# Everis - Project

Mykhaylo Marfeychuk - ist194039

June

Github Project:
`https://github.com/Mika412/Everis-Challenge`
Faster R-CNN Colab:
`https://colab.research.google.com/drive/1DNU4MOWGNFNI38IdHZ1Y915huenltQkD`
Sliding Windows Colab:
`https://colab.research.google.com/drive/1vVSYrPI4icGTDK7ixjdlftE2KQqr3IeN`

## 1 Introduction

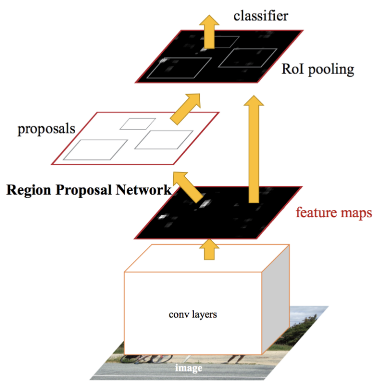
The goal of this challenge is to implement a Neural Network based algorithm to detect and classify fruits in the image. To attack this challenge two different approaches were used, Sliding Windows and Faster R-CNN. Sliding Windows approach is an extension of the homework 5 for the Machine Learning class where the goal was to implement a model to classify fruits. To this model was added a sliding windows algorithm to classify different areas of the image and combine them into a solution. The Faster R-CNN is a more sophisticated algorithm to detect objects. Due to Region of Interest approach and the ResNet50 model it enables to reliably detect objects in the image in a fast manner.

## 2 Approach 1 - Faster R-CNN

Note: This approach requires Keras 2.2.0. or older. Won't work with newer version.

### 2.1 Algorithm Description

Faster R-CNN is a algorithm build based on previous iterations, R-CNN & Fast R-CNN, which uses the same process, but saves on region proposal time. The algorithm is split into 3 parts, feature map extraction, region proposal and the classifier. The feature extraction bit, receives the image and passes throw a CNN to generate a feature map, this map is then passed to a Regional Proposal map, which generates coordinates for possible objects. Each of this proposals is passed through a classification network, which classifies the type of fruit. This is the general process of the Fast R-CNN. The Faster R-CNN differs where the classifier receives the proposed regions plus Region-of-Interest(RoI) Pooling data.



## 2.2 Dataset

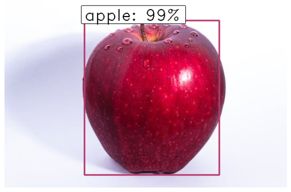
The Faster R-CNN algorithm receives the image, and the (x, y, width, height, class) for each object in the image. To generate the dataset you need to use a labeling tool, like LabelImg tool[1], which saves the annotations as XML files in PASCAL VOC format. These XML files still need to be converted to the correct format, for this I've written a python script called *convert.py* which does this.
As the goal of this project is the algorithm and the lack of time, I've used a pre-annotated fruit dataset [2]. The dataset contains three fruits, Apple, Banana and Orange, which were still annotated in PASCAL VOC format, so I used the python script to convert into a proper format.

## 2.3 Implementation

The implementation is based on Keving Bardool implementation[3] of the original paper. The implementation is written in Keras. For the feature extraction, a ResNet50 model is used, there is also the possibility of using the VGG model, but the ResNet50 is a lot faster and smaller in size. The model receives a array of images, which it then resizes to a predefined size, in this case 600x600. Then it does data augmentation by flipping the images into different orientation. After it is processed it outputs a vector of coordinates and it's respective class and confidence.

## 2.4 Results and Improvements

With the small dataset that was provided, the algorithm was able to recognize the desired fruits around 90% of the time, even if the fruits are drawings of the original. To improve the accuracy further it needs to receive a bigger dataset, or additional methods of data augmentation.
One of noticed problems is when the proposed regions in the algorithm overlap, this creates overlapping detections of the same fruit, as shown in the figure blah, this occurs in some small sized images. To solve this a region filter could be added to solve the problem.



# 3 Approach 2 - Sliding Windows

## 3.1 Algorithm Description

This approach relies on two algorithms, a Classification Neural Network and a Sliding Window approach. The classification network, is a custom network that was build for the Homework 5 of Machine Learning class. A sliding window with varying sizes is run through the image. Each of the windows are passed to the classification model which outputs the fruit that is present on the image. These are then clustered to draw the bounding boxes.

## 3.2 Dataset

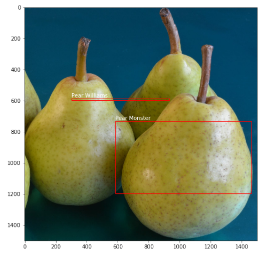
For the dataset, a modified Fruit-360 dataset was used. This dataset usually is just for classification and has only the images of the fruits. For this a solution had to be implemented.

## 3.3 Implementation

To classify the image, a simple CNN model was used which had a high accuracy. The network received a image of 100x100 pixels and gave a softmax value vector with the probabilities of the detected fruit. To this vector, was also added a class of "None" which means no fruit were found. To train this "None" class random images were used. To detect the objects, a sliding window approach was used. In this approach the image is subdivided into varying smaller areas which are run through the network, which at the end gives a matrix of probabilities for the fruit detected in the area. This is run with different window sizes, and the all the probability matrices are overlayed. The probability is clustered which gives the area where the specific fruit is.

## 3.4 Results and Improvements

The results are not very impressive, it is able to recognize that there is a fruit, but has difficulty localizing it correctly because of the algorithm that was used. One of the biggest improvements would be to improve the dataset. The images are extremely similar and all have the white background. Another improvement would be to the sliding window approach, the speed could be improved, if it was run on the GPU, or instead the approach would be replaced by the OverFeat algorithm that would greatly improve the detection and pinpointing more precisely the location of the fruits.



## References

[1] LabelImg Tool,
`https://github.com/tzutalin/labelImg`

[2] Annotated fruit dataset,
`https://www.kaggle.com/mbkinaci/fruit-images-for-object-detection`

[3] Faster R-CNN implementation,
`https://github.com/kbardool/keras-frcnn`