UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Recognition and tracking of vehicles in highways using deep learning

Ludwin Lope Cala

Dissertação de Mestrado do Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional (PPG-CCMC)

ICMC USP
SÃO CARLOS

Ludwin Lope Cala

Recognition and tracking of vehicles in highways using deep learning

Master dissertation submitted to the Institute of Mathematics and Computer Sciences – ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Profa. Dra. Roseli Aparecida Francelin Romero

USP – São Carlos
May 2019

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados inseridos pelo(a) autor(a)

LOPE CALA, LUDWIN
L864r Recognition and tracking of vehicles in highways using deep learning / LUDWIN LOPE CALA; orientador ROSELI APARECIDA FRANCELIN ROMERO. -- São Carlos, 2019.
70 p.

Dissertação (Mestrado - Programa de Pós-Graduação em Ciências de Computação e Matemática Computacional) -- Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, 2019.

1. Computer vision. 2. Deep learning. 3. Detection and classification. 4. Tracking. 5. Drone. I. FRANCELIN ROMERO, ROSELI APARECIDA, orient. II. Título.

Bibliotecários responsáveis pela estrutura de catalogação da publicação de acordo com a AACR2:
Gláucia Maria Saia Cristianini - CRB - 8/4938
Juliana de Souza Moraes - CRB - 8/6176

Ludwin Lope Cala

Reconhecimento e rastreamento de veículos em rodovias usando aprendizagem profunda

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências – Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientadora: Profa. Dra. Roseli Aparecida Francelin Romero

USP – São Carlos
Maio de 2019

Este trabalho é dedicado para minha familia e amigos.

A person who never made a mistake never tried anything new. — Albert Einstein

ACKNOWLEDGEMENTS

Gostaria de agradecer primeiramente a minha mãe Teresa, meu pai Fortunato e a meus irmãos, a minha orientadora, Profa. Dra. Roseli Aparecida Francelin Romero, pela orientação ao longo deste projeto, ao meus professores do ICMC pelas excelentes aulas. Aos familiares pelo apoio e conforto durante esses anos de trabalho e estudo.

Aos meus queridos amigos do ICMC e São Carlos por terem compartilhado comigo amizade e conhecimentos (Ernesto, Oscar, Aurea, Kevin, Silvia, Fabiana, Marcela, Alain, Gustavo, Paul, Joel, Danilo, Alex, Juan Pablo, Guilherme, Raphael, Milagros e muitos mais) estarei sempre muito agradecido com todos eles.

Ao equipe da secretaria de pós graduação e os demais funcionários do ICMC e da USP pelo suporte prestado.

E finalmente, agradeço à CNPq pelo apoio financeiro.

RESUMO

LOPE, L. C. **Reconhecimento e rastreamento de veículos em rodovias usando aprendizagem profunda**. 2019. 70 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Veículos aéreos não tripulados têm se tornado cada vez mais populares e sua capacidade de analisar imagens coletadas em tempo real tem chamado a atenção de pesquisadores quanto ao seu uso em diversas tarefas, como vigilância de ambientes, perseguição, coleta de imagens, entre outros. Esta dissertação propõe um sistema de rastreamento de veículos através do qual os UAV podem reconhecer um veículo e monitorá-lo em rodovias. O sistema é baseado em uma combinação de algoritmos de aprendizado de máquina bio-inspirados VOCUS2, CNN e LSTM e foi testado com imagens reais coletadas por um robô aéreo. Os resultados mostram que é mais simples e superou outros algoritmos complexos, em termos de precisão.

Palavras-chave: Visão Computacional, Aprendizado Profundo, Rede Neural Recorrente, Rastreamento, Detecção e Classificação, Drone.

ABSTRACT

LOPE, L. C. **Recognition and tracking of vehicles in highways using deep learning**. 2019. 70 p. Dissertação (Mestrado em Ciências – Ciências de Computação e Matemática Computacional) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos – SP, 2019.

Unmanned aerial vehicles (UAV) have become increasingly popular and their ability to analyze images collected in real time has drawn the attention of researchers regarding their use in several tasks, as surveillance of environments, persecution, collection of images, among others. This dissertation proposes a vehicle tracking system through which UAVs can recognize a vehicle and monitor it in highways. The system is based on a combination of bio-inspired machine learning algorithms VOCUS2, CNN and LSTM and was tested with real images collected by an aerial robot. The results show it is simpler and outperformed other complex algorithms, in terms of precision.

Keywords: Computer Vision, Deep Learning, Recurrent Neural Network, Tracking, Detection and Classification, Drone.

LIST OF FIGURES

1 ROLO architecture, Ning et al. (2016). 28
2 Overview of saliency system VOCUS2, Frintrop et al. (2015). 32
3 Structure of the Perceptron, Iyoda (2000). 34
4 MLP Architecture. 35
5 LSTM memory cell, Kyunghyun (2017). 37
6 LSTM and GRU, Chung et al. (2014). 38
7 LeNet convolutional network topology, LeCun et al. (1998). 38
8 Convolution: encoder, transformation matrix of pixels to array of characteristics, Li (2016). 39
9 Example convolution, with two-by-two kernel, Rocha (2015). 39
10 Pooling, two-by-two cells, with a max-pooling function, Rocha (2015). . . . 40
11 Mini-network replacing the 5x5 convolutions, Szegedy et al. (2015). 44
12 YOLO Architecture, Redmon et al. (2015). 45
13 The YOLO Model, Redmon et al. (2015). 45
14 Simplified overview ROLO, Ning et al. (2016). 46
15 Proposed System General Scheme. 50
16 Test VOCUS2 algorithm. 50
17 Saliency object by VOCUS2. 51
18 Architecture proposed for classification and tracking. 51
19 CNN Arch.1 52
20 CNN Arch.2 52
21 Example of data collected and classified manually in car and no-car. 56
22 Test Vehicle Tracking: Box green see our LSTM track and the white box is the real box (Video#1). 59
23 Vehicle Tracking: Box green show the proposed LSTM track and the white box is the real box. 60
24 Vehicle Tracking: The green box is proposed LSTM-tracker, the pink box is Correlation Filter Tracker and the white box is the real box. 61

LIST OF TABLES

1 Neural network architectures tested by (HUTTUNEN; YANCHESHMEH; CHEN, 2016) 26
2 Characteristics of the architectures 57
3 Table of errors in each Cross-Validation Folds 57
4 Average of accuracy and precision of architectures 1, 2 and Inception-v2 . . . 58
5 Training for Vehicle Tracking 58
6 Comparison between CFT and LSTM tracker 59
7 Confusion matrix average. 59
8 To train LSTM and GRU tracker by incremental epochs 60

LIST OF ABBREVIATIONS AND ACRONYMS

AP	Average Precision
CFT	Correlation Filter Tracker
CNN	Convolutional Neural Network
DNN	Deep Neural Networks
FFT	Fast Fourier Transform
GRU	Gated Recurrent Unit
IoU	Intersection over Union
LSTM	Long Short-Term Memory
MLP	Multi-Layer Perceptrons
NLP	Neuro-Linguistic Programming
RNN	Recurrent Neural Network
ROLO	Recurrent YOLO
SSD	Single Shot Multibox Detector
UAV	Unmanned aerial vehicles

CONTENTS

1 INTRODUCTION . . . 21
1.1 Contextualization and Motivation . . . 21
1.2 Objectives . . . 22
1.3 Organization . . . 23

2 RELATED WORKS . . . 25
2.1 Detection and Classification of Objects . . . 25
2.2 Tracking of Moving Object . . . 27
2.3 Final considerations . . . 28

3 THEORETICAL FOUNDATION . . . 31
3.1 Visual Attention . . . 31
3.1.1 VOCUS2 . . . *32*
3.2 Correlation Filter Tracker (CFT) . . . 32
3.3 Machine Learning . . . 33
3.3.1 Supervised learning . . . *34*
3.4 Foundations of Neural Networks . . . 34
3.4.1 Perceptron . . . *34*
3.4.2 Multi-Layer Perceptron Networks . . . *35*
3.4.3 Back-propagation . . . *35*
3.5 Deep Neural Networks . . . 35
3.5.1 Recurrent Neural Networks . . . *36*
3.5.1.1 *Long Short-Term Memory Network (LSTM)* . . . 36
3.5.1.2 *Gated Recurrent Unit Network (GRU)* . . . 37
3.5.2 Convolutional Neuronal Networks - CNN . . . *37*
3.5.2.1 *Convolution* . . . 38
3.5.2.2 Pooling . . . 39
3.5.2.3 *Neural Network Optimization* . . . 40
3.5.2.4 *SoftMax function* . . . 43
3.5.2.5 *Dropout* . . . 43
3.5.2.6 *Inception Network* . . . 44
3.5.2.7 *You Only Look Once (YOLO)* . . . 44
3.5.2.8 *Recurrent YOLO (ROLO)* . . . 45

3.5.2.9 Transfer Learning . . . 46
3.6 Evaluation techniques . . . 46
3.6.1 Cross-Validation . . . 46
3.6.2 Confusion Matrix . . . 47
3.6.3 Average Precision . . . 47
3.6.4 Intersection over Union . . . 47
3.7 Final considerations . . . 47

4 THE PROPOSED SYSTEM . . . 49
4.1 General Outline . . . 49
4.2 Saliency . . . 49
4.3 Recognition . . . 51
4.4 Tracking . . . 53
4.5 Final considerations . . . 53

5 RESULTS AND DISCUSSION . . . 55
5.1 Implementation Framework . . . 55
5.2 Data Collection . . . 55
5.3 Training for Vehicle Recognition . . . 56
5.4 Tracking of Vehicles . . . 58
5.5 Discussion of Results . . . 59
5.6 Final considerations . . . 61

6 CONCLUSION AND FUTURE WORKS . . . 63
6.1 Summary of Findings . . . 63
6.2 Limitations . . . 64
6.3 Future Works . . . 65

BIBLIOGRAPHY . . . 67

CHAPTER

1

INTRODUCTION

1.1 Contextualization and Motivation

Machines have been able to make decisions by themselves. A refrigerator, for example, uses an own application, called Smart Access, to send messages on foods about to run out or expire.

The autonomy of devices has been improved through the implementation of more sensors and more intelligent softwares, and those that accompany us every day, as smart phones, have made our lives easier. Their utilities comprehend control of feeding, daily routine, physical efforts, location maps, etc., and such devices can also work with others via Bluetooth (e.g., smart watches can control them for changing music or answering a call, apart from their own functionalities).

Another type of device, called Unmanned Aerial Vehicle (UAV), has become popular in the market. It has a camera, which provides a wider vision during the filming of a video or taking of photographs, and is also a tool that carries objects and searches and explores places of difficult access. UAVs can work, for instance, in police actions for persecution of fugitives, monitoring of environmental reserves, manifestations, rebellions in prison units and drug trafficking areas, and search and rescue.

Such activities developed by UAVs provide a better police service to the society and make their operations less risky. The use of those devices has been common in police stations in developed countries. They are sophisticated and equipped with GPS navigation systems, thermal cameras and long-range video transmitters, and some UAVs are surveillance aircraft at a low budget.

This type of a system covers a wider area faster and saves considerable time, which is a key factor in certain operations. Some of those gadgets are low-priced and highly useful in comparison to a large aerial vehicle; moreover, most of them are piloted by people. Recent

research has focused on equipping them with more sophisticated sensors, hardware and software for making them truly autonomous.

We simulated the recognition and persecution of a vehicle by a UAV according to two approaches, namely Convolutional Neural Network (CNN), for recognizing vehicles, and Long Short Term Memory (LSTM), for tracking, and propose a system that uses real data collected by a UAV.

The architecture is composed of three phases, namely saliency, recognition and tracking of a vehicle. VOCUS2 method used in the saliency phase detected the most salient object in an image, whereas CNN and LSTM were used for recognition and tracking, respectively.

The two final phases use Deep Learning (DL) method, which can be currently applied, due to its higher computational power. It has reached good records in world competitions, e.g., recognition and classification of objects in the competition of ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (DENG; SOCHER; LI, 2009).

Deep learning is a particular machine learning approach that achieves high power not only in images recognition and classification, but also in other areas of work, as autonomous video games, robots and drones, analyses of users' emotions, detection of mail problems, among others.

1.2 Objectives

Our main objective is to develop a system that uses real data of a UAV and simulates the recognition and tracking of a vehicle in a highway using bio-inspired techniques, as saliency system for detection and deep neural networks for recognition and tracking. The specific objectives are:

(a) Collection of data for training and testing;

(b) Segmentation of objects in an image by VOCUS2 visual attention algorithm;

(c) Implementation of a deep neural network, called CNN (Convolutional Neural Network), using Deep Learning TensorFlow library;

(d) Implementation of a robust tracking algorithm that accompanies a previously defined vehicle; and

(e) Testing of the software with real data collected by a UAV on a highway.

1.3 Organization

This text is structured as follows: Chapter 2 focuses on some related works; Chapter 3 describes the theoretical foundations; Chapter 4 introduces the methodology; Chapter 5 reports experiments, results and comparisons; and finally Chapter 6 addresses the conclusions and limitations of the proposed system and suggests some future works.

CHAPTER

2

RELATED WORKS

In this Chapter, some related works of the area are described: Chapter 2.1 involves the detection and classification of objects and Chapter 2.2 consists of the tracking of a moving object.

2.1 Detection and Classification of Objects

Object detection and classification is an active area of research. In these days, there are competitions to classify various kinds of objects. One of the most important competitions is ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The database was presented for the first time at the 2009 Conference on Vision and Pattern Recognition (CVPR), in Florida, by researchers from the Department of Computer Science at Princeton University (DENG; SOCHER; LI, 2009). This competition contains 1000 categories and 1.2 million images each year. Deep learning techniques were introduced in the competition in 2012. In 2015, it was possible to beat the human level of accuracy at classifying objects.

The proposed system in this thesis has a similar objective to (MONTANARI, 2015). In this work, the VOCUS2 technique, a visual attention technique, is used to obtain the saliency object segmentation.

The bag-of-features technique was applied for the classification of the images, and Camshift and Kalman filters were used to track the objects. The method obtained an accuracy of 79.82%. Specifically, as detector and a descriptor used SURF(Speeded-Up Robust Features) algorithm.

In (HUTTUNEN; YANCHESHMEH; CHEN, 2016), a system was proposed for recognizing 4 types of vehicles: buses, trucks, vans and small cars, one obtaining 97% of accuracy. In Table 1, are shown the different parameters of the CNN architectures that tested and obtained the best accuracy.

The parameters in the column **Selected Value** are the best parameters to obtain 97% of

Hyperparameter	Range	Selected Value
Number of Convolutional Layers	1 – 4	2
Number of Dense Layers	0 – 2	2
Input Image Size	{64, 96, 128, 160}	96
Kernel Size on All Convolutional Layers	{5, 9, 13, 17}	5
Number of Convolutional Maps	{16, 32, 48}	32
Learning rate	$10^{-5} - 10^{-1}$	0.001643

Table 1 – Neural network architectures tested by (HUTTUNEN; YANCHESHMEH; CHEN, 2016)

accuracy. In this architecture, the network receives an image input of 96 x 96 with 3 channels (RGB) and has the first convolutional layer with 32 feature maps, followed by a max-pooling (SCHERER; MULLER; BEHNKE, 2010), reducing the image dimension to 48 x 48. The second layer is also a convolutional layer with 32 feature maps, followed by a max-pooling, reducing the image dimension to 24 x 24. Finally, there are 2 fully connected layers of 100 neurons each one, to produce an output layer constituted by 4 output neurons with the SoftMax function (DUAN *et al.*, 2003).

In (RIVEROS; CACERES; CHáVEZ, 2016), a system was proposed for automobiles classifier based on CNN, which obtained 95.6% of accuracy for images collected by a security camera. The network architecture is based on LeNet-5 and it was tested with different activation functions (RELU, sigmoid, PreRELU), different functions for the pooling layer (AVG, MAX, STO), different ways to initialize weights from CNN (Xavier, Uniforme, Gaussian) were applied for getting the best results with RELU, MAX and XAVIER, respectively. All these functions are described in details on (RIVEROS; CACERES; CHáVEZ, 2016).

In (WANG *et al.*, 2016), the AlexNet model was used to recognise the vehicle brake lights. It was implemented in the library Open Source Caffe, the parameters as the learning factor had been defined in AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), the accuracy obtained was 89% and the results with well-focused image were 99%. The dataset was built from collections carried out in the street, the input of the data ran in real time, a GPU was used, algorithm ROI (Region Of Interest) was used which tried to clear the noises.

Another form of classifying car images with CNN is using transfer learning (CaffeNet and GoogLeNet), a Classification of Vehicle Make and Model Using Convolutional Neural Networks and Transfer Learning, proposed by (LIU, 2015). They implement, train, and test several state-of-the-art classifiers trained on domain general data sets for the task of identifying the maker and models of cars from various angles and different settings, but not from UAVs perspective. They experimented with different levels of transfer learning for fitting these models over to their domain. State-of-the-art results were compared with their model, and discuss the advantages of their approach about transfer learning. They reported: *the transfer learning was almost necessary to achieve decent performance*. For this reason, we used transfer learning from GoogLeNet(InceptionNet) (SZEGEDY *et al.*, 2015) with our collected data.

In (SATAR; DIRIK, 2018), it was proposed Deep Learning Based Vehicle Make-Model Classification, which combined a Single Shot Multibox Detector (SSD) (LIU *et al.*, 2015) model, one of the most popular detectors with a CNN model (ResNet (HE *et al.*, 2015)). To train on the database, they first detected the vehicles by SSD algorithm which reduces the time for annotation and then they used CNN model(ResNet). This model reached approximately 4% better accuracy results than others conventional CNN models.

2.2 Tracking of Moving Object

Object tracking is an important task in the field of computer vision. In its simplest form, tracking can be defined as the problem of estimating the trajectory of an object in an environment or around a scene. Almost all tracking algorithms require the *detection of objects* (YILMAZ; JAVED; SHAH, 2006), for this reason we discussed about it in the previous sub-section 2.1.

Tracking of objects was proposed in (VIDAL, 2010) using Particle Swarm (KENNEDY; EBERHART, 1995), which is a heuristic optimization method of non-linear functions; it works with the principle of social intelligence. The particles initially are without any prior guidance. Those particles move until one of them can find the best position (best state) according to a predefined goal (cost function), which eventually attracts the nearest particles. Then, these particles find a region of interest in the successive frames of a sequence of images. Despite requiring a significant computational cost, it has a greater ability to traverse the search space, increasing the probability of obtaining global optimal.

A system was proposed in (M.A.AMARAL *et al.*, 2015), for detection and tracking of multiple moving vehicles in the environment around an autonomous vehicle using a Light Detection and Ranging (LIDAR) 3D sensor. It operates in four steps: pre-processing, segmentation, association and tracking. In the tracking step, the states of the objects are estimated using a bootstrap particle filter (ARULAMPALAM *et al.*, 2002). At each iteration, the bootstrap filter operates in three phases: prediction, correction and re-sampling. In the results, the proposed system was able to detect and track objects with velocities above 3.0 m/s, in some moments, the system detected false positives. Of these, there are three causes for the detection of false positives: changing the angle of observed static objects, proximity of the observed static objects and rapid wind movement of components of observed static objects.

In (MONTANARI, 2015), they used Camshift technique (BRADSKI, 1998) with Kalman filter (WELCH; BISHOP, 1995). On the other hand in (RIVEROS; CACERES; CHáVEZ, 2016) they used Camshift with Correlation Filter Tracker (DANELLJAN *et al.*, 2014).

The authors observed that the Camshift algorithm has been kept for years as the most robust technique for tracking objects in real time by keeping tagged the object despite its variation in size throughout its trajectory. However, the Camshift algorithm decays in performance in images where the variation of illumination and occlusion are present. In these aspects, highlighted

algorithms such as correlation filters or Kalman filters are quite useful. For more information about tracking algorithms, the reader may consult (YILMAZ OMAR JAVED, 2006).

Discussing for this type of tracking algorithm is not yet robust for our object tracking. A robust object tracking requires knowledge and understanding of the object being tracked (GORDON; FARHADI; FOX, 2017). For this reason, in this work we will focus on Recurrent YOLO (ROLO) (NING *et al.*, 2016), a tracking framework that uses a highly efficient image detector called YOLO (REDMON *et al.*, 2015), which is a fast CNN detector (45 frames per second). The ROLO framework receives the outputs of the last fully connected layer of the YOLO as it can be seen in Fig. 1.

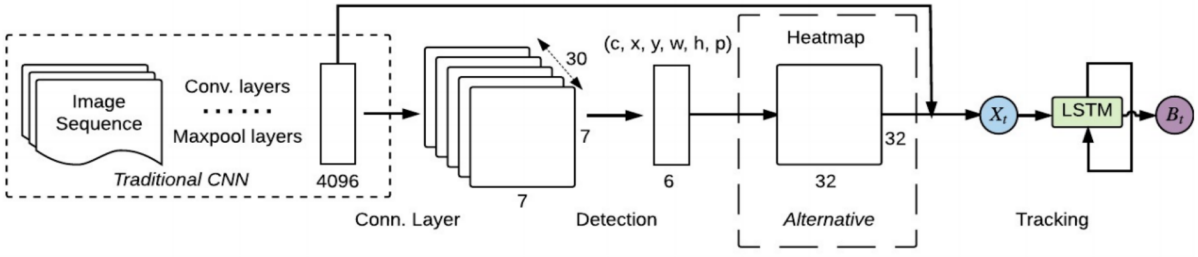


Figure 1 – ROLO architecture, Ning et al. (2016).

This layer has 4096 neurons. Its output signals are sent into the recurrent network LSTM and for a layer that will extract the position coordinates, represented by the vector (0, x, y, w, h, 0) of each object that is being tracked, where (x,y) are central position and (w,h) the width and height of the tracked object, these parameters of vectors are added in the input of the recurrent network LSTM, and therefore it receives 4102 inputs coming from YOLO. The LSTM network will be described in more detail in subsection 4.4.

The framework ROLO has two phases. The first phase does not solve occlusion and does not use Heatmap, that is a YOLO output reshape (32x32 size). The second phase includes Heatmap to solve occlusion. Thus, the framework ROLO predicts the new position of the object. In our case, we have implemented only the first phase, therefore our proposal do not solve occlusions.

In (GHAEMMAGHAMI, 2017), supervised recurrent convolutional neural networks are used for visual human tracking. In the part of recognition, the author used different architectures of CNN, which are YOLO detection and SSD detection, both with LSTM tracker, getting better results with YOLO detection.

2.3 Final considerations

In this Chapter, the main techniques and algorithms for recognition and tracking based in classical techniques and modern techniques such as the last based in deep learning algorithms

were exposed.

We focus on Deep Neural Networks (DNN) techniques, the two main types of DNN are Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). These types of algorithms will help us detect and track the vehicle in the highway. The other classical techniques will allow us to compare them with our proposed system.

CHAPTER

3

THEORETICAL FOUNDATION

In this Chapter, Visual Attention, Deep Learning, Transfer Learning, among others are introduced. The metrics used for training and performance evaluation are also presented.

3.1 Visual Attention

In recent years, visual attention has been studied following two approaches: Bottom-up and Top-down processes. Frequently, both are focused on the selection of the most relevant stimuli in the search field. The two approaches are explained in the next items:

(a) **Bottom-up process:** Use direct experience in the sensory receptors to arrive at the perception. It presents a processing basically marked by physiological parameters of stimulus detection (*e.g.,* spatial and temporal aspects).

(b) **Top-down process:** Use previous knowledge to arrive at the perception. It presents processes centered on the internal guidelines of the subjects, being estimated by parameters and targets established by the cognitive system under direct influence of mnemonic processes and mental representations (ROSSINI, 2006).

There are algorithms based on a single process, such as the work of (ITTI, 2005), which is based on bottom-up only. Also, there is some research using both approaches as the Saliency System called VOCUS (Visual Object detection with a CompUtational attention System) (FRINTROP; WERNER; GARCIA, 2015a). VOCUS is a bottom-up and a top-down system capable of automatically selecting regions of interest in images and detecting specific objects.

There is another algorithm called VOCUS2 that is a successor of the previous VOCUS system and it is presented in the next section.

3.1.1 VOCUS2

The VOCUS2 algorithm avoids searching an object through the complete image, reducing the computational time to analyze an image. Thus, VOCUS2 can be of significant use for real-time applications (FRINTROP; WERNER; GARCIA, 2015b). This technique provides the most salient parts of an image, dividing the actual image in 3 images as shown in Figure 2. The image is separated by the intensity channel blue-yellow, then the same image by the red-green channel, and finally, by the intensity of the 3 colors. From this, we obtain 3 different images, each image generates two pyramids of images which are the central part (the object) and the neighbor parts (the neighbors of the object), so at the end the 3 merged images get the most salient part of the image.

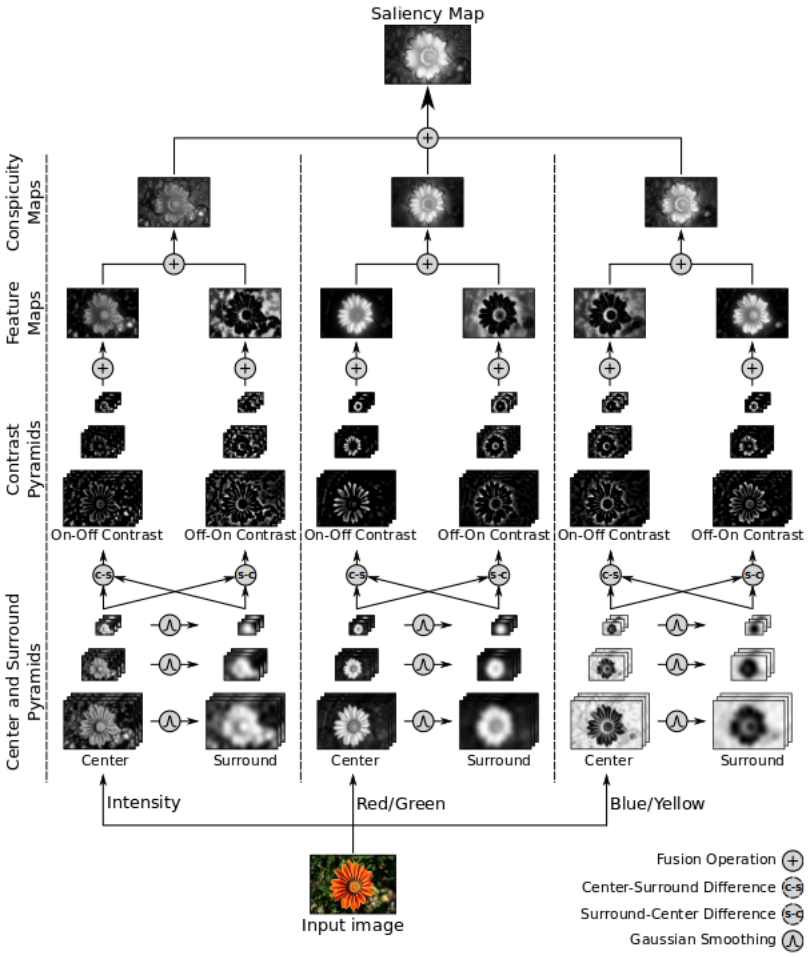


Figure 2 – Overview of saliency system VOCUS2, Frintrop et al. (2015).

3.2 Correlation Filter Tracker (CFT)

In recent years, correlation filters have gained attention in the area of visual tracking due to their computational efficiency and competitive performance (NAM; HAN, 2016), thus several

variations of correlation filter tracker have been subsequently investigated to improve tracking performance.

To create a fast tracker, correlation is computed in the Fourier domain Fast Fourier Transform (FFT) (W. Press *et al*, 1988). First, the 2D Fourier transform of the input image: $F = \mathscr{F}(f)$, and of the filter: $H = \mathscr{F}(h)$ are computed. The Convolution Theorem states that correlation becomes an elementwise multiplication in the Fourier domain. Using the $\odot$ symbol to explicitly denote element-wise multiplication and * to indicate the complex conjugate, correlation takes the form (BOLME *et al.*, 2010):

$$G = F \odot H^* \tag{3.1}$$

The correlation output is transformed back into the spatial domain using the inverse FFT. The bottleneck in this process is computing the forward and inverse FFTs.

3.3 Machine Learning

Machine learning is essentially a form of applied statistics with increased emphasis on the use of computers to statistically estimate complicated functions and a decreased emphasis on proving confidence intervals around these functions (GOODFELLOW; BENGIO; COURVILLE, 2016) (frequency estimators, Bayesian inference, etc).

A central task in machine learning is feature extraction (WIATOWSKI; BÖLCSKEI, 2015). The idea behind feature extraction is feeding characteristic features of the signals rather than the signals themselves to a trainable classifier. There are three paradigms of machine learning: supervised, unsupervised and reinforcement learning, described as follows:

(a) **Supervised learning:** The agent has inputs and expected outputs and learns a function to perform the mapping.

(b) **Unsupervised Learning:** Learn without feedback. The most common are clustering algorithms.

(c) **Semi-supervised Learning:** It is a combination of the previous items *(a)* and *(b)*. It is due to noise or lack of labels.

(d) **Reinforcement Learning:** The agent learns through a series of reinforcements (rewards or punishments).

In this work, supervised learning has been used and reinforcement learning approach will be left to the future works section.

3.3.1 Supervised learning

Given a training set $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ where each y_j is generated by $y = f(x)$, where f is unknown; the challenge is to find a $h(x)$ that approximates f; x and y can adopt any type of data, x_j is a vector of attributes, h is a hypothesis (RUSSELL; NORVIG, 1995). Learning consists of looking for a hypothesis that fits better with the data.

A test set is used to measure the accuracy of a hypothesis. A hypothesis generalizes well, when it predicts correctly with unfamiliar or unknown inputs. When the output is a value of a finite set, the problem is called classification and when the output is a real number the problem is called regression.

3.4 Foundations of Neural Networks

3.4.1 Perceptron

The most basic form of a neural network is a perceptron, shown in Figure 3, which is the artificial representation of a neuron. It has three important components: the dendrites, the soma and the axon. It is important to highlight the existence of a bias that serves to increase the degrees of freedom, allowing a better adaptation by the artificial neuron.

A perceptron has been characterized by a binary classifier, which can be able to have multiple input variables (a vector). This artificial neuron can learn different values (input vectors), generating just a binary answer to classify. Each time it is trained, the weights can vary and the generated answer is the same at the end. Thus, it let us know that the same problem have several solutions. The basic components of a perceptron are: input, weight, activate function and output, where the weight is the main component.

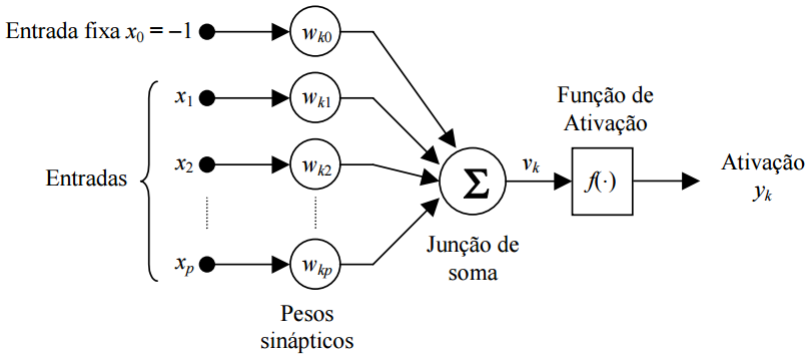


Figure 3 – Structure of the Perceptron, Iyoda (2000).

3.4.2 Multi-Layer Perceptron Networks

Multi-Layer Perceptrons (MLP) are characterized by the presence of at least one intermediate layer, shown in Figure 4. Therefore, the required network to be an MLP, must have two layers as minimum. This type of network has even more applications than a single Perceptron in different areas of knowledge (SILVA DANILO HERNANE SPATTI, 2010). The learning algorithm of MLP network is called Back-Propagation algorithm.

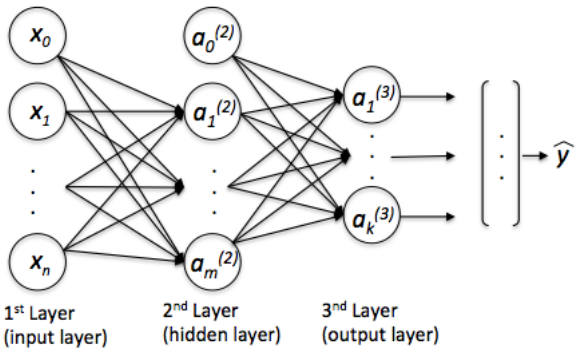


Figure 4 – MLP Architecture.

3.4.3 Back-propagation

In order to understand back-propagation algorithm, it is necessary to comprehend the successive phases that compose it, which are forward and backward propagation.

In the forward propagation, the input data is changed with multiplication operations, sum, activation function, layer to layer until production of respective outputs. Then, the outputs are compared with their real label (desired answers), the error is obtained and it depends on these error values whether the next phase (backward propagation) is applied or not.

In the backward propagation, a different mathematical calculation is made to readjust the weights, using partial derivatives, mean square error or cross-entropy. For example, the quadratic error can be applied in the output layer, while for intermediate layers, it changes their values (SILVA DANILO HERNANE SPATTI, 2010).

3.5 Deep Neural Networks

The usage and relevance of deep learning is increasing and each time better results are obtained in the recognition and classification of images. It learns hierarchical characteristics, separated at various levels of data representation; in other words, each layer learns a level of representation of the characteristics of the data (OLIVEIRA, 2014).

Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), the two main types of Deep Neural Networks (DNN) architectures, are extensively exploited to handle various Neuro-Linguistic Programming (NLP) tasks (YIN *et al.*, 2017). RNN and CNN will be explained in the next sections.

3.5.1 Recurrent Neural Networks

The main idea behind RNN is to use sequential information e.g., video processing, language processing. In this type of network, output and input are dependent, which implies that many tasks work efficiently i.e., if it is necessary to know the next prediction, sequence or action, the RNN network will provide the answer(GHAEMMAGHAMI, 2017). Thus, the main reason to explore this kind of architecture is its capacity to learn how to process spatio-temporal data in three basic ways (RIBEIRO; ALQUEZAR, 2002).

(a) *Sequence Recognition and Classification:* the network produces a particular output pattern once the whole input sequence is seen.

(b) *Sequence Reproduction and Prediction:* the network can generate the rest of a sequence when it sees part of it.

(c) *Temporal Association:* the network will produce an output sequence in response to a specific input sequence.

The most commonly used type of RNN are LSTM and GRU networks and they will be explained through the next sections.

3.5.1.1 Long Short-Term Memory Network (LSTM)

A Long Short-Term Memory (LSTM) (HOCHREITER; SCHMIDHUBER, 1997) is a special type of RNN network, capable of learning long-term dependencies. In some cases, it was necessary to have long term memory for the prediction.

LSTM was created in 1997 by Hochreiter and Schimdhuber, but its popularity has grown in recent years for different applications due to the good accuracy obtained.

LSTM is composed by memory cell block (Fig. 5) and three multiplicative gates, called the input gate, output gate or inference and forget gate. While the cells are responsible for maintaining information over long periods of time, the responsibility for deciding what information to store, and when to apply that information lies with the input and output gate units, respectively. Finally, the forget gate feeds the self–recurrent connection with its output activation and is responsible for not allowing the internal state values of the cells to grow without bound by resetting the internal states as long as it is needed. (RIBEIRO; ALQUEZAR, 2002).

The forget gate $f_t = \sigma\left(W_f.[h_{t-1}, x_t] + b_f\right)$ represent what information is going to throw away from the cell state. Then, the input gate $i_t = \sigma\left(W_i.[h_{t-1}, x_t] + b_i\right)$ and candidate values $\hat{C}_t = \tanh\left(W_C.[h_{t-1}, x_t] + b_C\right)$ decide what new information is going to store in the cell state. The old cell state is updated $C_t = f_t \odot C_{t-1} + i_t \odot \hat{C}_t$. Finally, output gate $o_t = \sigma\left(W_o.[h_{t-1}, x_t] + b_o\right)$ and cell state through *tanh* decide what is going to output $h_t = o_t \odot \tanh\left(C_t\right)$.

The parameters W denote weight matrices (e.g. W_i is the matrix of weights from the input gate to the input), the b terms denote bias vectors (e.g. b_i is the input gate bias vector), σ is the logistic sigmoid function, and *i, f, o* and *C* are respectively the input gate, forget gate, output gate and cell state, all of which are the same size as the cell output activation vector h_t, $\odot$ is the element-wise product of the vectors. (SAK; SENIOR; BEAUFAYS, 2014)

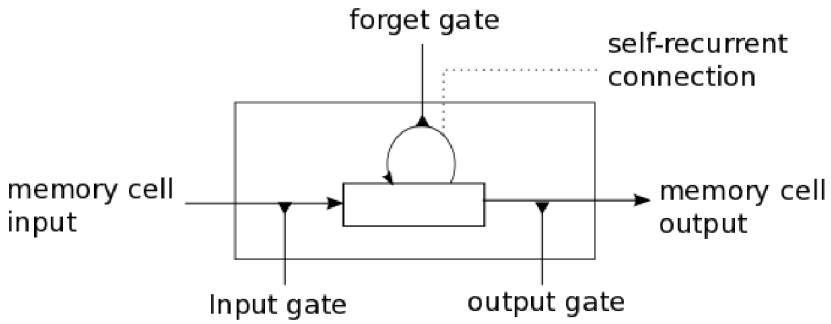


Figure 5 – LSTM memory cell, Kyunghyun (2017).

3.5.1.2 Gated Recurrent Unit Network (GRU)

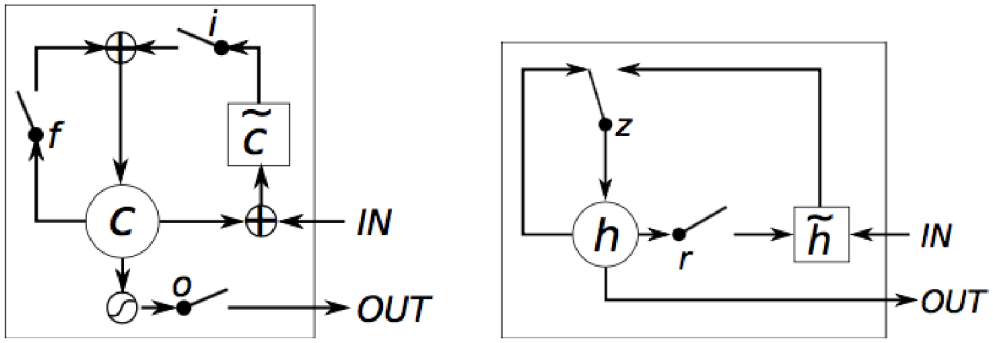
A Gated Recurrent Unit (GRU) was proposed by Cho et al(2014), in (CHUNG *et al.*, 2014). Similarly to LSTM unit, the GRU has gating units modulating the flow of information inside the unit. However, without having a separate memory cell. In the Fig. 6, it is shown (a) LSTM and (b) GRU. According to this:

(a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content, respectively.

(b) r and z are the reset and update gates, respectively. h and $\tilde{h}$ are the activation and the candidate activation, respectively.

3.5.2 Convolutional Neuronal Networks - CNN

CNN are used commonly for object recognition. They are a specialized type of vision network that learn from previous experiences. In Figure 7, it is shown a topology of a convolu-



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 6 – LSTM and GRU, Chung et al. (2014).

tional network, which has two layers of convolution, two pooling layers (Max-Pooling) and a completely connected layer.

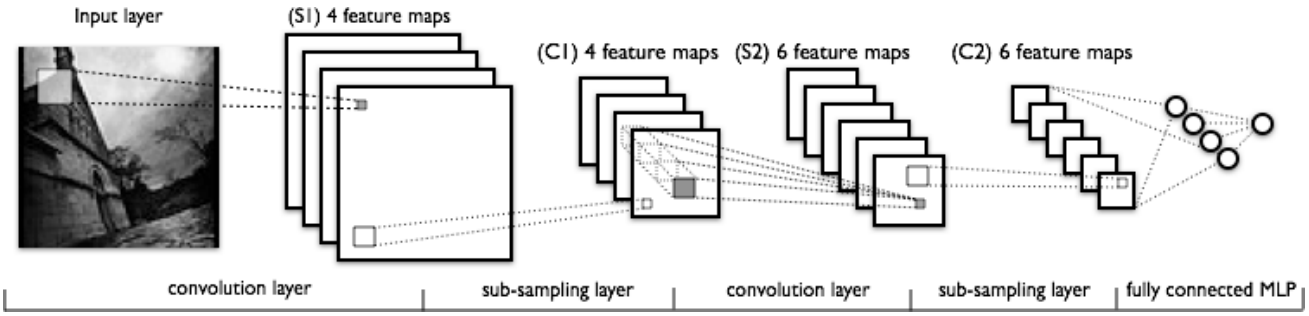


Figure 7 – LeNet convolutional network topology, LeCun et al. (1998).

CNN contain some main parameters called hyper-parameters. They are:

a) Filter Size: It is the kernel that can be 5x5, 7x7, etc. in the convolution layer.

b) Number of feature maps: It is the amount of K filters required in the convolution layer.

Convolution combined with pooling in neural networks are very important. For example, if it has a centralized face image and then the same face image slightly displaces, then the network recognizes as non-similar. Therefore, in order to avoid this kind of error, convolution and pooling are the right choice.

3.5.2.1 Convolution

It is based on how our neurons are actually structured in the visual cortex. The idea is that if a circle is moved from one position to another position, it remains as a circle. Consequently, it can train autoencoders with stains on the image, and then slide the encoder all over the image as

a digitizer, looking for features. This process makes the transform of the pixel matrix into another matrix of characteristics, which are produced by the decoder. The matrix of characteristics is much more than just single pixels.

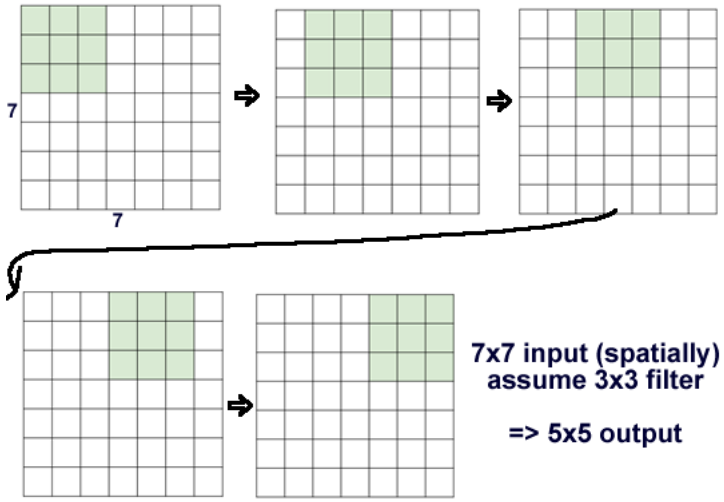


Figure 8 – Convolution: encoder, transformation matrix of pixels to array of characteristics, Li (2016).

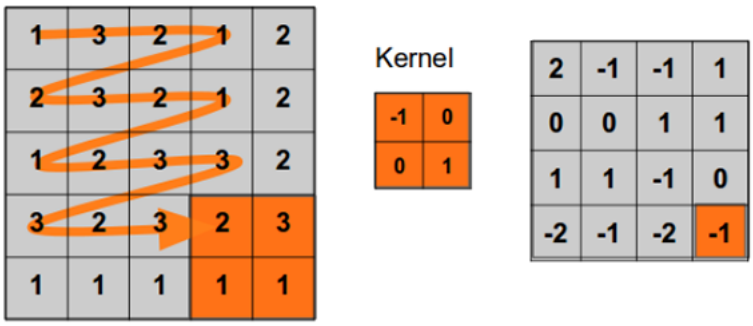


Figure 9 – Example convolution, with two-by-two kernel, Rocha (2015).

3.5.2.2 Pooling

Pooling reduces the dimensionality of the input by a constant factor. The scope of this layer is not only to reduce the computational burden, but also to perform feature selection. The input images are tiled in non overlapping sub-regions from which only one output value is extracted. Common choices are maximum or average, usually shortened as Max-Pooling and Avg-Pooling.

The main objective of pooling is to reduce the sensitivity of the network with regard to small changes in the image (ROCHA, 2015). Commonly in the convolution, the matrix cells are overlapped which does not occur in the pooling.

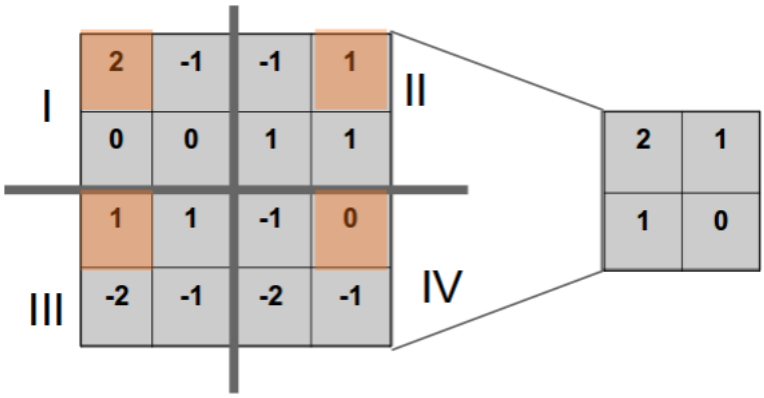


Figure 10 – Pooling, two-by-two cells, with a max-pooling function, Rocha (2015).

3.5.2.3 Neural Network Optimization

Optimization in general is an extremely difficult task. Traditionally, machine learning has avoided the difficulty of general optimization by carefully designing the objective function and constraints to ensure that the optimization problem is convex (GOODFELLOW; BENGIO; COURVILLE, 2016).

A global minimum, also known as an absolute minimum, is the smallest overall value of a set, function, etc., over its entire range. It is impossible to construct an algorithm that will find a global minimum for an arbitrary function. A local minimum, also called a relative minimum, is a minimum within some neighborhood that need not be (but may be) a global minimum. Most deep learning algorithms are based on an optimization algorithm called stochastic gradient descent.

Stochastic Gradient Descent(SGD): Its variants are probably the most used optimization algorithms for machine learning in general and for deep learning in particular. This can use a fixed learning rate. The learning rate may be chosen by trial and error, but it is usually better to choose this rate by monitoring learning curves that plot the objective function as a function of time. Theoretically, Batch gradient descent enjoys better convergence rates than stochastic gradient descent.

Momentum: The method of momentum (Polyak, 1964) is designed to accelerate learning, especially in the face of high curvature, small but consistent gradients, or noisy gradients. The momentum algorithm accumulates an exponentially decaying, moving average of past gradients and continues to move in their direction.

Parameter Initialization Strategies: All the weights in the model are frequently initialized to values drawn randomly from a Gaussian or uniform distribution. However, the scale of the initial distribution does have a large effect on both the outcome of the optimization procedure and the ability of the network to generalize. Saxe et al. (2013) recommended initializing to random orthogonal matrices, with a carefully chosen scaling or gain factor g that accounts for

the nonlinearity applied at each layer. But unfortunately, these optimal criteria for initial weights often do not lead to optimal performance.

Algorithms with Adaptive Learning Rates: Neural network researchers have long realized that the learning rate is reliably one of the most difficult to set hyperparameters because it significantly affects the model. The momentum algorithm can mitigate these issues somewhat, but it does so at the expense of introducing another hyperparameter.

- **AdaGrad:** AdaGrad Algorithm individually adapts the learning rates of all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of the gradient. The parameters with the largest partial derivative of the loss function have a correspondingly rapid decrease in their learning rate, whereas parameters with small partial derivatives have a relatively small decrease in their learning rate (see algorithm 1).

- **RMSProp:** Modifies AdaGrad to perform better in the nonconvex setting by changing the gradient accumulation into an exponentially weighted moving average. This is designed to converge rapidly when applied to a convex function. When applied to a nonconvex function to train a neural network, the learning trajectory may pass through many different structures and eventually arrive at a region that is a locally convex bowl (see algorithm 2).

- **Adam:** In the context of the earlier algorithms, it is perhaps better seen as a variant on the combination of RMSProp and momentum with a few important distinctions. Adam includes bias corrections to the estimates of both the first-order moments and the second-order moments to account for their initialization at the origin (GOODFELLOW; BENGIO; COURVILLE, 2016) (see algorithm 3).

Algorithm 1 – The AdaGrad algorithm

Require: Global learning rate ε
Require: Initial parameter θ
Require: Small constant δ, perhaps 10^{-7}, for numerical stability.
Initialize gradient accumulation variable $r = 0$

while *stopping criterion not met* **do**
  Sample a minibatch ofmexamples from the training set $\left\{x^{(1)},...,x^{(m)}\right\}$ with corresponding targets $y^{(i)}$.
  Compute gradient: $g \leftarrow \frac{1}{n} \bigtriangledown_\theta \sum_i L\left(f\left(x^{(i)};\theta\right),y^{(i)}\right)$.
  Accumulate squared gradient: $r \leftarrow r + g \odot g$
  Compute update: $\Delta\theta \leftarrow -\frac{\varepsilon}{\delta+\sqrt{r}} \odot g$. (Division and square root applied element-wise)
  Apply update: $\theta \leftarrow \theta + \Delta\theta$
end
Source: Goodfellow, Bengio and Courville (2016)

Algorithm 2 – The RMSProp algorithm

Require: Global learning rate ε, decay rate ρ
Require: Initial parameter θ
Require: Small constant δ, usually 10^{-6}, used to stabilize division by small numbers.
Initialize accumulation variables $r = 0$
while *stopping criterion not met* **do**
  Sample a minibatch ofmexamples from the training set $\left\{x^{(1)},...,x^{(m)}\right\}$ with corresponding targets $y^{(i)}$.
  Compute gradient: $g \leftarrow \frac{1}{n} \bigtriangledown_\theta \sum_i L\left(f\left(x^{(i)};\theta\right),y^{(i)}\right)$.
  Accumulate squared gradient: $r \leftarrow \rho r + (1-\rho)\, g \odot g$.
  Compute parameter update: $\Delta\theta \leftarrow -\frac{\varepsilon}{\sqrt{\delta+r}} \odot g$. ($\frac{1}{\sqrt{\delta+r}}$ applied element-wise)
  Apply update: $\theta \leftarrow \theta + \Delta\theta$
end
Source: Goodfellow, Bengio and Courville (2016)

Algorithm 3 – The Adam algorithm

Require: Step size ε (Suggested default: 0.001)
Require: Exponential decay rates for moment estimates, ρ_1 and ρ_2 in [0,1). (Suggested defaults: 0.9 and 0.999 respectively)
Require: Small constant δ used for numerical stabilization (Suggested default:10^{-8}).
Require: Initial parameter θ
Initialize 1st and 2nd moment variables $s = 0, r = 0$
Initialize time step $t = 0$
while *stopping criterion not met* **do**
  Sample a minibatch of m examples from the training set $\left\{x^{(1)},...,x^{(m)}\right\}$ with corresponding targets $y^{(i)}$.
  Compute gradient: $g \leftarrow \frac{1}{n} \bigtriangledown_\theta \sum_i L\left(f\left(x^{(i)};\theta\right),y^{(i)}\right)$.
  $t \leftarrow t+1$
  Update biased first moment estimate: $s \leftarrow \rho_1 s + (1-\rho_1)\, g$.
  Update biased second moment estimate: $r \leftarrow \rho_2 r + (1-\rho_2)\, g \odot g$.
  Correct bias in first moment: $\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$
  Correct bias in second moment: $\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$
  Compute update:: $\Delta\theta \leftarrow -\varepsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$. (operations applied element-wise)
  Apply update: $\theta \leftarrow \theta + \Delta\theta$
end
Source: Goodfellow, Bengio and Courville (2016)

Currently, the most popular optimization algorithms actively in use include SGD, SGD with momentum, RMSProp, RMSProp with momentum, AdaDelta and Adam. The choice of which algorithm to use, at this point, seems to depend largely on the user's familiarity with the algorithm.

Approximate Second-Order Methods: Optimization algorithms that use only the gradient, such as gradient descent ($\theta^* = \theta_0 - \varepsilon \nabla_\theta J(\theta_0)$), are called first-order optimization algorithms. Optimization algorithms that also use the Hessian matrix (this is symmetric at such points and containing second derivative) such as Newton's method ($\theta^* = \theta_0 - H^{-1} \nabla_\theta J(\theta_0)$), are called second-order optimization algorithms (Nocedal and Wright, 2006).

The difference in these methods is that the Hessian matrix is denoted by H and the learning rate is denoted by ε, then both have $\nabla_\theta J$ as the gradient, θ^* as the new value and θ_0 as the last value.

Cost Function Algorithms: The main algorithms of cost function are

- Mean square error:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \tag{3.2}$$

  where m is the number of predictions, the input is x, h_θ is the vector of the last neuronal network output, and y is the vector of the corresponding desired output.

- Cross-entropy:

$$C = -\frac{1}{n} \sum_x [y \ln a + (1-y) \ln(1-a)))] \tag{3.3}$$

  where n is the total number of items of training data, the sum is over all training inputs, x, and y is the corresponding desired output.

When the output is a value of a finite set, the problem is called classification and when the output is a real number the problem is called regression. A recommendation indicated by researchers of the area is that mean square error is used for regression and cross-entropy for classification.

3.5.2.4 SoftMax function

It is used in the last layer of the CNN architecture. To be able to recognize and make sure of an answer, the SoftMax function generates a possible answer in percentages. Therefore, one can interpret the output because one property is that the sum of the outputs is 1 and all outputs are positive (ROCHA, 2015). Thus, the one having the highest percentage is the answer provided by this function.

3.5.2.5 Dropout

Dropout is a new regularization technique that has been more recently employed in deep learning. Pioneering work by Hinton et al.,(2012), dropout was only applied to fully connected layers. For that reason, they found that the convolutional shared-filter architecture had a drastic reduction in the number of parameters and thus **reduced its possibility to overfit** in convolutional layers.

3.5.2.6 Inception Network

The Inception-v3 from Google (SZEGEDY *et al.*, 2014) was inspired by the article: "Network In Network" (LIN; CHEN; YAN, 2013).

This inception-v3 architecture was implemented in the winning ILSVRC 2014 submission GoogLeNet (SZEGEDY *et al.*, 2015). The main contribution with respect to Network in Network is the application to the deeper nets needed for image classification. From a theoretical point of view, Google's researchers observed that some sparsity would be beneficial to the network's performance, and implemented it using today's computing techniques.

A simple example of how inception works would be a 5x5 kernel generating a 1x1 kernel. The idea of inception is to create intermediate layers in that transition from 5x5 to 1x1 as shown in Fig. 11. That is, in the middle between 5x5 and 1x1 kernels, another convolution of 3x3 is created. Thus, it generates a one more layer of 3x3.

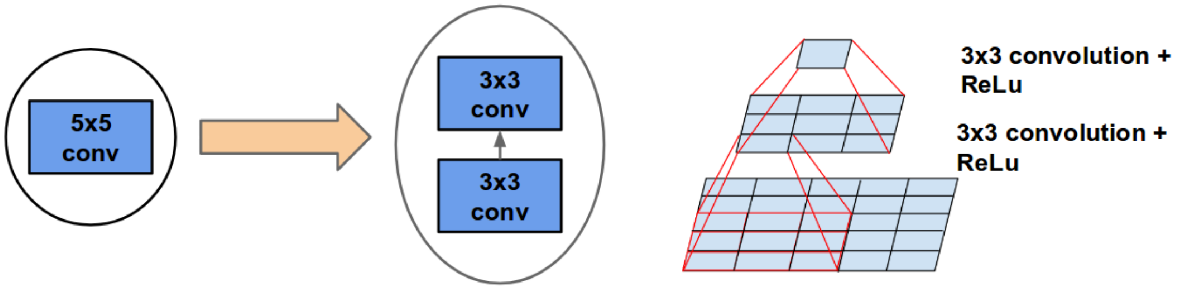


Figure 11 – Mini-network replacing the 5x5 convolutions, Szegedy et al. (2015).

3.5.2.7 You Only Look Once (YOLO)

YOLO is one of the best real-time object detection systems. YOLO predicts bounding boxes and class probabilities directly from full images in one evaluation. For this, it divides the input image into an S x S grid, each grid cell predicts only one object. YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolution layers use 1 x 1 reduction layers alternatively to reduce the depth of the features maps. For the last convolution layer, it outputs a tensor with shape (7, 7, 1024). The tensor is then flattened. Using 2 fully connected layers as a form of linear regression, it outputs 7x7x24 parameters and then reshapes to (7, 7, 24). YOLO architecture is shown in Fig. 12

The system divides the input image into a 7 x 7 grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts a bounding box and class probabilities associated with that bounding box. Each grid cell predicts 20 conditional class probabilities, and 4 bounding box coordinates, that are shown in Fig. 13

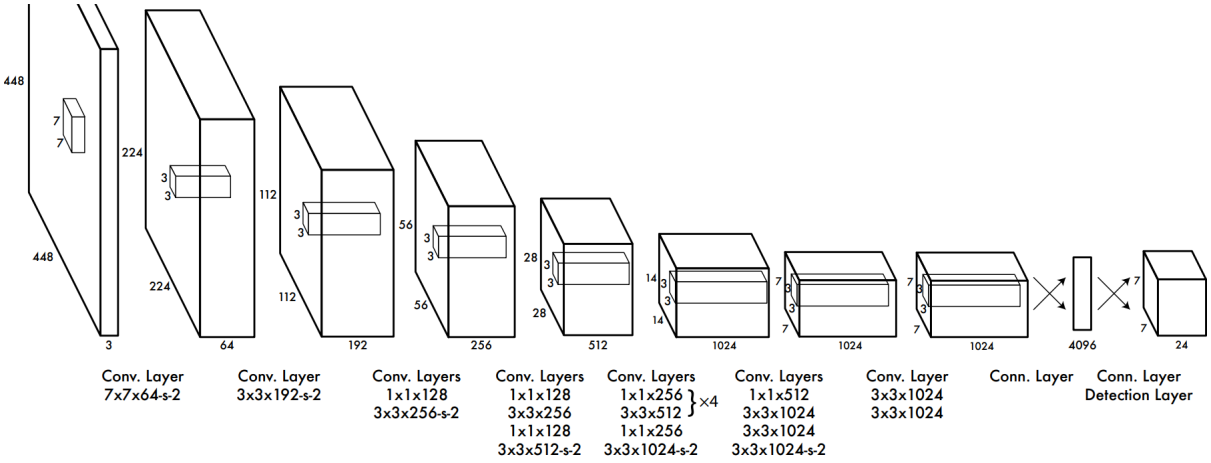


Figure 12 – YOLO Architecture, Redmon et al. (2015).

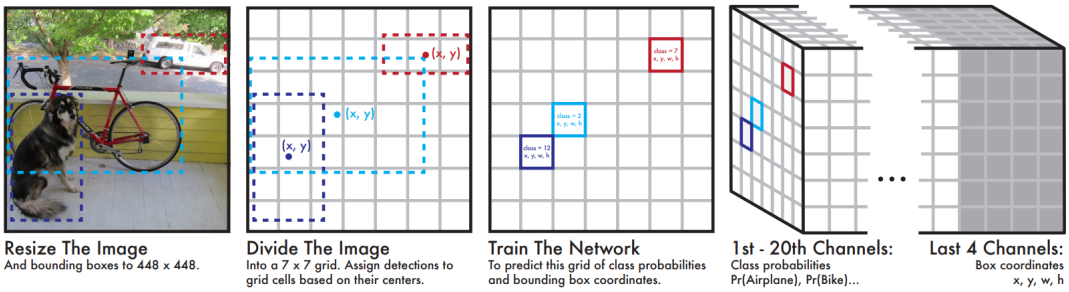


Figure 13 – The YOLO Model, Redmon et al. (2015).

3.5.2.8 Recurrent YOLO (ROLO)

Deep Trackers term represent a new paradigm in tracker today. In ROLO were used LSTM's interpretation and regression capabilities of high-level visual features, the proposed network is both accurate and efficient with low complexity. The major innovation of this tracker is LSTM network, its memory cell which essentially acts as an accumulator of the state information. The cell is accessed, written and cleared by several self-parameterized controlling gates. Thus, LSTM network uses memory cells to store and output information, allowing it to discover better long-range temporal relations.

In Fig. 14, YOLO has a role of collecting rich and robust visual features, as well as preliminary location inferences; and LSTM network is used in the next stage as it is spatially deep and appropriate for sequence processing.

The architecture is shown in Fig. 1. ROLO framework receives the outputs of the last fully connected layer from YOLO. This fully connected layer has 4096 neurons. Its output signals are sent into the recurrent network LSTM and for a layer that will extract the position coordinates, represented by the vector (0, x, y, w, h, 0) of each object that is being tracked. The parameters x, y, w, h are the object position. These parameters of the vector are added in the

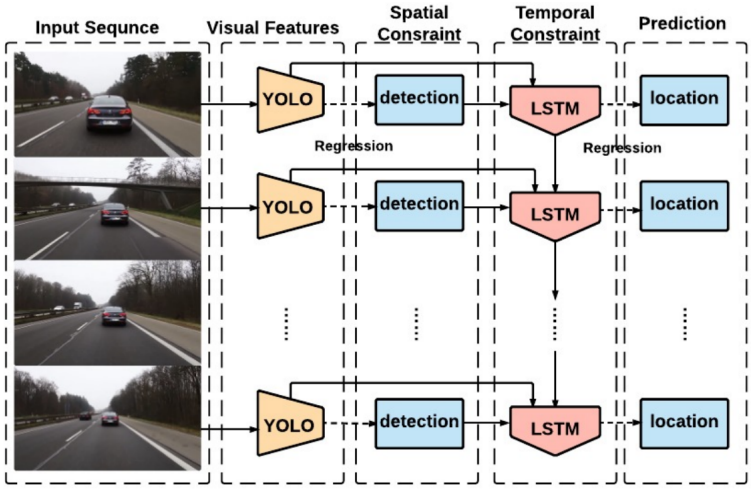


Figure 14 – Simplified overview ROLO, Ning et al. (2016).

input of the recurrent network, and therefore, it receives 4102 inputs. The ROLO does regression for direct prediction of the tracking locations.

3.5.2.9 Transfer Learning

Transfer learning is the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned. Today, transfer learning methods appear in several top venues, most notably in data mining and machine learning.

In these days, the transfer learning is quite used either to reduce or increase the number of objects to classify. In many cases, this type of technique prevents the weights of the layers of a neural network to start from zero and needs few epochs to be trained.

There are cases in which it is necessary not to change the number of classes, rather to maintain it and only specialize on it for some specific objective.

3.6 Evaluation techniques

3.6.1 Cross-Validation

Cross-Validation is a statistical method of evaluating and comparing learning algorithms by dividing data into two segments: one used to learn or train a model and the other used to validate the model. In typical cross-validation, the training and validation sets must cross-over in successive rounds such as each data point has a chance of being validated against. The basic form of cross-validation is k-fold cross-validation.

3.6.2 Confusion Matrix

A confusion matrix (Kohavi and Provost, 1998) contains information about actual and predicted classifications done by a classification system. Performance of such systems is commonly evaluated using the data in the matrix. The following table shows the confusion matrix for a two-class classifier.

3.6.3 Average Precision

The Average Precision (AP) is often used as an indicator for evaluating ranked output of documents in standard retrieval experiments.

$$p_m = \frac{1}{m} \sum_{k=1}^{m} x_k \tag{3.4}$$

3.6.4 Intersection over Union

The Intersection over Union (IoU) is usually used to measure the performance of any object category segmentation method. This method allows us to measure the region of the object that is being tracked. Given an image, the IoU measure gives the similarity between the predicted region and the ground-truth region for an object presented in the image. It is defined as the size of the intersection divided by the union of the two regions:

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{3.5}$$

3.7 Final considerations

In this chapter, the main techniques and algorithms used in this work were presented. This includes visual attention techniques, based on DL algorithms for the classification, recurrent networks for the tracking and finally, the evaluation techniques. In the next chapter, the methodology of the proposed system for recognizing and tracking objects from images collected by a drone is introduced.

CHAPTER

4

THE PROPOSED SYSTEM

In this Chapter, the system proposed for recognition and tracking of vehicles in a highway is described in details. It is composed by three main steps: **Saliency detection, Recognition and Tracking** which will be described below.

4.1 General Outline

The scheme of functionality of the proposed system is shown in Fig. 15. In Video-KeyFrame Module, the system receives a video, and it processes one image for each 15 frame.

Immediately after the image is selected, it is sent to Saliency Module, then subsequently it is processed by VOCUS2 algorithm. The highlighted objects are sent to Vehicle recognition Module. More specifically, the goal here is to find just a single object from any frame captured from the UAV, aiming to guarantee that the object can be identified with precision. In this case, it is necessary to separate a small vehicle from all other objects. Once that single vehicle is recognized, this information is sent to Vehicle tracking Module. The goal is to track just one vehicle, since the proposed system is used for the persecution of suspicious vehicles with a drone (UAV).

4.2 Saliency

This step consists in highlighting that part of the image in which the vehicle is located. The main point is to reduce the searching of the object in the image to be analyzed, avoiding to look for it in the whole image. For this purpose, we will use an algorithm to detect the most salient object in the image.

An algorithm of saliency called VOCUS2 (FRINTROP; WERNER; GARCIA, 2015a) has been used to highlight the objects in the images. It is an improvement of VOCUS algorithm. This

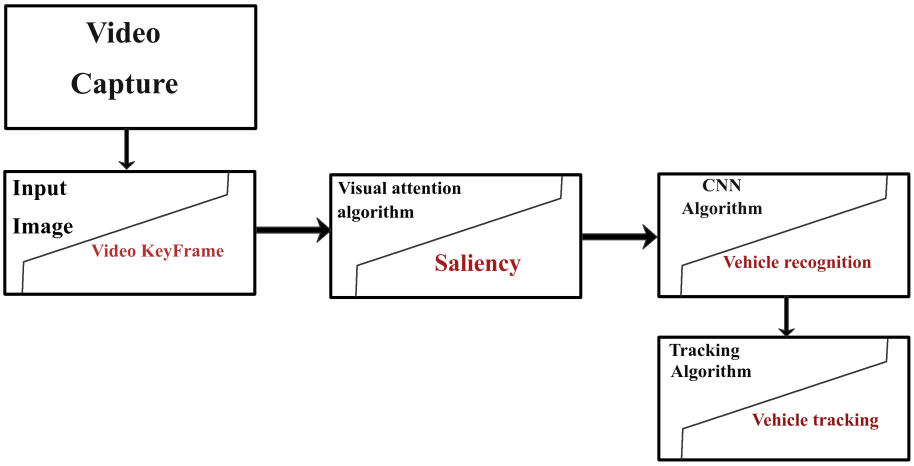


Figure 15 – Proposed System General Scheme.

algorithm calculates the feature channels of the image in parallel and the center-neighbor contrast is computed by Difference-of-Gaussians. It is based on the concepts from human perception, which is useful to obtain an object of greater saliency in an image. The salient segment is the input for the next stage which will be described in the next section. Some examples are shown in Fig. 16.

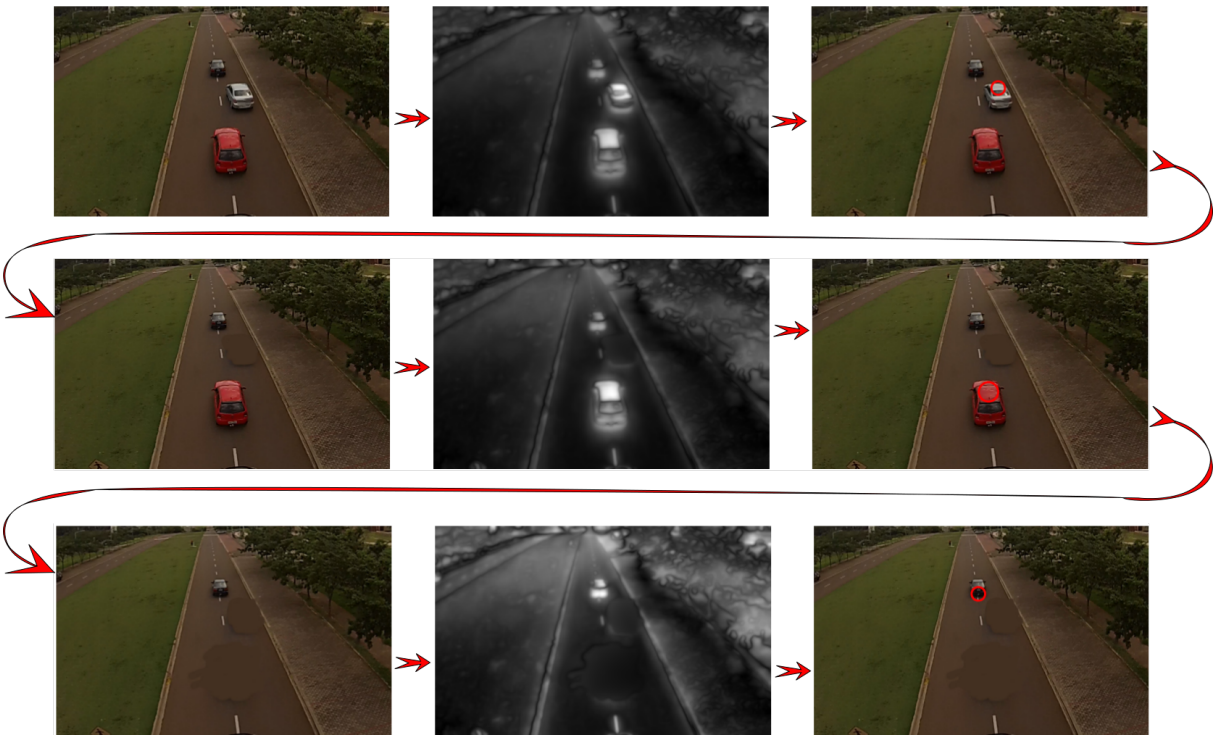


Figure 16 – Test VOCUS2 algorithm.

The system finds in each image, corresponding to 01 frame, seven more salient objects. The system generates images as those shown in Fig. 17. The most salient object gets cropped. The process to find the next more salient image is to color with a neutral color the previous salient segment. As a result, algorithm VOCUS2 can find all the seven objects. The reason why

we selected this number (7) is a result of try and error.

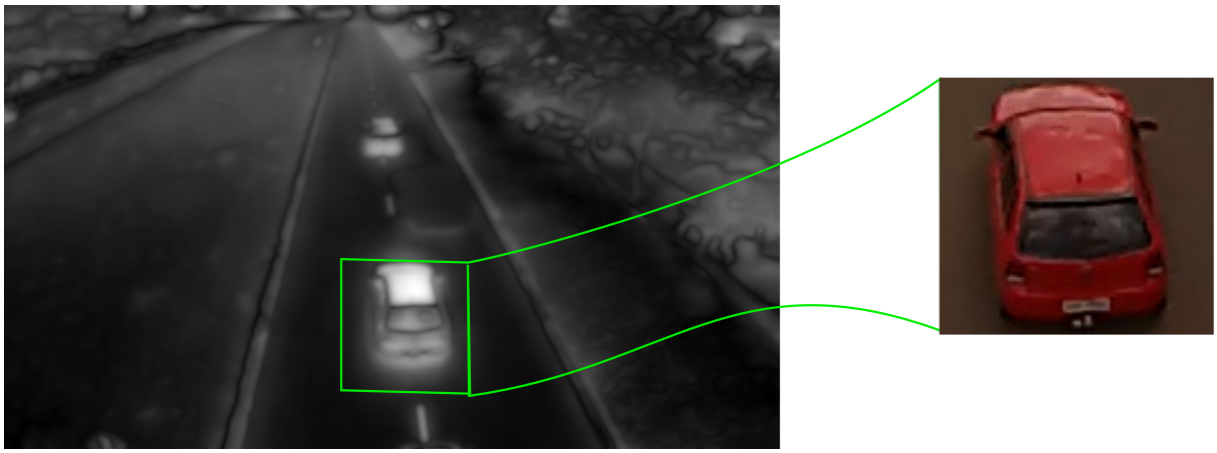


Figure 17 – Saliency object by VOCUS2.

4.3 Recognition

In this subsection, it is described how the salient object will be recognized. For this purpose, it is necessary to explain how a CNN (Le Cun *et al.*, 1998) can recognize or classify the salient object. A CNN is biologically inspired by the visual cortex of animals. After the input of an image, it processes the image in the convolutional layers and then it classifies the image in the fully connected layers, as it is shown in Fig. 18. A typical CNN processes the image and produces an output. In our case, this output is binary, representing the object: car or not car. The components of CNN are inputs, weights, activation functions and outputs. The main components in CNN are the weights, because they are updated during the training process.

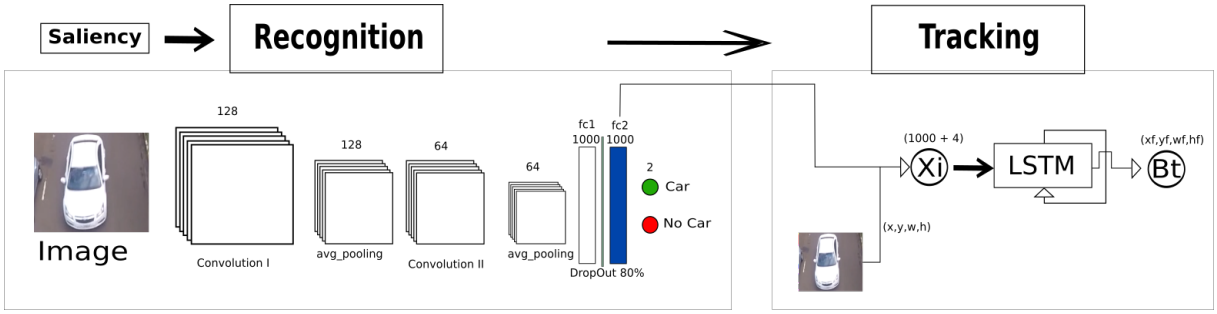


Figure 18 – Architecture proposed for classification and tracking.

In the first tentative to decide which architecture of CNN to use, Arch.1 was proposed. This architecture can be seen in Fig. 19. It contains two convolutional layers of dimension 96 x 96 (with 32 feature maps), and 48 x 48 (with 32 feature maps), respectively. Each convolutional layer is followed by an average_pooling layer. Furthermore, there are two fully connected layers containing 100 neurons each one. This architecture was inspired by (HUTTUNEN;

YANCHESHMEH; CHEN, 2016) with some modifications such as feature maps dimension and input dimension.

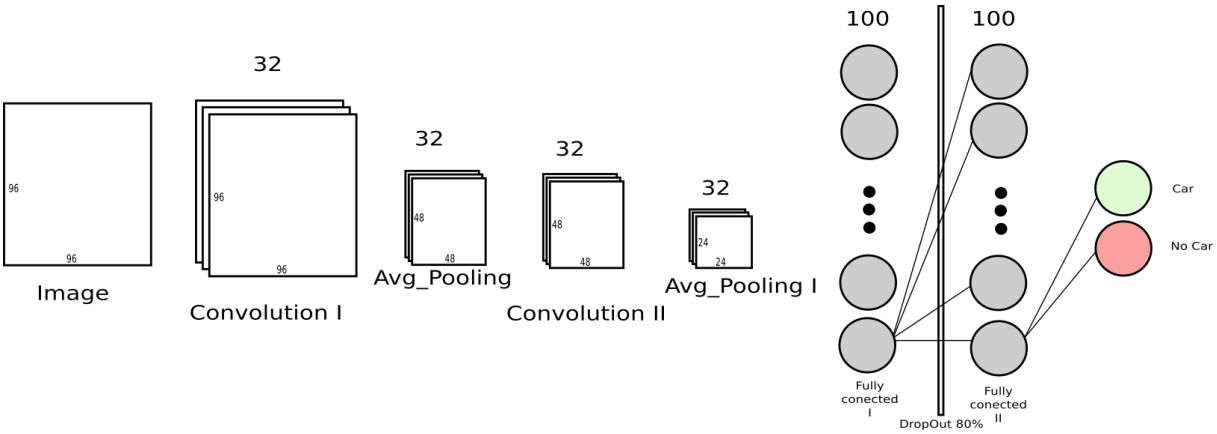


Figure 19 – CNN Arch.1

To obtain Arch.2, the second tentative proposed to CNN network, several topologies were investigated considering more convolutional layers, different number of neurons in each layers, different learning rates, variations of dropout and different optimization algorithms of the learning rates. It is constituted by two convolutional layers of dimension 96 x 96 (with 128 feature maps) and 48 x 48 (with 64 feature maps), respectively. Each convolutional layer is followed by an average_pooling layer. Furthermore, there are two fully connected layers containing 1000 neurons each one. The architecture Arch.2, is presented in Fig. 20.

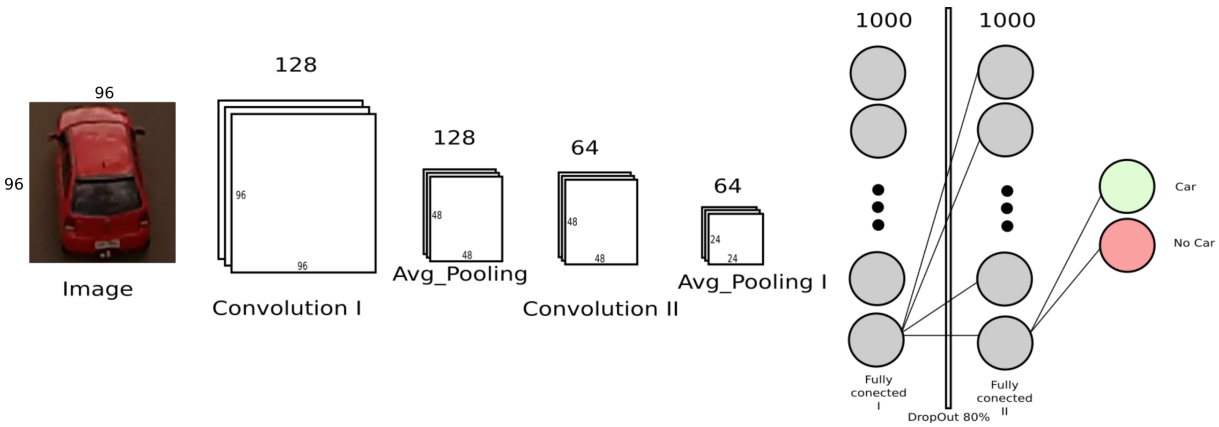


Figure 20 – CNN Arch.2

Once the detection of the object is completed by using the CNN proposed, the next stage is to track the object and this process will be described in Section 4.4. To sum up, in this section we describe how the system classifies the salient images in car or no-car, as it is shown in Fig. 19 and Fig. 20.

4.4 Tracking

To track a vehicle, it is necessary a method for memorizing the final positions of the vehicle for a short period and then predict a future position of the vehicle. In the literature, it can be found several techniques such as the recurrent neural network LSTM (Long Sort Term Memory), presented in section 3.

The new architecture is being proposed in this work for recognition and tracking. It is based on ROLO framework, which is constituted by YOLO, as presented in section 3.

It is worth to emphasize that ROLO framework has been adapted by us. For this, instead of YOLO, two new networks are being investigated for object recognition: Arch.1 and Arch.2, presented in the previous section. They are simpler than YOLO.

Therefore, the last fully-connected layer from the simpler CNN architecture (Arch.2) is inserted in the input from LSTM network, as it is shown in Fig. 18. In addition to the components of each data, the coordinate (x,y,w,h) of the image is obtained by another algorithm called Correlation Filter Tracker (CFT), described in section 3.

All these algorithms: VOCUS2, CNN and LSTM are useful for detecting, recognizing and tracking a vehicle respectively.

4.5 Final considerations

In this Chapter, the main techniques were presented, which were used to construct the proposed system for recognizing and tracking of vehicles. They are VOCUS2 for detecting the object present in the image, CNN for recognizing the object, if it is car or not, and finally, LSTM for tracking the vehicle during a certain time. The proposed architecture, obtained results and discussion will be presented in the next chapter.

CHAPTER

5

RESULTS AND DISCUSSION

In this Chapter, it will be described how the data has been collected and how the training for vehicle recognition and tracking has been performed. For the classification task, two architectures of CNN proposed are tested and compared to Inception-v2 network, in terms of accuracy and precision. For vehicle tracking, two videos have been taken, to compare LSTM and CFT, in terms of IOU measure. Finally, to further assess the viability of the trained model, more videos were considered to compare the tracking performance between LSTM and GRU.

5.1 Implementation Framework

The software was trained and developed in Geforce GTX 1060 3GB, intel(R) Core(TM) i7-6700 CPU-3.40GHz, architecture x86_64, linux version 4.15.0-43-generic, ubuntu 16.04.1, TensorFlow version 1.2.1, OpenCv version 3.2.0, Python version 3.5.2.

The framework TensorFlow has been used for implementing the networks CNN and LSTM. The CNN networks were trained and tested with images of 3 videos collected by a drone of our Laboratory. The OpenCv has been used for running VOCUS2 algorithm and OpenCv with c++ code called from Python.

5.2 Data Collection

For collecting videos, a drone existing in our Laboratory took images from inside of our university. The images collected were processed using the saliency algorithm VOCUS2. Seven objects were highlighted in each image frame. Each object of the image was manually classified between car and no-car. This process was repeated for each frame of a video. Afterwards, the seven most salient objects in all frames from videos were saved as an image.

The number of collected images is 21122, including figures of car and no-car. From these

images, 75% have been separated to train and 25% to test. Cross validation has been performed considering 4 folds.

In Fig. 21, examples of images present in the data collected are shown, where it can be seen some figures of car and no-car. This data was used to train the two architectures mentioned in section 4.

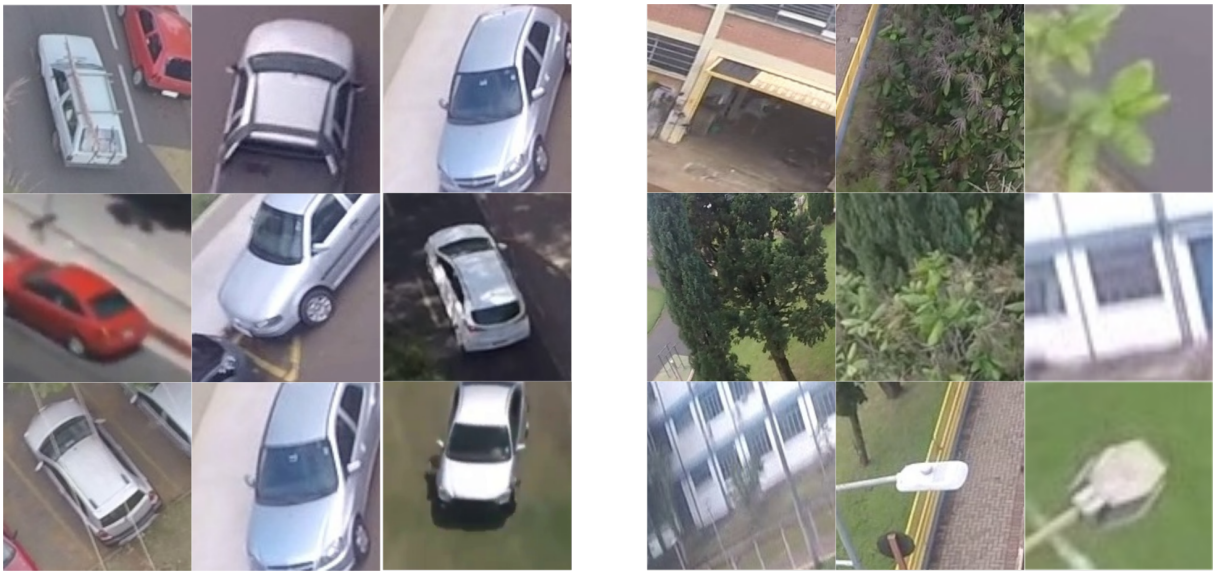


Figure 21 – Example of data collected and classified manually in car and no-car.

5.3 Training for Vehicle Recognition

The data was trained with different architectures of networks. The characteristics of two different architectures mentioned, Arch.1 and Arch.2, for recognizing the vehicle are shown in Table 2.

We had tested these networks with different number of epochs, other optimizer algorithms with different Dropout rate. Finally, the networks were trained with 5000 epochs each one, using the Adagrad Optimizer as the optimizer algorithm and with a dropout rate of 20%.

	Arch. 1	**Arch. 2**
Number of training epoch.	5000	5000
Number of Convolutional layers	2	2
Number of layers fully connected	**2(100, 100)**	**2(1000, 1000)**
Input image size	96x96	96x96
Number of feature maps	**{32, 32}**	**{128, 64}**
Kernel size of the convolutional layers	5x5	5x5
Pooling kernel size	2x2 (avg)	2x2 (avg)
Cost function	Mean Squared Sum	Mean Squared Sum
Optimizer	AdagradOptimizer	AdagradOptimizer
Learning Rate	0.001643	0.001643
Dropouts	20%	20%

Table 2 – Characteristics of the architectures

Another architecture, used for comparison of performance with Arch.1 and Arch.2, was the Inception from google (SZEGEDY *et al.*, 2014), which has 22 layer deep CNN, but reduced the number of parameters from 60 million (AlexNet) to 4 million of parameters. This architecture was trained with 1000 epochs of transfer learning training, with the collected data.

For the validation of the proposed models, the cross-validation technique have been used. From a total of 21122 images collected, classified manually between whether is a car or not, the data was fragmented in 4 folds, considering approximately 25% of data for testing and 75% for training, of both groups (car and no car). The training was performed with 5000 epochs. The results obtained by using Arch.2, for each fold, are shown in the Table 3.

It can be seen that an average accuracy of 90.55% was obtained.

Fold	Error %
I	14.75
II	12.08
III	9.27
IV	1.73
Avg. Error	**9.45**

Table 3 – Table of errors in each Cross-Validation Folds

This shows a good performance of the proposed approach. We trained also our data considering the Arch.1. However, only Fold was considered for training, because it was the fold that presented the higher error rate (14.75%) in Arch.2. When using Arch.1, the results presented an error of 16.11%, greater than the one obtained in Arch.2. Because of this result, Arch.1 was not tested with other folders.

After that, Fold I was trained again, with another more complex type of architecture, which is called Inception-v2. It has been trained with 1000 epochs of training, giving an approximate error of 0.12%.

To sum up, the three neural networks were trained with the Fold I. In Table 4, it is shown the mean values of accuracy and precision, tested with different videos collected by the drone of LAR laboratory, considering the architectures: Arch.1, Arch.2 and Inception-v2. It can be seen a difference of approximately 17% of accuracy between the proposed architectures (1 and 2) and Inception-v2, which implies that Inception-v2 was better than our architectures (1 and 2), in terms of accuracy.

	Arch. 1	Arch. 2	Inception-v2
Avg. accuracy	0.4158	0.4192	0.5775
Avg. precision	0.9313	0.9572	0.8546

Table 4 – Average of accuracy and precision of architectures 1, 2 and Inception-v2

It is worth noting that Inception-v2 has already trained weights and it has used the transfer learning technique to train Fold I. Furthermore, Inception-v2 takes more time to classify. On the other hand, it is necessary to highlight the good precision obtained by our architectures (1 and 2). They presented almost 10% better precision than Inception-v2.

These results demonstrate that the true positives are good in our architectures (1 and 2). They also indicate that it is necessary to increase the data to improve the accuracy for any video. Comparing the architectures 1 and 2, they are not different statistically, according Test-T. However, as the precision of architecture 2 was better than architecture 1, according Table 4, it will be considered for the task of tracking of vehicles presented in the next section.

5.4 Tracking of Vehicles

In Table 5, it is shown the number of epochs used to train the LSTM recurrent network and the average of Intersection-over-Union (IoU). It may also be noticed that the increment of the number of epochs on a video has a good IoU. For video#1, it is observed that are necessary 1000 epochs for getting a IoU value approximately equals to 70%, whereas for video#2, only 200 epochs are needed.

No. Epochs	IoU
Video#1	
200	0.37077
1000	0.70266
Video#2	
200	0.75823
500	0.79602

Table 5 – Training for Vehicle Tracking

In Table 6, it is shown a comparison between CFT and LSTM tracker proposed by us. It can be seen that CFT in video#2 obtained a better IoU value but in the video#1, it obtained the worst. On the other hand, considering an average of IoU values obtained in these two videos, the proposed system was better than CFT for the tracking of vehicles.

	CFT	Our Tracker
IoU of Video#1	0.3154	0.7027
IoU of Video#2	0.9163	0.7960
Avg. IoU	**0.6159**	**0.7494**

Table 6 – Comparison between CFT and LSTM tracker

In Fig. 22, 23 and 24, some examples of vehicle tracking from the perspective of a drone are shown, using the Arch.2 trained, as described in the previous chapter.

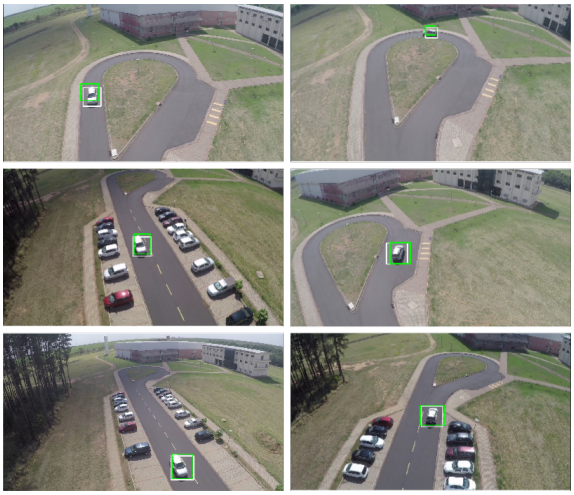


Figure 22 – Test Vehicle Tracking: Box green see our LSTM track and the white box is the real box (Video#1).

5.5 Discussion of Results

Inception-v2 from Google outperformed our system in accuracy. However, a test without labels was done. Then, making the confusion matrix (Table 7), we realized that the two proposed architectures were better in precision than Inception-v2. Another point in favor of our architectures is the classification time of an image. The mean response time in recognizing an image using Inception-v2 is 4.53 seconds and considering Arch.2 is 0.225 seconds.

Incep.-v2	Car	Nocar
NoCar-pred.	180.0	28.6
NoCar-pred.	140.3	51.0

Arch. 2	Car	Nocar
NoCar-pred.	88.6	1.6
NoCar-pred.	230.6	79.0

Arch. 1	Car	Nocar
NoCar-pred.	89.3	2.3
NoCar-pred.	231.3	77.0

Table 7 – Confusion matrix average.

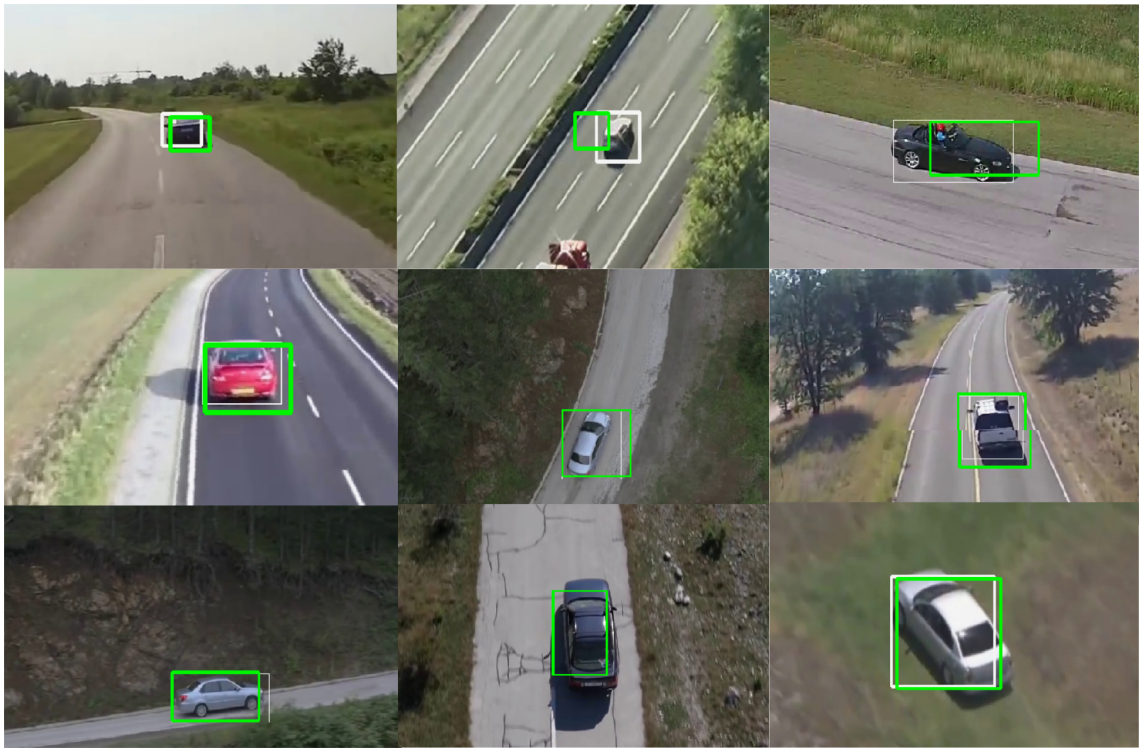


Figure 23 – Vehicle Tracking: Box green show the proposed LSTM track and the white box is the real box.

In Table 8, it is shown the IoU measure of 11 videos. Some of these videos were obtained from the website (LUQUE, 2017), and some of those are shown in Figure 24.

The trainings were done by epochs, starting with 500 epochs until 2500, for both LSTM and GRU. The goal was to compare how different or equal these two are. It can been seen in Table 8, that it does not make much difference; in fact, by an average of 2%, the LSTM network wins GRU in the first 2 cases (500 and 1200 epochs), and in the third case (2500 epochs), by 4% approximately.

Epochs	500	1200	2500
IoU LSTM	0.4833	0.5850	0.6840
IoU GRU	0.4617	0.5602	0.6285

Table 8 – To train LSTM and GRU tracker by incremental epochs

All of these tracking results were tested incrementally, first with 2 videos, and to further assess the viability of the trained model, considering 11 more videos. As it was mentioned, the algorithms proposed with LSTM and GRU had a different behavior when the vehicle is located in the curve of the highway and when the camera performs abrupt movements. In Figure 24, in the cyan contour, it can be seen the abrupt movement performed by the camera.

In this case, the CFT technique loses the vehicle. On the other hand, in the box marked by yellow color, it can be seen a vehicle that passes at high speed, then the vehicle passes so fast

that the CFT also loses the vehicle. Finally, it can be noted, in the red box, that the vehicle goes on the curve, where the CFT also stops tracking the vehicle. These problems did not occur with the LSTM tracker proposed in this work.

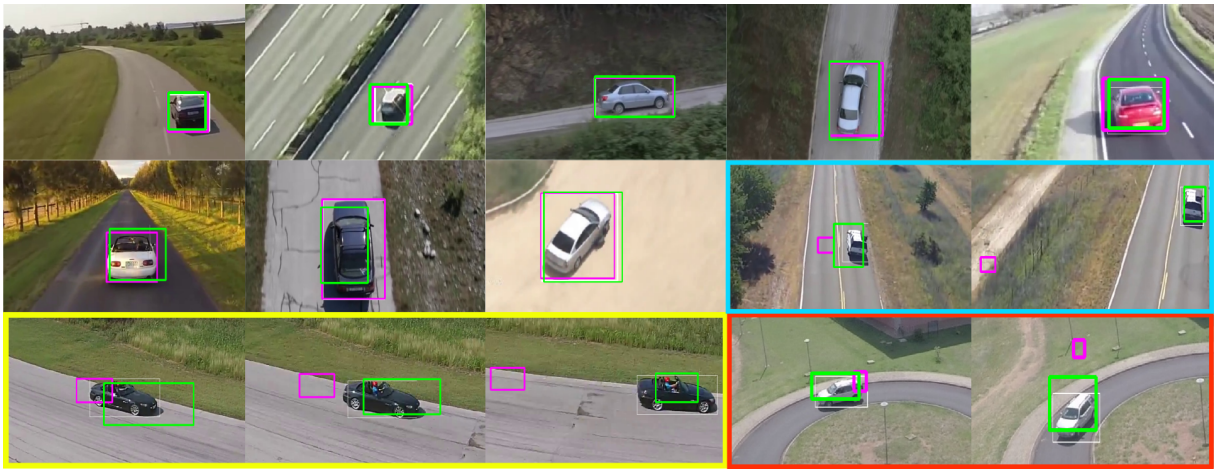


Figure 24 – Vehicle Tracking: The green box is proposed LSTM-tracker, the pink box is Correlation Filter Tracker and the white box is the real box.

5.6 Final considerations

In this chapter, the proposed system was tested and the results obtained were evaluated in terms of the accuracy and precision metrics for recognition, using cross validation with 4 folds. In order to evaluate the tracking in the videos, the IOU measure was applied. The comparison results indicated that our proposal was better than CFT for the tracking of vehicles. In the next chapter, the conclusions obtained about the proposed system and future works will be presented.

CHAPTER

6

CONCLUSION AND FUTURE WORKS

In this Master Dissertation, it has been proposed a model for robust vehicle tracking. This model has been implemented using the TensorFlow framework. It achieved good results and outperformed the Inception-v2 in terms of precision in the classification and it was better than Correlation Filter Tracker technique in tracking vehicles in videos. Different approaches have been tried for different tasks during the development of the proposed algorithms implementation which was discussed in the previous chapters. In this Chapter, it will be presented the advantages and disadvantages of the proposed system.

6.1 Summary of Findings

This work presented an architecture based on deep learning networks for recognizing and tracking a vehicle in a highway. The proposed system can receive and analyze images captured by an UAV (drone), aiming to track a vehicle. It is constituted by bio-inspired algorithms: VOCUS2, CNN and LSTM. The obtained results showed 90.55 % of average accuracy in the tested data classification, which demonstrates a good performance. Two aspects should be highlighted.

The first one is related to classification results. From the comparative results in the classification, in terms of accuracy, Inception-v2 network was better than the architecture proposed, but it is important to point out that Inception-v2 network took longer. On the other hand, in terms of precision, the proposed architecture had a better performance compared to Inception-v2 network.

The second aspect is related to the vehicle tracking task. In this task, the proposed approach was compared with the Correlation Filter Tracker technique. The obtained result showed that the proposed system got an IoU measure of approximately 75% compared to 62% obtained for the CFT. When comparing LSTM to GRU trackers, it was noted that by an average of 2% the LSTM wins the GRU in the first 2 cases of the tests performed, and in the third case,

by 4% approximately.

Some difficulties were found to arrive to these results. Certainly, it was not an easy task to obtain this system for tracking vehicles. At the beginning, there was not much data. Some videos were collected by using a drone existing in LAR-ICMC-USP Laboratory, to increase the data. Afterwards, many tests with different neural network topologies were taken with data augmentation, which was trained with different epochs.

In the state-of-the-art was found that many neural network architectures were prominent, where one of these architectures is almost similar to the Arch.1. Changing some parameters (applying a *dropout* between fully connected layers, and some variations in a number of feature maps), we obtained the Arch.2.

The Arch.2 gave better accuracy and precision than the Arch. 1. On the other hand, other topologies with much more layers as Inception-v2 from Google were investigated, which outperformed our system in accuracy using transfer learning. However, once trained the proposed neural networks, Arch.1 and Arch.2, a test without labels was performed. Then, after making the confusion matrix, it was realized that in precision, the two proposed architectures were better than Inception-v2, which means that it classified more vehicles that are really vehicles. The conclusion is that the Inception-v2 has better accuracy as it has before, but not the best precision. Since the Arch.2 had a better performance in precision than others, it was used to do the vehicle tracking. In order to do that, the data was obtained from the penultimate layer of the neural network (Arch.2). Those characteristics plus the position of the vehicle were the inputs for the recurrent network to do the tracking.

At the beginning, two videos were taken and, after comparing their average IoU measure, it provided a better result to LSTM compared to CFT. Besides, aiming to confirm this result, more videos were taken. Furthermore, it was also compared with another recurring algorithm, GRU, which is considered simpler than LSTM. The obtained results confirmed a better performance, demonstrating more stability of the proposed approach mainly in videos presenting curves and when the camera performs abrupt movements compared to GRU.

6.2 Limitations

The main disadvantages of the proposed system are:

- The software does not work properly with streetlights,
- It needs high computational capacity and needs a large number of images to be trained,
- When an occlusion occurs as for example, a car passing under the bridge or among the trees, etc.

- From our knowledge, it could not be found a database available on the internet yet taken at the perspective of an UAV.

6.3 Future Works

There are many possibilities to enhance the proposed system, which are listed as follows:

- Increase the data with large vehicles, since the data built for the tests in this work contains only images of small cars.
- Use transfer learning in the proposed architecture.
- Install the software into single board computers, such as, Jetson TX2, Raspberry pi, Banana pi, etc. and test them in a UAV from laboratory.
- Train the network track LSTM in an specific car on real time.
- Use reinforcement learning in the part of salient images, to cut the salient object autonomously.
- Solve the occlusion when the car pass under the bridge or among the trees and finally embed the proposed system into a UAV.

BIBLIOGRAPHY

ARULAMPALAM, M. S.; MASKELL, S.; GORDON, N.; ; CLAPP, T. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. 2002. Citation on page 27.

BOLME, D. S.; BEVERIDGE, J. R.; DRAPER, B. A.; LUI, Y. M. Visual object tracking using adaptive correlation filters. 2010. Citation on page 33.

BRADSKI, G. R. **Computer Vision Face Tracking For Use in a Perceptual User Interface**. 1998. Citation on page 27.

CHUNG, J.; GÜLÇEHRE, Ç.; CHO, K.; BENGIO, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. **CoRR**, abs/1412.3555, 2014. Citation on page 37.

DANELLJAN, M.; HAGER, G.; KHAN, F. S.; FELSBERG, M. Accurate scale estimation for robust visual tracking. In: **Proceedings of the British Machine Vision Conference**. [S.l.]: BMVA Press, 2014. Citation on page 27.

DENG, W. D. J.; SOCHER, L.-J. L. R.; LI, L. F.-F. K. Imagenet: A large-scale hierarchical image database. **conference on Computer Vision and Pattern Recognition**, 2009. Citations on pages 22 and 25.

DUAN, K.; KEERTHI, S. S.; CHU, W.; SHEVADE, S. K.; POO, A. N. Multi-category classification by soft-max combination of binary classifiers. In: **In 4th International Workshop on Multiple Classifier Systems**. [S.l.: s.n.], 2003. Citation on page 26.

FRINTROP, S.; WERNER, T.; GARCIA, G. M. Traditional saliency reloaded: A good old model in new shape. In: **The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2015. Citations on pages 31 and 49.

FRINTROP, S.; WERNER, T.; GARCIA, G. M. Traditional saliency reloaded: A good old model in new shape. **IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, 2015. Citation on page 32.

GHAEMMAGHAMI, M. P. **Tracking of Humans in Video Stream Using LSTM Recurrent Neural Network**. Master's Thesis (Master's Thesis), 2017. Citations on pages 28 and 36.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Citations on pages 33, 40, 41, and 42.

GORDON, D.; FARHADI, A.; FOX, D. Re3 : Real-time recurrent regression networks for object tracking. **CoRR**, abs/1705.06368, 2017. Available: <http://arxiv.org/abs/1705.06368>. Citation on page 28.

HE, K.; ZHANG, X.; REN, S.; SUN, J. Deep residual learning for image recognition. **CoRR**, abs/1512.03385, 2015. Available: <http://arxiv.org/abs/1512.03385>. Citation on page 27.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, MIT Press, v. 9, n. 8, p. 1735–1780, 1997. Citation on page 36.

HUTTUNEN, H.; YANCHESHMEH, F. S.; CHEN, K. Car type recognition with deep neural networks. **IEEE Intelligent Vehicles Symposium (IV)**, 2016. Citations on pages 15, 25, 26, and 52.

ITTI, L. Models of bottom-up attention and saliency. In: **Tsotsos (Eds.), Neurobiology of Attention, Elsevier**. [S.l.]: Elsevier, 2005. p. 576–582. Citation on page 31.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. 1995. Citation on page 27.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. **Proceedings of 26th Annual Conference on Neural Information Processing Systems**, 2012. Citation on page 26.

Le Cun, Y.; BOTTOU, L.; BENGIO, Y.; HAFFNER, P. Gradient based learning applied to document recognition. **Proceedings of IEEE**, v. 86, n. 11, p. 2278–2324, 1998. Available: <http://leon.bottou.org/papers/lecun-98h>. Citation on page 51.

LIN, M.; CHEN, Q.; YAN, S. Network in network. **CoRR**, abs/1312.4400, 2013. Available: <http://arxiv.org/abs/1312.4400>. Citation on page 44.

LIU, D. Monza : Image classification of vehicle make and model using convolutional neural networks and transfer learning. In: . [S.l.: s.n.], 2015. Citation on page 26.

LIU, W.; ANGUELOV, D.; ERHAN, D.; SZEGEDY, C.; REED, S. E.; FU, C.; BERG, A. C. SSD: single shot multibox detector. **CoRR**, abs/1512.02325, 2015. Available: <http://arxiv.org/abs/1512.02325>. Citation on page 27.

LUQUE, B. **Spatio-Temporal Road Detection from Aerial Imagery using CNNs**. 2017. Available: <https://github.com/imatge-upc/videolabeler>. Accessed: 01 Feb. 2019. Citation on page 60.

M.A.AMARAL, E.; BADUE, C.; OLIVEIRA-SANTOS, T.; SOUZA, A. F. Detecção e rastreamento de veículos em movimento para automóveis robóticos autônomos. 2015. Citation on page 27.

MONTANARI, R. **Detecção e classificação de objetos em imagens para rastreamento de veículos**. Master's Thesis (Master's Thesis), 2015. Citations on pages 25 and 27.

NAM, H.; HAN, B. Learning multi-domain convolutional neural networks for visual tracking. 2016. Citation on page 32.

NING, G.; ZHANG, Z.; HUANG, C.; HE, Z.; REN, X.; WANG, H. Spatially supervised recurrent convolutional neural networks for visual object tracking. **CoRR**, abs/1607.05781, 2016. Available: <http://arxiv.org/abs/1607.05781>. Citation on page 28.

OLIVEIRA, T. P. **PREDIÇÃO DE TRÁFEGO, USANDO REDES NEURAIS ARTIFICIAIS, PARA GERENCIAMENTO ADAPTATIVO DE LARGURA DE BANDA EM ROTEADORES.** Master's Thesis (Master's Thesis), 2014. Citation on page 35.

REDMON, J.; DIVVALA, S. K.; GIRSHICK, R. B.; FARHADI, A. You only look once: Unified, real-time object detection. **CoRR**, abs/1506.02640, 2015. Available: <http://arxiv.org/abs/1506.02640>. Citation on page 28.

RIBEIRO, S.; ALQUEZAR, R. Incremental construction of lstm recurrent neural network. 12 2002. Citation on page 36.

RIVEROS, E. R. L.; CACERES, J. C. G.; CHáVEZ, J. G. Analyzing the effect of hyperparameters in a automobile classifier based on convolutional neural networks. 2016. Citations on pages 26 and 27.

ROCHA, R. H. S. **Reconhecimento de Objetos por Redes Neurais Convolutivas**. Master's Thesis (Master's Thesis), 2015. Citations on pages 39 and 43.

ROSSINI, C. G. J. C. Visual attention: behavioral studies about location-based and object-based selection. **Estudos de Psicologia (Natal)**, 2006. Available: <http://www.scielo.br/pdf/epsic/v11n1/10.pdf>. Citation on page 31.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence A Modern Approach**. [S.l.: s.n.], 1995. Citation on page 34.

SAK, H.; SENIOR, A. W.; BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In: **INTERSPEECH**. [S.l.: s.n.], 2014. Citation on page 37.

SATAR, B.; DIRIK, A. E. Deep learning based vehicle make-model classification. **CoRR**, abs/1809.00953, 2018. Available: <http://arxiv.org/abs/1809.00953>. Citation on page 27.

SCHERER, D.; MULLER, A.; BEHNKE, S. Evaluation of pooling operations in convolutional architectures for object recognition. In: **Proceedings of the 20th International Conference on Artificial Neural Networks: Part III**. Berlin, Heidelberg: Springer-Verlag, 2010. (ICANN'10), p. 92–101. ISBN 3-642-15824-2, 978-3-642-15824-7. Available: <http://dl.acm.org/citation.cfm?id=1886436.1886447>. Citation on page 26.

SILVA DANILO HERNANE SPATTI, R. A. F. Ivan Nunes da. **Redes Neurais Artificiais para Engenharia e Ciencias Aplicadas**. [S.l.: s.n.], 2010. Citation on page 35.

SZEGEDY, C.; LIU, W.; JIA, Y.; SERMANET, P.; REED, S. E.; ANGUELOV, D.; ERHAN, D.; VANHOUCKE, V.; RABINOVICH, A. Going deeper with convolutions. **CoRR**, abs/1409.4842, 2014. Available: <http://arxiv.org/abs/1409.4842>. Citations on pages 44 and 57.

SZEGEDY, C.; VANHOUCKE, V.; IOFFE, S.; SHLENS, J.; WOJNA, Z. Rethinking the inception architecture for computer vision. **CoRR**, abs/1512.00567, 2015. Available: <http://arxiv.org/abs/1512.00567>. Citations on pages 26 and 44.

VIDAL, F. de B. Rastreamento visual de objetos utilizando otimização por enxame de partículas. **VI Workshop de Visão Computacional**, 2010. Citation on page 27.

WANG, J.-G.; ZHOU, L.; PAN, Y.; LEE, S.; SONG, Z.; HAN, B. S.; SAPUTRA, V. B. Appearance-based brake-lights recognition using deep learning and vehicle detection. **IEEE Intelligent Vehicles Symposium (IV)**, 2016. Citation on page 26.

WELCH, G.; BISHOP, G. **An Introduction to the Kalman Filter**. 1995. Citation on page 27.

WIATOWSKI, T.; BÖLCSKEI, H. A mathematical theory of deep convolutional neural networks for feature extraction. **CoRR**, abs/1512.06293, 2015. Available: <http://arxiv.org/abs/1512.06293>. Citation on page 33.

YILMAZ, A.; JAVED, O.; SHAH, M. Object tracking: A survey. **ACM Computing Surveys**, 2006. Citation on page 27.

YILMAZ OMAR JAVED, M. S. A. Object tracking: A survey. **ACM Comput. Surv. 38**, 2006. Citation on page 28.

YIN, W.; KANN, K.; YU, M.; SCHÜTZE, H. Comparative study of CNN and RNN for natural language processing. **CoRR**, abs/1702.01923, 2017. Available: <http://arxiv.org/abs/1702.01923>. Citation on page 36.


ICMC
USP
SÃO CARLOS