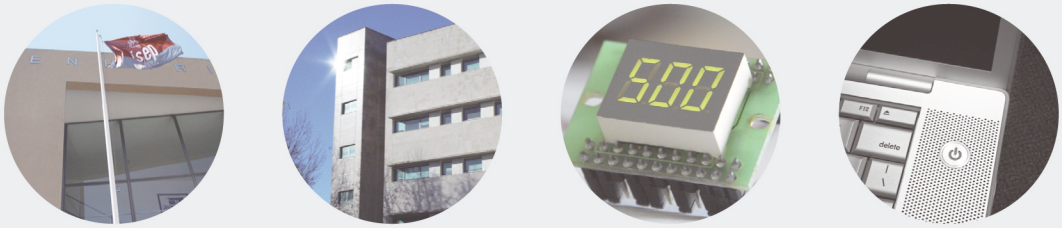
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



Inspecção Visual de Isoladores Eléctricos - Abordagem baseada em Deep Learning

DANIEL DOS SANTOS OLIVEIRA
Outubro de 2017

POLITÉCNICO
DO PORTO

Instituto Superior de
Engenharia do Porto

Visual Insulator Powerline Inspection

Approach Using Deep Learning

Master Thesis

To obtain the degree of master at the
Instituto Superior de Engenharia do Porto,
public defend on November by 2017

Daniel Dos Santos Oliveira

Supervisor: Prof. Dr. André Dias

Degree in Electronics and Computer Science
Porto, Portugal.

Agradecimentos

Em primeiro lugar, agradeço aos meus pais pelo esforço que fizeram e pela oportunidade que me deram.

À Laura pelo apoio e compreensão.

Agradeço ao meu orientador Prof. Dr. André Dias pelo desafio proposto e pela possibilidade de estudar este tema.

A todo o grupo LSA pela ajuda ao longo do mestrado e em especial aos elementos da *Aerial Team* André Ferreira, Miguel Moreira e Tiago Miranda.

Obrigado aos meus amigos e colegas de mestrado mas também aos meus amigos e colegas da licenciatura em Engenharia Electromecânica da UBI pelo apoio e motivação durante todo o (longo) percurso académico.

O trabalho desenvolvido apenas foi possível graças ao apoio de todos. O meu sincero obrigado.

Abstract

To supply the electrical population's demand is necessary to have a good quality power distribution systems. Electrical asset inspection, like electrical towers, dam or power line is a high risk and expensive task. Nowadays it is done with traditional methods like using a helicopter equipped with several sensors or with specialised human labour.

In the last years, the Unmanned Aerial Vehicle (UAV) exponential growth (most common called drones) make them very accessible for different applications. They are cheaper and easy to adapt. Adopting this technology will be in the future the next step on electrical asset inspection. It will provide a better service (safer, faster and cheaper), particularly in power line distribution.

This thesis brings forward an alternative to traditional methods using a UAV for images processing during the insulator visual inspection.

The developed work implement real-time insulators visual detection using an Artificial Neural Network (ANN), You Only Look Once (YOLO) in this case, on medium and high voltage power lines. YOLO was trained with different types and sizes of insulators. Isn't always possible to see what the UAV is recording so it has a gimbal system which controls the camera orientation/position. It will centre the insulator on the image and this way getting a better view of it. All the training and tests were performed on board Jetson TX2.

Keywords: Deep learning, YOLO, power line inspection, UAV, gimbal control, real-time, object detection

Resumo

A inspeção de ativos elétricos, sejam eles torres elétricas, barragens ou linhas elétricas, é realizada com recurso a helicópteros, equipados com sensores para o efeito ou, de uma forma mais minuciosa, com o recurso a mão-de-obra especializada. Ambas as situações são trabalhos de risco elevado.

Nos últimos anos temos assistido a um enorme crescimento de veículos aéreos não tripulados, vulgarmente chamados de *drones*. Estes sistemas estão bastante desenvolvidos e são economicamente acessíveis, o que os torna perfeitos para variadíssimas funções. A inspeção de linhas elétricas não é exceção.

Esta dissertação, pretende ser uma primeira abordagem à utilização de *drones* para uma inspeção autónoma de linhas elétricas, nomeadamente no processamento de imagem para inspeção visual de isoladores.

O trabalho desenvolvido, consiste na implementação de um sistema que funciona em tempo real para a deteção visual de isoladores. A deteção é feita com recurso a uma rede neuronal, neste caso específico a fico a *You Only Look Once* (YOLO), que foi treinada com isoladores de diferentes tamanhos e materiais. Uma vez que nem sempre é possível acompanhar o que está a ser filmado, o *drone* consta de um sistema capaz de orientar a câmara, chamado *gimbal*, para centrar o isolador na imagem e assim conseguir obter um melhor enquadramento do ativo a ser inspecionado. Todos este desenvolvimentos e consequentes testes foram realizados com a utilização de processamento paralelo, que neste caso foi utilizada a placa Jetson TX2.

Palavras-chave: Rede neuronal, YOLO, inspeção elétrica, veículos aéreos não tripulados, controlo de gimbal, aplicação em tempo real, deteção de objetos

Contents

List of Figures

List of Tables

Acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
ILSVRC	ImageNet Large-Scale Visual Recognition Challenge
INS	Inertial Navigation System
IoU	Intersection over Union
LR	Learning Rate
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NN	Neural Network
RNN	Recurrent Neural Network
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
YOLO	You Only Look Once

1 Introduction

1.1 Contextualisation

Due to the population growth, is necessary to create more and better infrastructures to supply all population's demand. **Every single infrastructures need to be in good status to be operational**, as is the case of power electrical systems. With this sentence in mind, preventing electrical faults on power line systems is a crucial task. Nowadays the power line inspection is done with helicopters [1] or with humans climbing the towers, which is a very expensive and very dangerous method. Every year, all around the world, there are helicopter crashes or accidents during the inspection.

Robotic systems have increased their importance and performance in several and various fields, since military to sea exploration passing through medicine. This work aims to be an alternative to the traditional methods, avoiding human's life risk with a cheaper and faster solution using an Unmanned Aerial Vehicle (UAV) for visual inspection and data record on power line inspection.

1.2 Motivation

The Autonomous Systems Laboratory of Instituto Superior de Engenharia do Porto (ISEP) has many works on robotics field, specifically in the field of marine mission and aerial inspection. All this know-how creates a good environment to

develop new methods and use cutting edge technologies for the most diversified challenges.

1.2.1 Electrical asset inspection project - EDP Lablec

One project developed at laboratory in partnership with EDP Lablec, which is a Portuguese electrical inspection company, consists in developing an UAV, Figure 1.1, capable of recording images and data of electrical structures such as electrical tower, dams or wind towers for inspection [2].

The UAV is equipped with visual and thermal cameras to analyse the power lines looking for hot spots. In addition the UAV is equipped with an LiDAR, RTK-GPS and Inertial Navigation System (INS). It can operate manually or in autonomous mode.

This thesis aims to be a contribute adding a new feature to improve the visual insulators inspection.

Figure 1.1: *EDP Lablec's done for electrical asset inspection.*

1.2.2 Contests

The Autonomous Systems Laboratory also participates in contests using its own developed technology. The most successful participation was on **Eurathlon 2015**, in which won the *Grand Challenge (Land + Air +Sea)* plus the *Sub-challenge (Land + Air): Survey the building and search for a missing worker* and in **Eurathlon 2017** won the *Grand Challenge (Scenario 1: land, sea and air)*, plus *Pipe inspection and search for missing workers (Scenario 3: sea and air)* and get the third place on *Survey the building and search for missing workers (Scenario 2: land and sea)*. These victories prove the knowledge's quality that exists in the laboratory.

1.3 Goals

The purpose is to develop a system capable of automatically detect, in real-time, insulators on the electric pole. The images are captured using a UAV with a camera mounted on a gimbal system. This system has to be able to adjust the gimbal orientation in order to centre the image on insulator in a way that a better view during the visual inspection is obtained.

Insulators can be made of several types of materials, like porcelain or glass, each one which have their own modes of light reflection making detection process difficult. In some cases, their inspection involves image capturing combining visible, infrared and ultraviolet range [3]. To avoid all these differences and perform a better detection, it was trained an Artificial Neural Network (ANN) with the purpose of differentiating the object from the background. The ANN chosen was the You Only Look Once (YOLO) [1] [2] architecture, developed on Darknet framework, created by Joseph Redmon.

The goals for this work are:

- Create the dataset with several images of insulators
- Train an ANN (this case YOLO) for insulator detection.
- Develop a method to centre the insulator on image, with the control of the gimbal.
- Integrate the system on a drone

1.4 Thesis structure

On Chapter 2 it is discussed the State of Art of the technologies used and the visual detection methods. Chapter 3 describes some Deep Learning concepts, necessary to understand how ANN works. Chapter 4 is about how the works were implemented. The results are presented on Chapter 5. The conclusions and future work are discussed on Chapter 6.

[1] Official documentation: `https://pjreddie.com/darknet/yolo/`

[2] TED Talk by Joseph Redmon: `https://www.ted.com/talks/joseph_redmon_how_a_computer_learns_to_recognize_objects_instantly`

2

State of Art

This chapter is a brief resume of works in the visual detection and power lines inspection. For visual detection is approached features detection methods and the use of ANN. Some CNN models are referred, not because they were used in this work but because of their importance on the CNN field. There are already robots developed for power lines inspection and some of them are also referred in this chapter.

2.1 Non-Neural Network Methods

The first ways to do inspection and detection of components were with traditional computer vision techniques (edge and corner detection, histogram analysis, image segmentation, etc).

Poles detection were studied using image segmentation and simple techniques like edge detection and corner detection ([4],[5]). In the same way, there are works on detecting insulators based on edge-detector, image threshold and erosion filter ([6],[7]).

The weather conditions influence the detection and it is necessary to develop methods to contour that. In [8], the method of histogram specification it is proposed to eliminate the influence of fog using image stitching to get the whole tower and with the help of Gestalt perceptual algorithm, the tower, insulator and transmission lines are detected.

Instead of doing the inspection with a vehicle, it is possible to install multiple

cameras on poles to picture the insulators periodically [9].

In [10] they not only detect the insulator but go further and also perform an evaluation of the status of the insulator in each individual caps by using a descriptor with elliptical spatial support. The work presented on [11] also detect individual caps but uses texture features to create a lattice model and search for multiple insulators jointly.

2.2 Neural Network Methods

P. Campy[12][13] presents an approach for aerial power line inspection for real-time autonomous detection of electric towers. The strategy combines classic computer vision, as Sobel filter, and machine learning using a Multilayer Perceptron (MLP) neuronal network.

In [14] the insulators are localised in the image using a MLP neural network and classified regarding its status.

Yue Liu and Jun Yong [15] also presented a method based on CNN for insulator recognition. They used a 6 layer convolution network and remove the false positives applying the Non-Maximum Suppression (NMS) and line fitting methods. The dataset was composed of 3000 images but because they were very similar they applied some transformations (rotations and translations) on the images to get the insulators in different angles, resulting in a dataset of 40000 images.

2.3 CNN models

Talking about CNN is talking about an important achievement in computer vision. This section presents papers with an important role that helped to get a step further on image detection/classification.

AlexNet [16]

Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton created a "large, deep convolutional neural network" that was used to win the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). The network called AlexNet get a top 5 test error rate of 15.4% where the second place had 26.2% error rate. This mark starts the beginning of the CNN uses for image detection/classification.

The network contains 8 layers (5 convolutional layers and 3 fully-connected layers) and applies the ReLU on all outputs. The first convolutional layer has 96 filters (kernels) of size $11 \times 11 \times 3$.

The network was trained on ImageNet dataset and used augmentation techniques (applied some transformations on images) like what was done by [15].

Region Based CNN: R-CNN [17], Fast R-CNN [18], Faster R-CNN [19]

The idea of R-CNN is to use region proposals instead of sliding windows to find an object on the image. This change has improved the time detection of the CNN. The process can be divided into two parts: the region proposal phase and the classification phase. Selective Search looks for potential objects, generating bounding boxes. Next, the CNN extracts the features and passes to the SVM for region classification scores.

VGG Net [20]

The contribute of VGG was the simplicity and depth. The network is composed of 19 layers using 3×3 filters and 2×2 max-pooling layer. The input images have a fixed-size of 224×224 RGB colour.

GoogLeNet [21]

Google presents a really deep CNN with inception model. It contains 100 layers with several filters (1×1, 3×3 and 5×5). Google won the 2014 ILSVRC with this work.

YOLO [22][23]

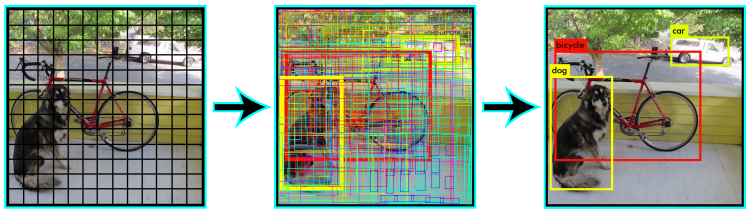


Figure 2.1: *YOLO image division and bounding boxes (Source: [22]).*

A new novel approach was presented by J. Redmon which pretended to be capable of running in real-time applications. YOLO divides the image into $S \times S$ grid and each grid cell predicts bounding boxes associated with a confidence score (Figure 2.1). The confidence score shows how confident the algorithm is about existing an object in that bounding box.

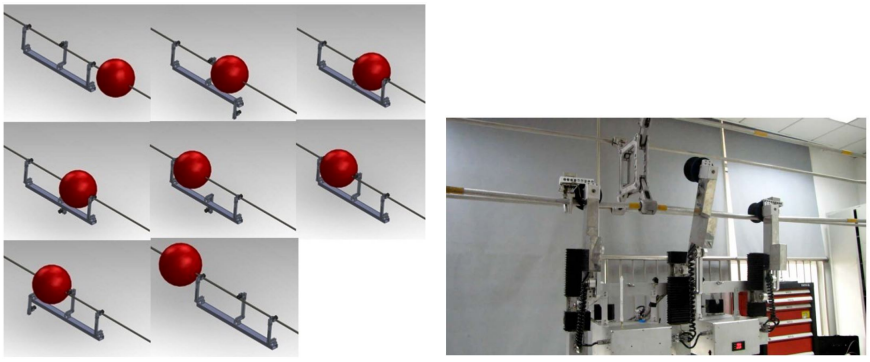
YOLO is faster than Region-proposal methods because it predicts bounding boxes and class probabilities in one evaluation at the same time by a single network.

The paper [23] won the "CVPR 2017 Best Paper Honorable Mention Awards".

2.4 On cable robots for power line inspection

Many works had been done in making and developing robots capable of performing power line inspection, most of them in High Voltage power systems. The firsts robots were used for cable inspection and were very similar among them. They are devices that hold themselves on the cable and move horizontally along it ([24],[25],[26],[27],[28],[29]). These devices are all equipped with cameras and some of them with resistor testers to record data to be analysed after the inspection. The Figure 2.2 and Figure 2.3 shown some examples.

Also, a team of robots was proposed in [30]. They used a three-arms robot for horizontal line inspection and an UAV for vertical tower line inspection and data transmission.



(a) CAD design of Robonwire - cable inspection robot (Source: [25]).

(b) Cable inspection robot (Source: [26]).

Figure 2.2: *Two examples of robots for line inspection. The robot is held on the cable and moves along the cable, recording data or send it to a ground station.*

Figure 2.3: *Insulator inspection robot. This robot has cameras installed for visual inspection and also perform electrical tests (Source: [29]).*

2.5 Aerial robots for power line inspection

Other methods, faster than cable robots, are using aerial vehicles like a helicopter. The inspection requires the robot/vehicle to be stable during the operation, which obligates a good control of the position and attitude of the robot/vehicle relative to the overhead obstacle. Some works, studied in the laboratory, tested the visual control of the robot/vehicle [31] where a rotorcraft draws its power from the power lines and use Hough transform to control the position and attitude relative to the overhead line.

ANN also has a role in control, as shown in [32], where an ANN was used to stabilise a radio control helicopter just with the INS. Due to safety issues, to keep a safe distance to the power lines is needed in order to avoid collisions [33].

Nowadays UAV are very popular and they are starting to be used on inspection jobs, in a way to replacing the helicopters. Once they have a good degree of liberty it is possible to get a stable image and observe parts with more detail. It is also possible to add some sensors like a LIDAR or thermal camera [34] without be necessary an aircraft manufacturer certificate, rather what happens with helicopters.

Was started the study of using few robots, working in a team, to cover bigger areas. Some robots do the inspection while others establish the communications between the inspection robots and the ground station [35]

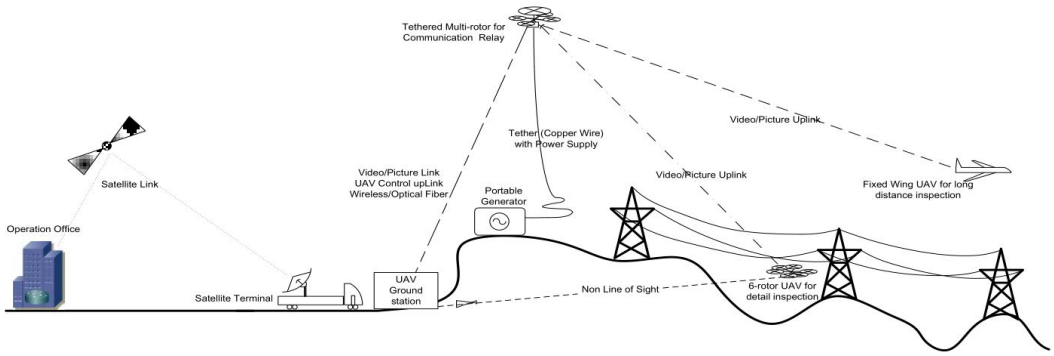


Figure 2.4: *The proposed cooperative UAV systems for power line inspection (Source: [35]).*

3

Deep Learning concepts

In this chapter are presented some basic concepts of ANN. There is many different types of ANN (discussed in section 3.1) and it isn't possible to cover them all. This work focus on image processing and so this chapter will cover only the CNN models that, by their nature, are strongly used for image processing.

3.1 What is Deep Learning?

"Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstractions. (...)", *Deep Learning, Nature, May 2015* [36]

Deep Learning is a sub-field of Artificial Intelligence that uses algorithms in a way inspired by the brain. Deep Learning is composed by an ANN with several layers. ANN is a set of layers is are responsible for some function and a layer is a set of neurons in which each neuron is responsible for a more specific function/detection.

Layers can be grouped into three families: the Input layer, Hidden Layer and Output Layer. Input and Output Layers are single layers. The first receive the data (input) and the second delivery the result of the neuronal network (output). Between this two layers exist one or more Hidden Layer(s) which is responsible for applying calculus/filters apply (see Figure 3.1). The terms Deep Learning and Machine Learning are very similar and the difference differs from authors. If there

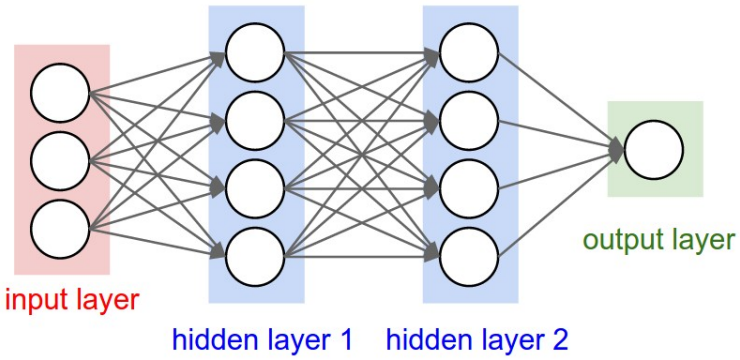


Figure 3.1: *Illustration of a simple ANN. The input layer (red) receives the input data and send to the hidden layers (blue) which at the end send to the output layer (green) (Source: [37]).*

is only one Hidden Layer the ANN is called Machine Learning and if there is more than one it's called Deep Learning.

The Hidden Layer never interacts directly with the input or the output data as that jobs for the Input and Output layers the name implies. These layers can be "trained" to a purpose for example, if the network was trained for boat detection on an image, it is able to classify/detect the image as a boat or not.

Why Deep Learning?

Deep Learning has a good performance advantage over traditional Computer Vision algorithms. With Computer Vision algorithms it's possible to improve the performance with a limit while with ANN the performance becomes better as data is provided as shown by Andrew Ng. at ExtractConf 2015 (Figure 3.2).

Another advantage is is that extraction features are not designed by humans. The features are selected by the ANN learning process according to the data sourced [36].

Applications

Deep Learning is being used in various fields. Where there is data to organise/process probably there is a neuronal network behind. Most of the common uses for deep learning are:

- Image processing - Classify objects, face recognition, medicine diagnosis, etc;
- Speech recognition - Translations, Virtual assistants, etc;
- Data analysis - Marketing services, Whether forecast, etc

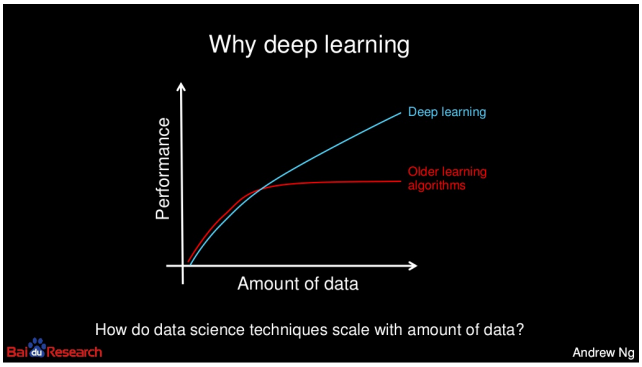


Figure 3.2: *Deep Learning performance related with the amount of data. The performance of Computer Vision methods ends up stabilising rather than Deep Learning (Source: Andrew Ng. in "What data scientists should know about deep learning" at ExtractConf 2015)*

Models of ANN

There is many models of ANN and is important to choose well the better one. Even with a good dataset (quality and number of samples), if the model isn't appropriated for the problem, the ANN will never get good results. The Figure 3.3 shows some ANN models.

The CNN model (models with pink circles on Figure 3.3) works well with adjacent data and this is one reason for why it is commonly used in image processing (work with adjacent pixels).

3.2 How does it works?

3.2.1 Perceptron

A perceptron is an artificial neuron. This concept was developed in the 1950s and 1960s as shown in Figure 3.4 show an example. The perceptron takes the input, x_1, x_2, x_3, multiplies it by the weights w_1, w_2, w_3, and do the sum of all operations. After, it adds what is called the *bias* and returns the neuron's output, which is 1 if it's equal or greater than 0 or 0 if it's lower.

$$Output = \begin{cases} 1 & \text{if } \sum_i w_i x_i + b \geq 0 \\ 0 & \text{if } \sum_i w_i x_i + b < 0 \end{cases}$$

Adjusting the values of weights (or bias) we get different outputs. Deep Learning, during the training, adjusts this values to get a good model fit.

The problem of perceptrons is that small values changes can cause a big change on the next neurons. To avoid that it is used other types of activation, which

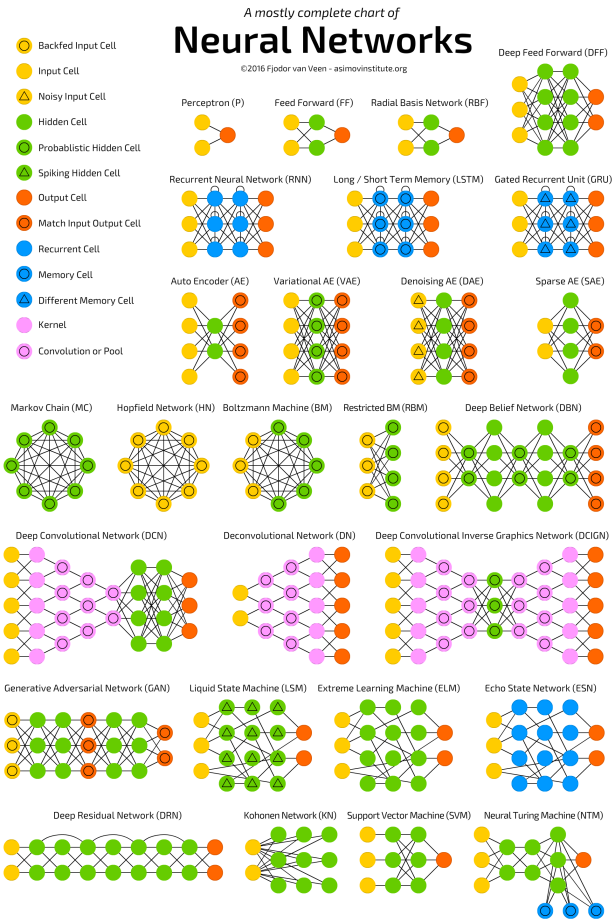


Figure 3.3: *Some models of ANN. In this work it is used the CNN model (pink colour) (Source: `www.asimovinstitute.org/neural-network-zoo`)*

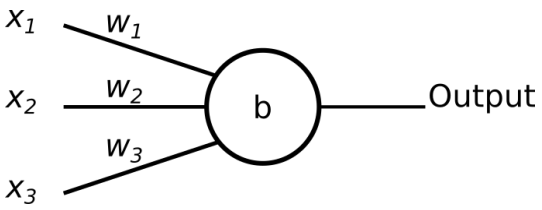


Figure 3.4: *Representation of a perceptron. The x_i are the inputs, w_i weights and b is the bias neuron.*

allow applying small changes without getting big changes on the next step [38]. This problem is also called "Vanishing gradient problem". To do that Sigmoid or Rectified linear Unit (ReLu) functions are used and for values between 0 and 1 can be considered instead of a binary output, Figure 3.5.

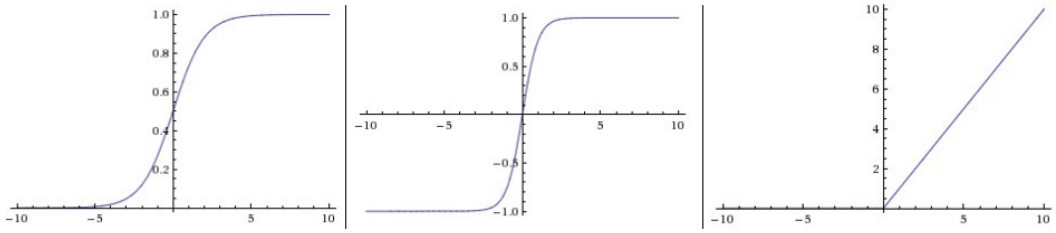


Figure 3.5: *Example of perceptron's activation methods. On left is the Sigmoid, centre tanh and on the right ReLU (Source [37]).*

3.2.2 Convolution

On CNN the neuron output is the result of the convolution between the image and the filter to apply. The more similar the image is with the filter, bigger is the output value of the convolution.

The Figure 3.6 shows an example of a convolution between the image (left) and the filter (centre). The filter search for diagonal lines, from top left to down right. As it is shown, the more the filter matches the image the more activated the neuron is.

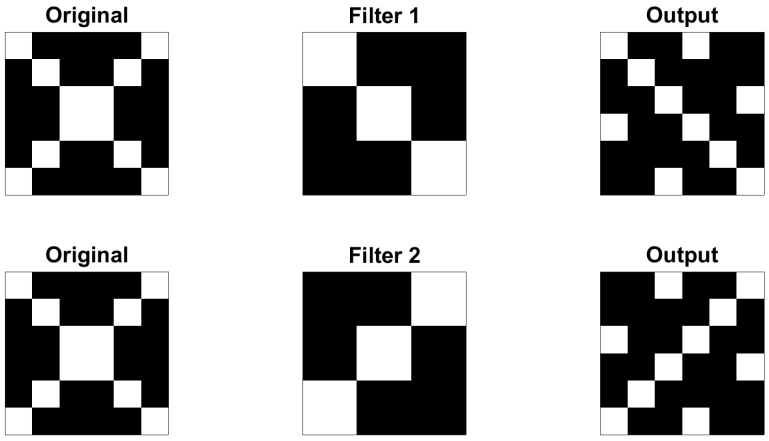


Figure 3.6: *Example of an convolution between image and filter. This filter looks for diagonal lines on the image. It's possible to see where the filter is equal to the image, the output is high.*

After the filter is applied to the image, and the output "normalise" with ReLu or Sigmoid functions, what is called **Activation Map** is obtained. The Activation Map shows how the neuron was activated by a particular region on the image.

3.2.3 Type of Training

To train an ANN is necessary to give them a lot of data. This data has to have labels, a way how what is pretended to know is marked. If the data are images, it is necessary to indicate where the object is on the image and what object it is (which class it belongs). This type of learning is called **Supervised**. It's a long process because it's necessary to label all data. To contour this problem researchers are developing another kind of network where isn't necessary to label the data called **Unsupervised**. This field is still in development but it is the next step in the improvement of training ANN.

3.2.4 Back-propagation

In some way, the CNN needs feedback of their predictions to update itself and fit in the model. This kind of feedback process is called Back-propagation. Back-propagation can be separated into four distinct sections, the Forward Pass (FWP), the Loss Function (LF), the Backward Pass (BWP), and the Weight Update (WU). The sequence is listed below:

1. **Forward Pass** Pass the data through the network, always to the next layer. In this step, the CNN do the predictions.

2. **Loss Function** The Loss Function tell how far the predictions to the label are.In most of the cases, the Loss Function is calculated with Mean Squared Error (MSE).

3. **Backward Pass** Determining which weights contributed most to minimise the error, meaning how much a neuron needs to be updated with new weight value. This value is obtained by deriving Loss Function in order to the weight which needs to be minimised, called **Gradient Descent**.

4. **Weight Update** Update the weight value and repeat the process. The new weight value is calculated this way:

$$Weight_{New} = W_{Initial} - \eta \frac{\partial LF}{\partial Weight}$$

, where η is the learning rate and LF is the loss function.

Learning rate allow the CNN, during the training phase, to learn faster or slower. If the Learning Rate is too big the training is faster but it can never reach the minimum error. If the Learning Rate is too small the training will be very slow.

To reach a balance what is usually done is starting with a big Learning Rate, since, in the beginning, there is a big error so it is possible to do "big steps" and during the training reduce the Learning Rate to reach the minimum error. For each set of training images, **batch**, the program will repeat this process for a fixed number of times, **iteration**. When all data have passed through the CNN it is called an **epoch**. As an example, if the dataset is composed of 1000 images and batch is 500, in 2 iterations is counted 1 epoch.

3.3 CNN Architectures

CNN are designed to work with data that comes in form of multiple arrays, as is a colour image (three arrays of 2D). Other data type can be 1D signals or 2D audio spectrograms. For this reason, they are widely used in processing images [36].

3.3.1 Layers on CNN

CNN can be composed of some types of layer listed bellow [39]:

- **Convolutional** The layer compute the convolution between the image and the filter (neuron output) and pass the output to the next layers. After the convolution an **Activation** (e.g. ReLu, sigmoid, tanh) is done. The purpose of activation is to introduce non-linearity. ReLu is faster than other methods and it also helps to alleviate the vanishing gradient problem, which is the issue where the lower layers of the network train very slowly because the gradient decreases exponentially through the layers.

- **Pooling** This layer is used to reduce the amount of data. It reduces the computational cost, which makes the CNN faster and helps with overfitting. The most common option is the max-pooling, which is selected the highest value.

- **Network in Network** [40] Is a convolutional layer with a 1×1 size filter. It is used to spanning a certain depth $1 \times 1 \times N$ where N is the number of filters applied in the layer.

- **Fully Connected** Applied on the end of the network. This layer computes the class scores and output an N dimensional vector, where N is the number of classes and correlate to a particular class. Each neuron has full connections to all activation's in the previous layer

4

Implementation

In this chapter it is described the dataset and the YOLO architectures and how the training and detection tests were performed.

4.1 What is an insulator?

The insulator (Figure 4.1) is a fundamental component of power line distribution. Its function is to support the cable on poles, isolating the cable from the poles. It is made of glass, porcelain or composite material and it's size (number of caps) is dependent of the voltage where it is installed. When it is in bad condition the isolation performance decrease or it could even break and fall down, compromising the electrical distribution.

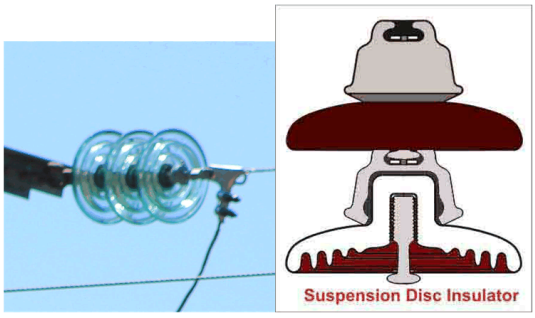
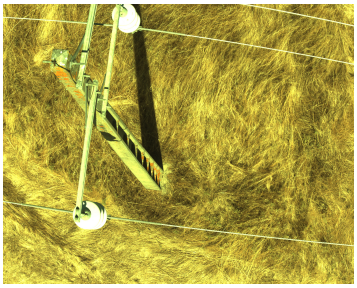


Figure 4.1: *On left an image of a glass insulator. On right an illustration how an insulator is built.(Source: www.electrical4u.com).*

4.2 Dataset

For training YOLO it was necessary a big dataset with insulator images. This dataset contains images with Low/Medium and High Voltage insulators, different materials (glass and ceramic), different backgrounds (blue sky, kind yellow ground and green forest and poles/towers), different views (top/bottom and front) and with insulators in several positions (vertical, horizontal and diagonal). Some images have more than one insulator and also insulators in a different position on the same image. The Figure 4.2 shows some dataset images. The light conditions aren't always constant and the ANN should be invariant of that. To improve it the images have different light conditions as Figure 4.3 shows.

(a) Green background and long insulators.



(b) Top view of white ceramic insulators.

(c) Sky and ground in the background with small insulators.

(d) Bottom view of glass insulators.

Figure 4.2: *Some images used to train the CNN.*

The dataset was initially composed by +3 000 images of insulators. Since the images are very similar, rotations and crop regions were applied on the images to obtain more images (Figure 4.4). Images were rotated ±5° (simulating the UAV's oscillations) and were cropped around the insulators, resulting in more than 90 000 images with different resolutions and orientations, from $600 \times 600px$ cropped images up to $3000 \times 3000px$. This "technique" was inspired by [41], who did this to raise

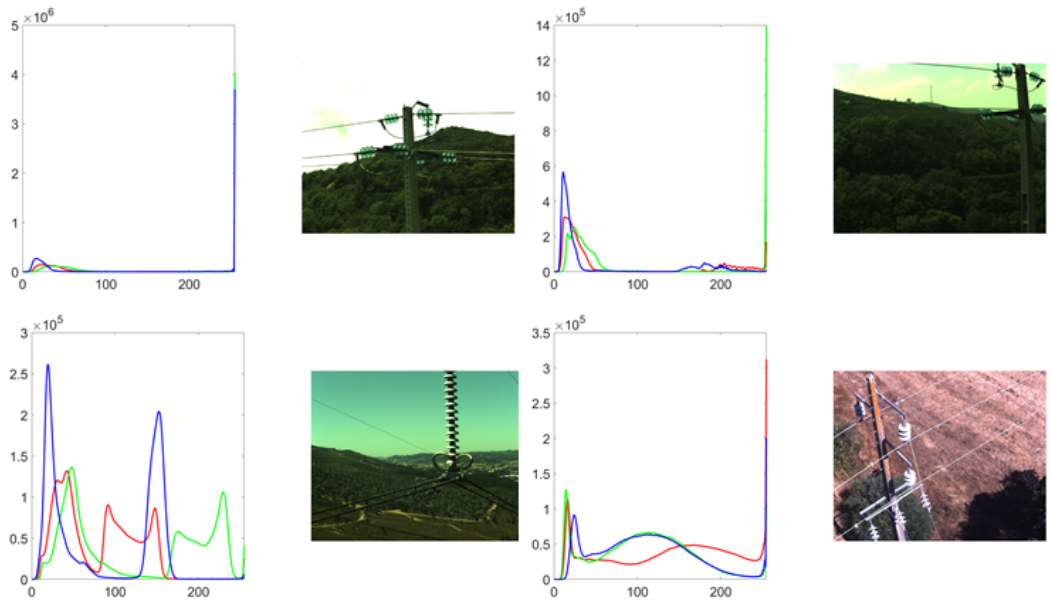


Figure 4.3: *Histogram of some dataset images. Some images don't have the white balance corrected and others are dark. The graphs show the number of pixels, yy axis, with a certain intensity, xx axis in the three colour Red, Green and Blue.*

the dataset images number to participate on ILSVRC 2010.

4.2.1 Bounding boxes

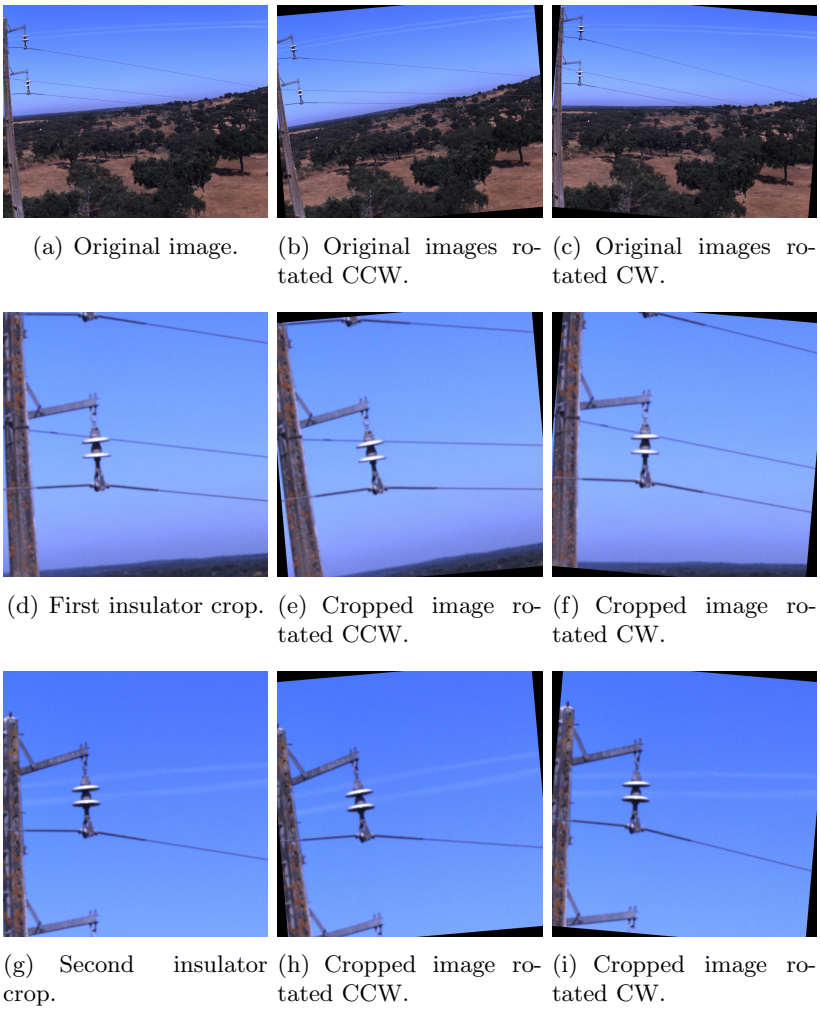
YOLO is an ANN supervised learning type what means that it is necessary to indicate where the insulators on all the images are (labels). To do that it was used a program called YOLO-Mark[1] (see Figure 4.5), developed for YOLO. To optimise the process, first it was "only" labelled the original +3 000 images and then, using MatLab it was generated the labels of the rotated and cropped images.

4.3 Hardware

4.3.1 Processing platform

For this work it was used a NVIDIA Jetson TX2 (on the NVIDIA Jetson TX2 Developer Kit) to train the YOLO network and perform detection. Due to its small size and low power consumption, it's perfect to be used in robotics and mobile platforms. On the Table 4.1 are listed some features of this GPU.

[1] Available here: `https://github.com/AlexeyAB/Yolo_mark`



(a) Original image. (b) Original images rotated CCW. (c) Original images rotated CW.

(d) First insulator crop. (e) Cropped image rotated CCW. (f) Cropped image rotated CW.

(g) Second insulator crop. (h) Cropped image rotated CCW. (i) Cropped image rotated CW.

Figure 4.4: *From the original image 4.4(a) it was possible to generate 8 "new" images with different orientations.*

Table 4.1: *NVIDIA Jetson TX2 specifications*

GPU	NVIDIA Pascal^{TM}, 256 NVIDIA CUDA cores
CPU	HMP Dual Denver 2/2MB L2 + Quad ARM A57/2MB L2
Memory	8 Gb
Power	7.5W / 15 W
Size	50 mm x 87 mm

4.3.2 Images capture platform

The UAV used to test the concept is shown in Figure 4.6. It's a hexacopter equipped with a Point Grey Chameleon3 plus a Fujinon YV2.8×2.8SA-2 lens (image

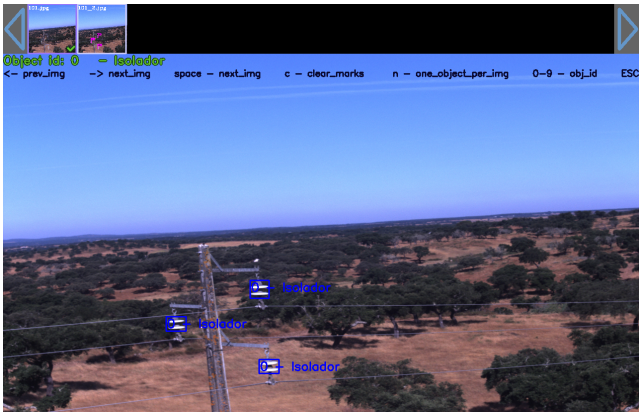


Figure 4.5: *Label insulators on the image with YOLO-Mark program. The YOLO-Mark converts the boxing drawn by the user into the YOLO format.*

resolution of 1280 × 960), mounted on a gimbal system actuated by servomotors. With this setup was possible to test the gimbal system and the YOLO detection.

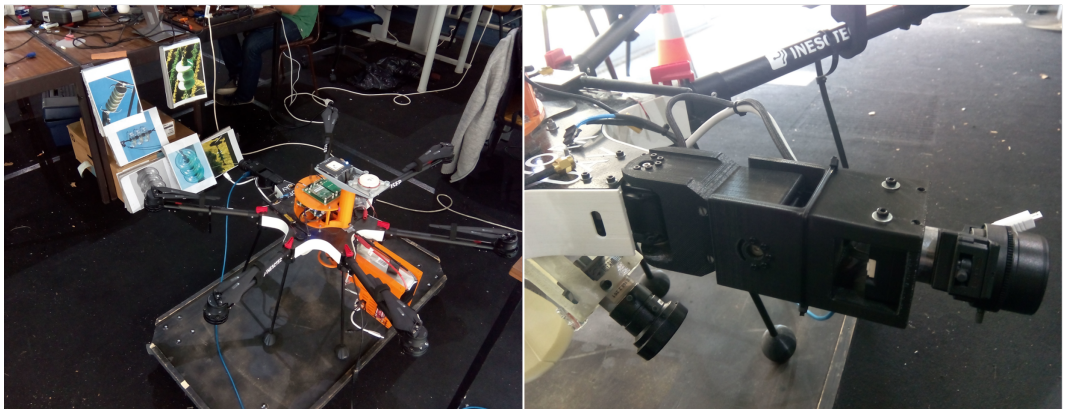


Figure 4.6: *Image capture platform (UAV + camera + gimbal) used during the tests. On the right is a close up of gimbal system.*

4.4 YOLO architecture

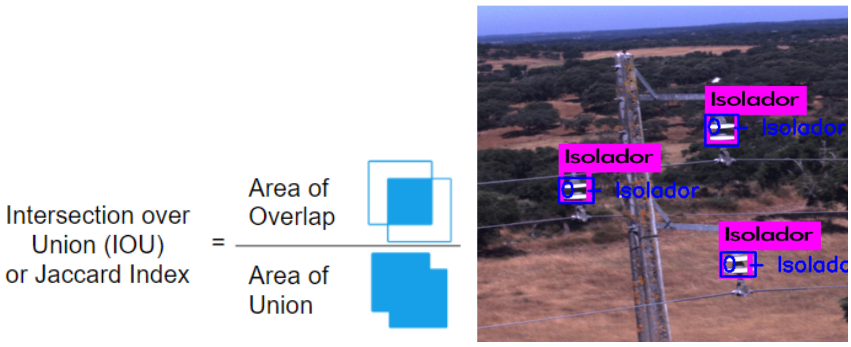
YOLO offers two architectures: the YOLOv2 and the Tiny-YOLO. YOLOv2 is a CNN architecture, based on Darknet-19, composed by 21 convolution layers, 5 max-pooling layers and 2 fully-connected layers. It uses 3 × 3 and 1 × 1 kernels/filters. Tiny-YOLO is a "short version" of YOLOv2, a small architecture that is less precise but more faster. It has 7 convolution layers, 6 max-pooling layers and 2 fully-connected layers and uses the same filters of YOLOv2.

The comparison between this two architectures is approached in the section 4.4.1.

Error calculation

YOLO calculate the error using Jaccard index method also know as **Intersection over Union (IoU)**, which compares the given label area with what it thinks it is an object. The model does a ratio between the Overlap area (detection overlapped on the label) and Union area (detection and label areas).

The Figure 4.7(a) illustrate IoU ratio and Figure 4.7(b) shows a real example of labels (blue box) overlapped with the YOLO predictions (pink boxes).



(a) IoU illustration (Source: Github - AlexeyAB/darknet).

(b) Labels vs predictions.

Figure 4.7: *IoU illustration and Label (blue) vs Prediction (pink).*

Darknet outputs the error at each iteration, as shown in Figure 4.8. This output shows how the loss is evolving at each iteration and indicates how the YOLO perform in each sub-batch. The output of these lines can be analysed on file *detector.c ln. 136*.

The analyse of the last line of the output on Figure 4.8 is on Table 4.2.

```
Region Avg IOU: 0.769098, Class: 1.000000, Obj: 0.754678, No Obj: 0.002056, Avg Recall: 1.000000,  count: 6
Region Avg IOU: 0.764838, Class: 1.000000, Obj: 0.589976, No Obj: 0.002440, Avg Recall: 1.000000,  count: 3
Region Avg IOU: 0.719910, Class: 1.000000, Obj: 0.736155, No Obj: 0.002267, Avg Recall: 1.000000,  count: 11
Region Avg IOU: 0.665226, Class: 1.000000, Obj: 0.481652, No Obj: 0.001467, Avg Recall: 0.833333,  count: 6
Region Avg IOU: 0.764160, Class: 1.000000, Obj: 0.785857, No Obj: 0.002057, Avg Recall: 1.000000,  count: 6
Region Avg IOU: 0.850125, Class: 1.000000, Obj: 0.804475, No Obj: 0.003427, Avg Recall: 1.000000,  count: 5
Region Avg IOU: 0.816231, Class: 1.000000, Obj: 0.647789, No Obj: 0.002810, Avg Recall: 1.000000,  count: 3
Region Avg IOU: 0.705806, Class: 1.000000, Obj: 0.446752, No Obj: 0.001501, Avg Recall: 0.714286,  count: 7
Region Avg IOU: 0.722957, Class: 1.000000, Obj: 0.573790, No Obj: 0.000701, Avg Recall: 1.000000,  count: 3
Region Avg IOU: 0.735202, Class: 1.000000, Obj: 0.725597, No Obj: 0.002979, Avg Recall: 1.000000,  count: 8
Region Avg IOU: 0.635548, Class: 1.000000, Obj: 0.618481, No Obj: 0.002639, Avg Recall: 0.750000,  count: 8
Region Avg IOU: 0.760738, Class: 1.000000, Obj: 0.443256, No Obj: 0.001704, Avg Recall: 1.000000,  count: 5
Region Avg IOU: 0.812140, Class: 1.000000, Obj: 0.598036, No Obj: 0.002503, Avg Recall: 1.000000,  count: 4
Region Avg IOU: 0.885746, Class: 1.000000, Obj: 0.749944, No Obj: 0.001917, Avg Recall: 1.000000,  count: 4
Region Avg IOU: 0.863389, Class: 1.000000, Obj: 0.648980, No Obj: 0.001712, Avg Recall: 1.000000,  count: 3
Region Avg IOU: 0.816444, Class: 1.000000, Obj: 0.657818, No Obj: 0.002966, Avg Recall: 1.000000,  count: 4
116280: 3.160083, 2.096787 avg, 0.000500 rate, 33.916916 seconds, 7441920 images
```

Figure 4.8: *YOLO output during the training.*

The other lines are the result of each sub-division. In this case, the subdivision is 16 which means it will divide the batch into 16 groups and process it, so if $batch = 64$

Table 4.2: *Darknet output for each iteration*

116280	is the iteration number
3.160083	the loss in that iteration
2.096787	the actual average loss
0.0005	the learning rate
33.916916	the iteration time consumption in seconds
7441920	total images seen by the network

and *subdivision* = 16, the training iteration will have 16 groups of 4 image each allowing a better performance on systems with low memory. The output description is on Table 4.3.

Table 4.3: *Darknet output for each batch*

Region Avg IOU: 0.816444	Is the average of the IOU in the current subdivision. The last subdivision has an overlap of 81.64
Class: 1.0000	The relation of classes classified correctly. In this case it's only one class.
Obj: 0.657818	In code it is the relation between the average objects detected and the true positives (count)
No Obj: 0.002966	Relation between all objects detected with the number of real true positives (count)
Avg Recall: 1.00000	The average recall on subdivision.
count: 4	Is the amount of real true positives on subdivision.

Re-size images option

YOLO has an option to re-size the images automatically during the training (independently of the image size) at every 10 iterations. This option is enabled on *CFG* file, parameter "random=1" and allows a better training performance compared although the iteration speed is slightly lower but overall worth it.

The average loss is less linear because the detection is affected by the image size, as is shown on Figure 4.9 however, the ANN became more invariant to the object size. The differences are shown on subsection 4.4.1.

Hardware consumption

Darknet framework does a full use of the GPU resources (it uses 99% of GPU capacity at 1.12 GHz). The RAM memory, 8 Gb total on Jetson TX2, is shared

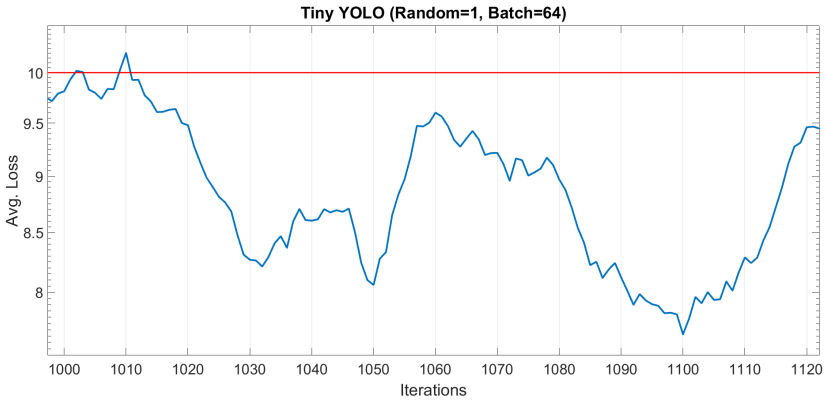


Figure 4.9: *Re-sizing effect(red line value is* 10*). For every 10 iterations, the input image size is changed and the error also changes. Batch=64 Subdivisions=16, random=1.*

between the GPU and CPU and the total use is a little more than 5 *Gb*. The resource consumption is shown on Figure 4.10.

```
RAM 5141/7854MB (lfb 1x2MB) cpu [100%@1859,off,off,1%@1871,2%@1874,1%@1871] EMC 69%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1880,off,off,1%@1882,1%@1885,1%@1885] EMC 65%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1851,off,off,1%@1869,2%@1874,0%@1873] EMC 67%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1856,off,off,2%@1873,0%@1872,3%@1874] EMC 68%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1854,off,off,0%@1868,2%@1873,2%@1874] EMC 66%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1881,off,off,0%@1883,0%@1882,1%@1884] EMC 73%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1807,off,off,2%@1824,1%@1826,1%@1827] EMC 65%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1883,off,off,2%@1884,2%@1883,2%@1881] EMC 71%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1881,off,off,1%@1884,1%@1884,1%@1883] EMC 64%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1880,off,off,2%@1884,3%@1889,3%@1884] EMC 68%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1882,off,off,1%@1883,1%@1882,0%@1884] EMC 64%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1879,off,off,1%@1884,2%@1882,1%@1883] EMC 68%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1881,off,off,1%@1883,2%@1884,2%@1884] EMC 65%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1860,off,off,1%@1875,1%@1866,2%@1882] EMC 66%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1881,off,off,1%@1883,1%@1884,1%@1883] EMC 70%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1856,off,off,2%@1873,2%@1874,3%@1874] EMC 66%@1600 APE 150 VDE 1203 GR3D 99%@1122
RAM 5140/7854MB (lfb 1x2MB) cpu [100%@1853,off,off,1%@1872,0%@1873,1%@1879] EMC 74%@1600 APE 150 VDE 1203 GR3D 99%@1122
```

(a) GPU Load.

```
nvidia@tegra-ubuntu:~/darknet$ sudo ~/jetson_clocks.sh --show
SOC family:tegra186  Machine:quill
Online CPUs: 0,3-5
CPU Cluster Switching: Disabled
cpu0: Gonvernor=schedutil MinFreq=345600 MaxFreq=2000000 CurrentFreq=1881600
cpu1: Gonvernor=schedutil MinFreq=345600 MaxFreq=2035200 CurrentFreq=2035200
cpu2: Gonvernor=schedutil MinFreq=345600 MaxFreq=2035200 CurrentFreq=2035200
cpu3: Gonvernor=schedutil MinFreq=345600 MaxFreq=2000000 CurrentFreq=1881600
cpu4: Gonvernor=schedutil MinFreq=345600 MaxFreq=2000000 CurrentFreq=1881600
cpu5: Gonvernor=schedutil MinFreq=345600 MaxFreq=2000000 CurrentFreq=1881600
GPU MinFreq=140250000 MaxFreq=1120000000 CurrentFreq=1122000000
EMC MinFreq=40800000 MaxFreq=1866000000 CurrentFreq=1600000000 FreqOverride=0
Fan: speed=80
```

(b) GPU Frequencies.

Figure 4.10: *GPU usage during the training stage. On top is marked the GPU consumption and on bottom is the GPU frequency range..*

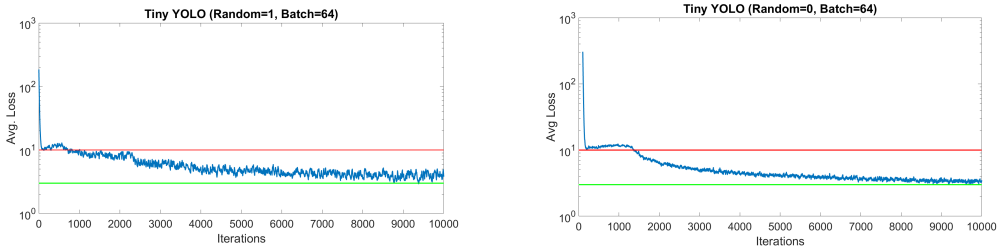
4.4.1 YOLO vs Tiny YOLO

Because of a matter of time consumption, it was performed 10000 iterations to see which model should be used. As the objective is to run in a real-time set, the speed detection is very important and it is the most valuable point.

Comparison graphics

The next figures (Figure 4.11 and Figure 4.12) shows the training comparison result. The tests were performed with two differences: one using or not the Random option, which changes the image size before feeding the network; second changing the number of batch size, batch=1 and batch=64 images. The xx axis is the number of iterations and the yy is the Average Loss (Error) in logarithmic scale.

The random option changes the image size at every 10 iterations allowing a more size independent training. On Figure 4.11 the training with the random option enabled (Figure 4.11(a)) has a less stable average loss and the error is a little higher when compared with its disabled counterpart. However, the the CNN learn with different images sizes, making it more size tolerant.



(a) Re-size option enable (random=1).

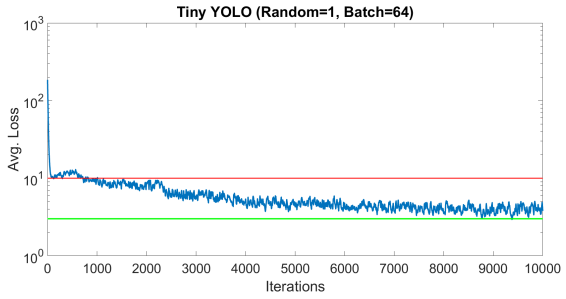
(b) Re-size option disable (random=0).

Figure 4.11: *Random option influence during the training. The green line marks* 3 *and red line mark* 10*.*

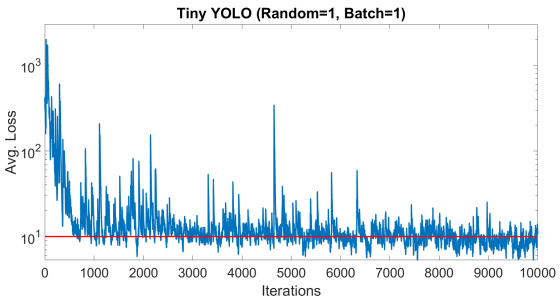
Batch size is the number of images that the CNN "sees" per iteration. The bigger that number is the more images the CNN uses to adjust the weights. The Figure 4.12 compares one training with batch=64 (Figure 4.12(a)) vs batch=1 (Figure 4.12(b)). The conclusions are immediate with an average loss decreasing much faster with a bigger batch size.

Training time

Training a network is a very time consumption task. It was measured the total time with the two architectures, including with changes in batch size and



(a) Batch size of 64.



(b) Batch size of 1.

Figure 4.12: *The batch size influence during the training. The green line marks* 3 *and red line mark* 10.

with/without the resize option. The results are shown on Table 4.4.

The hardware used for this test was Jetson TX2 for the most cases. To understand the Jetson TX2 performance was also tested with Nvidia Tesla K80, Quadro K2000 and with Intel Xeon E5-1650 v2.

The performance of Jetson TX2 is very similar to the Quadro K2000 as expected because they have piratically the same CUDA Cores. However, when Jetson TX2 is compared with Tesla K80 the scenario is totally different. **Tesla K80 is** $3\times$ **faster than Jetson TX2**. The use of GPU is very important to accelerate the training time. The prove of that is using only the CPU, an Intel Xeon E5-1650 v2 the time per iteration is more than 500 seconds, while with GPU is about 20 seconds or 8 seconds with Tesla K80.

Note: The case of using only CPU wasn't executed all the 10000 iterations.

Note 2: Nvidia Tesla K80 has 4992 CUDA Cores and 24 Gb of memory however, in the Amazon Web Services (AWS) it only provides half the graphical resources, 2496 CUDA cores and 12 Gb memory.

	Batch = 1 Subdivisions = 1	Batch = 64 Subdivisions = 16	
Random = 1	12 758 s (3.54 hours) Average: 1.27 s/iteration	517 470 s (143 hours) Average: 51.75 s/iteration‡	YOLO 2.0
Random = 0	11 933 s (3.31 hours) Average: 1.19 s/iteration	417 653 s (116 hours) Average: 41.76 s/iteration	
Random = 1	2 970 s (0.82 hours) Average: 0.29 s/iteration	229 272 s (63.68 hours) Average: 22.92 s/iteration⋆†∗	Tiny YOLO
Random = 0	2 738 s (0.76 hours) Average: 0.27 s/iteration	200 242 s (55.62 hours) Average: 20.02 s/iteration	

‡ On Nvidia Tesla K80, 2496 CUDA Cores (px2.xlarge AWS), the average is 16.63 s/iteration.
⋆ On Nvidia Quadro K2000, 384 CUDA Cores, the average is 22.34 s/iteration.
† On Intel Xeon E5-1650 v2, @6 × 3.50GHz, the average is > 500 s/iteration.
∗ On Nvidia Tesla K80, 2496 CUDA Cores (px2.xlarge AWS), the average is 7.80 s/iteration.

Table 4.4: *YOLO and Tiny YOLO time comparison. Time consumption to perform 10 000 iterations.*

4.5 System architecture

The system is composed of two main parts (Figure 4.13): the UAV part (orange boxes), has the camera which captures the images sending them to YOLO and is responsible to receive the new coordinates for the gimbal and to point the camera to the right position; second the Jetson TX2 part (blue boxes) is the brain of the systems, running YOLO which detects and classify the insulators and perform calculus to correct the gimbal with the gimbal control software.

The second part uses the ROS framework, explained in section 4.6, but it also works without ROS however, isn't possible to control the gimbal.

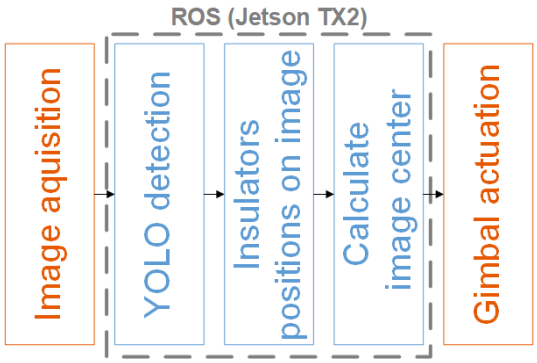


Figure 4.13: *System architecture overview - Processing pipeline*

4.6 ROS architecture

ROS is a very popular framework in robotics field and in this work, it's used to connect the YOLO with the gimbal control software. To integrate YOLO on ROS it was used a Darknet_ROS package[2]. Figure 4.14 represents how package Darknet_ROS works.

First of all, the image captured is published, in this case on topic */camera/image_raw*. Then the node *darknet_ros* subscribes the image's topic and run the YOLO on the image. The output are the number of objects detected, published on topic */darknet_ros/found_object*, and the bounding boxes of the objects detected, published on topic */darknet_ros/bounding_boxes*, in pixels coordinates.

Next the node */gimbal_control* calculates the error between the image centre and the insulator centre (bounding box centre) and send the new servo position values to the node */mavros/vision_pose/pose* which will actuate the servo. These last two steps are discussed more in detail on the next section 4.7.

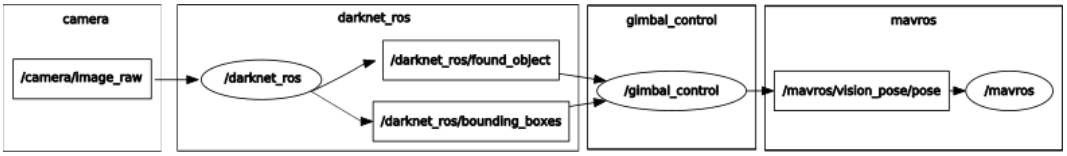


Figure 4.14: *ROS architecture.*

4.7 Gimbal Control

In order to get an image centred on the insulator it is necessary to adjust the camera position/orientation. The gimbal system allows this adjustment. To do that it is published the values, in angles, on topic */mavros/vision_pose/pose*, and the system will adjusts the servos position to point the camera.

These values are in function of the distance of image's centre and the insulator centre bounding box, measured in pixels. To minimise this distance, since the goal is to have the insulator on the centre of the image, is applied an P controller on this error.

The calculus is done on node */gimbal_control* (red box on Figure 4.15) and published on an mavros topic */mavros/vision_pose/pose*. Mavros[3] is a ROS package used to convert the MAVLink protocol to ROS framework.

[2]Darknet Ros package developed by Marko Bjelonic, Robotic Systems Lab, ETH Zurich, `https://github.com/leggedrobotics/darknet_ros`

[3]More information about mavros `http://wiki.ros.org/mavros`

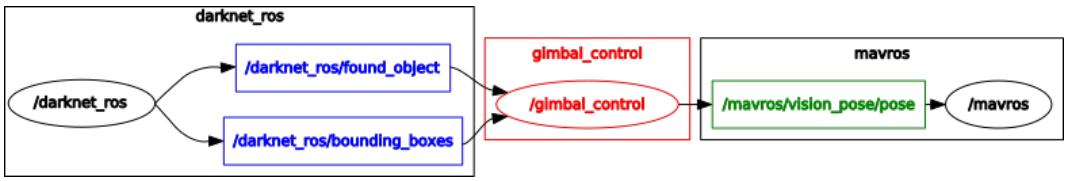


Figure 4.15: *ROS package /gimbal_control to control the gimbal position. It subscribes two topics from /darknet_ros and publish on topic /mavros/vision_pose_pose*

However, case the YOLO doesn't detect any insulator on the image, the gimbal system will go to the horizontal position one degree a time. Case it detect an insulator during this phase, it will centre the image on it. State transition is illustrated on Figure 4.16.

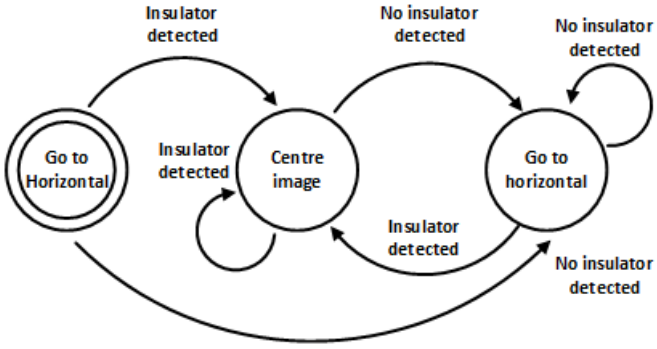


Figure 4.16: *Gimbal Control - State machine. The systems start with gimbal in horizontal position. If an insulator is detected, it will centre the image on it, else it will go to the horizontal position.*

5 Results

In this chapter it is reported the experiments done in a controlled environment. It shows the comparison between YOLO and Tiny-YOLO performance the results.

5.1 Error analysis

A way to evaluate the training quality is representing a Precision-Recall graph which is calculated as shown on Figure 5.1. The the **Recall** is the ratio of **True Positives** and **True Positives plus False Negatives**, which means True Positives with everything matters (in this case is insulators). **Precision** is the ratio between **True Positives** and **True Positives plus False Positive**, which is the True Positives with everything CNN thinks is a true object. The Precision-Recall graph is shownn on Figure 5.2(a) where Tiny-YOLO and YOLO are compared. As a complement of this graph it is illustrated the Recall and Precision of each architecture in Figure 5.2(b).

The validation test was made with 120 images which the CNN have never seen. The graphs show a deficient training as can be seen by very low precision with high recall levels (e.g. 1% precision at 80% recall).

On the Figure 5.2(b) it is demonstrated that if a precision about 70% is pretended the threshold has to be 20%. The threshold is applied on the boxes proposed by the YOLO.

The training phase is the most important and also the most time consuming cost

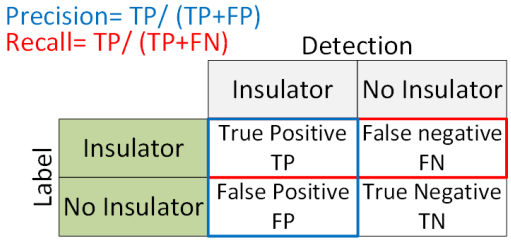
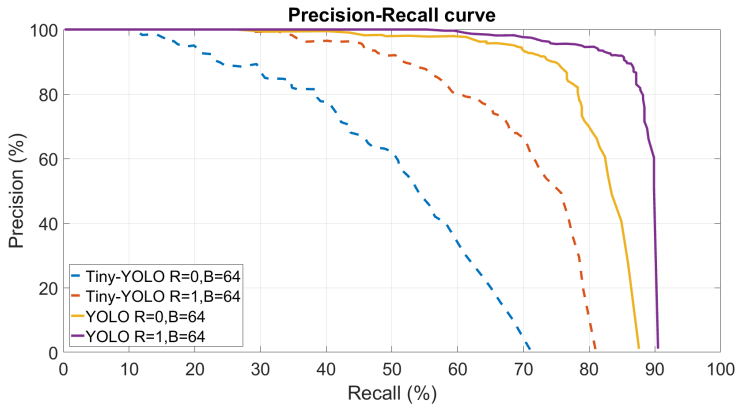
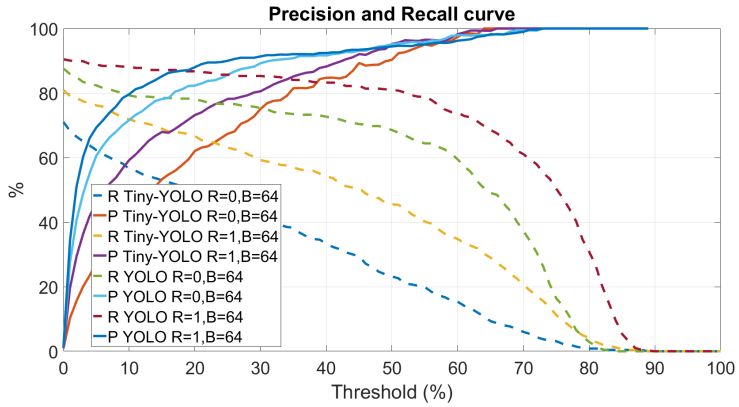


Figure 5.1: *Relation of Recall and Precision (adapted from [42])*



(a) Precision-Recall curve



(b) Precision (solid line) and Recall (dash line) curves

Figure 5.2: *YOLO and Tiny-YOLO error analysis. Figure 5.2(a) is the Precision-Recall curve and the Figure 5.2(b) is the Precision and Recall in function of the proposal threshold.*

of the use of an ANN. The training session with Tiny-YOLO (batch=64 images and re-size images option active) took approximately 20 days on Jetson TX2. After more than 72 000 iterations, the value of average loss (error) is about 2.0 with a learning rate of 0.01. **The average loss should be as close to zero as possible**.

However, it wasn't possible to achieve a lower value, even changing the learning rate to lower values.

The variation of the average loss at each iteration is shown in Figure 5.3.

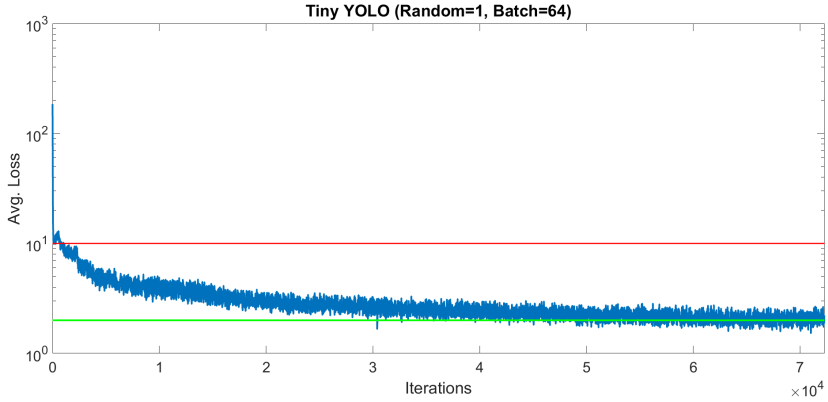


Figure 5.3: *AVG Loss variation of Tiny YOLO along the training. After more than 72 200 iterations, approximately 471 hours (almost 20 days), the average loss it's about* 2.0 *(green line) and the red line marks* 10.0.

5.2 YOLO kernels

As explained in Chapter 3, the first layer detects low-level features. The Figure 5.4 is the Tiny-YOLO $3 \times 3 \times 16$ kernels (filters) applied on the first convolution layer at different training stage. In the case of YOLO (Figure 5.5) the kernels on first convolution layer are $3 \times 3 \times 32$.

Both Figures show that the kernel slightly changes during the training phase according to the dataset.

(a) Initial kernel's Tiny YOLO. The kernels was trained on VOC dataset.

(b) Kernel's Tiny YOLO after 72 000 iterations.

Figure 5.4: *Tiny-YOLO kernel's evolution. Before, Figure 5.4(a), and after, Figure 5.4(b), the training. Is possible to see a slight difference on the most left kernels.*

(a) Initial Kernels of YOLO.

(b) Kernel's YOLO after 10 000 iterations.

Figure 5.5: *YOLO's kernels on the first convolution layer. In this case, the kernels didn't change during the training.*

5.3 Insulator detection speed

As this work pretends to be a real-time solution, the YOLO and Tiny-YOLO performance was tested in two ways: one the performance on a video file; second using ROS with a rosbag and a camera.

5.3.1 Running from video file

The video with a resolution of 1280×720 *px* was filmed on the ground. The Figure 5.6 shows some frames with insulator detection running between 10 and 12 frames per second (FPS). The GPU usage varies during the detection but wasn't as intense as in the training (the *GR3D* parameter on the bottom of Figure 5.7). As the background is mostly sky the detection was performed quite well.

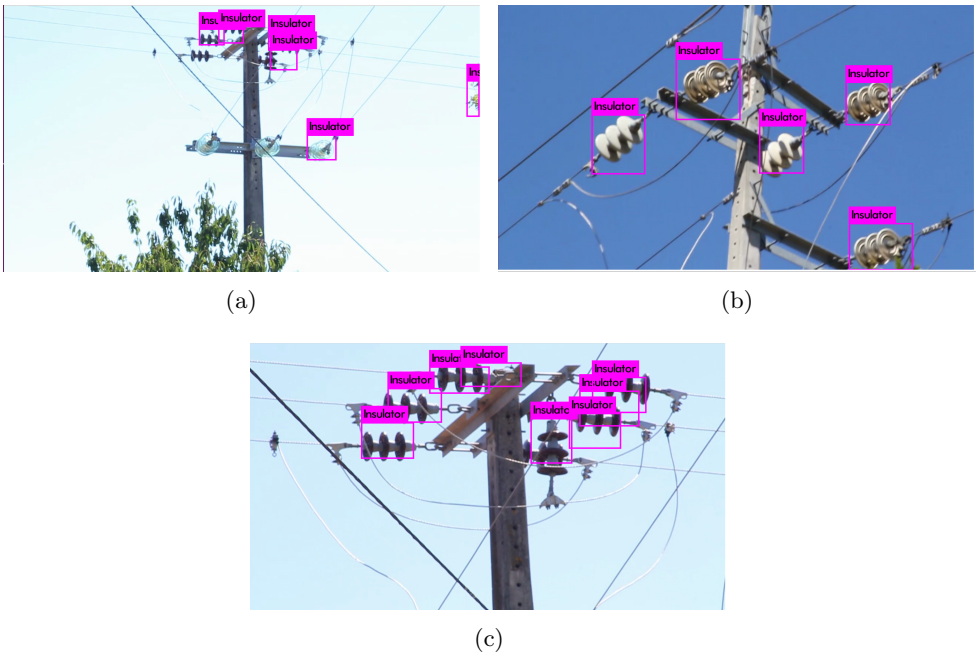
5.3.2 Running on ROS

The system was also tested in ROS environment in two ways: one with a rosbag recorded with a UAV, stored on an HDD connected via USB; second subscribing an image topic from a USB camera pointing to printed images of insulators.

Images from a Rosbag

The Figure 5.8 shows the detection running from the rosbag. On the left is the image with the bounding boxes marking the insulators detected; on the top right is the output of *darknet_ros* package with frame rate (in this case 0.8 FPS), number of objects detected and probabilities of detection; on the bottom right is the usage of CPU and GPU during the detection. As it happen with video file method, the GPU wasn't as much used as in the training mode.

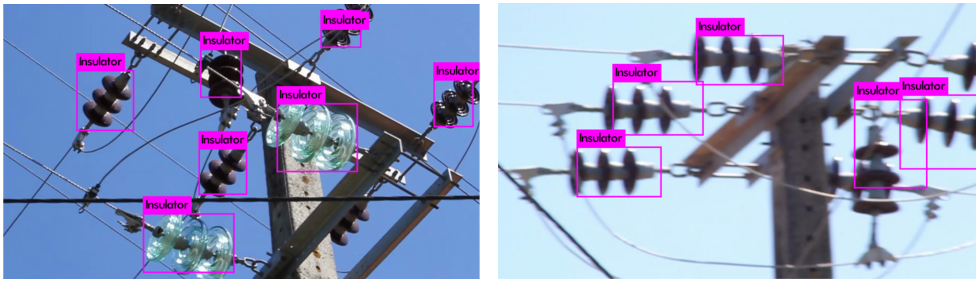
The speed processing/detection was very slow (running at less than 1 FPS) regardless of whether it was with YOLO or Tiny-YOLO architecture.



(a) (b)

(c)

Figure 5.6: *The Tiny-Yolo detection with a video file (*$1280 \times 720px$*) runs between 10 and 12 FPS on Jetson TX2. The images on video were not viewed by CNN during the training.*



(a) 9.5 FPS; Probability: 34% – 71% (b) 11.5 FPS; Probability: 34% – 82%

Figure 5.7: *Probabilities of detection and GPU usage from video file detection.*

Images from a camera

When getting images from the camera, via USB, the average was 3 FPS (with Tiny-YOLO, 2 FPS with YOLO) which was faster than the rosbag method despite */camera/image_raw* is publishing at 7 FPS.

To avoid false positives, the bounding boxes threshold was adjusted to 55% (left image on Figure 5.9). The centre and right images were taken with a threshold of 15% and it was possible to detect all the insulators on the image.

Figure 5.8: *ROS detection from a rosbag. On top is the image with the bounding boxes, in the middle is probability detection of each bounding boxes and on the bottom is the resources usage (Memory, CPU and GPU).*

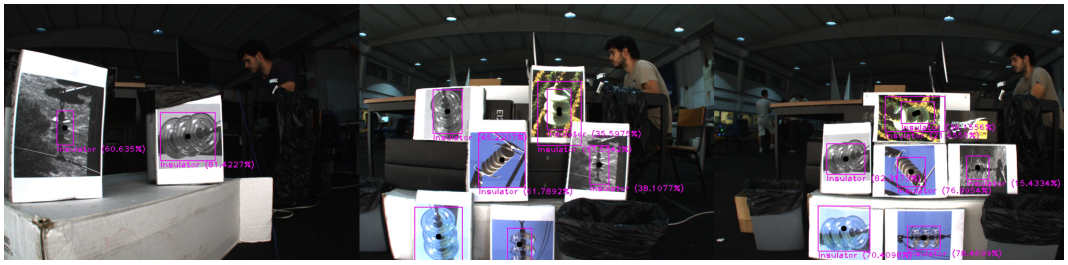


Figure 5.9: *Tiny-YOLO running on ROS with the image coming from a camera. The insulators are in different positions/orientation to test the invariant of it.*

5.3.3 False positives

As is possible to see on Precision-Recall graphs in Figure 5.2, the YOLO/Tiny-YOLO performance is poor, with a lot space for False Positives. On Figure 5.10 is show some cases when the detection fails, with figures where there are some insulators and other figures where there aren't insulators at all.

The tests did indicate that most of the false positives occur when there are blue spaces (sky) between the green vegetation. A way to avoid this situation is to use a higher threshold on bounding boxes because the majority of false positives has probability values around 40%.

Figure 5.10: *Some false positives detected by YOLO and Tiny-YOLO.*

5.4 Gimbal control

5.4.1 Insulator tracking

The system only tracks the insulator with the highest probability, independently of how many insulators are detected. The Figure 5.11 shows the sequence of tracking the most right side insulator because it's the insulator with the higher probability (77% vs 69%).

The tests demonstrated that the system is capable of tracking the insulator correctly with low frame rate. If the frame rate is increased probably it will be necessary to adjust the controller parameter.

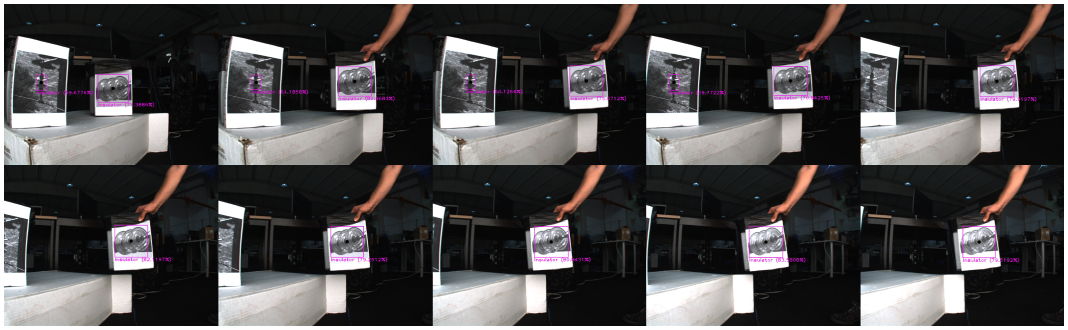


Figure 5.11: *ROS package /gimbal_control tracking the insulator with the higher probability (77% vs 69%). The UAV is static just the insulator print is moving.*

5.4.2 Insulator not found

In case the YOLO doesn't detect any insulator, the */gimba_control* send orders for gimbal to go to the horizontal position (Figure 5.12). If meanwhile the YOLO detects some insulator, it will centre the image on the insulator.

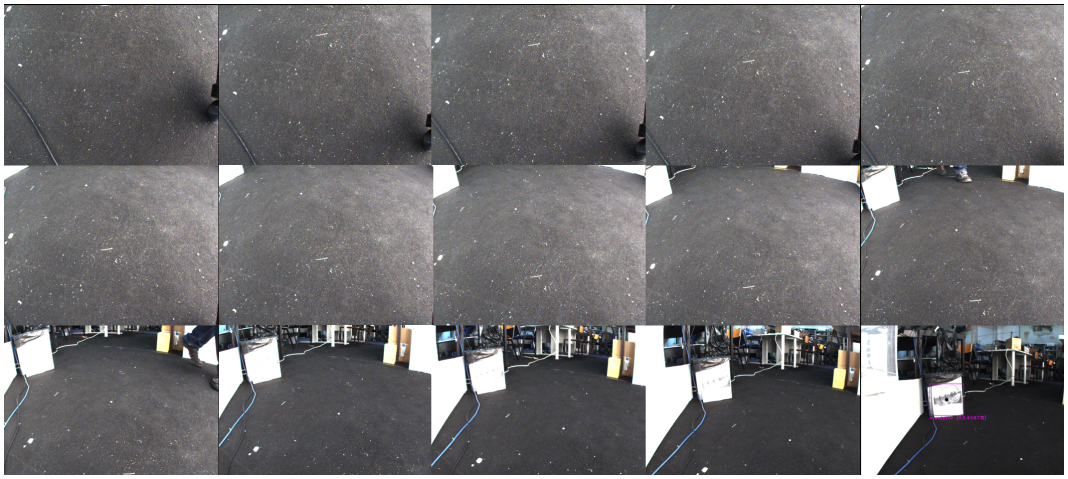


Figure 5.12: *The gimbal is moving to the horizontal position. In this case, after the gimbal is in the horizontal position, YOLO detected an insulator. The next step will be to centre the image on it.*

5.4.3 Tracking issues

During the tests some failures occured, mostly because of the false positive detection. Some single frame false positives have higher probabilities than the printed insulators, so the gimbal will centre the image on that false positive, ignoring the real insulators.

Another issue occured when the threshold is low, it detects a lot of false positives or insulators with very similar probabilities. This can cause the tracking system to fail or not perform so well because the insulator reference is always changing.

6

Conclusions and Future Work

The work developed had as mission developing a system capable of helping the insulator inspection making it more efficient, faster and safer than traditional methods.

The proposed system is capable of detecting insulators either during the inspection by capturing images from a camera or after the inspection with the video recorded (in post-processing).

The YOLO training process is efficient. With good GPU hardware is possible to train it in about a week. The YOLOv2 shows a better performance over Tiny-YOLO but it is slightly slower.

Insulators are centred in image by controlling the gimbal and this way the UAV isn't obligated to be right in front of the insulator as it may be slightly next to them (according to the gimbal limits). If the UAV is moving the gimbal will follow the insulator until it finds an object with higher probability of being an insulator.

However, detection has low precision and detect many false positives, falling short of the expected results. For the same threshold, YOLOv2 has less false positives than Tiny-YOLO. Since early, it was clear the importance of having a good dataset for training the CNN.

One of the problems was the similarity between images, having few variations of positions, distance, solar reflection etc. Other problem was the white balance of the images which in some of them weren't adjusted so that there were many images with high gains in green colour and others were too dark. As a consequence,

CNN considered the green background (like trees or leaves) as an insulator. During the marking of bounding boxes it was only considered insulators with two or more caps visible. Maybe the dataset shouldn't contains images with partial insulators or partially overlapped insulators to not be ambiguous to YOLO. In the continuation line of this work, it is extremely important to get a good dataset in way to validate the influences of such images. It can pass by review of the actual dataset and/or uses different images.

Once there are some different insulators types, it could be interesting during the detection to differentiate the insulators by the material type or size. This feature can provide feedback for statistical analyses of the quality type, adjust automatically camera parameters or adjust the UAV behaviour.

The next steps could be planning the UAV's trajectories around the insulators, in such a way it can automatically record images of insulators and provide a better and detailed image. This trajectories can be pre-planned with waypoints or develop an algorithm to keep the distance and looking for possible collisions.

Detect faulty insulator is a good and important feature for the system. However, it can be difficult to achieve if the cracks/defects are small and impossible to see from an UAV. Other point is how to train an CNN, or other model, to detect defective/break insulators. Probably it is necessary help of some electrical field companies to get images for the training.

Other field where this can be applied is in the railroad power line inspection, since the insulators are very similar to the electrical power line distribution and maybe the training would be faster.

In brevi, YOLO and Jetson TX2 seems to be a good choice for this application but it is necessary to work on the training phase. Personally, this work gave me the opportunity to learn about deep learning and all the difficulties to train an ANN with a non-ideal dataset. I hope the work continues and more features are added to develop a possible commercial product. Would this system, in the future, stop the crashes and people injuries?

Bibliography

[1] C. C. Whitworth, A. W. G. Duller, D. I. Jones, and G. K. Earp, "Aerial video inspection of overhead power lines," Power Engineering Journal **15,** 25–32 (2001).

[2] J. Formiga, J. Dinis, J. Fialho, F. Moreira, J. Almeida, A. Dias, E. Silva, M. Moreira, and T. Santos, "Field experiments in power line inspection with an unmanned aerial vehicle," 24th International Conference on Electricity Distribution (CIRED) (2017).

[3] F. Mirallès, N. Pouliot, S. Montambault, and H.-q. Ireq, "State-of-the-Art Review of Computer Vision for the Management of Power Transmission Lines," 3rd International Conference on Applied Robotics for the Power Industry (CARPI) pp. 1–6 (2014).

[4] W. Cheng and Z. Song, "Power Pole Detection Based on Graph Cut," 2008 Congress on Image and Signal Processing **3,** 720–724 (2008).

[5] I. Golightly and D. Jones, "Corner detection and matching for visual tracking during power line inspection," Image and Vision Computing **21,** 827–840 (2003).

[6] Z. Jian and Y. Ruqing, "Insulators recognition for 220kv/330kv high-voltage live-line cleaning robot," Proceedings - International Conference on Pattern Recognition **4,** 630–633 (2006).

[7] X. Zhang, J. An, and F. Chen, "A Simple Method of Tempered Glass Insulator Recognition from Airborne Image," 2010 International Conference on Optoelectronics and Image Processing **1,** 127–130 (2010).

[8] F. Zhang, W. Wang, Y. Zhao, P. Li, Q. Lin, and L. Jiang, "Automatic diagnosis system of transmission line abnormalities and defects based on UAV," 2016 4th International Conference on Applied Robotics for the Power Industry (CARPI) pp. 1–5 (2016).

[9] I. Y. H. Gu, U. Sistiaga, S. M. Berlijn, and A. Fahlström, "Automatic Surveillance and Analysis of Snow and Ice Coverage on Electrical Insulators of Power Transmission Lines," Computer Vision and Graphics: International Conference, ICCVG 2008 Warsaw, Poland, November 10-12, 2008 Revised Papers pp. 368–379 (2009).

[10] M. Oberweger, A. Wendel, and H. Bischof, "Visual Recognition and Fault Detection for Power Line Insulators," Computer Vision Winter Workshop (2014).

[11] J. Zhao, X. Liu, J. Sun, and L. Lei, in *Intelligent Computing Technology: 8th International Conference, ICIC 2012, Huangshan, China. Proceedings*, D.-S. Huang, C. Jiang, V. Bevilacqua, and J. C. Figueroa, eds., (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012), pp. 442–450.

[12] C. Martinez, C. Sampedro, A. Chauhan, and P. Campoy, "Towards autonomous detection and tracking of electric towers for aerial power line inspection," 2014 International Conference on Unmanned Aircraft Systems (ICUAS) pp. 284–295 (2014).

[13] C. Sampedro, C. Martinez, A. Chauhan, and P. Campoy, "A supervised approach to electric tower detection and classification for power line inspection," 2014 International Joint Conference on Neural Networks (IJCNN) pp. 1970–1977 (2014).

[14] M. J. B. Reddy, B. K. Chandra, and D. K. Mohanta, "A DOST based approach for the condition monitoring of 11 kV distribution line insulators," IEEE Transactions on Dielectrics and Electrical Insulation **18,** 588–595 (2011).

[15] Y. Liu, J. Yong, L. Liu, J. Zhao, and Z. Li, "The method of insulator recognition based on deep learning," 2016 4th International Conference on Applied Robotics for the Power Industry (CARPI) pp. 1–5 (2016).

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems pp. 1097–1105 (2012).

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition pp. 580–587 (2014).

[18] R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV) pp. 1440–1448 (2015).

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," IEEE Transactions on Pattern Analysis and Machine Intelligence **39,** 1137–1149 (2017).

[20] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," CoRR abs/1409.1556 (2014).

[21] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1–9 (2015).

[22] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 779–788 (2016).

[23] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017).

[24] L. Wang and H. Wang, "A survey on insulator inspection robots for power transmission lines," 2016 4th International Conference on Applied Robotics for the Power Industry (CARPI) pp. 1–6 (2016).

[25] S. R. Milan Jayatilaka, Madhavan Shanmugavel, "Robonwire: Design and development of a power line inspection robot," 1st International & 16th National Conference on Machines and Mechanisms (2013).

[26] Guo Rui, Zhang Feng, Cao Lei, and Yong Jun, "A mobile robot for inspection of overhead transmission lines," Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry pp. 1–3 (2014).

[27] J.-y. Park, J.-k. Lee, B.-h. Cho, and K.-y. Oh, "Development of Advanced Insulator Inspection Robot for 345kV Suspension Insulator Strings," Proceedings of the International MultiConference of Engineers and Computer Scientists **II,** 17–20 (2010).

[28] W. Hongguang, J. Yong, L. Aihua, F. Lijin, and L. Lie, "Research of power transmission line maintenance robots in SIACAS," 2010 1st International Conference on Applied Robotics for the Power Industry pp. 1–7 (2010).

[29] L. Zhong, J. Jia, R. Guo, J. Yong, and J. Ren, "Mobile robot for inspection of porcelain insulator strings," Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry pp. 1–4 (2014).

[30] S. Pirbodaghi, D. Thangarajan, T. H. Liang, M. Shanmugavel, V. Ragavan, and J. S. Sequeira, "A cooperative heterogeneous Unmanned Autonomous Systems solution for monitoring and inspecting power distribution system," pp. 495–502 (2015).

[31] I. Golightly and D. Jones, "Visual control of an unmanned aerial vehicle for power line inspection," pp. 288–295 (2005).

[32] G. Buskey, G. Wyeth, and J. Roberts, "Autonomous helicopter hover using an artificial neural network," Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164) **2,** 1635–1640 vol.2 (2001).

[33] S. Antunes and K. Bousson, "Safe flight envelope for overhead line inspection," Proceedings of the 2014 3rd International Conference on Applied Robotics for the Power Industry pp. 1–6 (2014).

[34] M. Malveiro, "Inspection of High Voltage Overhead Power Lines With UAV's," pp. 3–5 (2015).

[35] C. Deng, S. Wang, Z. Huang, Z. Tan, and J. Liu, "Unmanned aerial vehicles for power line inspection: A cooperative way in platforms and communications," Journal of Communications **9,** 687–692 (2014).

[36] Y. B. Yann LeCun and G. Hinton, "Deep learning," Nature pp. 436–444 (2015).

[37] "CS231n: Convolutional Neural Networks for Visual Recognition,", `http://cs231n.github.io`, university of Standford - Stanford Computer Vision Course, Accessed: 2017-09-05.

[38] M. A. Nielsen, in *Neural Networks and Deep Learning*, D. Press, ed., (2015).

[39] A. Deshpande, "A Beginner's Guide To Understanding Convolutional Neural Networks,", `https://adeshpande3.github.io/A-Beginner's-Guide-To-Understanding-Convolutional-Neural-Networks`, accessed: 2017-09-05.

[40] S. Y. Min Lin, Qiang Chen, "Network In Network," CoRR abs/1312.4400 (2013).

[41] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds., (Curran Associates, Inc., 2012), pp. 1097–1105.

[42] J. Chan, "Quora - What is the best way to understand the terms "precision" and "recall"?,", `https://www.quora.com/What-is-the-best-way-to-understand-the-terms-precision-and-recall`, accessed: 2017-09-25.