Fruit recognition from images using deep learning

Horea MUREŞAN[1]
Mihai OLTEAN[2]

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 this kind of neural network.

Keywords: *Deep learning*, *Object recognition*, *Computer vision*

1 Introduction

The aim of this paper is to propose a new dataset of images containing popular fruits. Having a high-quality dataset is essential for obtaining a good classifier. Most of the existing datasets with images (see of instance the popular CIFAR dataset [12]) contain both the object and the noisy background. This will 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 neural network 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 [14] 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

[1] Faculty of Computer Science, "Babeş-Bolyai" University, Str. Mihail Koglniceanu, Nr. 1 400084, Cluj-Napoca, România, Email: mhsd1292@scs.ubbcluj.ro

[2] Faculty of Engineering and Exact Sciences, "1 Decembrie 1918" University of Alba Iulia, Strada Gabriel Bethlen Nr.5, 510009, Alba Iulia, România, Email: mihai.oltean@gmail.com

application is to correctly identify the objects. Currently the identification is based on a deep neural network.

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.

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 detail the objectives of this project, followed by a presentation of the concept of deep learning. Secondly we will shortly discuss a few outstanding achievements obtained using deep learning. In the second part we will present the framework used in this project - TensorFlow[10] 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 Deep Learning

Deep learning is a class of machine learning algorithms that use multiple layers that contain nonlinear processing units [5][6]. Each layer uses the output from the previous layer as input. Deep learning[9] algorithms use more layers than shallow learning algorithms. Convolutional neural networks are classified as a deep learning algorithm. These networks are composed of multiple convolutional layers with a few fully connected layers. They also make use of pooling. This configuration allows convolutional networks to take advantage of bidimensional representation of data. Another deep learning algorithm is the recursive neural network. In this kind of architecture the same set of weights is recursively applied over some data. Recurrent networks have shown good results in natural language processing. Yet another model that is part of the deep learning algorithms is the deep belief network. 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.

In the area of image recognition and classification, the most successful results were obtained using artificial neural networks [7][8]. This served as one of the reasons we chose to use a deep neural network in order to identify fruits from images. Deep neural networks have managed to outperform other machine learning algorithms. They also achieved the first superhuman pattern recognition in certain domains. 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. We will present a few results on popular datasets and the used methods.

Among the best results obtained on the MNIST [11] dataset is done by using multi-column deep neural networks. As described in paper [4], 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 [3] 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. 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 [1] an all convolutional network that gains very good performance on CIFAR-10 [12] 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.

3 Data set

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. 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 100x100 pixels. In order to diversify the image dataset, we created a program that fills the background of the images with uniform colors. We are currently developing a program that can generate non uniform backgrounds for the images. We also introduced a class of negative images, so that the network can classify items that are not fruits.In this category we introduced images with uniform colors that match the backgrounds used in the train and validation images as well as non uniform images to simulate various backgrounds on which fruits can be placed.

The resulted dataset used for training consists of 85260 images of fruits spread across 26 labels. The testing data is made of 38952 images. The images were obtained by filming the fruits while they are rotated by a motor and then extracting frames. The data set is available on GitHub [13]. The labels and the number of images for training are given in Table 1.

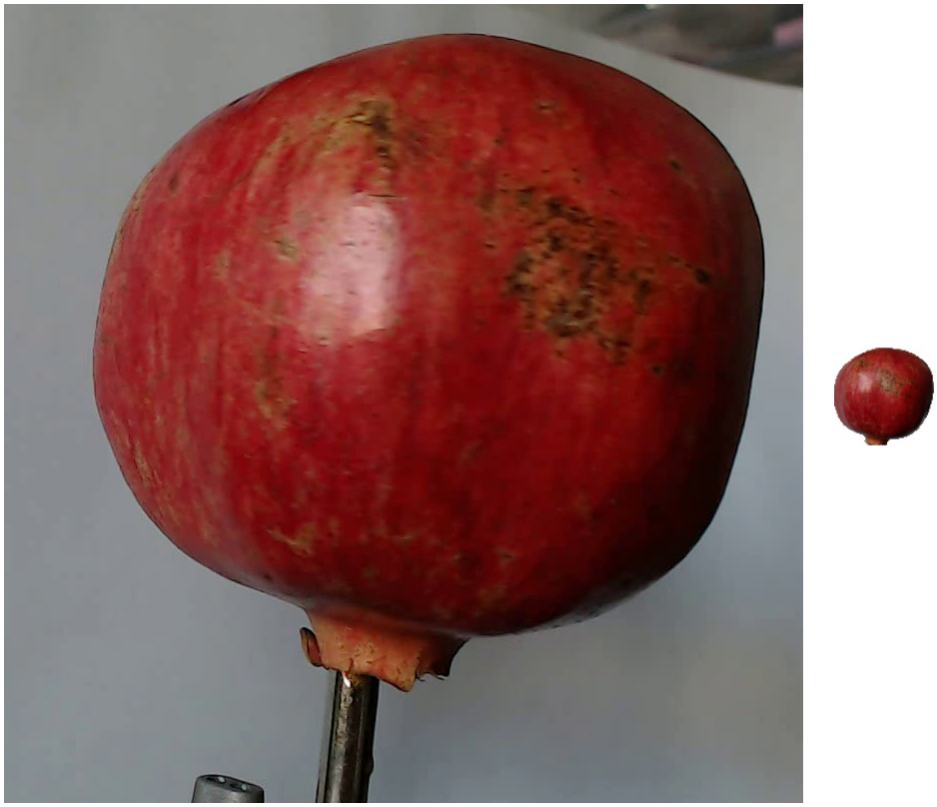

Figure 1: Left-side: original image. Notice the background and the motor shaft. Right-side: the fruit after the background removal and after it was scaled down to 100x100 pixels.

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 popular names 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**
Not Fruits	15750	15750
Apple Red 1	2952	984
Apple Red 2	2952	984
Apple Red 3	2574	864
Apple Red Yellow	2952	984
Apricot	2952	984
Avocado	2562	858
Braeburn (Apple)	2952	984
Cherry	2952	984
Apple Golden 1	2952	984
Apple Golden 2	2952	984
Apple Golden 3	2886	966
Granny Smith (Apple)	2952	984
Grape	2952	984
Grapefruit	2952	984
Kiwi	2796	936
Lemon	1476	492
Nectarine	2952	984
Orange	2874	960
Papaya	2952	984
Peach	2952	984
Peach Flat	2952	984
Pear	2952	984
Plum	2682	906
Pomegranate	1476	492
Strawberry	2952	984

4 Neural network structure and utilized framework

For this project we used a convolutional neural network. Such a network can be composed of convolutional layers, pooling layers, ReLU layers, fully connected layers and loss layers.

Convolutional layers consist 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 have a small 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. 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 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.

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. 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. 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.

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). 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.

The first layer is a convolutional layer of shape 5 x 5 x 3 with 128 outputs. On this layer we apply max pooling with a filter of shape 2 x 2 with stride 2. The second convolutional layer is of shape 5 x 5 x 128 with 64 outputs. We apply on this layer the same kind of max pooling as on

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

Layer type	**Dimensions**	**Outputs**
Convolutional	5 x 5 x 3	128
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 128	64
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 64	32
Max pooling	2 x 2 — Stride: 2	-
Convolutional	5 x 5 x 32	16
Fully connected	13 x 13 x 16	64
Fully connected	64	32
Softmax	32	26

the first layer, shape 2 x 2 and stride 2. The third convolutional layer is of shape 5 x 5 x 64 with 32 outputs. Following is a max pool layer of shape 2 x 2 and stride 2. The fourth convolutional layer is of shape 5 x 5 x 32 with 16 outputs. Because of the three max pooling layers, the dimensions of the representation have each been reduced by a factor of 8, therefore the fifth layer, which is a fully connected layer, has 13 x 13 x 16 inputs and 64 outputs. This layer feeds into another fully connected layer with 64 inputs and 32 outputs. The last layer is a softmax loss layer with 32 inputs. The number of outputs is equal to the number of classes.

In order to create our convolutional neural network we used TensorFlow [10]. 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.

5 Numerical experiments

The data was bundled into 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.

In order to be able to detect fruits from images we used the previously described neural network which was trained over 1000 iterations with batches of 150 images selected at random from the train 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 94.59Some 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.

Apple Golden 2  Apple Golden 3	Apple Golden 3  Granny Smith (Apple)	Braeburn(Apple)  Apple Red 2	Peach  Apple Red Yellow
Pomegranate  Nectarine	Peach  Apple Red 1	Peach  Apple Golden 2	Pomegranate  Braeburn(Apple)

6 Conclusions

This project tries to set up a start to an area that is less explored at the current time. During this project we were able to explore part of the deep learning algorithms and compare strengths and weaknesses. We gained knowledge on deep learning and we obtained a software that can recognize fruits from images. We hope that the results and methods presented in this paper can be further expanded in a bigger project.

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 [1]. Another possibility is to replace the rectified linear units with exponential linear units. According to paper [2], 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 training and validation sets to include more items. This is a more time consuming process since we want to include items that were not used in most others examples.

References

[1] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, Martin A. Riedmiller. Striving for Simplicity: The All Convolutional Net. *CoRR* abs/1412.6806, 2014. Striving for Simplicity: The All Convolutional Net.

[2] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) *CoRR* abs/1511.07289, 2015. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)

[3] Ming Liang, Xiaolin Hu. Recurrent Convolutional Neural Network for Object Recognition *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Boston, pp. 3367-3375, 2015. doi:10.1109/CVPR.2015.7298958.

[4] Dan Cireşan, Ueli Meier, Jürgen Schmidhuber. Multi-column Deep Neural Networks for Image Classification *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* Providence, pp. 3642-3649, 2012. doi:10.1109/CVPR.2012.6248110.

[5] Stuart J. Russell, Peter Norvig. Artificial Intelligence - A Modern Approach. New Jersey, Simon and Schuster Company, 1995. Artificial Intelligence - A Modern Approach.

[6] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks vol. 61, 85-117, 2015* doi:10.1016/j.neunet.2014.09.003.

[7] Dan Claudiu Cireşan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. *Twenty-Second International Joint Conference on Artificial Intelligence*, pp. 1237-1242, AAAI Press, 2011, doi:10.5591/978-1-57735-516-8/IJCAI11-210.

[8] Rupesh K Srivastava, Klaus Greff, Jürgen Schmidhuber. Training very deep networks, Advances in neural information processing systems. *Twenty-Eight International Conference on Neural Information Processing Systems*, pp. 2377-2385, December 07-12, 2015, Montreal, Canada. Training very deep networks, Advances in neural information processing systems.

[9] Deep Learning article on Wikipedia. https://en.wikipedia.org/wiki/Deep_learning. last visited on 04.09.2017

[10] TensorFlow. https://www.tensorflow.org. last visited on 04.09.2017

[11] MNIST. http://yann.lecun.com/exdb/mnist. last visited on 04.09.2017

[12] CIFAR-10 and CIFAR-100 Datasets. https://www.cs.toronto.edu/~kriz/cifar.html. last visited on 04.09.2017

[13] Fruit Dataset. https://github.com/Horea94/Fruit-Images-Dataset. last visited on 04.09.2017

[14] Sarah Perez. Google Lens will let smartphone cameras understand what they see and take action. last visited on 04.09.2017