using end-to-end deep learning. It is simpler that other architectures and does not need hand designed components for background noise, reverberation or speaker variation.

We will present the most important utilized methods and data types from TensorFlow together with a short description for each of them.

A convolutional layer is defined like this:

```
conv2d(
    input,
    filter,
    strides,
    padding,
    use_cudnn_on_gpu=True,
    data_format='NHWC',
    dilations=[1, 1, 1, 1],
    name=None
)
```

Computes a 2-D convolution given 4-D *input* and *filter* tensors. Given an input tensor of shape [*batch, in_height, in_width, in_channels*] and a kernel tensor of shape [*filter_height, filter_width, in_channels, out_channels*], this op performs the following:

- Flattens the filter to a 2-D matrix with shape [*filter_height * filter_width * in_channels, output_channels*].

- Extracts image patches from the input tensor to form a virtual tensor of shape [*batch, out_height, out_width, filter_height * filter_width * in_channels*].

- For each patch, right-multiplies the filter matrix and the image patch vector.

```
tf.nn.max_pool(
    value,
    ksize,
    strides,
    padding,
    data_format='NHWC',
    name=None
)
```

Performs the max pooling operation on the input. The *ksize* and *strides* parameters can be tuples or lists of tuples of 4 elements. *Ksize* represents the size of the window for each dimension of the input tensor and *strides* represents the stride of the sliding window for each dimension of the input tensor. The *padding* parameter can be ''VALID'' or ''SAME''.

```
tf.nn.relu(
    features,
    name=None
)
```

Computes the rectified linear operation - max(features, 0). *Features* is a tensor.

```
tf.nn.dropout(
    x,
    keep_prob,
    noise_shape=None,
    seed=None,
    name=None
)
```

Applies dropout on input *x* with probability *keep_prob*. This means that for each value in *x* the method outputs the value scaled by 1 / *keep_prob* with probability *keep_prob* or 0. The scaling is done on order to preserve the sum of the elements. The *noise_shape* parameter defines which groups of values are kept or dropped together. For example, a value of [*k, 1, 1, n*] for the *noise_shape*, with *x* having the shape [*k, l, m, n*], means that each row and column will be kept or dropped together, while the batch and channel components will be kept or dropped separately.

6   Neural network structure used in experiments

For this project we used a convolutional neural network. As previously described this type of network makes use of convolutional layers, pooling layers, ReLU layers, fully connected layers and loss layers. In a typical CNN architecture, each convolutional layer is followed by a Rectified Linear Unit (ReLU) layer, then a Pooling layer then one or more convolutional layer and finally one or more fully connected layer.

Note again that a characteristic that sets apart the CNN from a regular neural network is taking into account the structure of the images while processing them. A regular neural network converts the input in a one dimensional array which makes the trained classifier less sensitive to positional changes.

The input that we used consists of standard RGB images of size 100 x 100 pixels.

The neural network that we used in this project has the structure given in Table 2.

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

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 32	64
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 64	128
Max pooling	2 x 2 — Stride: 2	-
Fully connected	5 x 5 x 128	1024
Fully connected	1024	256
Softmax	256	60

The first layer is a convolutional layer which applies 16 5 x 5 filters. On this layer we apply max pooling with a filter of shape 2 x 2 with stride 2 which specifies that the pooled regions do not overlap. This also reduces the width and height to 50 pixels each. The second convolutional layer applies 32 5 x 5 filters which outputs 32 activation maps. We apply on this layer the same kind of max pooling as on the first layer, shape 2 x 2 and stride 2. The third convolutional layer applies 64 5 x 5 filters. Following is another max pool layer of shape 2 x 2 and stride 2. The fourth convolutional layer applies 128 5 x 5 filters after which we apply a final max pool layer. Because of the four max pooling layers, the dimensions of the representation have each been reduced by a factor of 16, therefore the fifth layer, which is a fully connected layer, has 5 x 5 x 16 inputs. This layer feeds into another fully connected layer with 1024 inputs and 256 outputs. The last layer is a softmax loss layer with 256 inputs. The number of outputs is equal to the number of classes.

We present a short scheme containing the flow of the the training process:

```
iterations = 40000
```

```
read_images(images)
apply_random_hue_saturation_changes(images)
apply_random_vertical_horizontal_flips(images)
convert_to_hsv(images)
add_grayscale_layer(images)

define_network_structure(images, network,
    training_operation)

for i in range(1, iterations):
            sess.run(training_operation)
```

7 Numerical experiments

The dataset was split in 2 parts: training set - which consists of 37836 images of fruits and testing set - which is made of 12709 images.

The data was bundled into a TFRecords file (specific to TensorFlow). This is a binary file that contains protocol buffers with a feature map. In this map it is possible to store information such as the image height, width, depth and even the raw image. Using these files we can create queues in order to feed the data to the neural network. By calling the method $shuffle\_batch$ we provide randomized input to the network. The way we used this method was providing it example tensors for images and labels and it returned tensors of shape batch size x image dimensions and batch size x labels. This helps greatly lower the chance of using the same batch multiple times for training, which in turn improves the quality of the network.

On each image from the batch we applied some preprocessing in order to augment the data set. The preprocessing consists of randomly altering the hue and saturation, and applying random vertical and horizontal flips.

For the hue and saturation we use the TensorFlow methods: $random\_hue$ and $random\_saturation$. To further improve the accuracy of the network we converted each image from the batch to grayscale and concatenated it to the image. Thus the data that is fed into the network will have the size 100 x 100 x 4.

In order to be able to detect fruits from images we used the previously described neural network which was trained over 50000 iterations with batches of 50 images selected at random from the training set. Every 50

steps we calculated the accuracy using cross-validation. This showed steady improving of the network until reaching 100% accuracy on cross-validation. For the testing phase, we used the testing set and the calculated accuracy was 96.19%.

Some of the incorrectly classified images are given in Table 3.

Table 3: Some of the images that were classified incorrectly. On the top we have the correct class of the fruit and on the bottom we have the class that was given by the network and its associated probability

Apple Golden 2	Apple Golden 3	Braeburn(Apple)	Peach
Apple Golden 3 96.54%	Granny Smith (Apple) 95.22%	Apple Red 2 97.71%	Apple Red Yellow 97.85%
Pomegranate	Peach	Pear	Pomegranate
Nectarine 94.64%	Apple Red 1 97.87%	Apple Golden 2 98.73%	Braeburn(Apple) 97.21%

8 Conclusions and further work

We described a new and complex database of images with fruits. Also we made some numerical experiments by using TensorFlow library in order to classify the images according to their content.

From our point of view one of the main objectives for the future is to improve the accuracy of the neural network. This involves further experimenting with the structure of the network. Various tweaks and changes to any layers as well as the introduction of new layers can provide completely

different results. Another option is to replace all layers with convolutional layers. This has been shown to provide some improvement over the networks that have fully connected layers in their structure. A consequence of replacing all layers with convolutional ones is that there will be an increase in the number of parameters for the network [25]. Another possibility is to replace the rectified linear units with exponential linear units. According to paper [9], this reduces computational complexity and add significantly better generalization performance than rectified linear units on networks with more that 5 layers. We would like to try out these practices and also to try to find new configurations that provide interesting results.

In the near future we plan to create a mobile application which takes pictures of fruits and labels them accordingly.

Another objective is to expand the data set to include more fruits. This is a more time consuming process since we want to include items that were not used in most others related papers.

Acknowledgments

References

[1] S. Arivazhagan, N. Shebiah, S. Nidhyanandhan, L.Ganesan, Fruit Recognition using Color and Texture Features, Journal of Emerging Trends in Computing and Information Sciences, VOL. 1, NO. 2, Oct 2010 ⇒ 4

[2] S. Bargoti, J. Underwood, Deep fruit detection in orchards, IEEE International Conference on Robotics and Automation (ICRA), pp. 3626-3633, 2017. ⇒ 3

[3] R. Barth, J. Ijsselmuiden, J. Hemming, E. Van Henten, Data synthesis methods for semantic segmentation in agriculture: A Capsicum annuum dataset, Computers and Electronics in Agriculture, 144, pp.284-296, 2018. ⇒ 4

[4] T. Chan, L. Vese, Active contours without edges. IEEE Trans. Image Process. Vol. 10, pp. 266277, 2001 ⇒ 5

[5] H. Cheng, L. Damerow, Y. Sun, M. Blanke, Early Yield Prediction Using Image Analysis of Apple Fruit and Tree Canopy Features with Neural Networks, Journal of Imaging, Vol. 3(1), 2017. ⇒ 4

[6] D. C. Cireşan, L. M. Gambardella, A. Giusti, J. Schmidhuber, Deep Neural Networks Segment Neuronal Membranes in Electron Microscopy Images Proceeding NIPS'12 Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2, Pages 2843-2851, 2012 ⇒ 5

[7] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, J. Schmidhuber, Flexible, high performance convolutional neural networks for image classification, *Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 1237-1242, AAAI Press, 2011. ⇒ 5

[8] D.C. Cireşan, U. Meier, J. Schmidhuber, Multi-column Deep Neural Networks for Image Classification, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Providence, pp. 3642-3649, 2012. ⇒ 6

[9] D. Clevert, T. Unterthiner, S. Hochreiter, Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) *CoRR* abs/1511.07289, 2015. ⇒ 19

[10] S. Fernndez, A. Graves, J. Schmidhuber, An Application of Recurrent Neural Networks to Discriminative Keyword Spotting, Proceedings of the 17th International Conference on Artificial Neural Networks. ICANN'07. Berlin, Heidelberg: Springer-Verlag: pp. 220229, 2007. ⇒ 6

[11] A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, A. Y. Ng Deep Speech: Scaling up end-to-end speech recognition CoRR, abs/1412.5567, 2014 ⇒ 13

[12] J. Hemming, J. Ruizendaal, J. W. Hofstee. E J. Van Henten, Fruit Detectability Analysis for Different Camera Positions in Sweet-Pepper Sensors, Vol. 14(4), pp. 6032-6044, 2014. ⇒ 4

[13] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation. Vol. 9 (8): pp. 17351780, 1997 ⇒ 6

[14] K. Kapach, E. Barnea, R. Mairon, Y. Edan, O. Ben-Shahar, Computer vision for fruit harvesting robots state of the art and challenges ahead, Journal of Imaging, Vol. 3(1), pp. 4-34, 2017. ⇒ 4

[15] H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations Proceedings of the 26th International Conference On Machine

Learning, ICML 2009. 77. 10.1145/1553374.1553453, 2009 ⇒ 8

[16] M. Liang, X. Hu, Recurrent Convolutional Neural Network for Object Recognition, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Boston, pp. 3367-3375, 2015. ⇒ 6, 8

[17] D. Mumford, J. Shah, Optimal Approximations by Piecewise Smooth Functions and Associated Variational Problems, Commun. Pure Appl. Math. Vol. 42, pp. 577685, 1989. ⇒ 5

[18] P. Ninawe, S. Pandey, A Completion on Fruit Recognition System Using K-Nearest Neighbors Algorithm, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Vol. 3 (7), July 2014 ⇒ 4

[19] S. Puttemans, Y. Vanbrabant, L. Tits, T. Goedem, Automated visual fruit detection for harvest estimation and robotic harvesting, Sixth International Conference on Image Processing Theory, Tools and Applications, 2016 ⇒ 4

[20] M, Rahnemoonfar, C. Sheppard, Deep count: fruit counting based on deep simulated learning, Sensors, 17(4), p. 905-, 2017. ⇒ 4

[21] S. Ren, K. He, R. Girshick, J. Sun, Faster r-cnn: Towards real-time object detection with region proposal networks, In Advances in neural information processing systems, pp. 91-99, 2015. ⇒ 4

[22] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez & C. McCool, Deep-Fruits: A Fruit Detection System Using Deep Neural Networks, Sensors (Basel, Switzerland), Vol. 16(8), pp. 1222-, 2016. ⇒ 3, 4

[23] J. Schmidhuber, Deep learning in neural networks: An overview, *Neural Networks vol. 61, pp. 85-117, 2015* ⇒ 5

[24] Y. Song, C. Glasbey, G. Horgan, G. Polder, J. A. Dieleman, G. Van Der Heijden, Automatic fruit recognition and counting from multiple images, Biosystems Engineering, Vol. 118, pp. 203-215, 2014. ⇒ 3

[25] J. T. Springenberg, A. Dosovitskiy, T. Brox, M. A. Riedmiller, Striving for Simplicity: The All Convolutional Net, *CoRR* abs/1412.6806, 2014. ⇒ 6, 19

[26] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, Advances in neural information processing systems, *Twenty-Eight International Conference on Neural Information Processing Systems*, pp. 2377-2385, 2015, Montreal, Canada, 2015 ⇒ 5

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich Going Deeper with Convolutions CoRR, abs/1409.4842, 2014 ⇒ 13

[28] J. Xiong, Z. Liu, R. Lin, R. Bu, Z. He, Z. Yang, 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(4), pp. 969-, 2018. ⇒5

[29] H. M. Zawbaa, M. Abbass, M. Hazman, A. E. Hassenian, Automatic Fruit Image Recognition System Based on Shape and Color Features, In: Hassanien A.E., Tolba M.F., Taher Azar A. (eds) Advanced Machine Learning Technologies and Applications. AMLTA 2014. Communications in Computer and Information Science, vol 488. pp 278-290, Springer, Cham, 2014. ⇒4

[30] D. Li, H. Zhao, X. Zhao, Q. Gao, L. Xu, Cucumber Detection Based on Texture and Color in Greenhouse, International Journal of Pattern Recognition and Artificial Intelligence, Vol. 31 (08), August 2017 ⇒4

[31] Convolution in mathematics https://en.wikipedia.org/wiki/Convolution. last visited on 26.05.2018 ⇒6

[32] Deep Learning article on Wikipedia. https://en.wikipedia.org/wiki/Deep_learning. last visited on 05.05.2018 ⇒5

[33] TensorFlow. https://www.tensorflow.org. last visited on 05.05.2018 ⇒3, 13

[34] MNIST. http://yann.lecun.com/exdb/mnist. last visited on 05.05.2018 ⇒6

[35] CIFAR-10 and CIFAR-100 Datasets. https://www.cs.toronto.edu/~kriz/cifar.html. last visited on 05.05.2018 ⇒2, 6

[36] Fruits 360 Dataset on GitHub. https://github.com/Horea94/Fruit-Images-Dataset. last visited on 23.06.2018 ⇒1, 10

[37] Fruits 360 Dataset on Kaggle. https://www.kaggle.com/moltean/fruits. last visited on 23.06.2018 ⇒1, 10

[38] Britta O'Boyle and Chris Hall What is Google Lens and how do you use it?. last visited on 05.05.2018 ⇒2

[39] Google Lens on Wikipedia, $https://en.wikipedia.org/wiki/Google\_Lens$, last visited on 05.05.2018 ⇒2