[[1]](#footnote-1)

Iteration on the Fruit-360Dataset

Francesco Frassineti [fraf@itu.dk](mailto:fraf@itu.dk)

*Abstract*—In this paper I iterate o the work done on the Fruit-360 dataset available on Kaggle () to create a classifier of fruit and vegetables using convolutional neural networks. My project introduces improvements on the models proposed by the authors by trying new tunings of the parameters and by adding dropout layers between the fully connected layers of the Neural Network. I also train a model starting from the pre-trained model based on MobileNet V2. The results of the classifiers are then compared to the ones of the original paper and to a model based on logistic regression.

# INTRODUCTION

## The Dataset

T

he dataset used in this project is called Fruit-360 and can be downloaded from the addresses pointed by references. Currently, the dataset contains 90483 images of 131 different fruits and vegetables. At the time of writing *Mureșan, H., & Oltean, M. (2018)*, only 82213 images of 120 fruits and vegetables were available. The authors invite the reader to access the latest version of the dataset from the above indicated addresses.

The images were obtained by filming the fruits while they are rotated by a motor and then extracting the frames. A white sheet of paper was placed behind the fruits as background. Further work has been put to make sure the background was independent of the lighting conditions. Finally, fruits were scaled to fit a 100x100 pixels image. Each image contains one and only one fruit.

The dataset is already split between a training set (67692 images) and a test set (22688 images). The folder structure is the following:

* Images
  + Training
    - Apple Braeburn
    - Apple Crimson Snow
    - …
    - Watermelon
  + Test
    - Apple Braeburn
    - Apple Crimson Snow
    - …
    - Watermelon

## Motivation and Applications

I chose this dataset because I was interested in applying what I studied about Deep Learning to a real-life scenario starting from good quality data so that I could focus on the implementation, tuning and training of the machine learning model. Specifically, I wanted to work on Convolutional Neural Networks (CNNs) as they currently are the state-of-the-art classes of algorithms for image classification and detection. I also wanted to experiment with transfer learning, so I decided to train a network from scratch and compare it to a network that I could train from a pre-trained lightweight model such as MobileNet V2. I chose MobileNet V2 for its small size yet good performance, as I wanted my models to be small enough to work on mobile devices. Lastly, I decided to work on this dataset because, by reading *Mureșan, H., & Oltean, M. (2018)*, I realized that the authors did not use certain techniques in their model architecture that are known to improve generalization, so I wanted to see if by introducing them there would be improvements in the performance of the classifiers. Specifically, I’m referring to adding Dropout layers between each couple of consecutive fully connected layers.

My work may be applied across multiple domains. For example, the trained models could be inserted into a portable device to be used by visually impaired people to get help to recognize between different fruits and vegetables. It may also be applied to autonomous fruit harvesting in greenhouses or to the identification of out of place items in the aisles of stores.

# Theory

In this section I will briefly explain the key aspects of the theory used to back my work. For the sake of brevity, I will try to focus on the aspects of deep learning that are related to CNNs and image classification.

## Convolutional Neural Networks

Convolutional Neural Networks are a specialized kind of neural network for processing data with a grid-like topology (Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.). The most common application on CNNs is image classification and detection. Typical CNN architectures follow the following pattern:

* Sequence of Convolutional Layers (with ReLU as the activation function) and Pooling Layers one after the other
* Flatten Layer
* Sequence of Fully Connected Layers

Compared to fully connected NNs, CNNs take knowledge on the topology of the data into consideration and this allows the number of trainable parameters necessary to get the same performance of a traditional NN to be much lower.

### Convolutional Layer

Convolutional Layers are named after the convolution operation. A convolutional layer consists of groups of neurons that make up kernels. The kernels have a small size and they slide across the width and height of the input, extract high level features and produce a 2-dimensional activation map to be used as the input of the following layer.

### Pooling Layers

Pooling layers are used to:

1. Reduce the spatial dimensions of the representation
2. Reduce the amount of computation done in the network
3. Control overfitting

### Flatten Layer

Converts the output of the convolutional part of the CNN from a 2D to 1D representation so that it can be fed to the fully connected part.

### Fully Connected Layers

Each neuron from a fully connected layer is linked to each output of the previous layer.

## Dropout

Machine learning models suffer from overfitting when they achieve good classification results on the training set, but the model doesn’t manage to generalize well on the test set.

Overfitting is a very common problem when training a classifier and *Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014)* proposed Dropout as a possible way to reduce it. It’s a technique that has been proven to greatly improve the property of generalization in NN-based models. Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel. During training, some number of layer outputs are randomly ignored or “dropped out.” This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer. In effect, each update to a layer during training is performed with a different “view” of the configured layer. Dropout has the effect of making the training process noisy, forcing nodes within a layer to probabilistically take on more or less responsibility for the inputs.

This conceptualization suggests that perhaps dropout breaks-up situations where network layers co-adapt to correct mistakes from prior layers, in turn making the model more robust. As a rule of thumb, when introducing dropout to the network, it’s suggested to double the amount of neurons in the layer and use a *dropout-rate* of 0.5 for hidden layers and 0.8 for input layers.

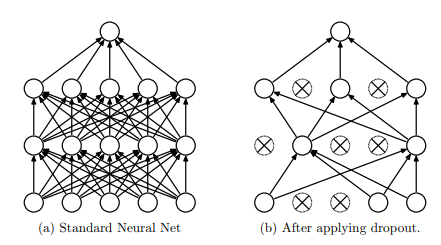


Figure : Dropout

# The Method

The Data

# Results

Fjshgdjahsd

# Discussion

Hsdfjshdfj

# Conclusion

References

1. Mureșan, H., & Oltean, M., “Fruit recognition from images using deep learning” (2018).

1. [↑](#footnote-ref-1)