Machine learning and image recognizance in renewable energies

GILLIER LILIAN

May 2, 2025

A dissertation submitted in the partial fulfillment of the requirements for the degree of

Master of Science

In Renewable Energy Development

School of Energy, Geoscience,
Infrastructure and Society

Heriot-Watt University Orkney

Contents

1 Introduction **4**

2 Renewable energy environment **5**

2.1 Biofouling 5

2.2 Identify damages on installations 5

2.3 Environmental studies 7

3 Machine learning ? **8**

3.1 Supervised learning 8

3.2 Reinforcement learning 9

3.3 Unsupervised learning 9

3.4 Operation 9

4 Used packages **13**

4.1 OpenCV 13

4.2 Tensorflow 14

4.2.1 How is it working ? 14

4.2.2 How to create a neural network ? 15

5 Setup **19**

5.1 Installation 21

6 Installation of all needed packages **22**

6.1 Tensorflow 22

6.2 OpenCV 24

6.3 Protocol buffers (Protobuf) 26

6.4 Setup Tensorflow directory structure 27

7 Creation of the model **30**

7.1 Collect data 30

7.2 Labeling 32

7.3 Training 36

7.4 First tests and behaviour of the model 37

8 Training in bigger scale **39**

9 Advantages and disadvantages **42**
9.1 Advantages 42
9.2 Disadvantages 43

10 Conclusion **44**

11 Annex 1 **46**

12 Annex 2 **47**

1 Introduction

For some years, technological advances have highlighted new computer and statistical means called machine learning. This technology has already had many approaches especially in the field of medicine where machine learning has found its place. It allows to identify and to diagnose diseases in compliment of the work of the doctor, to personalize the treatments according to the more or less positive reactions of a patient or in the field of the research of drug. More recently, we have seen the progress of some companies wanting to innovate in the field of transport through machine learning. This is the case of Waymo or Tesla which accelerates their research to make their vehicle autonomous and thus increase the safety of vehicles [1].

The future increasingly uses these technologies, not to replace men's work, but to assist, secure and help protect environments. If the ethical barrier is well placed and does not go beyond the fundamental human right, this technology has everything to be accepted by many organizations, societies, communities. Machine learning allows to detect what an human being can't. Because of the precision of the object observed, or simply because a human can be inattentive during few seconds. Technologies has always been here to reduce human failures and will continue in this way.

The purpose of this document is to study the possible applications of machine learning in the field of renewable energies, in particular on image recognition, the aim is to understand how it works in the first place, then to be able in a second time to create a model by myself. The aim is to be able thanks to an original database, to train a model of recognition whatever the object to serve a specific goal.

2 Renewable energy environment

The field of renewable energies and all related studies induces the need to authenticate many elements, whether in environmental studies or to help secure systems. Here are the main areas of applications that I have identified where image recognition could be a beneficial tool.

2.1 Biofouling

Biofouling is defined by the colonization of any surface, which may or may not be alive, in aqueous media by living organisms (Railkin 2004 [2]). The importance of studying biofouling is important in many areas such as aquaculture or fish farming where the environment can be colonized by many organisms, which is a form of biofueling. The colonization of environments by bacteria can cause corrosion of various elements depending on the substances consumed by these bacteria, by their presence or by blocking porous materials and this can disrupt the productivity of the media.

In the marine environment, each surface that is immersed in an environment that contains microorganisms is subjected to a biofouling process. This surface is found gradually covered by a primary biofilm until the installation of larvae, invertebrates or even algae.

The image recognition applied to biofouling could firstly, to identify the species that colonize the offshore installations sometimes reefing artificial reefs, this is the case for example oil platforms or wind turbines and in a second time to anticipate their development and spread.

2.2 Identify damages on installations

As we know, once a photovoltaic or solar installation is installed, the main costs are the maintenance costs. However fault detection on these types of installation is often detected during visits effected once a year or when the defect in question has repercussions on the electrical production. For example, a solar panel that suffers a weather and is dirty can lower the production of the entire solar power plant. But when is this problem detected? If the dust covering it is present in great quantities then the power output of the plant will drop

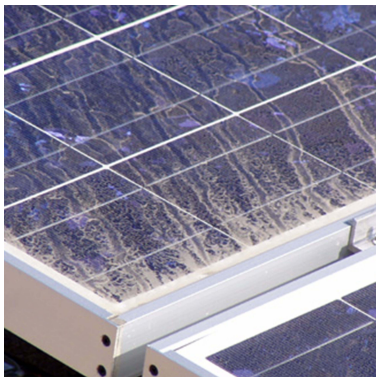


Figure 1: Dust disturbing production of a solar panel

but in cases where the plant is not equipped with a monitoring system to detect a possible fault, it will be necessary to wait until the problem is identified by an expert. Train a model to recognize the presence of obstruction in front a panel could increase the annual production of a solar central and identify defaults.

A similar approach is possible with wind turbines to which it is important to pay particular attention to the structural damage they may suffer. It is essential to be able to detect any defects that appear on the blades to prevent a bigger technical problem. A similar study (ASM Shihavuddin [3]) has already been carried out where its author has identified the different elements present on a wind turbine blade (Leading edge erosion (LE erosion), Vortex Generator panel (VG panel), VG panel with missing teeth, and lightning receptor) in order to to be able to draw information from it (see figure 2).

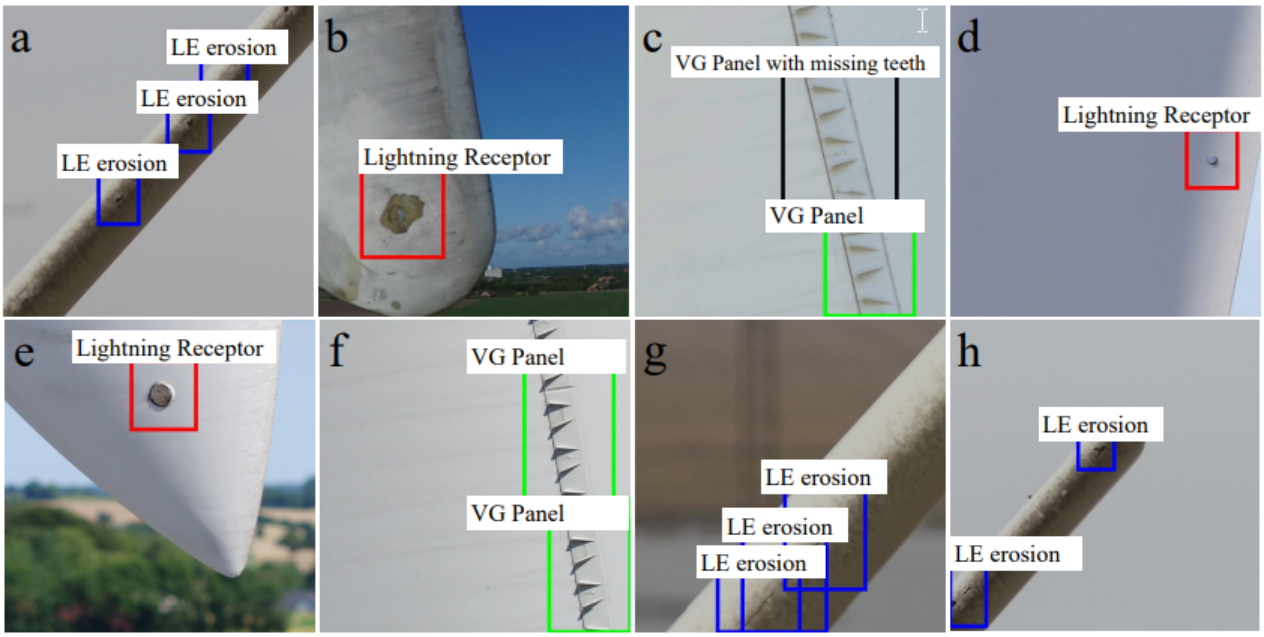


Figure 2: Different elements that can be identified on a wind turbine blade

2.3 Environmental studies

An environmental study aims to identify the fauna and flora present in an environment susptible to host a project, to list the species and assess the possible impacts and risk of the implementation of this project. It is then up to the specialist to assess on site the plant and animal species present and to study during behavior. The image recognition would firstly be able to list the species present in an effective way without forgetting, to know their exact number. For example, by recording bird passages throughout a day, this device could count and define the number of individuals and which species are present on the scene. On a larger scale, using multiple cameras, machine learning could define the movements of animal species.

During the operation of wind farms for example, it is also interesting to continue to observe the movements of animal species in the territory and the development of the vaginal species.

One more advantage is to use machine learning and image recognition during night. Using a thermal camera, is possible to use the images with machine learning. The interest of this system and to be able to observe the movement of animal species during the night and to identify a predatory behavior. Some species only hunt at night and machine learning can help predict these behaviors.

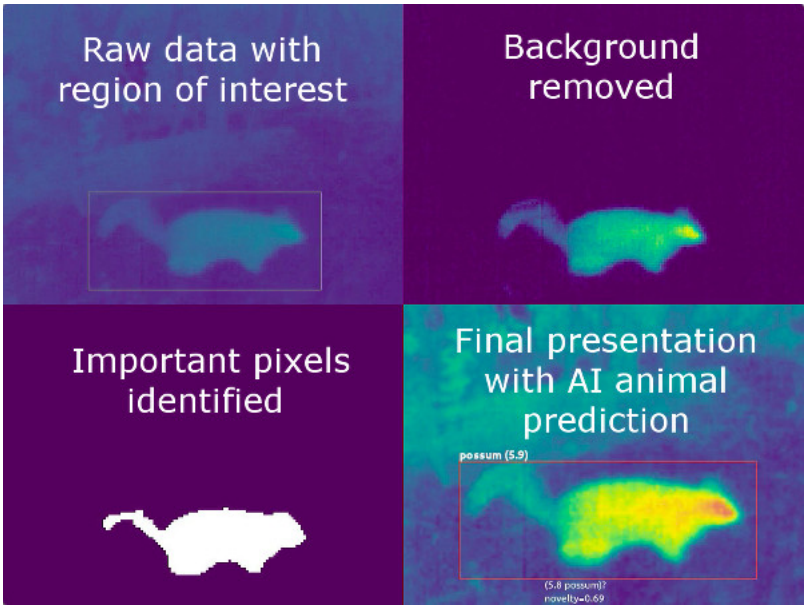


Figure 3: Possum recognition during the night [4]

3 Machine learning ?

Machine learning is a field of study based on a statistical approach to give to computer the ability to "learn" from data. In other words, improve its performances to find solutions to issues without be initially programmed to these tasks.
More often, machine learning is split in two phases. The first one is about to estimate a model from each data, called observations,available on a finite number, during the conception of the system. The estimation must be something simple as practical task as translate a text, estimate a density of probability or again to identify a dog on a picture, or again at a larger scale, participate to the drive of a autonomous vehicle. We call this phase the "training" or "learning" and it's realized before use the model. The second phase is to put it in production, with a determined model, we can now submit more data to obtain a desired result. Sometime, some systems can continue to learn from data during the production phase if they can obtain a feedback on the quality of the results.
We can characterize the first phase according different ways.

3.1 Supervised learning

If the answer is known to the task made by the system, we call that a supervised learning. To solve a problem with supervised problem, we perform the following steps :

- First is to determined the examples that will be used in the model. It's important to clearly know what will be the objective. For example, if we want to identify apples in a general term, it's important to have data not only from one variety but from different species to be able to identify any apple
- Then, we must determine the input feature representation of the learned function. The final precision will depend of how the data on the model are defined. In formations are stored in a vector for each object so it's important to defined the right number of characteristics we want to keep. Not too much to have an acceptable time calculation and not too few to be able to identify what we want.
- Determine the algorithm and how ithi will work
- The final step is to check the accuracy of the function. Because is this case we have the prediction of the algorithm and because we also know the real result, we can have an number of the accuracy of the model.

3.2 Reinforcement learning

However, if the model is learned incrementally base on a reward received by the program for each action made, we call it a reinforcement learning. In this scenario, the agent try several solutions (exploration), observe how the environment reacts and adapt its behaviour (variables) to find the best strategy. The thing is to find the right balance between the short and long term reward. It's use for example to learn to a reobot how to walk, in the case of autonomous car or again go game.

3.3 Unsupervised learning

The last category is the unsupervised learning. In this case, we have no initial models, so no class defined and no number of class. Here, we ask to the program to class by its own object in categories because it will identify similar characteristics on several non classified data.

3.4 Operation

To understand easily how machine learning is working, we will assume that we have to create a system able to answer if a mixture is lemon or strawberry syrup. To answer this question, we have to build a model during the training process. We want to create an accurate model able to answer to our question in most case as possible. But to train a model, we need to collect data. We need at this step to choose wich characteristics are interesting to be taken in account to make the distinction between the two drinks.

(a) Lemon syrup

(b) Strawberry syrup

Figure 4: Differentiation of syrups

For this case, we can choose two simple characteristics : the color (by knowing the wavelength of light) and the acidity (by knowing the pH). The best thing is to choose only one criteria, that is possible here because the color is quite characteristics for

each drink. The first thing to do will be to find bottles of strawberry syrup and the equipment needed to make our measures as a spectrometer to measure the color and a ph-meter to measure the ph.
Once we have all our equipment we will have to collect the data. It's one of the most important step because the quality and the quantity will influence on the results, in our case it will be the color and the ph.In our case for example, to have a good model, we can make a dilution with different concentration of syrup to have a larger quantity of data to be sure that we will be able to identify the syrup with a better accuracy.

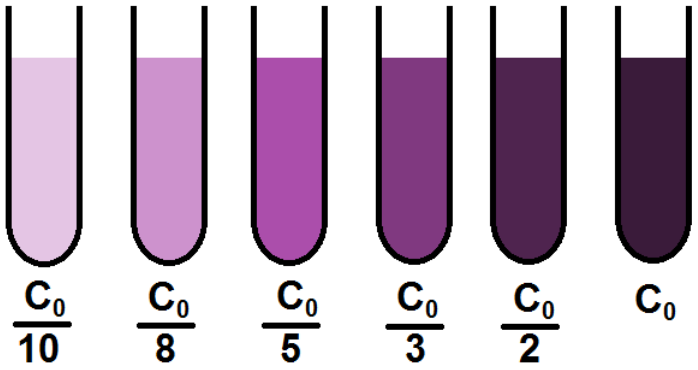


Figure 5: Scale dilution

Once we have our measures made, we can construct a tab of data as in 1.

Color (nm)	Acidity (ph)	Syrup ?
650	4.2	Strawberry
580	3	Lemon
600	3.1	Lemon
700	4	Strawberry

Table 1: Examples of measurements on syrups

This tab will be our training data. More analysis we do, more we will have precision in our result. So we have a large tabular full of data that represent or lemon or strawberry syrup in a random order (to not affect the classification). During this step, it's interesting to visualize the data we train to see if we don't create relations intentionally between measures. For example if we train more one data than another, the final model will identify this data twice than the other one and half of the prediction could be wrong.We also need to split the data in two parts. The first one will be used to create the model and the second one to evaluate the accuracy of our model. The aim is to not submit the same data to evaluate the model that the one use to create the

model, because the model could have memorized the data. Some kind of analyze also need sometime of some adjusting operation as normalization, error correction, vectorization,etc. In this case, we have a tab of data that don't need it.

We need then to chose a model. There are a lot of models created by the scientific society more or less complicated adapted to different cases as image data, music data, numerical data, etc. Be cause we have only two data to analyze, color and acidity, we can just use a linear model, easy to use.

The next step is the training, the heart of the machine learning. We will here use more and more data to improve the capacity of our prediction model. We can make here an analogy with human behaviour. For example when we learn traffic laws. First we make mistake by answering questions but more we practice, we learn from our mistake and we finally succeed to improve our performance and pass the exam. It's similar in this case with our drink. For example in our linear model, the corresponding function is :

$$y = a * x + b$$

To change the position of the curve, we need to adjust (or train) a and b values. It's the only way to adjust the output value (y) according the input value (x). In machine learning, these data are most of the time classify in matrix. The matrix with a value is named "weighs" matrix and matrix with b data the biases matrix. This step is repeat for each time we submit a data to adjust the matrix. If we take a look to our case, we first begin to draw a random line. Then, at each step (submit a new data), the line becomes closer and closer from a separation of the two kinds of syrup.

After this done, we can now check if the model has a good accuracy. We use the second set of data when we split the data in two parts at the beginning. Thanks the evaluation, we can test our model and we can see by this way how the model could behave with the unknown. The proportion between the number of data to train the model and evaluate it is of the order of 70/30 or 80/20.

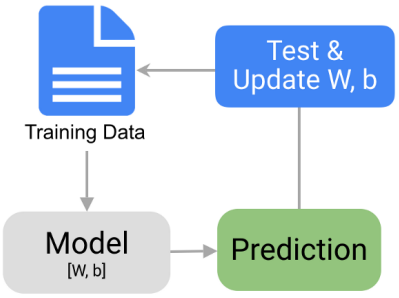


Figure 6: Training data

Tom improve the training, one possibility is to tuning the parameters as the number of data or again put several times the data in the training data. One important

parameter is the "learning rate" that defines how far we shift the line during each step. For complex models, initial conditions have a major role (difference between start with zero as initial value or a given data). But a good model is a subjective thing, we have to define what will be acceptable for us according the result we want obtain. For our case, it will be acceptable to have a success rate of 80% but this number will no be a good number in the case of an autonomous vehicle for example according all the security issues in this kind of project.

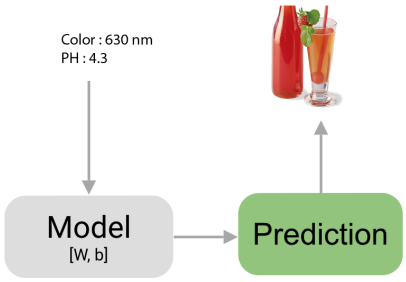


Figure 7: Prediction using the model

The final step is to use the model in prediction. We will be able to predict which kind of syrup is. In my case, I will use Tensorflow to do machine learning with images, I will detail in a following section how Tensorflow is working by taking the same concepts I described above. Here is the interest of machine learning, is quite easy for a person that had a formation to identify and classify some samples but there is a point where models are too complex to be evaluated by only one person or the data are too numerous to be dealt by just one human being. Machine learning allows us to deals with this complexity and number of data. Also, the possibility to improve models by training them is an advantage.

4 Used packages

4.1 OpenCV

OpenCV is a free graphics library, originally developed by Intel, specializing in real-time image processing. That's why I wanted to use it, to detect in real time the elements that I want to identify and an important element.

The first examples of use on OpenCV were applications dealing with image analysis on road transport. This is explained because the transport offers a large panel of possible data to analyze and make first tests. For example, we have seen applications to count the number of cars on the highway or to define the number of places available on a car park (see figure 8).

Figure 8: Parking application of OpenCV [5]

The benefits of OpenCV are numerous. First of all, it's a multiplatform library. It can be used on Linux, Windows, iOS and Android. In addition, OpenCV offers its users a multitude of algorithms for large tasks such as 3D model extraction from image, motion tracking or shape contouring. Because of its use by multinationals and many start-ups, it gives a trust to its user who knows that the technology he uses is part of the professional world. Finally, OpenCV is free of rights and has a very active community and can quickly get answers to a problem and quickly develop an operational tool.

4.2 Tensorflow

Tensorflow is an open source library of machine learning [6]. Created by Google in 2011, it permits to use Machine learning and deep learning applications. Basically, it's a tool permitting to solves complex mathematics issues. Searcher use it to create and develop learning architectures to turn them into software. We can create thanks to Tensorflow, programming systems in which calculation are represented as diagrams. Tensorflow is today the framework the more use for deep learning. Why ? First because it's multi-platform. We can use it on Linux, OS Mac and also on mobile as Android and iOS. Moreover, It's allows to create API (application programming interface used to create software) in Python, C++, Java and Go. Time f compilation are short in C and C++. Calculation can be made on CPU and GPU (CPU is the central processor of the computer and GPU is the processor equipped of the graphical card, it's important to be able to manage it for a good compilation). Additionally, Tensorflow has a large database with a lot of example and tutorials to be able to understand how it's working according several uses.

4.2.1 How is it working ?

The particularity of Tensorflow is that calculation can be represented as execution graph (see figure 9), each node representing an operation and each link representing a Tensor. An operation can be a basic math calculation or again a complex matrix issue in several dimension. A Tensor is for example, and it will be in our case, an set of images, that is a tensor in 4 dimensions (size -number of images-, high, width and number of channel of representation -3 for a RGB image-).

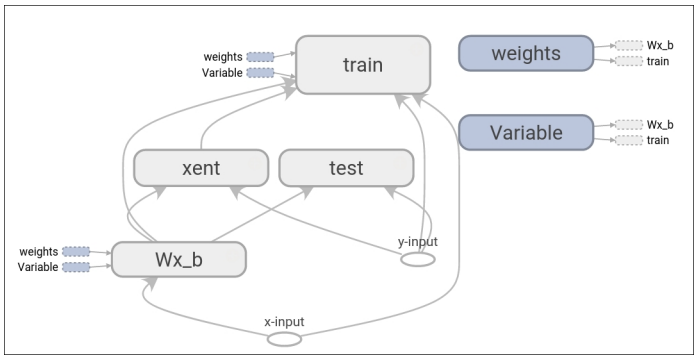


Figure 9: Example of calculation graph in Tensorflow

To make calculation, a graph as to be lunched. We have to create a session in which operation will be made. The session manage the computer memory and give methods to run the operations. To understand it, we will use the following figure :

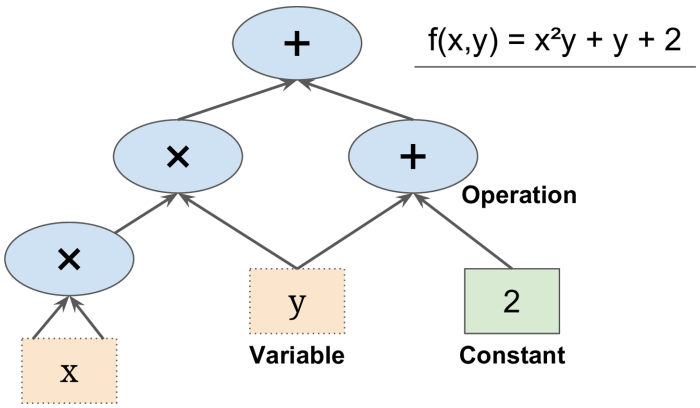


Figure 10: Example of simple operation in Tensorflow

Here, there are easy operation as multiplications an additions.Input variables are x and y. And once they are defined, calculations can be made. One other interest of this structure is that calculation can be split in several parts on several machines. For example here, the first calculation that is the multiplication of X by itself could be made on another computer and we could make the rest of the calculation on this one. Of course here it's not necessary because of the small system but when the complexity of the system is too important for one computer, it's possible to cut the model in several systems according the organization of the structure. We can by this way also win in compilation time.

Of course, when we are writing a program using Tensorflow, we are not creating directly graph like that but by coding. However, a tool named Tensorboard exists to visualize created graphs and control the evolution of machine learning.

4.2.2 How to create a neural network ?

A neural network is a mathematics model use to solve and optimize problems. A neural network is composed of neuron, that is an input, for example x, calculate something with this input (for example multiply it by a and add y) to produce an output (here ax +b). This result is then send to an activation function that is a non-linear function whose role is to activate the neuron. A lot of function are existing, for example the following one named Sigmoid :

$$y = \frac{1}{1 + e^{-z}} \tag{1}$$

Then, if stack neurons on a single line, we call that a layer. As we can see on the figure 11, greens neurons are representing one layer, forst layer of the network, that we can see as the input layer. In the same idea, we found the last layer that is the output

layer. Between these extremities, there are layers called hidden layers.

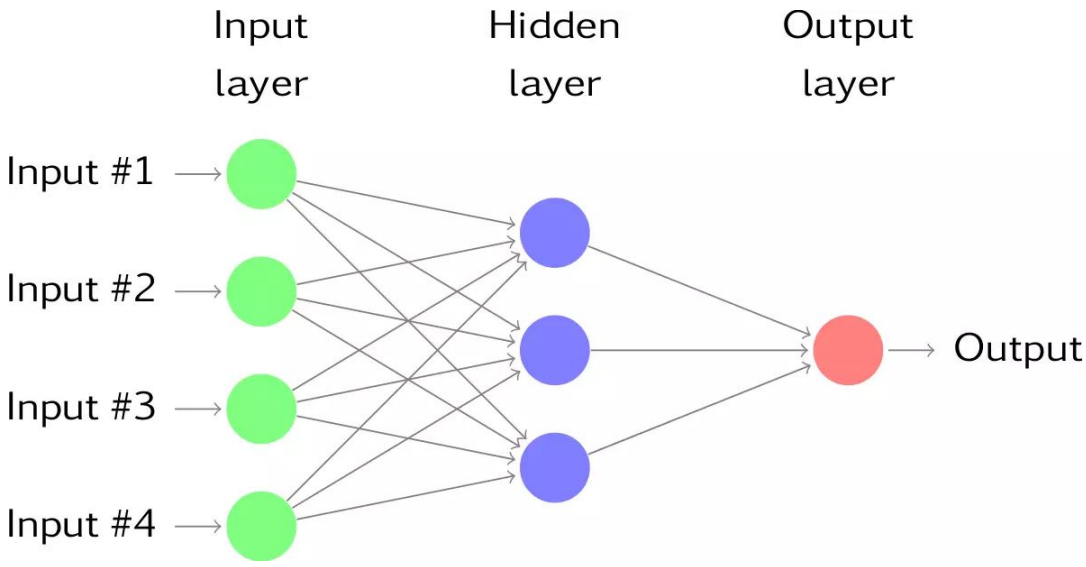


Figure 11: Formation of a layer

In a layer, all neuron compile the same kind of operation. However between layers, operations differs. I will describe the 3 most famous one. First, the convolution layer used to filters or find paterns in signals. The convolution operation made by this layer "change" the size of an initial data. For example, let's pretend the input data is an image so with 4 dimensions as we say before (size of 7x7x3) and that the filter is 3x3x3 size. The convolution operation will represent the image in a data set of the size of the filter. Let so consider the following case. Most of the time we use more than one filter in convolution operation to reduce more than one time the initial image.

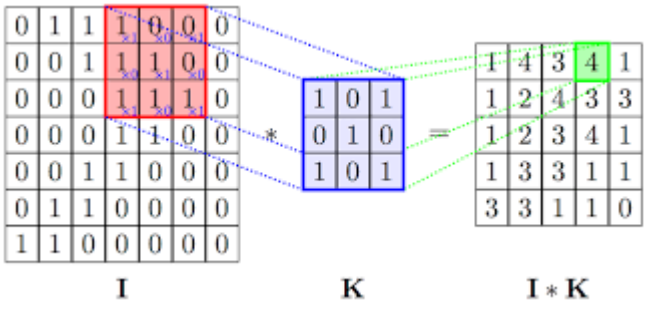


Figure 12: Convolution operation

On figure 12, K is the filter defined by 1 and 0. To create the convolution matrix, we slide this filter on the image I. If a 1 value of the image match with a 1 of the filter then we count one. We repeat this operation for the nine value of the filter and we add them all. We have a final value that we put in the convolution matrix. We repeat it as

many time we can slide the filter on the image. We finally obtain a new matrix with a smaller dimension than the initial one.

A second kind of layer is the pooling layer, often used just after a convolution layer to reduce the size of an image in width and height. It use to gain in time computation by reducing information size. It will not be used if a system has to recognize and locate object in a spatial environment (dimension, position), like play Tetris for example. In figure 13, maw pool is used. The operation is quite similar to convolution layer but here we don't move filter and we keep the higher value.

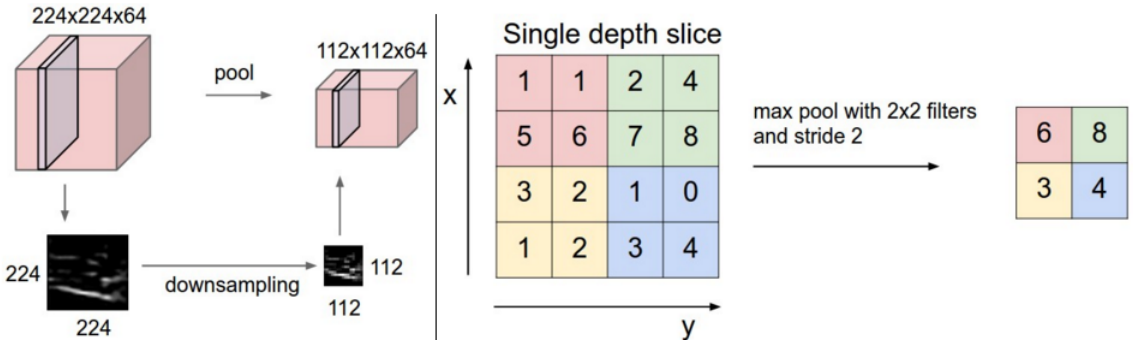


Figure 13: Pooling layer

Basically, use a 2x2 layer will reduce by half the size of the image. It's important to make the right choice of filter in convolution or in pooling, this will characterize the precision of our recognition.

A last important layer is the fully connected layer that is simply a neuron receiving as input, information of all neurons of the previous layer. This neurons operates a matrix multiplication followed by bias offset.

The architecture of a neural network is very important in machine and deep learning. It'"s a complex issue to find the right architecture for the right use. Without experience, it's impossible to create our own one. But standards exist as GoogleNet, VGG, AlexNet,etc. After have designed on neural network, the second step is to identify our main parameters (weight and biases). These parameters will depend of what we are looking for to recognize. As we talked earlier, color or size can be one of them. Backward propagation is a process that permits to identify these parameters. Basically it's started with random parameter and keep changing them until have the correct answer for each training image. More efficient methods are available but taken in charge by Tensorflow, it's not a thing we have to change by ourself.

If we imagine a case where, to back in our first example, that the output of a training image is 10% for strawberry and 90% for lemon. But if we go back to change parameters to change this probability ratio. A variable can be used to evaluate the accuracy how fast we change parameters during the training, this is the learning rate. The final goal is to maximize the number of correct answers of the network. We define a variable named cost able to show if we're taking the right direction. The cost is defined as more the cost is going down, more the precision of the network is raising. So we keep going iterations changing parameters until the cost is not going down anymore. The more easier to define the cost is mean root square cost. We define it by the following equation :

$$cost = 0.5 \sum_{i=0}^{n} (y_{actual} - y_{prediction})^2 \tag{2}$$

Where yprediction is the vector where there are outputs of training images and yactual is the vector with real values. So smaller the distance between this two vectors is near to zero, higher will be the precision.

In reality, calculation to evaluate the accuracy is more complicated than that, for example using entropy calculation, all managed by Tensorflow. Once we obtain the accuracy we want we can finally save the output binary file, this is the model. When we have a new iamge that we want to classify, we load the model that will do a prediction of what it is.

For compilation simplicity, we do not submit a large amount of image to train a model but we divide them in batches named batch-size not all submitted at the same time. Consequently, it will take a lot of time for the network to read all the data.

5 Setup

I wanted for this project, be able to have a device that will be transportable. The aim of this project is to move the system easily from a place to another. Moreover, I wanted a device with a Linux operating system, because I found it more simple to use all software a described before on Linux. A lot of devices exist to answer to my need. They are mini computers easily movable with a battery. This kind of computer is a raw computer without any device. On the other hand, they are equipped with jacks for peripherals such as USB jacks for connecting the mouse and keyboard or HDMI jacks for connecting a screen. So with this kind of computer, the aim will be to connect peripherals to configure the device home and then remove it to deploy the system.

For price reason and access facility, I chose to use a Raspberry Pi, a single-board computer. The first aim of the creation of Raspberry is to give to people the opportunity to have a cheap computer, teach computer sciences to students and in developing countries. This computer is sold "naked", we can install any operating system on it. Several Linux/GNU operating system has been developed for the Raspberry Pi (as desbian, I will describe it later).

The Raspberry B (model I use) is composed of the following components :

- Ethernet connector to do network connection in local or to connect it to internet
- Audio connector to plug speaker or headphones
- HDMI plug to connect a screen and have a visual interface
- 4 USB plug to connect keyboard, mouse, or other peripherals
- SD card reader where the operating system will be installed

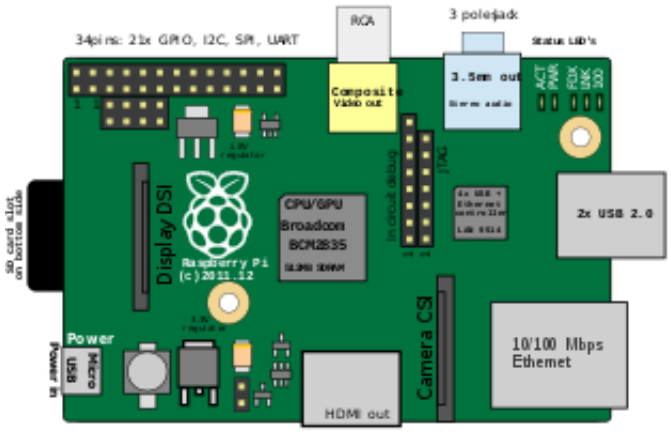


Figure 14: Components of Raspberry B

One thing I did not say before is that a Raspberry has no memory. But thanks the SD reader, it's possible add some. One other possibility that a Raspberry Pi offers me is the possibility to add a camera on the device. There is raspberry camera model, the pi camera v2 that can be connected on the CSI port. Once it's plugged, we just need to activate the camera in raspian settings and test the camera thanks the following command :

```
1 raspistill -o photoTest.jpg
```

Figure 15: Pi Camera connected to the Raspberry

5.1 Installation

To install the operating system on the Raspberry, it's necessary to mount an image on a SD card. This operation in made on my laptop operating on windows. On Raspberry website, I downloaded the image an to mount it on the SD card, it's needed to use a utility that can mount a disk image on an SD card like Etcher. On the software, we select the image, the destination and we mount the image on the SD card. Once the computation is done, we can insert the SD card in the Raspberry, connect a keyboard, a mouse and a screen via HDMI. Then we can supply in power the device and the configuration and installation are running.

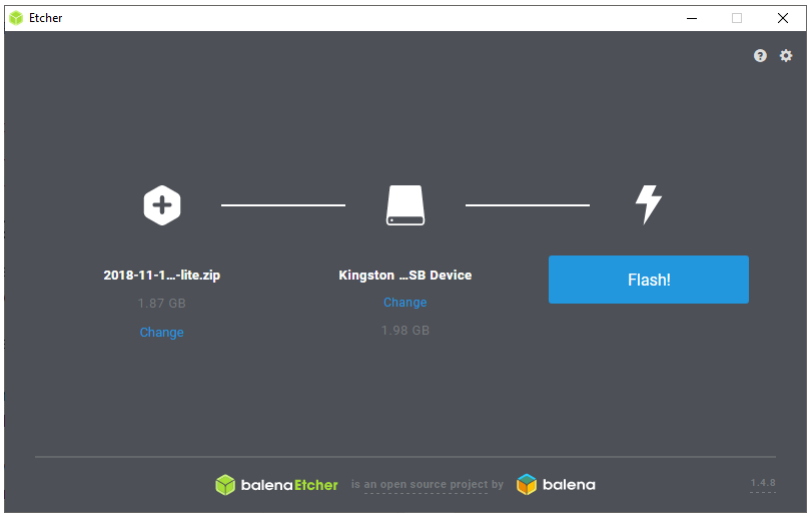


Figure 16: Etcher compilation

After this step, the system Raspian is installed. But because the memory of an SD card will not be enough, I chose to add to the Raspberry an external disk memory. I decided to divide the partition of the external memory in three parts :

- a SWAP partition that will be used to extend the virtual memory of the Raspberry. Basically, if we're using to many RAM in the Raspberry, SWAP partition will be used
- a Linux partition is used to extend the Linux operating system of the Raspberry
- a FAT32 partition that is a storage partition and can be used both on the Raspberry and on a Windows system so useful to transfer files from one system to another

After have spitted the external memory in these three parts and mount them on the Raspberry, the hardware system is completed an ready to used. The next step will be to install all software we need.

6 Installation of all needed packages

Once the system is configured, it is ready to support the installation of all packages needed to make machine learning.

6.1 Tensorflow

As I explained before, Tensorflow is the most important part of the project to make machine learning and to configure model recognition. It's more complicated than on a basic Linux machine to install Tensorflow on a raspberry. This is probably because one of the Google package can not be installed on the device. But there is another solution to install Tensorflow on a Rasperry Pi and I will describe it right now.

Before to do anything, we need to install python3, that will be the main language used for Tensorflow.

```
1 sudo apt-get install python3-dev libffi-dev libssl-dev -y
2 wget https:/www.python.org/ftp/python/3.6.3/Python-3.6.3.tar.xz ↘
    //Download the installation file
3 tar xJf Python-3.6.3.tar.xz //Extract the installation file
4 cd Python-3.6.3
5 ./configure
6 make
7 sudo make install
8 sudo pip3 install --upgrade pip
```

Tensorflow also need another package, the LibAtlas package that we can install inssuing :

```
1 sudo apt-get install libatlas-base-dev
```

Other package are required by Tensorflow to operate. There are the following :

- pillow it's the Python imaging library. It adds image processing capabilities for Python interpreter as more supported image formats. It's a strong image processing tool that will be used by Tensorflow for image computation
- lxml : a library that increase the speed computation and use in Tensorflow programs
- jupyter : an open source notebook use to create and share code, equation, visualization on a web environment
- matplotlib : it's a python 2D library from Matlab that can be used in Python to create figure, graphs, curves. It's very appreciated, easy to use and complete.
- cython : it's a language that makes C prgrams extension for Python language more easy to write than in C directly

We install all this packages as following :

```
1 sudo pip3 install pillow lxml jupyter matplotlib cython #Command↘
    to install packages
2 sudo apt-get install python-tk
```

The first step is to create an installation folder that will contain all the files for the installation of Tensorflow.

```
1 mkdir Tensorflow
2 cd Tensorflow
```

Then, there is a precompiled installation file made by one GitHub user name Ihekibtra. The use of this folder created by the community is very important.

```
1 wget https://github.com/lhelontra/tensorflow-on-arm/releases/↘
    download/v1.8.0/tensorflow-1.8.0-cp35-none-linux_armv7l.whl #↘
    command to dowload tensorflow, I chose version 1.8.0 because ↘
    it's a compatible version with the Raspberry
```

Once it is downloaded, we can install it by doing the following command :

```
1 sudo pip3 install /home/pi/tensorflow-1.8.0-cp35-none-↘
    linux_armvl.whl
```

Tensorflow installation is now done. We can test test if it has been well installed by typing the following commands :

```
python3 //to run python programming environment
import tensorflow as tf
tf.__version__ #This will return the version of Tensorflow installed
```

6.2 OpenCV

OpenCV is a free graphical library developed by Inter, particularly dealing with image processing in real time. The project has been lunched in 1999, the library has been developed particularly by Russian experts and first goals were to study real time tracing and 3D display walls. It's now a free library used and improved by the community.

We can take a look to the main packages of this library :

- core : it's composed of the basics functions, including matrix calculation, image drawing, save and load data in XML files
- imgproc : it's use to image processing, to transform images, create filters or again contour detection
- features2d : it's used with describer, in characterization of images
- objdetect : can be used to object detection, using Adaboost algorithm, me will use it to do object recognition
- video : video stream processing, used to segment and track moving objects in a video
- highgui : input-output and user interface. OpenCV integrates its own * library to open, save and display images and video streams. It also contains a number of functions to make graphical interfaces very simple
- calib 3d : calibration, pose estimation and stereovision. This module contains functions to reconstruct a 3D scene from images acquired with multiple cameras simultaneously.

Nowadays, image recognition and processing tends to develop more and more and OpenCV is the reference in this field. Whether for image recognition or for the creation of artificial intelligence, these tools are used in new technologies. Multiple operations are possible on OpenCV such as motion tracking, classifying objects, or creating learning algorithms.

We recently saw for example an application create with OpenCV using image recognition to lunch a music by seeing the picture of the album. The same technology is currently also controversial in China where facial recognition is used on the streets to track citizens.

In our case, we need it to do image processing using a Tensorflow model in real time using the Raspberry Pi and its camera.

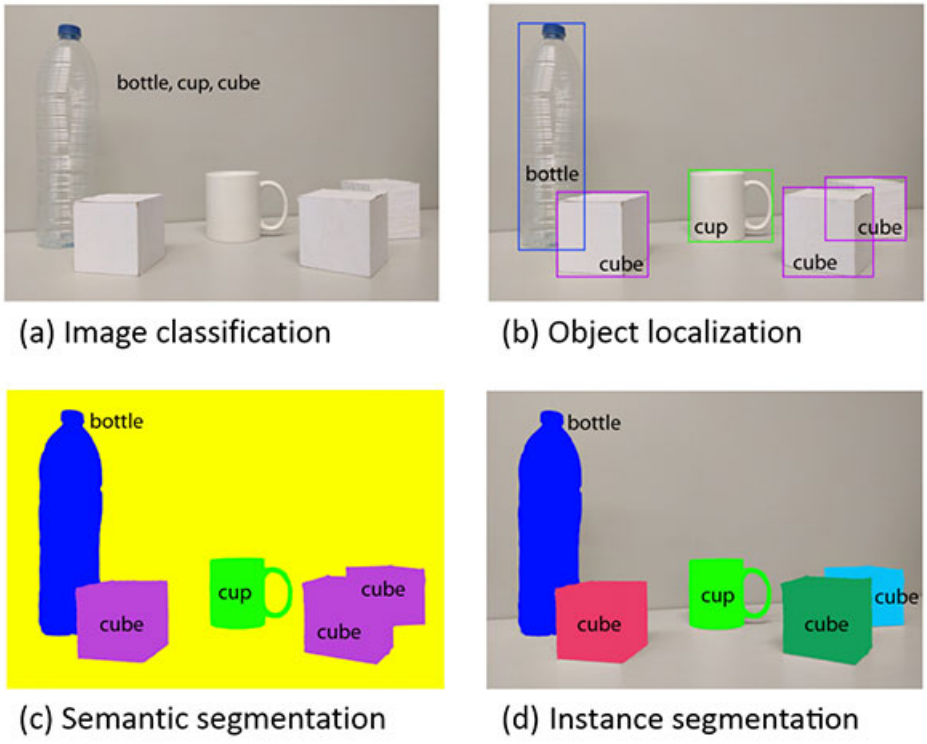


Figure 17: OpenCV examples

Before install OpenCV, it's needed to install some dependencies.

```
1 sudo apt-get install libjpeg-dev libtiff5-dev libjasper-dev ↘
    libpng12-dev
2 sudo apt-get install libavcodec-dev libavformat-dev libswscale-↘
    dev libv4l-dev
```

```
sudo apt-get install libxvidcore-dev libx264-dev
sudo apt-get install qt4-dev-tools
```

After all these packages installed, we can now installed OpenCV typing the command :

```
pip3 install opencv-python
```

6.3 Protocol buffers (Protobuf)

Protocol buffers is a serialization format and an interface description language (to communicate between differents languages) developed by Google. The aim of this protocol is to communicate between programs to store data. As it use interface description language, a program will be able to analyze messages in function of the description. It was at the begining just developed for internal use in Google to exchange data between different open-source programs but also to store and exchange from a lot of sources on all computers of the company. In our case, it will be use to store and classify the different object we want to identify in our program.

There is no easy way to install Protobuf on a Raspberry Pi. So I needed to compile and install it directly from the source. We first download the necessary packages as following :

```
sudo apt-get install autoconf automake libtool curl
```

Then we can download the Protobuf installation files and extract it :

```
wget https://github.com/google/protobuf/releases/download/v3.5.1/↘
    protobuf-all-3.5.1.tar.gz
tar -zxvf protobuf-all-3.5.1.tar.gz
cd protobuf-3.5.1
```

Then we can install it

```
./configure
```

```
make
make check
sudo make install
```

Then, it's necessary to export the library path found in the python directory

```
cd python
export LD_LIBRARY_PATH=../src/.lib
```

It's important to, after this step implement it and export the protocol :

```
python3 setup.py build --cpp_implementation
python3 setup.py test --cpp_implementation
sudo python3 setup.py install --cpp_implementation
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION=cpp
export PROTOCOL_BUFFERS_PYTHON_IMPLEMENTATION_VERSION=3
sudo ldconfig
```

We can check if it's well installed if we found the default message by typing the command :

```
protoc
```

6.4 Setup Tensorflow directory structure

Before to be able to test image recognition, we need to set up a Tensorflow directory.

```
mkdir tensorflow1
cd tensorflow1
```

and download the files from GitHub repository to make image recognition

```
git clone --recurse-submodules https://github.com/tensorflow/↘
    models.git
```

We need also to modify the PYTHONPATH to direct it to some Tensorflow folders. We want that each time we open a python terminal, these links were established. So we need to modify the PYTHONPATH file. We open this file like that :

```
sudo nano ~/.bashrc
```

We have to add the following line at the end of the file to establish the link.

```
export PYTHONPATH=$PYTHONPATH:/home/pi/tensorflow1/models/↘
    research:/home/pi/tensorflow1/models/research/slim
```

Once this is done, we save and close the file. Now, each time we will open a Python terminal, the command we insert will also be called. Then we can use one of the model named object detection. This file takes in inputs a model and a protobuf file (containing the name of all objects in the model) and can identify (make a prediction) of what he sees in real time. But for that we need a model. I will use the SSD Lite model from Google. It's a pre-trained with some objects as model, person or again bike. There are different SSD Lite models with different efficiency of prediction and computation time. Because the Raspberry is not the more powerful computer of the world, I chose a model that use less power as possible but faster.

So we just download and extract the model :

```
1  wget http://download.tensorflow.org/models/object_detection/↘
       ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
2  tar -xzvf ssdlite_mobilenet_v2_coco_2018_05_09.tar.gz
```

We can finally test the model, we need to download the object detection program in the same folder as the model

```
1 wget https://raw.githubusercontent.com/EdjeElectronics/TensorFlow↘
    -Object-Detection-on-the-Raspberry-Pi/master/↘
    Object_detection_picamera.py
```

and we can finally test it

```
1 python3 Object_detection_picamera.py
```

The pre trained object in the model contains keyboard, mouse or again bike. The video stream that we obtain is video in real time with 1-2 frame per second. Is quite enough for what we want to do (identify object) but for other application as movement analyze it would not be enough. Thanks OpenCV, we do not just identify which object is in the picture but also its position. Tensorflow also makes a percentage of prediction of the object detected.

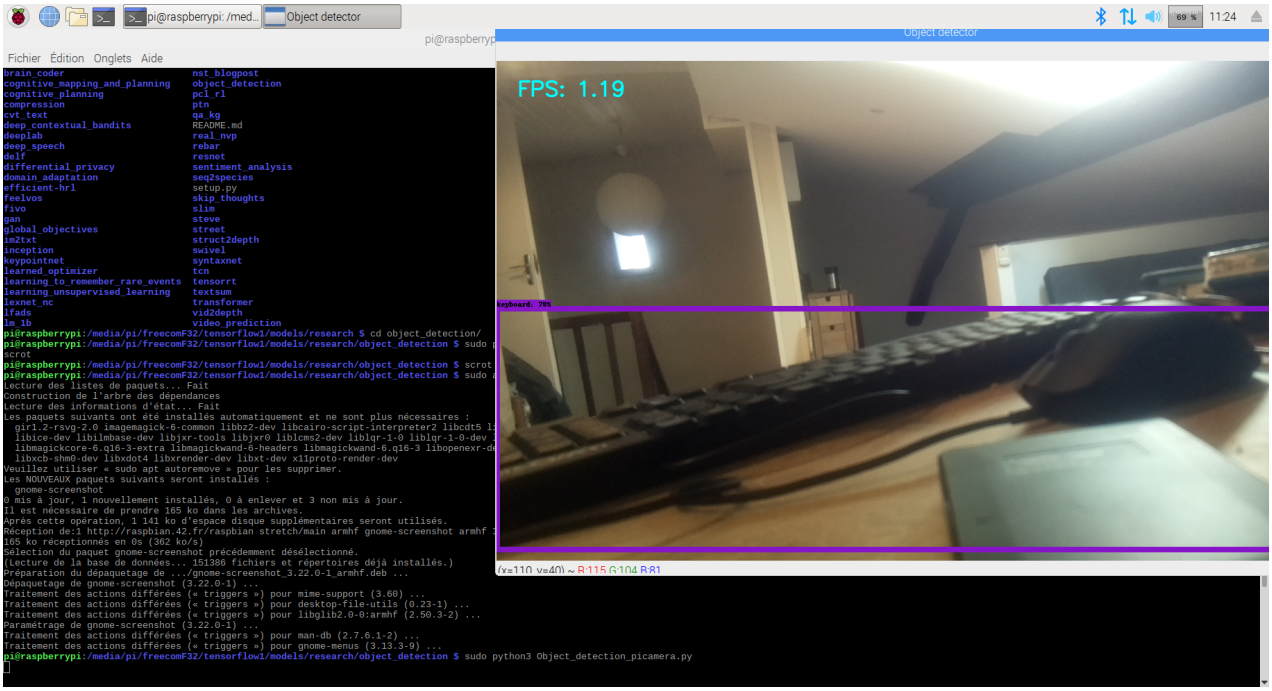


Figure 18: First Screen test

At this step I have done a big part of the configuration. I encountered a lot of difficulties in particular to run all software and packages all together. Because I use a Raspberry Pi, that is at the beginning "naked", without any software installed, I had to install them one by one. For each program and package and software I wanted to install, I always had issues, one or two package were missing. But after several tries, I finally obtain this result and I can now go to the next step that is to train my own model.

7 Creation of the model

To make my first model, I will use an easy object, playing cards. I think it's a good support to create a first model. I will be easy to see if the model is working or not and it will also be possible to collect myself the data and by this way, see the precision of the model according the number of data collected. The aim of this section is first to know how to create a model and then to try to evaluate which factors can influence the precision of the model.

7.1 Collect data

The first step is maybe the most important step of training a model. To have a strong classifier, we need a lot of data, more picture of the object we want to recognize, more the precision of the model will be high. I will use for that 4 different playing cards : 7 of diamonds, 9 of spades, ace of hearts and the lady of clover.

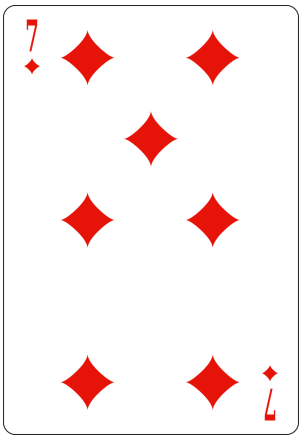


Figure 19: 7 of diamonds

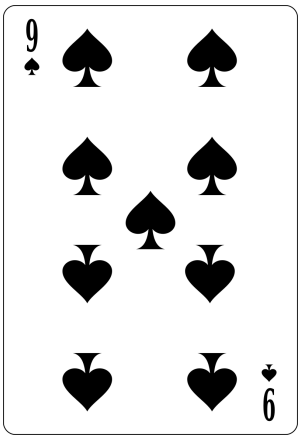


Figure 20: 9 of spades

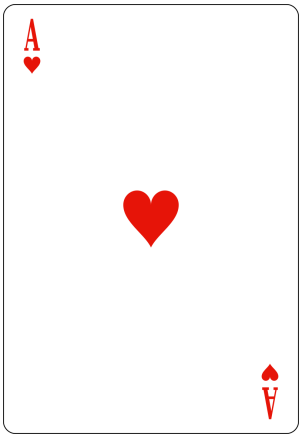


Figure 21: Ace of hearts

Figure 22: Lady of clover

First, I will take picture of all these cards in different positions and environments to

have initial data. I first chose to take 25, and see if I will be able do detect then them in real time using the Raspberry.I took them with my phone and consequently the size of these images is pretty big. The following script reduce the size of pictures.

```
from PIL import Image
import os
import argparse
def rescale_images(directory, size):
    for img in os.listdir(directory):
        im = Image.open(directory+img)
        im_resized = im.resize(size, Image.ANTIALIAS)
        im_resized.save(directory+img)
if __name__ == '__main__':
    parser = argparse.ArgumentParser(description="Rescale images")
    parser.add_argument('-d', '--directory', type=str, required=True, help='Directory containing the images')
    parser.add_argument('-s', '--size', type=int, nargs=2, required=True, metavar=('width', 'height'), help='Image size')
    args = parser.parse_args()
    rescale_images(args.directory, args.size)
```

Then according the orientation of the image (landscape or vertical) we lunch the following convert to compress the images :

```
python transform_image_resolution.py -d images/ -s 800 600
```

I reduced by this way the size of the images by 10. Then, as I have already explained before, we need to split the images in two parts. 80% in a train folder and 20% in a test directory.

7.2 Labeling

Now, we need to label the images according what my pictures. For that I use the labelImg software. With the "Open Dir" button, we select the test and train directory. Then, we need to create bouding box, selecting "Create RectBox". Then this is a long step that consists to select and name all object one by one for all train and test images.

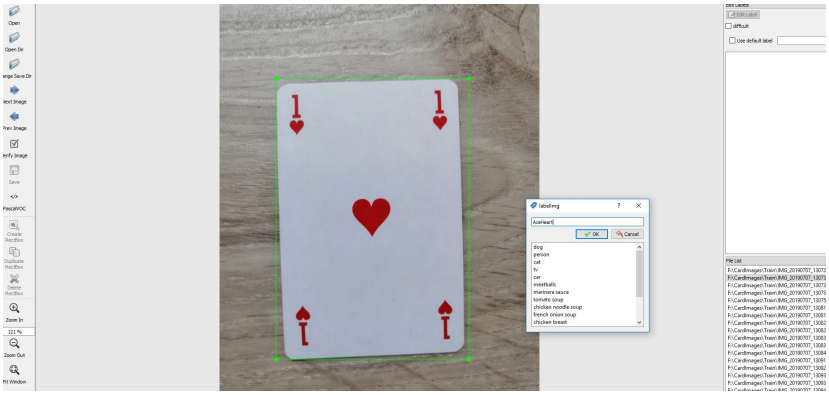


Figure 23: First Screen test

At the end of this step we have all the picture with xml files corresponding for each one that contains information of what we identify and its location on the picture. But these data can not be directly input to train the data, we need to convert them to be used. We will create a TFRecords by first converting the xml file to csv files and then generate the TFRecords. I will use for that the Dat Tran's racoon detector [4].

So we can first covnert xml file to csv just running the following script :

```
python3 xml_to_csv.py
```

Then we run the other program named generate tfrecords.py, we just need to declare the following class corresponding to our different cards :

```
def class_text_to_int(row_label):
    if row_label == 'AceHeart':
        return 1
    elif row_label == '7Diamonds':
        return 2
    elif row_label == 'QClocer':
        return 3
    elif row_label == '9Spades':
        return 4
    else:
        return None
```

Now by running the script, we create record files that will be use to train the model.

```
python generate_tfrecord.py --csv_input=images\train_labels.csv --image_dir=images\train --output_path=train.record
python generate_tfrecord.py --csv_input=images\test_labels.csv --image_dir=images\test --output_path=test.record
```

Next step is to configure the training of the model. For that, we need to create a label map that link the ID of the objects defined in the previous section to object. They have to be the same that in the object detectory python file.

```
item {
    id: 1
    name: 'AceHeart'
}

item {
    id: 2
    name: '7Diamonds'
}

item {
    id: 3
    name: 'QClocer'
}

item {
    id: 4
    name: '9Spades'
}
```

I put this file (labelmap.pbtxt) and the record files (train.record and test.record) in the same folder that I named annotation. Then, we need to download a pre-trained model. Create a model by myself is a complicated and non-necessary task. What I want here is just to use one existing model and train it with my objects. There is a large choice of model available in Github repository [7] with different characteristics (time computation, precision etc) available. I chose to use the ssd_inception_v2_coco that is a fast model. I downloaded it, extract its content and place it in a folder named "pre-trained-model".

We also need to find the corresponding config file and place it in a new folder naimed "training". It's important to modify this file according the situation. First we have to

defined in its code the number of object we want to train (corresponding to the number in the labelmap). We can also modify the nbatch size according the memory available on the computer. The last thing is to add the path to our file. So the summarize the location of all our file, we have the following configuration :

```
- annotations
  |labelmap.pbtxt #file with the labelling of cards
  |train.record # file use to train the model
  |test.record #file to test if the model is correct
- training
  |ssd_inception_v2_coco.config #model we are going to train
- pre-trained-model #path to the folder where is the model
- train.py #script used to train the model
```

So in the config file, we use the paths as they are defined just above.

```
}
  fine_tune_checkpoint: "pre-trained-model/model.ckpt" #path to the last checkpoint of the train model, we can stop training a model to train it more later
  from_detection_checkpoint: true
  # Note: The below line limits the training process to 200K steps, which we
  # empirically found to be sufficient enough to train the pets dataset. This
  # effectively bypasses the learning rate schedule (the learning rate will
  # never decay). Remove the below line to train indefinitely.
  num_steps: 200000 #number of iteration of the program
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
  data_augmentation_options {
    ssd_random_crop {
    }
  }
}
# In the folliwing section, we need to enter the path to annotations fodler containing the labeling of objects
train_input_reader: {
  tf_record_input_reader {
```

```
21      input_path: "annotations/train.record"
22    }
23    label_map_path: "annotations/labelmap.pbtxt"
24  }
25
26  eval_config: {
27    num_examples: 8000
28    # Note: The below line limits the evaluation process to 10 ↘
          evaluations.
29    # Remove the below line to evaluate indefinitely.
30    max_evals: 10
31  }
32  #This last section will permit to define the precision of the ↘
      model
33  eval_input_reader: {
34    tf_record_input_reader {
35    input_path: "annotations/test.record"
36    }
37    label_map_path: "annotations/labelmap.pbtxt"
38    shuffle: false
39    num_readers: 1
40  }
```

7.3 Training

Then, using the train python script, we can start to train the model.

```
python train.py --logtostderr --train_dir=training/ --↘
    pipeline_config_path=training/ssd_inception_v2_coco.config
```

Thanks the Tensorboard, that is an integrated tool in which one we can see a lot of information in real time in function of each interation about the training model. The most significant value I will look is the Loss value. For the first test, I stopped the model at 170 interations to run a firs ttime my model. But as we can see on the figure 24, the regularization of loss is always increasing, that could say that the model could better and could still me improved.

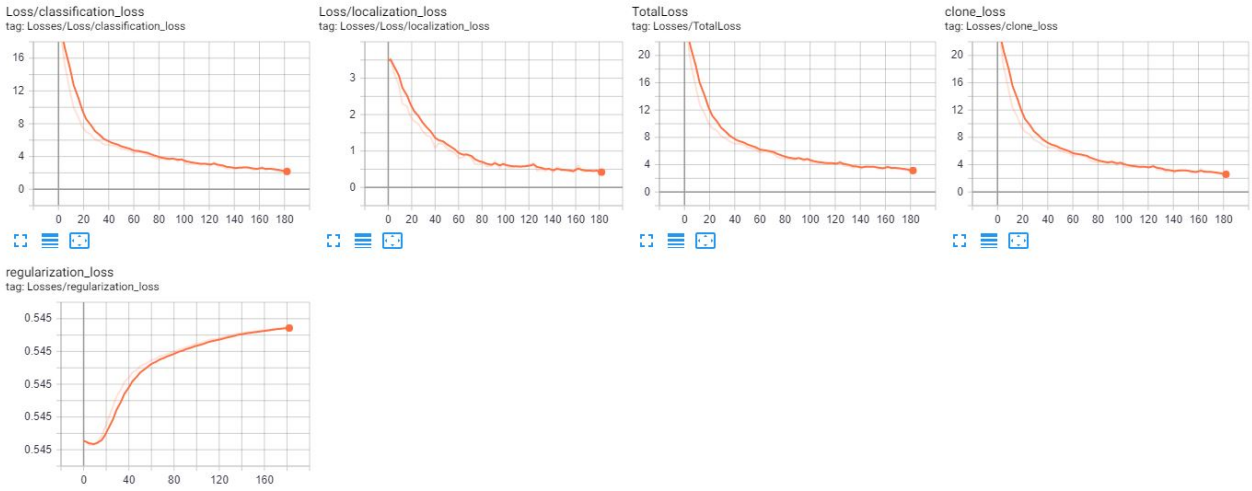


Figure 24: Tensorboard view with 170 interations

So I decided to create a second version of the model with more interations, to be able to compare the accuracy. In this second case, with 430 interations, we can notice on the Tensorflow (figure 25 that this time we reached the point in regularization of loss that this valueis decreasing. We could expect a better estimation.

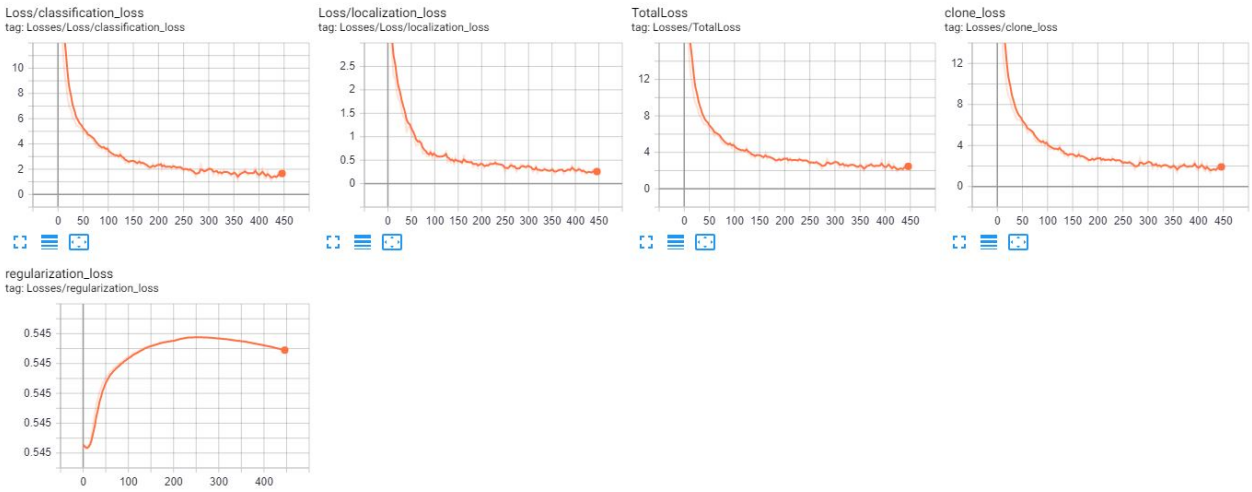


Figure 25: Tensorboard view

One final step before to do the first tests is to export the model thanks the following command :

```
python export_inference_graph.py --input_type image_tensor --pipeline_config_path training/ssd_inception_v2_coco.config --trained_checkpoint_prefix training/model.ckpt-XXXX --output_directory inference_graph
```

With XXX the number of iteration in the saved model.

7.4 First tests and behaviour of the model

At this point, I have two version of my models with the only difference, the number of interations during which I train the model. I will can version 1 the model with 170 interations and version 2 the model with 430 interations.
So I chose to test my model with new picture I took. 4 pictures with the 4 trained cards but also 4 pictures of other cards to see the behaviour of the model with something that the model doesn't know. Is this way I will be ableo to compare accuracy on a the image. The following table shows the results I had according the model on same pictures. Details can be found in Annex 1 and 2.

Card	Model v1	Model v2
9 of spades	99%	99%
Ace of Heart	87%	95%
7 of diamonds	94%	98%
Lady of clover	92%	98%
Ace of Heart 2	80%	93%
Lady of Heart	96%	99%
7 of clover	99%	99%
10 of clover	99%	99%

Table 2: Accuracy according different models

As we can see, the model is quite accurate for cards that I trained. Maybe we can see an exception fort the Ace of Heart that is less accurate than the other, in particular when we change the perspective of the card. We can explain that by the random repartition of train and test image when I defined the parameters of the model. Maybe, with the hazard, too few images were in the train folder and more in the test folder.
But the most interesting thing we can notice here, is the behaviour of the model in presence of something it doesn't know. Indeed, I presented to it cards it doesn't know and it tried to identify what is this card. It failed to do it. It seems to no be as precise that I wanted to. The model found some similarities between shape or color of cards it didn't know and though it was these cards.
It a point I want to put forward here. To identify something, we have to be sure that the object that we present to the model is known by the model. In the other case, In any case, even if it does not know the object, it will put a name of the object, and be wrong. It could be problematic with species authentication. If we present an specie that the model does not know, it will affect the same name to this specie than an other that it already know. Both species will be assimilate to the same name and this biases the authentication.

8 Training in bigger scale

As we saw before, the number of trained images and the time of training will influence the precision of our model. That's why the first step is to collect an high amount of images. That's why I wanted to directly use images that I could find online. I used the Google Images Download package to download the images corresponding to a Google search. This package, thanks to the command allows to download these images.

```
1 googleimagesdownload --keywords "Ace of Heart" --usage_rights ↘
    labeled-for-reuse --20
```

Where "Ace of heart" is the google research, the word from which one we want to get images. The second argument is to respect privacy and right on images. I want to use free image allowed to me reused. The final number is the number of picture we want to download.

This technique find its utility to have a first database available. As we know, the most difficult is to find the images necessary to train our model. I would like to see the precision of a new trained model if I had not taken all pictures by myself.

To be faster and use a maximal amount of images, I will try to skip the labelling procedure.

I wrote the following script to rename all the picture by number :

```
import os
import os.path
image_ext = ['.jpg', '.jpeg', '.JPG', '.JPEG','.png']list of extension that we want rename
image_files = [f for f in os.listdir('.')
               if os.path.splitext(f)[1] in image_ext] #contain all the pictures names

for i in range(0,len(image_files)) :
    #print(image_files[i])
    extension=os.path.splitext(image_files[i])[1]
    rename_name = str(i+1) + extension
    os.rename(str(image_files[i]), rename_name )# rename function
```

Now to be quicker I had to write a short script to create the csv file with test and train images. Instead to label images one by one and take a lot of time, I decided to use the all picture. So, I just needed to create a csv file that is a database file that coutain, for the case we are labeling images one by one the following information : the name of the picture, its size (width and height), the name of the class (so the name of the object we want to train) and extremities of the rectangle identifying the location of the object. In this case, the rectangle will be all the image.

```
import csv
import os
import os.path
import cv2

image_ext = ['.jpg', '.jpeg', '.JPG', '.JPEG','.png'] #list of extension suported by the script
image_files = [f for f in os.listdir('.')
               if os.path.splitext(f)[1] in image_ext]

with open('Test.csv', 'w',newline='') as csvfile:
    filewriter = csv.writer(csvfile, delimiter=',',
                            quotechar='|', quoting=csv.QUOTE_MINIMAL)
    filewriter.writerow(['filename','width','height','class','xmin','ymin','xmax','ymax'])
    for i in range(0,len(image_files)) :
        img = cv2.imread(image_files[i])
        height = img.shape[0]
        width = img.shape[1]
        filewriter.writerow([image_files[i],width,height,'AceOfHeart',0,0,width,height])
```

Then we create the record files as we did before and repeat the exactly same process to train the model.

Unfortunately, these experience was not a success. The convergence of the loss was quickly reached and it it was not possible to detect any object with this trained model. I supposed at this step that it was because I had not enough picture but I did it before with less than 20 pictures.

I repeat the same experience with more sample (50 images). But I encountered the same problem of identification by testing the model. I had several hypothesis about that. Maybe the model can not be trained with only one kind of image but hae at least to thing that it can detect. Another possibility is that the model has to be able to identify an object in a particular environment. If it can not do that it can not be able to detect the object in a new environment. For example here, the cards I used were on a white background but testing it, it was in a new environment and was so not able to detect the card.

9 Advantages and disadvantages

9.1 Advantages

We can find several advantages in the method I used. The main one, is that is possible to identify and train a model with anything we want. In the manner of the human being, the machine is able to learn from its experience. Like a person, she is in such a configuration that she can learn from her mistakes and her experience. The more we glow submits image examples with the exact location of the object to be identified, the more it is able to subsequently identify this object in a new environment.

Another advantage is that if we have a reliable model, we can be almost sure of not missing any information where the human being can make errors of inattention. To delegate this work to a machine allows the man to be able to concentrate on a more important task, on the methods of image capture for example in order to put in the best conditions the recognition algorithm.

9.2 Disadvantages

However, several obstacles can be a barrier to this learning. First, the main limitation is the number of items needed for learning. I have here realize small models with can of numbers of images which showed their limit. We can ask the question if in each case we have or not the sufficient number of images necessary for learning. I think in particular the proposal to identify the first damage affecting a wind turbine. Without a first experience of the man who would detect his defect and the takers took pictures, this type of approach is impossible. It is almost impossible to anticipate a new kind of problem. A past experience and in any case necessary, and an experience long enough to train a machine to recognize it. The size of the sample is very important.

A second major defect of this method is the impossibility of predicting the behavior of this recognition in the presence of an element which is unknown to it but close to what it knows. Train a model to recognize an aquatic plant present in a geographically precise marine environment and it could recognize this species. But what happens if this environment changes and then offers new conditions conducive to the development of a new species. Will the model be able to differentiate the two plants or will it equate the two plants with the same name? Additional training will be necessary for him to distinguish the two species.

Moreover, as we have seen with one of the experiments conducted, it is necessary to manually identify the location of the element to be identified for the model to train on an object present in an environment. The environment may change but the object of must not. Depending on the environment, the weather, or the perspective on the object, it will not appear in the same way but it must be identifiable by the model in each of these cases.

This technology can be improved without starting from scratch each time. A model can be improved to train more starting from an initial state. A model can be trained to infinity, the only limit is to define what rate of precision acceptable according to the desired use.

The tools made available by Google and therefore Tensorflow are free and thus develops within the scientific community quite quickly. We find for example the Tensorboard which allows to analyze the results in a powerful way by giving the evolution of the characteristics of the model according to the progress of the training.

10 Conclusion

We are in a world that uses more and more artificial intelligence and machine learning is part of this technology. This applies to many areas and renewable energies, especially in its approach to environmental preservation and habitat study can benefit from this development.

In this report, I have shown that it is possible, from a database of images and a work of prior identification by the human being, to train a model, taking into account its limits, to automatically recognize an object or a detail on a photo.

From a personal point of view, it was very instructive to work on this subject and I think to continue some experiments on my free time for other applications than those related to energy. To go beyond the scope of this dissertation, I sincerely think that this technology will be present in many areas in a few years. There is only one limit today on the conditions of its use. When this is used for scientific purposes to study animal behavior or plant development, this advances research scientifically but the problem arises when used for commercial purposes with databases on each individual. Or does the ethical limit lie? The debate must be brought before the development of this technology. And I think that should be the case for many technologies. Defining what is ethically and ethically acceptable must be studied before the drift to a technology can appear.

References

[1] *Learning from Data to Create a Safer and Smarter Self-Driving Experience*. Datanami. Aug. 16, 2019. URL: https://www.datanami.com/2019/08/16/learning-from-data-to-create-a-safer-and-smarter-self-driving-experience/ (visited on 08/16/2019).

[2] *Marine Biofouling: Colonization Processes and Defenses*. CRC Press. URL: https://www.crcpress.com/Marine-Biofouling-Colonization-Processes-and-Defenses/Railkin/p/book/9780849314193 (visited on 08/16/2019).

[3] A. S. M. Shihavuddin et al. "Wind Turbine Surface Damage Detection by Deep Learning Aided Drone Inspection Analysis". In: *Energies* 12.4 (Jan. 2019), p. 676. DOI: 10.3390/en12040676. URL: https://www.mdpi.com/1996-1073/12/4/676 (visited on 08/16/2019).

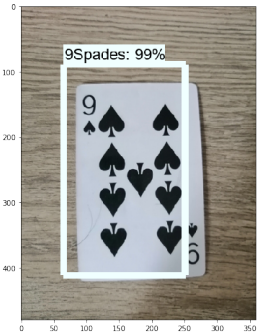
[4] Dat Tran. *The dataset is used to train my own raccoon detector and I blogged about it on Medium: datitran/raccoon_dataset*. original-date: 2017-07-27T19:04:10Z. July 7, 2019. URL: https://github.com/datitran/raccoon_dataset (visited on 07/07/2019).

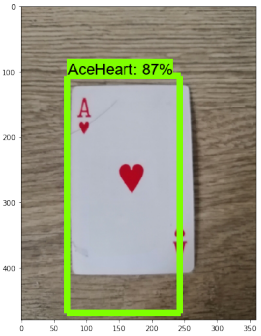
[5] Priya Dwivedi @ Deep Learning Analytics. *Find where to park in real time using OpenCV and Tensorflow*. Medium. Mar. 27, 2019. URL: https://towardsdatascience.com/find-where-to-park-in-real-time-using-opencv-and-tensorflow-4307a4c3da03 (visited on 08/17/2019).

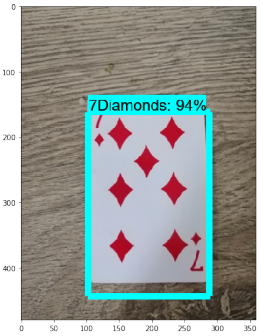
[6] *TensorFlow*. URL: https://www.tensorflow.org/ (visited on 08/22/2019).

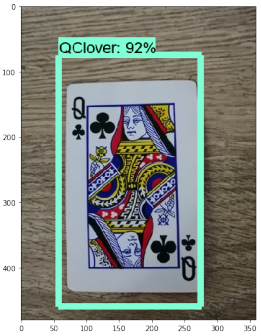
[7] *Models and examples built with TensorFlow. Contribute to tensorflow/models development by creating an account on GitHub*. original-date: 2016-02-05T01:15:20Z. July 12, 2019. URL: https://github.com/tensorflow/models (visited on 07/12/2019).

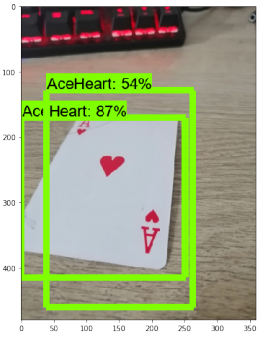
11 Annex 1

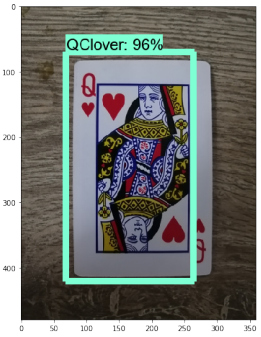

9Spades: 99%

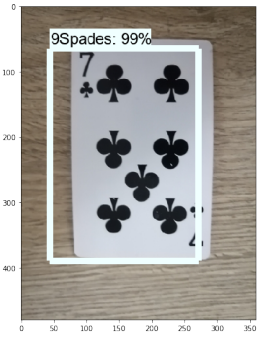

AceHeart: 87%

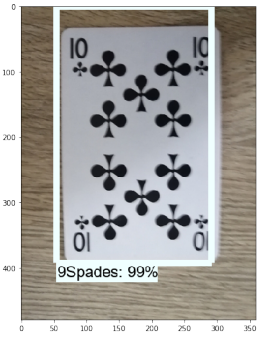

7Diamonds: 94%


QClover: 92%

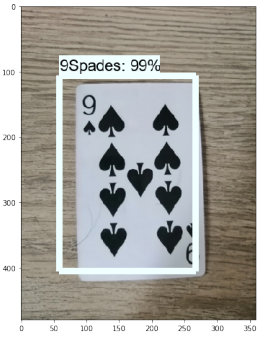

AceHeart: 54%
Ac Heart: 87%


QClover: 96%

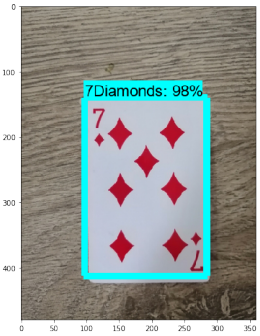

9Spades: 99%


9Spades: 99%

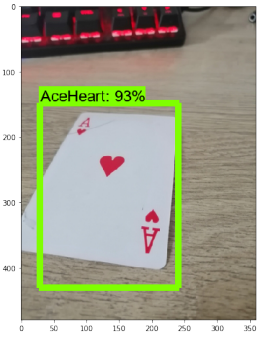
12 Annex 2

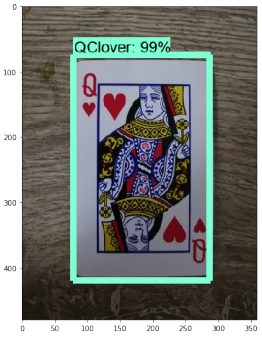

9Spades: 99%

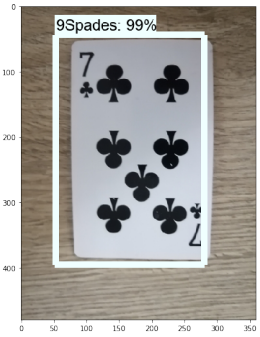

AceHeart: 95%

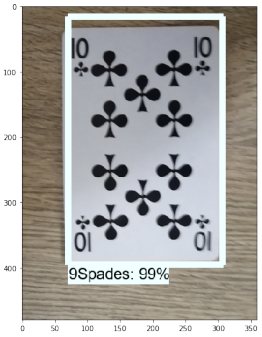

7Diamonds: 98%


QClover: 98%


AceHeart: 93%


QClover: 99%


9Spades: 99%


9Spades: 99%