MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE ABBES LAGHROUR KHENCHELA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département de Mathématiques et Informatique

N° de série : …

Mémoire de fin d'études

Pour l'obtention du diplôme de Master (L.M.D)

Spécialité : Informatique
Option : Sécurité et technologie web

Object detection and recognition using Deep learning

Réalisé par : BELHOUCHET Wail
HANI Achref bader eldine

Dirigé par : Dr. ABBAS Fayçal

Membres de jury :
Dr. HEMAM Mounir Président
Dr. BAKHOUCHE Abdelali Examinateur

Présenté le 26/06/2019

Dedication

BELHOUCHET Wail

I take this opportunity to express my gratitude to my family. To my dear parents, a source of love and affectation, who watch over my success and happiness. For the two being, who gave me life, who saw me grow up, the two that I could never thank enough.

To my dear sisters Mayar , Malak, Acil. I also dedicate to my friends.

HANI Achref bader eldine

to my parents for the sacrifices made in my respect, for their patience their love and their confidence in me they did everything for my happiness and my success. May they find in this modest work the testimony of my deep affection and unwavering attachment. No dedication can express what I owe them that god reserves for them good health and a long life. I dedicate to all my family .

Thanks

We want, first of all, to thank ALLAH for giving us strength and health, courage, willingness to accomplish this modest work .

This work of memory was accomplished under the direction of Dr Fycal Abbas . We thank him for his precious time provides, who we followed and derailed for the good realization of this project. .

We send our thanks to everyone who helped us to finalize this project including our parnet, our respective families who have supported and encouraged us throughout this journey .

Abstract

In the field of computer vision the problem of detection and recognition of different objects is imposed. The Convolutional Neural Networks (CNN) are a new concept for the deep learning way especially recognition and classification of images and videos, natural language processing and other applications. In particular vehicle detection for aerial images.

Detecting vehicles in aerial images provides important information for traffic management and urban planning. Detecting the cars in the images is challenging due to the relatively small size of the target objects and the complex background in man-made areas .

In this work, a vehicle detection method for aerial image is presented. Our method works in two steps: a data preparation step, it consists in applying treatments on the input image, the second step is to use the convolutional neural networks for the detection and the recognition of the vehicles in aerial image. Our method offers good results in terms of detection and recognition and produce results in real time.

Résumé

Dans le domaine de la vision par ordinateur, le problème de la détection et de la reconnaissance de différents objets s'impose. Les réseaux de neurones convolutionnels (CNN) constituent un nouveau concept d'apprentissage approfondi, notamment la reconnaissance et la classification d'images et de vidéos, le traitement du langage naturel et d'autres applications. En particulier, détection de véhicules pour des images aériennes.

La détection de véhicules sur des images aériennes fournit des informations importantes pour la gestion du trafic et la planification urbaine. Détecter les voitures dans les images est difficile en raison de la taille relativement petite des objets cibles et du fond complexe dans les zones artificielles..

Dans ce travail, nous présentons une méthode de détection des véhicules pour les images aériennes. Notre méthode fonctionne en deux étapes : une étape de préparation des données, elle consiste à appliquer des traitements sur l'image d'entrée, la seconde étape consiste à utiliser les réseaux de neurones à convolution pour la détection et la reconnaissance des véhicules dans une image aérienne. Notre méthode offre de bon résultats en terme de détection et de reconnaissance ainsi produire des résultats en temps réel.

ملخص

في مجال رؤية الكمبيوتر، يتم فرض مشكلة الكشف عن الأشياء المختلفة والتعرف عليها. تعد الشبكات العصبية التلافيفية مفهوماً جديداً لطريقة التعلم العميق وخاصة التعرف على الصور ومقاطع الفيديو وتصنيفها ومعالجة اللغة الطبيعية والتطبيقات الأخرى. على وجه الخصوص الكشف عن مركبة للصور الجوية.

يوفر اكتشاف المركبات في الصور الجوية معلومات مهمة لإدارة حركة المرور والتخطيط الحضري. يمثل اكتشاف السيارات في الصور تحديا بسبب الحجم الصغير نسبيا للأجسام المستهدفة والخلفية المعقدة في المناطق التي يصنعها الإنسان.

في هذا العمل، يتم تقديم طريقة الكشف عن مركبة للصورة الجوية. تعمل طريقتنا على خطوتين: خطوة إعداد البيانات، وهي تتمثل في تطبيق المعالجات على صورة الإدخال، والخطوة الثانية هي استخدام الشبكات العصبية التلافيفية للكشف عن المركبات والتعرف عليها في الصورة الجوية. تقدم طريقتنا نتائج جيدة من حيث الكشف والتعرف كذلك تحقيق نتائج في الوقت الحقيقي.

Contents

List of Figures

Introduction

In very high resolution (VHR) remote sensing images, vehicle detection is an indispensable technology in both civilian and military surveillance, traffic management . Therefore, vehicle detection from aerial images has attracted significant attention worldwide . However, automatic vehicle detection in aerial images still has a lot of challenges due to the relatively small size and variable orientation of vehicles .

As the key technology of intelligent transportation system, vehicle detection is the basis for realizing many important functions , such as measurement and statistics of traffic parameters such as traffic flow and density, vehicle location and tracking .

In this work , we will describe vehicle detection with deep learning on aerial imager based on YOLO .We will start with an introduction to machine learning and neural networks in Chapter 1.

In Chapter 2 , we will introduce deep learning and we will describe convolutional neural networks (CNN) architttecture of layers, and how every layer transforms one volume to another through differentiable function .

In Chapter 3 , we will talk about how object detection and recognition work and the diferent models of object detection (R-CNN , fast RCNN , faster R-CNN , SSD , YOLO) . In the second part of this chapter we will describe transfer learning and the architecture of VGG16 .

in chapter 4 , we will show the experimental part of our work and we discuss the different results obtained .

Finally , we will finish this work with general conclusion and future work .

Chapter 1

Machine Learning

1.1 Introduction

In this chapter , we will introduce machine learning and describe here types and approaches . In recent years, machine learning has become an important research area providing us with for example self-driving cars, speech recognition and improved understanding of the human genome . The interest is strong also due to the constantly increasing volume and variability of data and computational processing which has become cheaper and more powerful .

Machine learning is an area of artificial intelligence .The basic concept of machine learning is to use algorithms to parse data, learn and from that be able to make predictions about the subject studied. Instead of having to hand-code all procedures to be able to perform a certain task, a large amount of data is fed to the machine to make it capable to learn how to perform the task.

1.2 Artificial intelligence

AI is the intelligence exhibited by machines or software. It is also the name of the academic field of study which studies how to create computers and computer software that are capable of intelligent behavior[1]. Major AI researchers and textbooks define this field as "the study and design of intelligent agents", in which an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success[2].

AI research is highly technical and specialized, and is deeply divided

into subfields that often fail to communicate with each other. Some of the division is due to social and cultural factors: subfields have grown up around particular institutions and the work of individual researchers[3]. AI research is also divided by several technical issues. Some subfields focus on the solution of specific problems. Others focus on one of several possible approaches or on the use of a particular tool or towards the accomplishment of particular applications.

The central problems (or goals) of AI research include reasoning, knowledge, planning, learning, natural language processing (communication), perception and the ability to move and manipulate objects[4]. General intelligence is still among the field's long-term goals. Currently popular approaches include statistical methods, computational intelligence and traditional symbolic AI. There are a large number of tools used in AI, including versions of search and mathematical optimization, logic, methods based on probability and economics, and many others. The AI field is interdisciplinary, in which a number of sciences and professions converge, including computer science,

1.2.1 The history of artificial intelligence

1.2.1.1 The 'Dark Ages', or the birth of artificial intelligence (1943–56)

The first work recognised in the field of artificial intelligence (AI) was presented by Warren McCulloch and Walter Pitts in 1943[1]. McCulloch had degrees in philosophy and medicine from Columbia University and became the Director of the Basic Research Laboratory in the Department of Psychiatry at the University of Illinois. His research on the central nervous system resulted in the first major contribution to AI: a model of neurons of the brain.

McCulloch and his co-author Walter Pitts, a young mathematician, proposed a model of artificial neural networks in which each neuron was postulated as being in binary state, that is, in either on or off condition (McCulloch and Pitts, 1943). They demonstrated that their neural network model was, in fact, equivalent to the Turing machine, and proved that any computable function could be computed by some network of connected neurons. McCulloch and Pitts also showed that simple network structures could learn[1].

1.2.1.2 The rise of artificial intelligence, or the era of great expectations (1956–late 1960s)

The early years of AI are characterised by tremendous enthusiasm, great ideas and very limited success. Only a few years before, computers had been introduced to perform routine mathematical calculations, but now AI researchers were demonstrating that computers could do more than that. It was an era of great expectations[5].

John McCarthy, one of the organisers of the Dartmouth workshop and the inventor of the term 'artificial intelligence', moved from Dartmouth to MIT. He defined the high-level language LISP – one of the oldest programming languages (FORTRAN is just two years older), which is still in current use. In 1958, McCarthy presented a paper, 'Programs with Common Sense', in which he proposed a program called the Advice Taker to search for solutions to general problems of the world (McCarthy, 1958). McCarthy demonstrated how his program could generate, for example, a plan to drive to the airport, based on some simple axioms. Most importantly, the program was designed to accept new axioms, or in other words new knowledge, in different areas of expertise without being reprogrammed. Thus the Advice Taker was the first complete knowledgebased system incorporating the central principles of knowledge representation and reasoning [5].

1.2.1.3 The technology of expert systems, or the key to success (early 1970s–mid-1980s)

Probably the most important development in the 1970s was the realisation that the problem domain for intelligent machines had to be sufficiently restricted. Previously, AI researchers had believed that clever search algorithms and reasoning techniques could be invented to emulate general, human-like, problem-solving methods[1]. A general-purpose search mechanism could rely on k knowledge about domain. However, when weak methods failed, researchers finally realised that the only way to deliver practical results was to solve typical cases in narrow areas of expertise by making large reasoning steps.

The expert systems mentioned above have now become classics. A growing number of successful applications of expert systems in the late 1970s showed that AI technology could move successfully from the research laboratory to the commercial environment. During this period, however, most expert systems were developed with special AI languages, such as LISP,

PROLOG and OPS, based on powerful workstations. The need to have rather expensive hardware and complicated programming languages meant that the challenge of expert system development was left in the hands of a few research groups at Stanford University, MIT, Stanford Research Institute and Carnegie-Mellon University. Only in the 1980s, with the arrival of personal computers (PCs) and easy-to-use expert system development tools – shells – could ordinary researchers and engineers in all disciplines take up the opportunity to develop expert systems[3].

1.2.1.4 How to make a machine learn, or the rebirth of neural networks (mid-1980s–onwards)

In the mid-1980s, researchers, engineers and experts found that building an expert system required much more than just buying a reasoning system or expert system shell and putting enough rules in it. Disillusion about the applicability of expert system technology even led to people predicting an AI 'winter' with severely squeezed funding for AI projects. AI researchers decided to have a new look at neural networks[1].

By the late 1960s, most of the basic ideas and concepts necessary for neural computing had already been formulated (Cowan, 1990). However, only in the mid-1980s did the solution emerge. The major reason for the delay was technological: there were no PCs or powerful workstations to model and experiment with artificial neural networks. The other reasons were psychological and financial .

Neural network technology offers more natural interaction with the real world than do systems based on symbolic reasoning. Neural networks can learn, adapt to changes in a problem's environment, establish patterns in situations where rules are not known, and deal with fuzzy or incomplete information. However, they lack explanation facilities and usually act as a black box[1]. The process of training neural networks with current technologies is slow, and frequent retraining can cause serious difficulties.

1.3 Machine Learning

Machine learning is a subfield of computer science that is concerned with building algorithms which, to be useful, rely on a collection of examples of some phenomenon[6]. These examples can come from nature, be handcrafted by humans or generated by another algorithm.

Machine learning can also be defined as the process of solving a practical problem by gathering a dataset and algorithmically building a statistical model based on that dataset, that statistical model is assumed to be used somehow to solve the practical problem [7].

1.4 Types of Learning

The four different types of learning :

- Supervised Learning .
- Unsupervised Learning.
- Semi-Supervised Learning.
- Reinforcement Learning.

1.4.1 Supervised Learning

The goal of a supervised learning algorithm is to use the dataset to produce a model that takes a feature vector x as input and outputs information that allows deducing the label for this feature vector. For exemple, the model created using the dataset of people could take as input a feature vector describing a person and output a probability that the person has cancer[8].

1.4.2 Unsupervised Learning

the goal of an unsupervised learning algorithm is to create a model that takes a feature vector x as input and either transforms it into another vector or into a value that can be used to solve a practical problem. For example, in clustering, the model returns the id of the cluster for each feature vector in the dataset[14]. In dimensionality reduction, the output of the model is a feature vector that has fewer features than the input x; in outlier detection, the output is a real number that indicates how x is different from a "typical" example in the dataset[8].

1.4.3 Semi-Supervised Learning

The goal of a semi-supervised learning algorithm is the same as the goal of the supervised learning algorithm. The hope here is that using many unlabeled examples can help the learning algorithm to find (we might say "produce" or "compute") a better model. It could look counter-intuitive that

learning could benefit from adding more unlabeled examples. It seems like we add more uncertainty to the problem. However, when you add unlabeled examples, you add more information about your problem: a larger sample reflects better the probability distribution the data we labeled came from. Theoretically, a learning algorithm should be able to leverage this additional information[9].

1.4.4 Reinforcement Learning

Reinforcement learning is a subfield of machine learning where the machine "lives" in an environment and is capable of perceiving the state of that environment as a vector of features. The machine can execute actions in every state. Different actions bring different rewards and could also move the machine to another state of the environment. The goal of a reinforcement learning algorithm is to learn a policy.[9] A policy is a function (similar to the model in supervised learning) that takes the feature vector of a state as input and outputs an optimal action to execute in that state. The action is optimal if it maximizes the expected average reward[10]. Reinforcement learning solves a particular kind of problem where decision making is sequential, and the goal is long-term, such as game playing, robotics, resource management, or logistics. The input examples are independent of one another and the predictions made in the past.

1.5 Tasks of machine learning

Examples of common tasks that can be resolved by machine learning algorithms are classification, Regression.

Classification : The purpose of classification is to associate a label y in $1, \cdots, K$ to a data x by learning a function f such that $y = f(x)$. Some well-known classification tasks are object detection, face recognition or fraud detection[8].

Regression : Similar to classification, in a regression task a function f such that given an input x will predict a numeric value y in R such that $y = f(x)$. An example of a regression task is to predict the fuel price given information about production level [8].

1.6 Approaches of machin learning

1.6.1 Decision tree learning

Decision tree learning uses a decision tree as a predictive model, which maps observations about an item to conclusions about the item's target value [11].

1.6.2 Association rule learning

Association rule learning is a method for discovering interesting relations between variables in large databases [12].

1.6.3 Bayesian networks

A Bayesian network, belief network or directed acyclic graphical model is a probabilistic graphical model that represents a set of random variables and their conditional independencies via a directed acyclic graph (DAG)[13]. For example, a Bayesian network could represent the probabilistic relationships between diseases and symptoms. Given symptoms, the network can be used to compute the probabilities of the presence of various diseases. Efficient algorithms exist that perform inference and learning [14].

1.6.4 Representation learning

Several learning algorithms, mostly unsupervised learning algorithms, aim at discovering better representations of the inputs provided during training. Classical examples include principal components analysis and cluster analysis. Representation learning algorithms often attempt to preserve the information in their input but transform it in a way that makes it useful, often as a preprocessing step before performing classification or predictions, allowing to reconstruct the inputs coming from the unknown data generating distribution, while not being necessarily faithful for configurations that are implausible under that distribution [15].

1.6.5 Artificial neural networks

An artificial neural network (ANN) learning algorithm, usually called "neural network" (NN), is a learning algorithm that is inspired by the structure and functional aspects of biological neural networks [16]. Computations

are structured in terms of an interconnected group of artificial neurons, processing information using a connectionist approach to computation. Modern neural networks are non-linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, to find patterns in data, or to capture the statistical structure in an unknown joint probability distribution between observed variables[17] .

1.6.5.1 Biological and artificial neurons

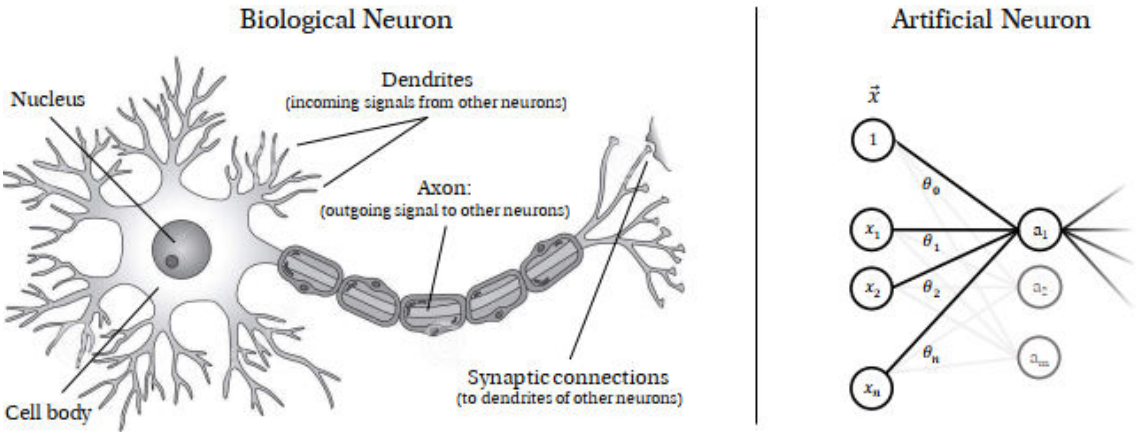

Figure 1.1: A schematic of a biological neuron on the left, and a schematic of an artificial neuron on the right .

Figure 1.1 shows a schematic view of a biological neuron The biological neuron receives incoming signals from other neurons via its Dendrites. These signals are accumulated in the Cell Body. When this accumulated signal surpasses a certain threshold, the Axon fires an output signal. This outgoing signal flows via Synaptic connections to the Dendrites of other neurons where the proces repeats itself.

The strength of the synaptic connection between the Axon and the Dendrite of the other neuron is called a synaptic weight. It reflects how much influence the firing of the sending neuron has on the receiving neuron. The free parameters in the artificial neural network are supposed to represent these synaptic weights. But how these synaptic weights are changed inside the brain and how humans learn is only poorly understood[16].

An artificial neuron is a mathematical description of how the biological

neuron works. Its connectivity to other artificial neurons is often depicted in schematics such as the one in figure 1.1. In this connectivity scheme each node represents a neuron and the connections between them represent the synaptic connections.

1.6.5.2 Network architectures

Single-layer Feedforward networks :

In a layered neural network the neurons are organized in the form of layers in the simplest form of a layered network,we have an input layer of source nodes that projects onto an output layer of neurons [18] .it is illustrated in this figure :

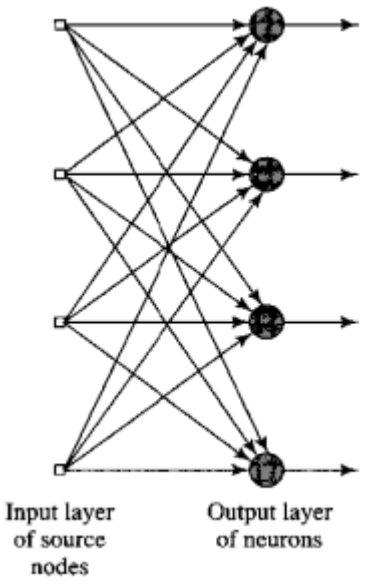

Figure 1.2: Feedforward with a single layer of neurons.

Multilayer Feedforward Networks :

The second classs of a feedforward neural network distinguiches itself by the presence of one or more hidden layers .The function of hidden neurons is to intervene between the estrnal input and the network output in some useful manner [18].By adding one or more hidden layers , the network is enabled to extract higher-order statistics.

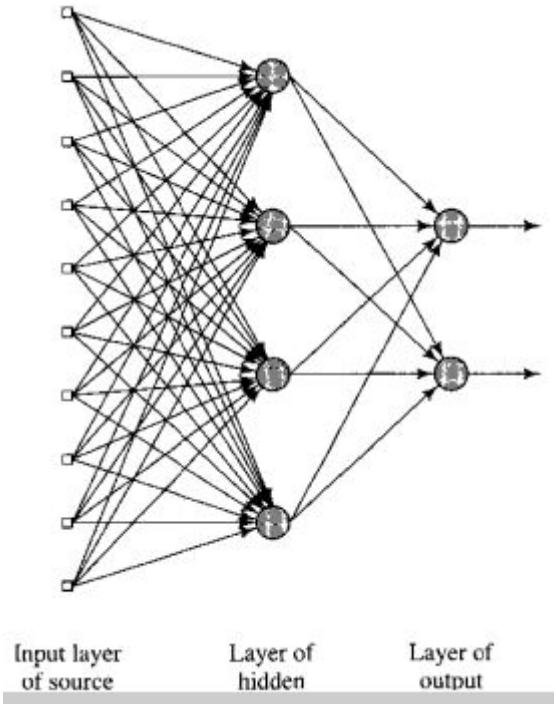

Figure 1.3: Feedforward with one hidden layer and one output laye.r

1.6.5.3 Real-life applications

The tasks artificial neural networks are applied to tend to fall within the following broad categories:[19]

- Function approximation, or regression analysis, including time series prediction, fitness approximation and modeling.
- Data processing, including filtering, clustering, blind source separation and compression.
- Robotics, including directing manipulators, prosthesis.
- Classification, including pattern and sequence recognition, novelty detection and sequential decision making.

1.7 Conclusion

Machine learning is a very vast field. In this chapter we have seen the important concepts in machine learning, of course there are other concepts and other learning techniques, we have presented the mane concepts in this field and talked about artificial neural networks and their architecture .

In the next chapitre , we will introduce deep learning wich is subfiled of machine learning and describe convolutional neural networks CNN .

Chapter 2

Deep Learning

2.1 Introduction

In this chapter, we will review the basic concepts of deep learning. Next, we are going to explain convolutional Neural Network (CNN), which is widely used in image recognition.

In recent years, deep learning have become a very popular and important research field .When speaking about deep learning it is commonly thought that deep learning is only relevant to use when there is a large amount of data available to train on. This is partly true but not always. Indeed, deep learning needs to be capable of learning features automatically and this usually requires a great amount of training data, especially in the case of images which is a high dimensional problem. Nonetheless, the main algorithm within deep learning, convolutional neural network (CNN), is one of the best models for computer vision problems even for very little training data . Using a small amount of training data to train a CNN from scratch will still yield reasonable results. Moreover, deep learning algorithms are also reusable.

2.2 What is deep learning

Deep learning (deep machine learning, or deep structured learning, or hierarchical learning, or sometimes DL) is a branch of machine learning based on a set of algorithms that attempt to model high-level abstractions in data by using model architectures, with complex structures or otherwise, composed of multiple non-linear transformations [20][21].

Deep learning is part of a broader family of machine learning methods

based on learning representations of data[22]. An observation (an image) can be represented in many ways such as a vector of intensity values per pixel, or in a more abstract way as a set of edges, regions of particular shape, Some representations make it easier to learn tasks (face recognition or facial expression recognition).

Research in this area attempts to make better representations and create models to learn these representations from large-scale unlabeled data[23]. Some of the representations are inspired by advances in neuroscience and are loosely based on interpretation of information processing and communication patterns in a nervous system, such as neural coding which attempts to define a relationship between the stimulus and the neuronal responses and the relationship among the electrical activity of the neurons in the brain[24][25].

Deep learning is an area within machine learning research and a subset of artificial intelligence, Figure 2.1 .

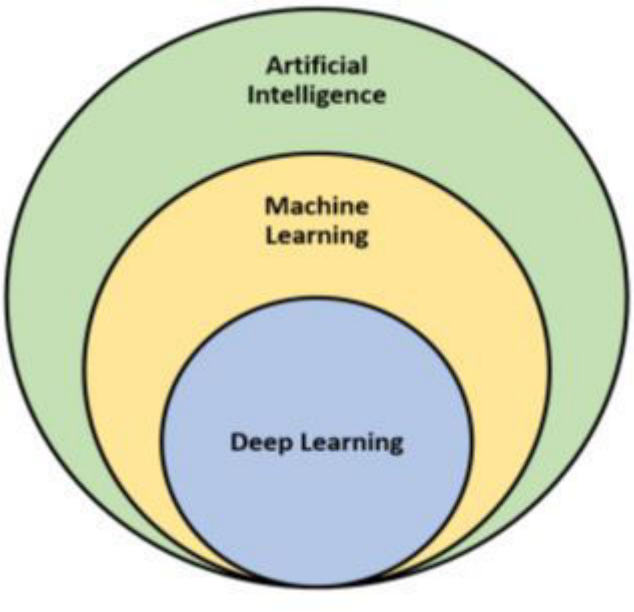

Figure 2.1: Scope of artificial learning, with machine learning, and deep learning .

2.3 Deep neural networks

A deep neural network (DNN) is an artificial neural network with multiple hidden layers of units between the input and output layers. Similar to shallow ANNs, DNNs can model complex non-linear relationships. DNN architectures, for object detection and parsing generate compositional models where the object is expressed as layered composition of image primitives [26]. The extra layers enable composition of features from lower layers, giving the potential of modeling complex data with fewer units than a similarly performing shallow network .

DNNs are typically designed as feedforward networks, but recent research has successfully applied the deep learning architecture to recurrent neural networks for applications such as language modeling. Convolutional deep neural networks (CNNs) are used in computer vision where their success is well-documented. More recently, CNNs have been applied to acoustic modeling for automatic speech recognition (ASR), where they have shown success over previous models [27] .

2.4 Convolutional neural network

The success of convolutional neural networks (CNN) is one reason of the recent popularity of neural networks. Instead of applying for each input the sum over all the parameters wi, we consider the parameters as a small grid called a kernel that is shared across the input xi [16]. Those parameters will force the network to learn local features that correspond to some meaningful parts of an object (in case where the inputs are images). For example, first layers of a CNN will learn efficient edge detection and the last layers will learn more specific features.[28]

many neurally inspired models of computational vision: the NeoCognitron , HMAX and LeNet-5 . While they may differnet in the details of their implementation, all these models share the same basic architecture, an example of which is shown in(figure 2.2) They alternate layers of simple and complex units , arranged in 2D grids to mimic the visual field. Each unit at layer L is connected to a local subset of units at layer L-1.

LeNet-5 additionally adds the constraint that all neurons at a given layer are "replicated" across the entire visual field. These form feature maps which are shown in figure 2.2 as stacks of overlapping rectangles. neurons within the

same feature map share the same parameters and each feature map contains a unique offset . The result is a Convolutional Neural Network (CNN).

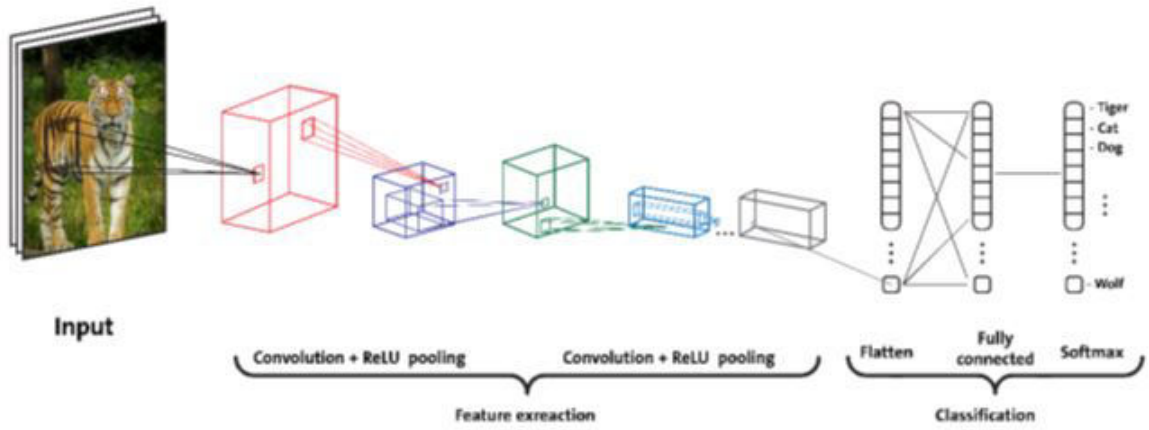

Figure 2.2: An example of a convolutional neural network .

2.5 Architecture of convolutional neural networks

The convolutional, pooling and fully-connected layers are the core building blocks of convolutional neural networks. Like a fully-connected neural network, a CNN is a type of feedforward network which can be written as a composite function of these high-level building blocks [16]. The output of the previous layer is fed-forward as input for the next layer, in this way a CNN gradually transforms the input image into the desired output .

The most common form of a CNN architecture stacks a few convolutional layers, follows them with local pooling layers, and repeats this pattern until the tensor has been reduced to a small spatial size 2. At some point, it is common to use a global pooling layer to make the transition to fully-connected layers of which there are usually only one or two. The last fully-connected layer has no activation function. It outputs the regressed values in the case of a regression problem, and the distances to the decision boundaries in the case of a classification problem. A final softmax layer normalizes these distances into proper class probabilities that sum to 1. The class that holds the heighest probability is then selected as the predicted class[29].

2.6 Convolutional layers

The main building block of a CNN is the Convolutional layer. Like it is convenient to think of neurons in fully-connected layers as vectors, it is convenient to represent the neurons in convolutional layers as 3-dimensional tensors with spatial dimensions: width £ height £ channels. For a color image in the input layer, these channels could be the RGB color channels [27].

A convolutional layer transforms a 3D input tensor into a 3D output tensor. To avoid confusion, we refer to the output channels as featuremaps and to the input channels simply as channels. The name 'featuremap' is particularly well chosen ,the output values of neurons in a featuremap indicate the absence or presence of a particular image feature at a particular spatial location in the image. Neurons in different featuremaps detect different image features, while neurons in the same featuremap detect the same image feature, but at different spatial locations [28]. One could imagine an image feature as an small image patch of which the content displays an object of interest. For example, a nose when you want to detect faces.

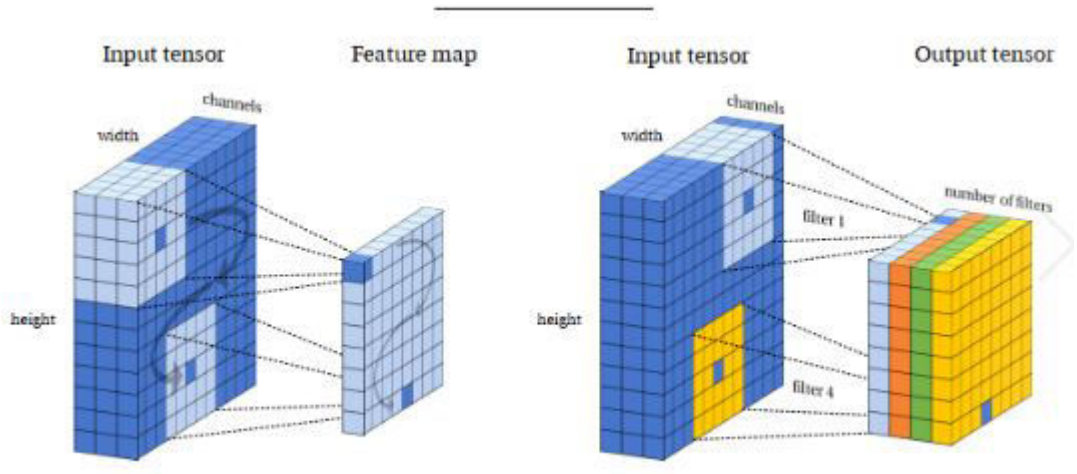

Figure 2.3: Example of how a convolution layer works

in this figure above The left part shows how a 5x5x3 filter slides through the input tensor and produces an output featuremap. Each output neuron in the featuremap takes as input the signals coming from a local neighborhood of 5x5x3=75 input neurons. A convolutional layer outputs a stack of multiple

featuremaps which are all constructed with different filter weights. This can be seen in the right part of the figure.

A convolutional layer outputs a stack of featuremaps which are all constructed with different filter weights, see the right part of figure 2.3. A filter only slides along the spatial dimensions (width and height) of the input tensor, and always extends along the full length of the input channels. This ensures that all features detected by the previous layer, can be combined to detect new features in the current layer. For example, if the previous layer contains featuremaps that have detected eyes, ears, mouths and noses, then the current convolutional layer can combine these features to detect faces.

2.7 Pooling layers

A pooling layer is a commonly used building block in CNNs. It reduces the spatial dimensions (the width and the height) of a tensor by a applying a subsampling operation to each input channel, indepently. There are many variants of this subsampling operation but the most common variants are average and max pooling. The pooling layer works in a similar way as the convolutional layer, it slides a local neighbordhood over the input channel, but instead of taking a weighted sum of this local neighboorhood it takes the maximum or the average. To produce a smaller output tensor these neighborhoods are strided with a stepsize that is equal or greater than two [27]. Summarizing the differences with a convolutional layer, the local neighborhood covers a single channel only and does not extend along the full length of the input channels; the local neighborhoods are strided with a stepsize that is equal or greater than two; the pooling layer has no learnable parameters and computes a statistic (the average or the maximum) instead of a weighted sum.

Figure 2.4 shows an illustration of how a pooling layer works. This example shows a max pooling layer in it most common form: a 2x2 neighborhood with a stride of two downsamples every input channel by a factor two along both width and height. In the downsampled featuremap the maximum value of every 2x2 neighborhood in the input tensor is preserved, while the other three activations are discarded. There are only two commonly seen variations of the pooling layer in practice, a pooling layer with neighborhood of 3x3 and a stride of two (also called overlapping pooling), and more commonly a neighborhood of 2x2 and a stride of two. Pooling with larger sizes

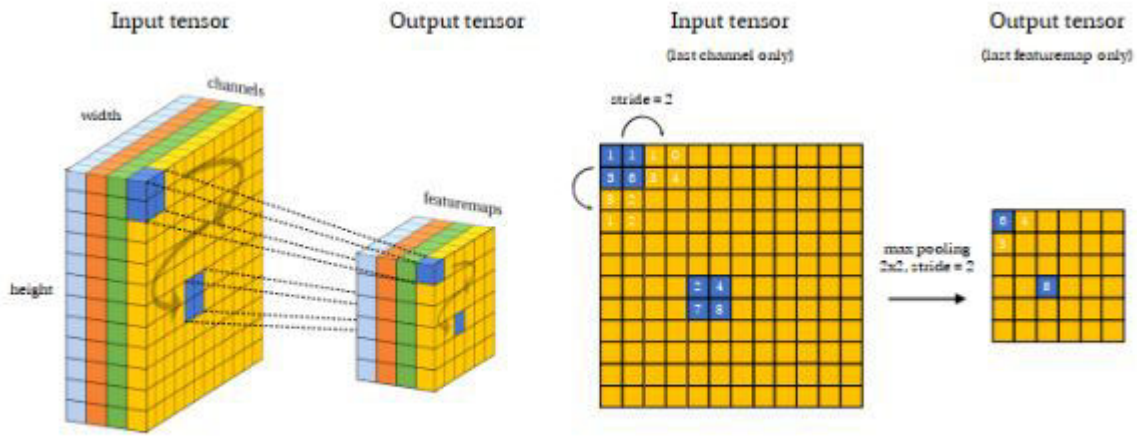

Figure 2.4: Example of how a pooling layer works .

and strides are too destructive. It is worth noting that there is currently also a trend to use normal convolutional layers with a stride of two instead of pooling layers [52].

2.8 Additional layers

The convolutional layer typically includes a non-linear activation function, such as a rectified linear activation function . Activations are sometimes described as a separate layer between the convolutional layer and the pooling layer.

Some systems, such as [27], also implement a layer called local response normalization, which is used as a regularization technique. Local response normalization mimics a function of biological neurons called lateral inhibition, which causes excited neurons to decrease the activity of neighbouring neurons.

The final hidden layers of a CNN are typically fully-connected layers [30] [31]. A fully-connected layer can capture some interesting relationships parameter-sharing convolutional layers cannot. However, a fullyconnected layer requires a sufficiently small data volume size in order to be practical. Pooling and stride settings can be used to reduce the size of the data volume that reaches the fully-connected layers. A convolutional network that does

not include any fully-connected layers, is called a fully convolutional network (FCN) [30]. If the network is used for classification, it usually includes a softmax output layer [31]. The activations of the topmost layers can also be used directly to generate a feature representation of an image. This means that the convolutional network is used as a large feature detector [32].

2.9 Activation function types

The activation function determines the final output of each neuron. It is important to select the function properly in order to create an effective network.

Early researchers found that perceptrons and other linear systems had severe drawbacks, being unable to solve problems that were not linearly separable, such as the XOR-problem. Sometimes, linear systems can solve these kinds of problems using hand-crafted feature detectors, but this is not the most advantageous use of machine learning. Simply adding layers does not help either, because a network composed of linear neurons remains linear no matter how many layers it has [26].

2.9.1 Sigmoid

The main reason why we use sigmoid function is because it exists between (0 to 1). Therefore, it is especially used for models where we have to predict the probability as an output.Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice [27].

$$A = \frac{1}{1 + e^{-x}}$$

The function is differentiable.That means, we can find the slope of the sigmoid curve at any two points.

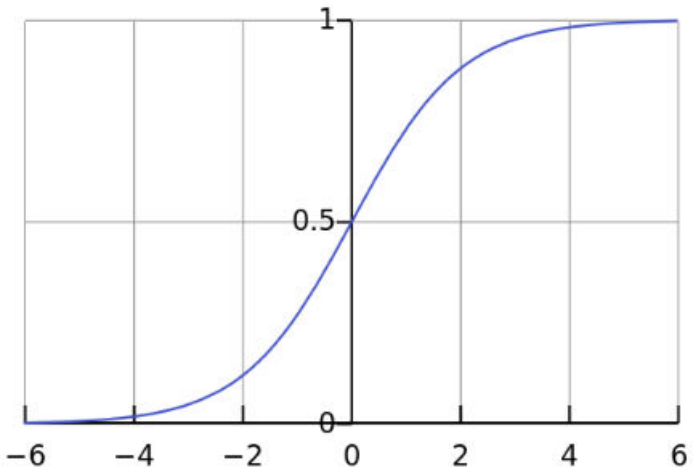

Figure 2.5: Graph of sigmoid function .

2.9.2 Tanh or hyperbolic tangent

tanh is also like logistic sigmoid but better. The range of the tanh function is from (-1 to 1). tanh is also sigmoidal (s - shaped) [16].

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

The advantage is that the negative inputs will be mapped strongly negative and the zero inputs will be mapped near zero in the tanh graph.

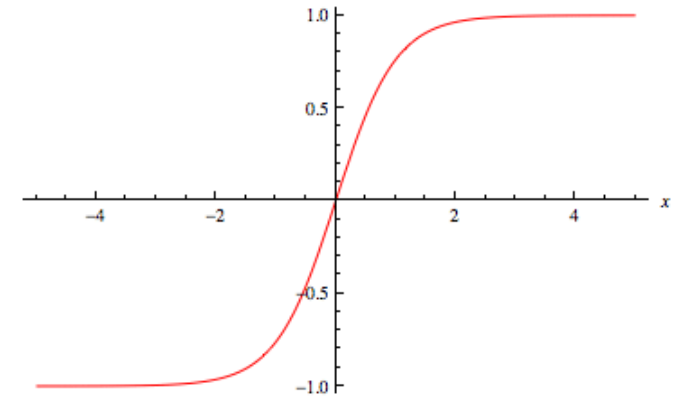

Figure 2.6: Graph of Tanh function .

2.9.3 ReLU (Rectified Linear Unit)

The ReLU is the most used activation function in the world right now.Since, it is used in almost all the convolutional neural networks or deep learning [16].

$$A(x) = max(0, x)$$

A light-weight and effective way of creating a non-linear network is using rectified linear units (ReLu) [22]. the range is [0 to infinity).The ReLu function is as shown above it given an output x if x is positive and 0 otherwise.

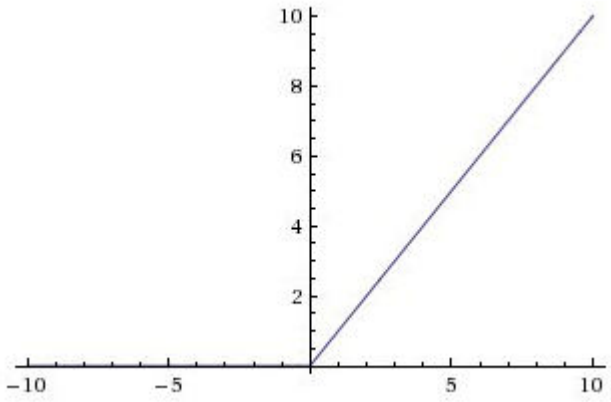

Figure 2.7: Graph of ReLu function .

2.9.4 Leaky ReLU

The leak helps to increase the range of the ReLU function. Usually, the value of a is 0.01 or so. Therefore the range of the Leaky ReLU is (-infinity to infinity).[16]

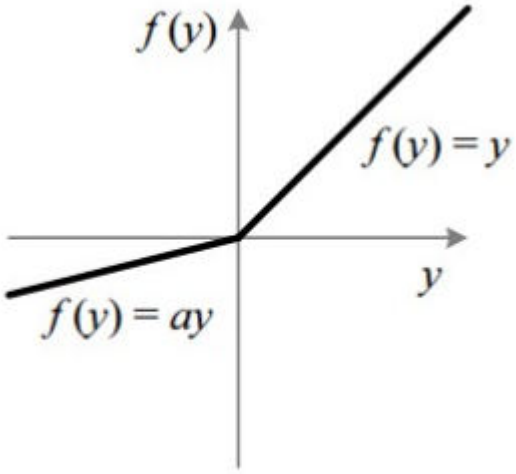

Figure 2.8: Graph of leaky ReLu function .

2.9.5 Softmax

For multi-class classification problems, the softmax activation function [31] is used in the output layer of the network.

The softmax function takes a vector of K arbitrarily large values and outputs a vector of K values that range between 0...1 and sum to 1. The values output by the softmax unit can be utilized as class probabilities.

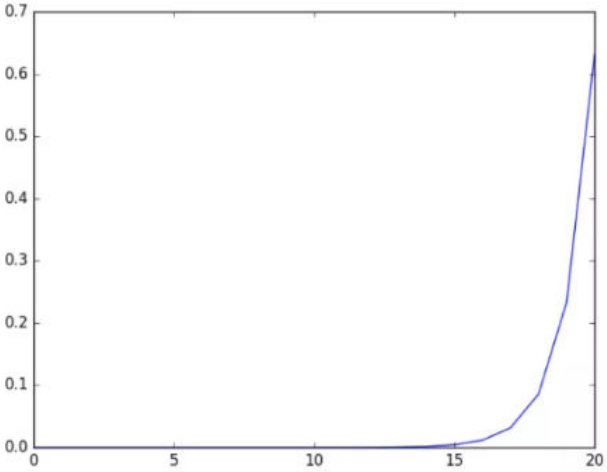

Figure 2.9: Graph of softmax function .

2.10 Conclusion

In this chapter we presented deep learning and convolutional neural networks CNN , also we described in detail the architecture of CNN starting with convolution layer and it end up with fully connected layer .

In next chapter we will see object detection and recognition concepts, models and how CNN work in that models .

Chapter 3

Object detection and Transfer Learning

3.1 Introduction

In this chapter, we will defines a number of important terminology commonly used in image processing related research which are vital to prevent reader confusion. In particular the terms object detection, object recognition and transfer learning .

In computer vision terminology, the term object detection generally refers to the classification of a perceived object into any human identifiable type (a car, human, animal, house, tree) .

In contrast, in computer vision terminology, object recognition refers to the classification of a perceived object into a particular group of the human identifiable types(car, human, animal, house, tree) Object recognition requires a more detailed analysis of features for the purpose of classification into one of the sub groups of identifiable objects.

Transfer learning consists in transferring the parameters of a neural network trained with one dataset and task to another problem with a different dataset and task . Many deep neural networks trained on natural images exhibit a curious phenomenon in common: on the first layers they learn features that appear not to be specific to a particular dataset or task, but general in that they are applicable to many datasets and tasks.

3.2 Object Detection in Computer Vision

The ability to identify the objects present in an image or scene is one of the most basic requirements when it comes to interacting with ones environment. While it seems completely effortless with humans and in fact most animals, trying to teach computers to see and also understand what they are seeing has proven extremely difficult [33].

The key to understanding visual scenes are three closely related subproblems. The easiest one will be called classification in the following. For classification, the one dominant object in a given image should be determined and labelled. The next more demanding task is object localisation: In addition to labelling the dominant object, it also needs to be localised in the image, usually by determining a bounding box around the image region that is occupied by the object. The difficulty of this task again increases if not only one but all objects in an image need to be labelled and multiple objects of the same category can appear in one image. This task is called object detection.

In addition, object detection is a big and very active field of research, so the following paragraph will only give a very brief overview about recent advances.

3.3 Object Localisation with CNN

For image classification, the output of the CNN is a 1000-dimensional vector that contains a probability of being the most prominent object for each of the possible classes. If only this one most prominent object should be localised, the network can simply be expanded to produce bounding box coordinates (either only one bounding box or one bounding box per possible object class). This was first done in [29]. is still used successfully by recent models like [27].

The big drawback of this approach is that it is not applicable if the number of bounding boxes that need to be predicted (the number of objects detectable in the scene) is not previously known. But this is precisely the case in object detection tasks. A simple cure would be to apply a network for classification and localisation to every part of the image (at different scales) successively and to accumulate the boxes with the best results along the way.

This sliding window approach is of course extremely expensive due to the huge search space. To reduce the amount of times that the classifier (a CNN) needs to run, it could be applied on a coarser grid, but this has the risk of missing the ideal bounding box in some cases. Ideally, one would like to have an initial set of somehow proposed regions that need to be evaluated. It should be minimal while still containing all ground truth bounding boxes of objects in the image. This approach was successfully introduced by Girshick et al. [33] in their work called R-CNN.

3.4 Convolutional object detection

Unlike image classification, object detection requires multi-object localization. Because regression tool is used to predict fixed number of outputs, the classification network cannot be used to directly regress from the input image to objects' location. In this case, the network does not know how many objects are present in the input images. Thus, researchers repurposed the current single object classifier to perform multiple objects detection and came up with region proposal-plus-classification solutions to change the new task to familiar single object classification.Many models were proposed for object detection that utilize convolutional neural networks .[34]

3.4.1 R-CNN

In 2012, Krizhevsky et al. [35] achieved promising results with CNNs for the general image classification task In 2013, Girshick et al. published a method [29] generalizing these results to object detection. This method is called R-CNN .

R-CNN models first select several proposed regions from an image and then label their categories and bounding boxes . Then, they use a CNN to perform forward computation to extract features from each proposed area. Afterwards, R-CNN use the features of each proposed region to predict their categories and bounding boxes.

The method is trained in multiple stages, beginning with the convolutional network [54]. After the CNN has been trained, the SVMs are fitted to the CNN features. Finally, the region proposal generating method is trained

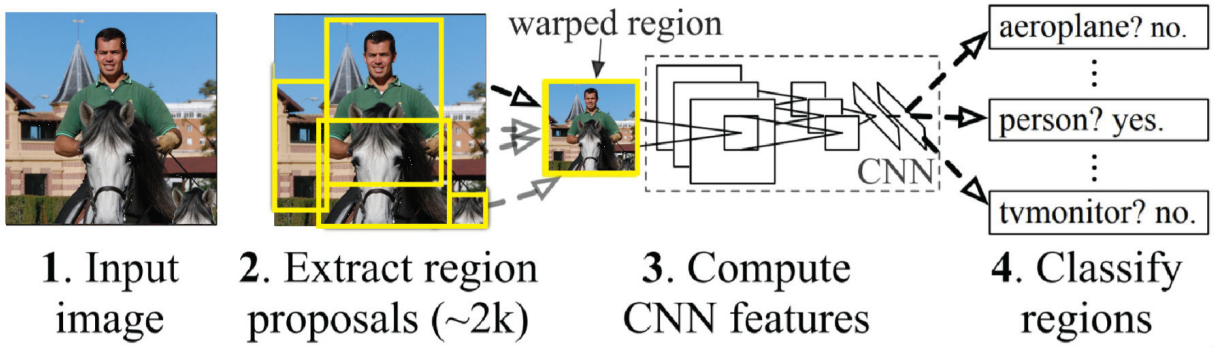

Figure 3.1: Stages of R-CNN forward computation .

3.4.2 Fast R-CNN

Fast R-CNN [36] published in 2015 by Girshick provides a more practical method for object recognition. The main idea is to perform the forward pass of the CNN for the entire image, instead of performing it separately for each RoI.

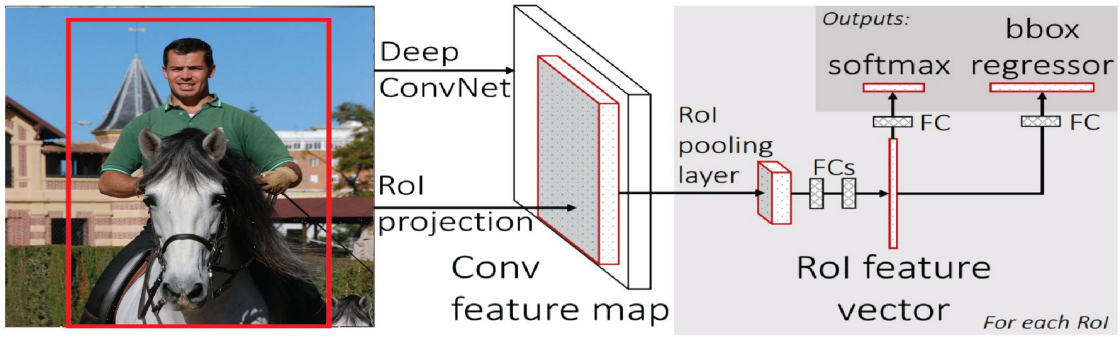

Figure 3.2: Stages of Fast R-CNN forward computation .

The general structure of Fast R-CNN is illustrated in the figure. The method receives as input an image plus regions of interest computed from the image. As in R-CNN, the RoIs are generated using an external method. The image is processed using a CNN that includes several convolutional and max pooling layers

the network processes the entire image with multiple convolutional and pooling layers producing a convolutional feature map. This first operation

is one of the gains, in terms of velocity, that Fast R-CNN achieves when compared with R-CNN since instead of running a CNN for each region of interest, it runs a single CNN for the entire image producing at the end the mentioned feature map . the second part of the framework begins, where for each object proposal a RoI pooling layer, using max pooling, gets a small fixed size vector from the feature map and then mapped to a feature vector by a sequence of fully connected layers that finally splits into two output vectors per RoI: one for the classifier (usually softmax) to estimates the probability of each object class, and other with the bounding box regressor that outputs the coordinates for each object class .

3.4.3 Faster R-CNN

Faster R-CNN [30] by Ren et al. is an integrated method. The main idea is to use shared convolutional layers for region proposal generation and for detection.

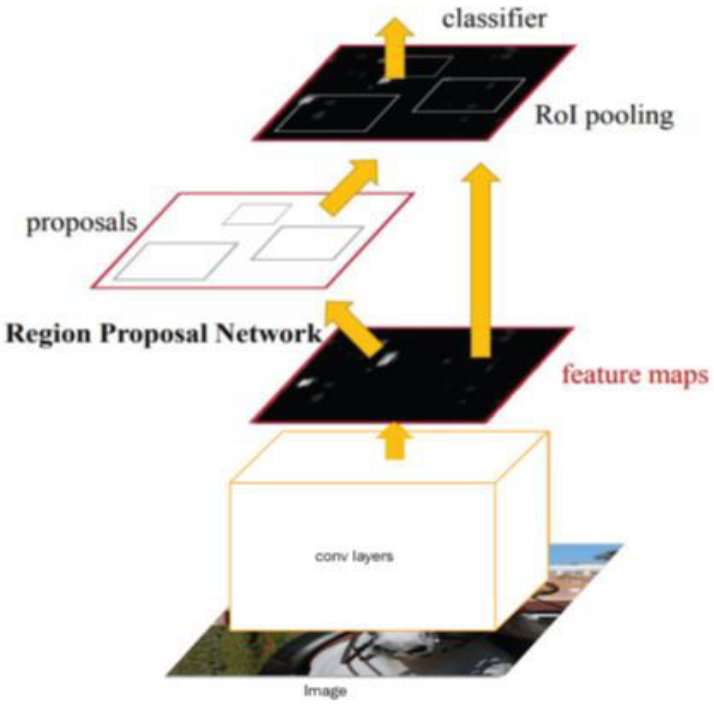

Figure 3.3: Architcture of Faster R-CNN .

Faster R-CNN has two networks: region proposal network (RPN) for

generating region proposals and a network using these proposals to detect objects. The main different here with Fast R-CNN is that the later uses selective search to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, when RPN shares the most computation with the object detection network. Briefly, RPN ranks region boxes (called anchors) and proposes the ones most likely containing objects.

3.4.4 SSD

The Single Shot MultiBox Detector [37] (SSD) takes integrated detection even further. The method does not generate proposals at all, nor does it involve any resampling of image segments. It generates object detections using a single pass of a convolutional network.

The SSD approach is based on a feed-forward convolutional network that produces a fixed-size collection of bounding boxes and scores for the presence of object class instances in those boxes, followed by a non-maximum suppression step to produce the final detections. The early network layers are based on a standard architecture used for high quality image classification (truncated before any classification layers) .

3.4.5 YOLO

You only look once (YOLO) is an object detection system targeted for real-time processing uses deep learning and convolutional neural networks (CNN) for object detection, it stands out from its "competitors" because, as the name indicates it only needs to "see" each image once. This allows YOLO to be one of the fastest detection algorithms .

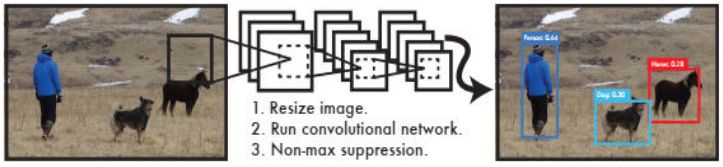

Figure 3.4: The YOLO Detection System .

This figure above show that the Processing images with YOLO is simple and straightforward. the system 1 resizes the input image to 448 x 448, 2 runs a single convolutional network on the image, and 3 thresholds the resulting detections by the model's confidence.

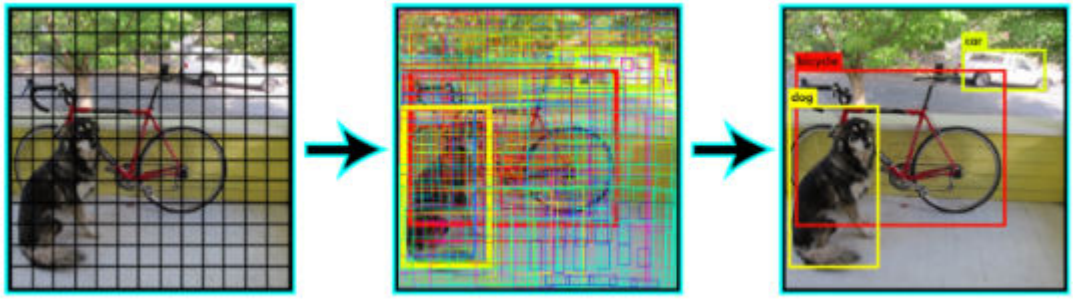

Figure 3.5: Object detection with YOLO .

In figure 3.5 the image is divided in a grid of SxS (left image). Each one of the cells will predict N possible "bounding boxes" and the level of certainty (or probability) of each one of them (image at the center), this means SxSxN boxes are calculated. The vast majority of these boxes will have a very low probability, that's why the algorithm proceeds to delete the boxes that are below a certain threshold of minimum probability. The remaining boxes are passed through a "non-max suppression" that will eliminate possible duplicate objects and thus only leave the most exact of them (image on the right).

3.4.5.0.1 Network Design :

YOLO model implement as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset [38]. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinate

YOLO architecture is inspired by the GoogLeNet model for image classification [39]. His network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, YOLO simply use 1 x 1 reduction layers followed by 3 x 3 convolutional layers, similar to Lin et al [40].

3.4.5.0.2 YOLO Loss Function :

YOLO uses ones single loss function for both bounding box and the classification of the object. The loss function is:

$$loss = \lambda coord \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{obj}^{ij} (x_i - \bar{x_i})^2 + (y_i - \bar{y_i})^2 \quad (1)$$

$$+\lambda coord \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{obj}^{ij} (\sqrt{w_i} - \sqrt{\bar{w_i}})^2 + (\sqrt{h_i} - \sqrt{\bar{h_i}})^2 \quad (2)$$

$$+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{obj}^{ij} (c_i - \bar{c_i})^2 \quad (3)$$

$$+\lambda noobj \sum_{i=0}^{s^2} \sum_{j=0}^{B} 1_{noobj}^{ij} (c_i - \bar{c_i})^2 \quad (4)$$

$$+ \sum_{i=0}^{s^2} 1_{obj}^{i} \sum_{c \varepsilon classes} (p_i(c) - \bar{p_i}(c))^2 \quad (5)$$

The loss function can be parsed into 5 parts, where parts (1) and (2) are focusing on the loss of the bounding box coordinates, parts (3) and (4) are penalizing the differences in confidence of having an object in the grid and part (5) is penalizing for the difference in class probability. It is interesting to note that the loss function for the bounding box size is based on the square root of the dimensions. This is used to address that the small deviations in larger bounding boxes should incur less of a penalty than in smaller bounding boxes. We also note that there are $\lambda coord$ and $\lambda noobj$, which are set to be 5 and 0.5 by the original authors. The $\lambda coord$ hyper-parameter is set to ensure "fair" contribution of the bounding box location penalty and the classification penalty to the overall loss function, and the $\lambda noobj$ is set to penalize less for the confidence of identifying an object when there is not one.

3.5 Classical Object Recognition

Object recognition has its foundations on computer vision history around the years of the 1970s where the pioneers of artificial intelligence and robotics

viewed this as an ambitious challenge that could at the end achieve another huge step to replicate the human intelligence and behavior and endow the robots with it [41].

It can be defined as the task of discovering a certain object in an image or even in a video sequence. It is a fundamental vision problem since unlike humans that can detect and identify with almost no effort a huge range of objects in images or videos that might diverge from the viewpoint, color, size or even when the object it is partially obstructed this task continues to be a real challenge for object recognition engines [42].

3.6 Object recognition system

The problematic of recognizing an object in an image is defined as a labeling problembased on models of known objects. Essentially, given a generic image, that contains the objects of interest and a set of labels corresponding to a set of models available in the system, the system may be capable to assign properly the labels to the respective regions in the image [41].

In computer vision the way it recognizes what objects are presented in a certain image is not a linear process, there are multiple techniques, however, the generic system may be represented as shown in Figure [42].

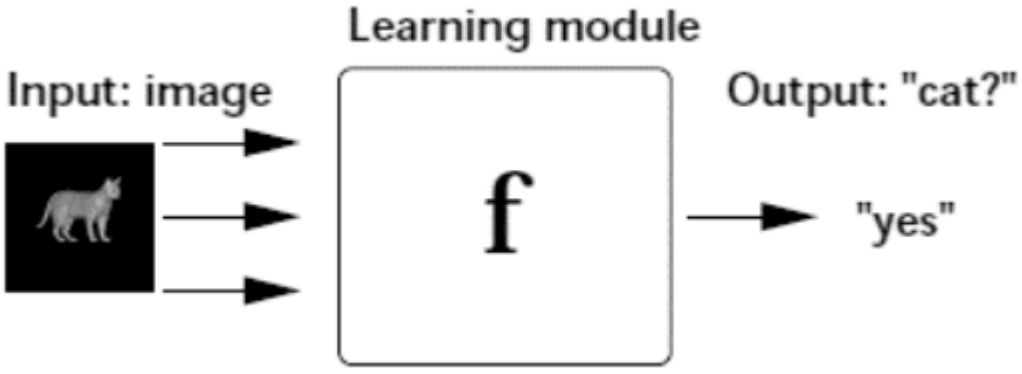

Figure 3.6: Object recognition system .

In the previous figure, we have the basic architecture of an object recognition system. Essentially, the learning module is trained with a set of ex-

amples, corresponding to images previously labeled and can be described as a binary classifier. Based on an input image it retrieves as an output "yes" or "no", for either the respective class of the object, like cat or dog or for the individual identity where the image belongs, for example, a face [42].

3.7 Neural network for object recognition

Object recognition has been studied for over four decades [43]. "Neocognition" [44] was an early biologically-inspired shift-invariant model for object recognition but lacked a supervised training process. LeCun et al. [45] introduced gradient descent through backpropagation on convolution neural networks, which makes the training process much more effective than previous models. CNNs were heavily used in object recognition in the 1990s, but then fell out of fashion with the rise of support vector machines. In 2012, AlexNet [46], a deep neural network with CNN, won the object classification competition ILSVRC 2012, and kindled research on deep convolutional neural networks.

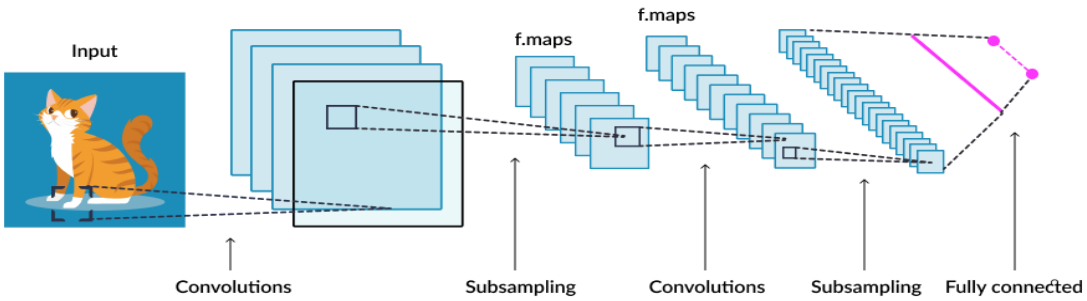

Figure 3.7: Neural network architecture for object recognition .

Figure 3.7 shows a canonical neural network architecture for object recognition. While max-pooling stages and fully-connected (FC) layers have been considered indispensable components, an all-convolutional network [43] that replaces max-pooling with convolutions has competitive or state-of-the-art performance. Further, global average pooling [47] is used to eliminate FC layers on top of the convolutional layers .

3.8 Transfer Learning

To train a convolutional neural network is very expensive: the more layers stack up, the higher the number of convolutions and parameters to be optimized.The computer must be able to store several gigabytes of data and efficiently perform the calculations.That's why hardware manufacturers are stepping up their efforts to provide high-performance graphics processors (GPUs) that can quickly drive a deep neural network by paralleling calculations[48].

Transfer Learning allows you to do Deep Learning without having to spend a month of calculations. The principle is to use the knowledge acquired by a neural network when solving a problem in order to solve another more or less similar. This provides a transfer of knowledge[49].

In addition to accelerating the training of the network, Transfer Learning makes it possible to avoid overfitting [52] . Indeed, when the collection of input images is small, it is strongly disadvised to train the neural network starting from zero (that is to say with a random initialization): the number of parameters to be learned being much greater than the number of images.

Transfer Learning is a technique that is widely used in practice and simple to implement. It requires having a network of neurons already trained, preferably on a problem close to the one we want to solve. Nowadays, we can easily retrieve one from the Internet, especially in Deep Learning libraries like Keras.[49][51]

We can exploit the pre-trained neural network in a number of ways, depending on the size of the input dataset and its similarity to that used in pre-training.

3.8.1 Fine-tuning total :

The last fully-connected layer of the pre-trained network is replaced by a classifier adapted to the new problem (SVM, logistic regression ...) and randomly initialized. All layers are then dragged onto the new images.

This method should be used when the new image collection is large: in this case, one can afford to drive the entire network without running the risk of overfitting. In addition, since the parameters of all the layers (except the last) are initially those of the pre-trained network, the learning phase will be

done more quickly than if the initialization had been random.

3.8.2 Feature extraction :

This method is to use the features of the pre-trained network to represent the images of the new problem. For this purpose, the last fully-connected layer is removed and all other parameters are fixed. This truncated network will calculate the representation of each input image from the features already learned during the pre-training. We then train a classifier, randomly initialized, on these representations to solve the new problem.

This method should be used when the new image collection is small and similar to the pre-workout images. Indeed, training the network on as few images is dangerous since the risk of overfitting is important. In addition, if the new images look like the old ones, then they can be represented by the same features.

3.8.3 Partial fine tuning :

This is a mix of the first and the second method, the last fully-connected layer is replaced again by the new classifier randomly initialized, and the parameters of some layers of the pre-trained network are fixed. Thus, in addition to the classifier, the new images are dragged onto the non-fixed layers, which generally correspond to the highest levels of the network.

This method is used when the new image collection is small but very different from the pre-workout images. On the one hand, as there are few training images, the method one which involves driving the entire network is not feasible because of the risk of overfitting.

On the other hand, we also eliminate the second method since the new images have very little in common with the old ones: using the features of the pre-trained network to represent them is not a good idea. But the features of the lower layers are simple and generic (so can be found in two very different images), while those of the upper layers are complex and specific to the problem. Thus, the method of fixing the lower layers and driving the classifier and the high layers is a good compromise.

3.9 VGG16 Architecture

The VGG network architecture was initially proposed by Simonyan and Zisserman [27]. The VGG models with 16 layers (VGG16) and with 19 layers (VGG19) were the basis of their ImageNet Challenge 2014 submission, where the Visual Geometry Group (VGG) team secured the first and the second places in the localization and classification tracks respectively.

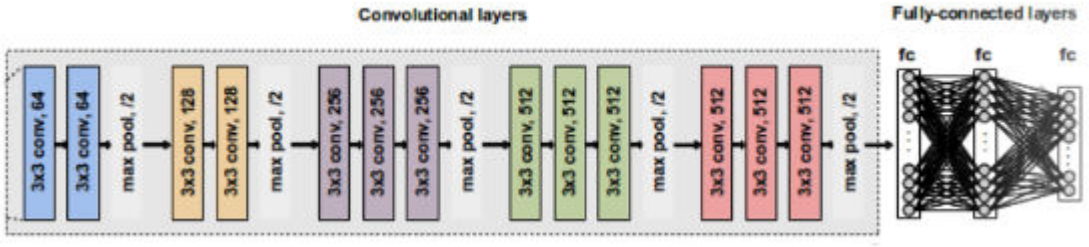

Figure 3.8: Architecture of VGG16 .

The VGG16 architecture, shown in the Figure above is structured starting with five blocks of convolutional layers followed by three fully-connected layers. Convolutional layers use 3*3 kernels with a stride of 1 and padding of 1 to ensure that each activation map retains the same spatial dimensions as the previous layer. A rectified linear unit (ReLU) activation is performed right after each convolution and a max pooling operation is used at the end of each block to reduce the spatial dimension. Max pooling layers use 2*2 kernels with a stride of 2 and no padding to ensure that each spatial dimension of the activation map from the previous layer is halved. Two fully-connected layers with 4,096 ReLU activated units are then used before the final 1,000 fully-connected softmax layer.

3.10 Advantages of transfer learning

We utilize knowledge from source models to improve learning in the target task [52]. Apart from providing capabilities to reuse already-built models, transfer learning may assist learning the target task in the following ways:

Improved baseline performance :

When we augment the knowledge of an isolated learner (also known as an ignorant learner) with knowledge from a source model, the baseline performance might improve due to this knowledge transfer.

Model-development time :

Model-development time: Utilizing knowledge from a source model might also help in fully learning the target task, as compared to a target model that learns from scratch. This, in turn, results in improvements in the overall time taken to develop/learn a model.

Improved final performance :

Higher final performance might be attained by leveraging transfer learning.

3.11 Conclusion

In this chapter we have presented objecte detection different technics and models with their architecture . We have seen the important concepts of object recognition , also we described transfer learning and seen his different models . In this work we used convolutional neural networks CNN architecture in aerial image detection .We will see it in the next chapter with detail .

Chapter 4

Implimentaiton And Result

4.1 Introduction

In this chapter, we will define the architecture of the model we have created, then we will apply this model on a data set. For this, we used Python, C, OpenCV and other library to make our work more flexible .

4.2 System configuration

For the best result we used a powerfull configuration which is :

- Prossesor: Intel(R) Core (TM) I7-3612QM CPU @ 2.10 GHz x 4.
- RAM: 6 GO.
- Operating System : Linux Debian 9

4.3 Work environmrnt

4.3.1 Python

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms [53].

4.3.2 C

C is a general-purpose programming language, and is used for writing programs in many different domains, such as operating systems, numerical computing, graphical applications, etc. It is a small language, with just 32 keywords . It provides "high-level" structuredprogramming constructs such as statement grouping, decision making, and looping, as well as "lowlevel" capabilities such as the ability to manipulate bytes and addresses[54] .

4.3.3 Cypthon

Cython is an extension to the Python language that allows explicit type declarations and is compiled directly to C. This addresses Python's large overhead for numerical loops and the difficulty of efficiently making use of existing C and Fortran code, which Cython code can interact with natively. The Cython language combines the speed of C with the power and simplicity of the Python language[55] .

4.3.4 Anaconda

The open source version of Anaconda is an easy-to-install high performance Python and R distribution with a package manager, environment manager and collection of 720+ open source packages with free community support[56].

4.3.5 Spyder

Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging and introspection features[56].

4.3.6 OpenCV

OpenCV is a state of the art computer vision library that is used for a variety of image and video processing operations. Some of the more spectacular and futuristic features, such as face recognition or object tracking, are easily achievable with OpenCV 3. Learning the basic concepts behind computer vision algorithms, models, and OpenCV's API will enable the development of all sorts of real-world applications, including security and surveillance tools[57] .

4.3.7 TensorFlow

TensorFlow is currently the leading open-source software for deep learning, used by a rapidly growing number of practitioners working on computer vision, Natural Language Processing (NLP), speech recognition, and general predictive analytics. This book is an end-to-end guide to TensorFlow designed for data scientists, engineers, students and researchers[58].

4.4 Case study

Aerial image can be a solution to many problems , one of this problems is the car traffic jams, this model can solve this problem notice other drivers so they can avoid it, other problems is to track a car in the road , police can find some problems to track a car in the road beceause to traffic ,this model can help the police to track any specific car all over the city using drone .

4.5 Our model

We used transfer learning to get pre trained model from darknet YOLOv3 , this model is so powerful compared to other models especially to use ability to detect small size object with big accuracy and small time ,And this is the original architecture of the darknet YOLOV3:

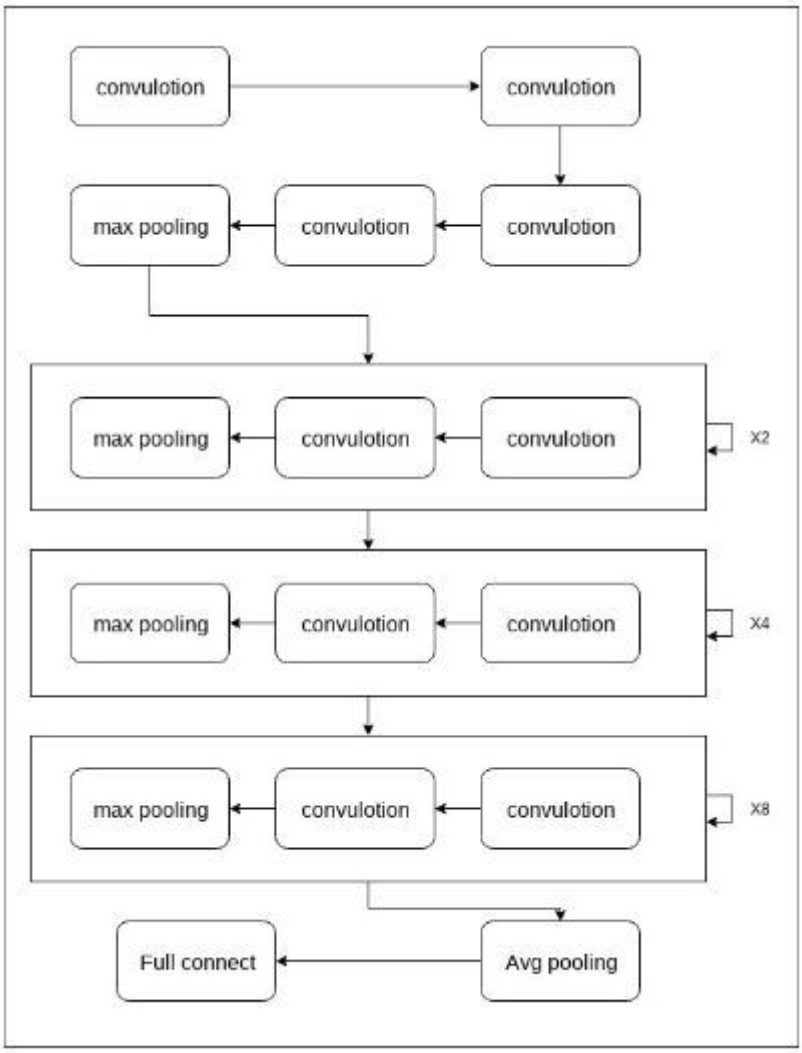

Figure 4.1: architcture of YOLOv3

In our architectute , after we get that model we changed the architecture of original model to be compatible with aerial image :

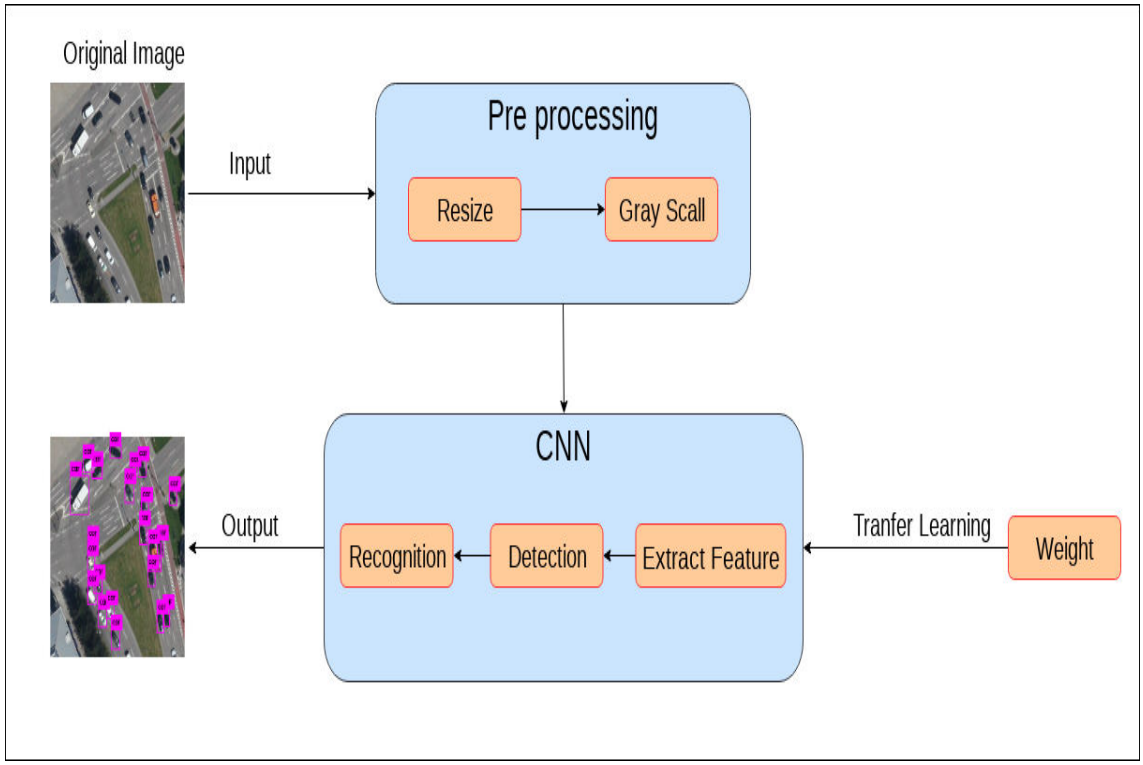

Figure 4.2: architcture of our model

Our model compose of many convolution layers and pooling layers and finally we end it with full connected layer , the first step which is pre processing is to resize the input image to 416x416 so it can go to our model.

Figure 4.3: Resize the input image

After resizing the image we will do gray scal to detect edge and make the image easy to recognize:

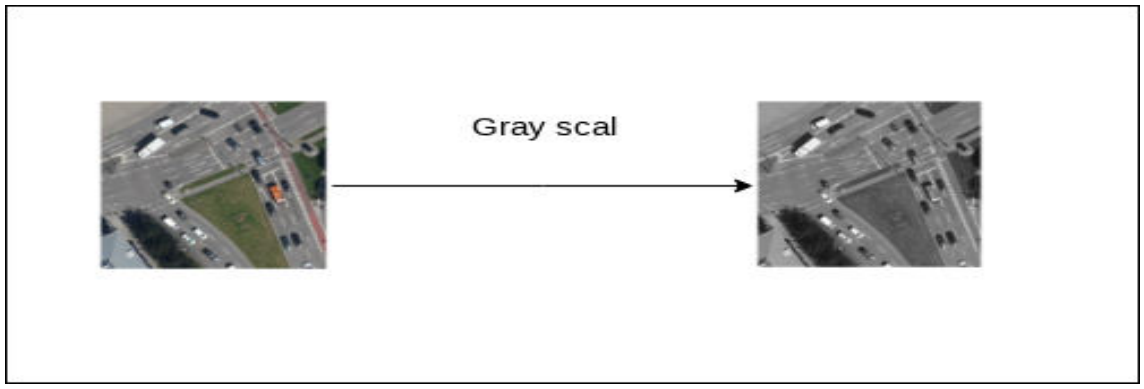

Figure 4.4: applying gray scall on image

After gray scall the image we put it to CNN:

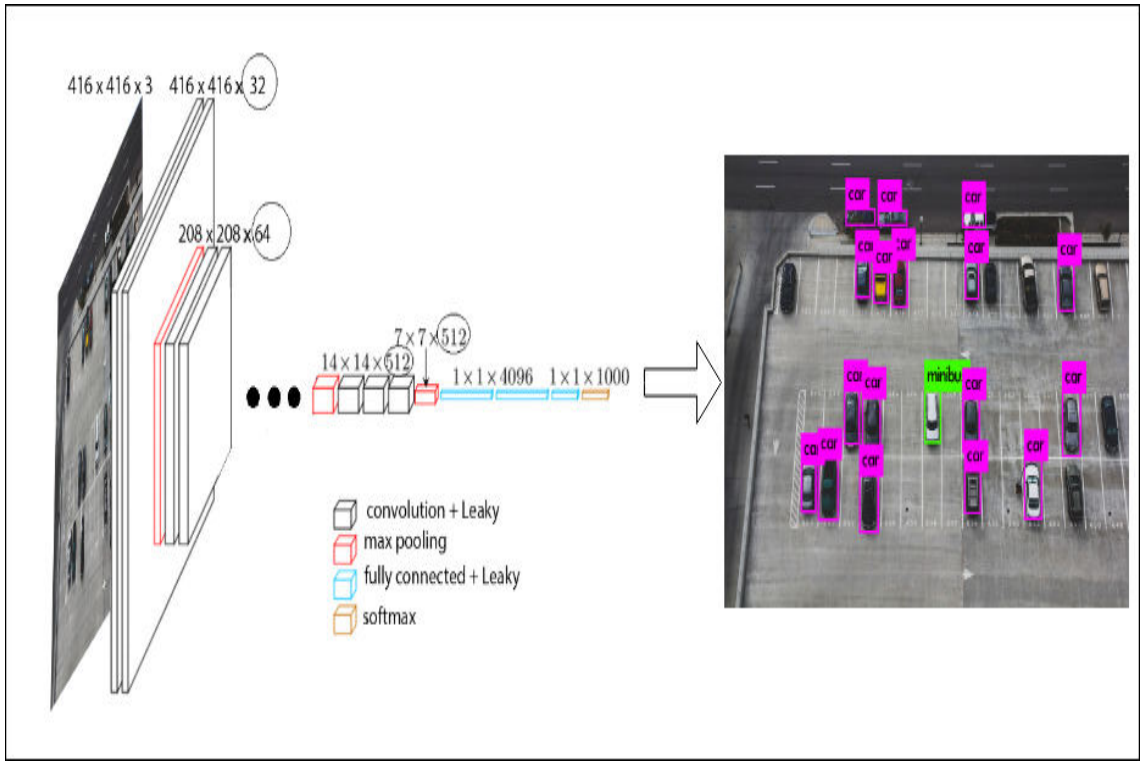

Figure 4.5: architecture of our convolutional neural network

Convolution layer : After resize the image we will put it in the first convolutional layer which contain 32 feature maps (kernels) with size of 3x3 matrix for each feature map and it end up with a leaky activation function after the convolution layer finish the job

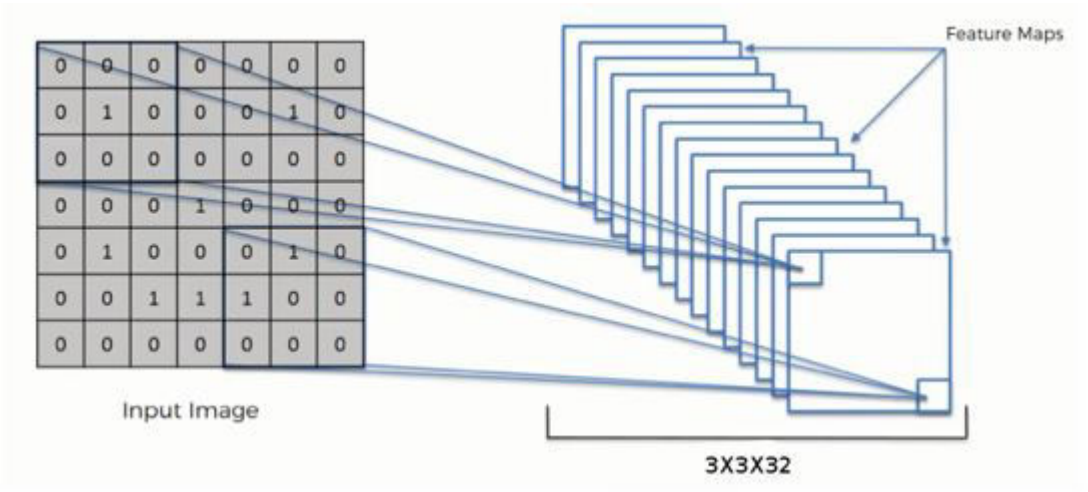

Figure 4.6: Convolution layer

Max Pooling layer: After serie of convulation layer we get to the max pooling layer which reduce the size of the image to make it easy and fast for training the model , we use 2x2 matrix for maxing pooling which mean it will only take 25 % of the image .

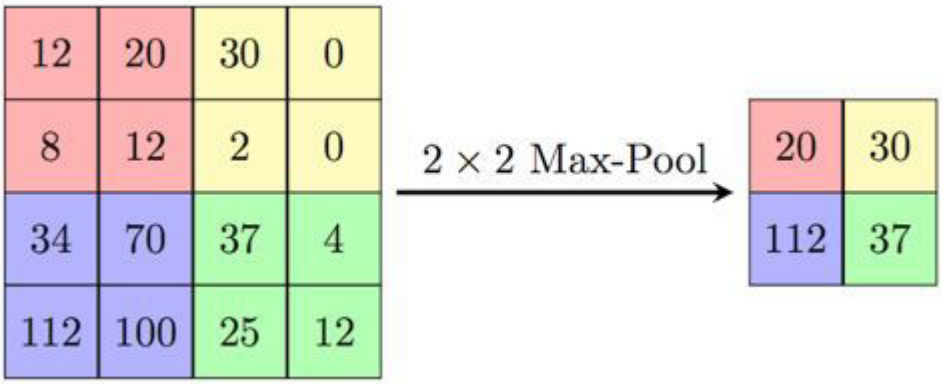

Figure 4.7: Max Pooling layer

Full Connection layer: This is the last layer in our architecture which will implement the out of the previous layer to the neural network with many hidden layers to select the right weight for each neural and finaly we will use softmax activation function for the out put .

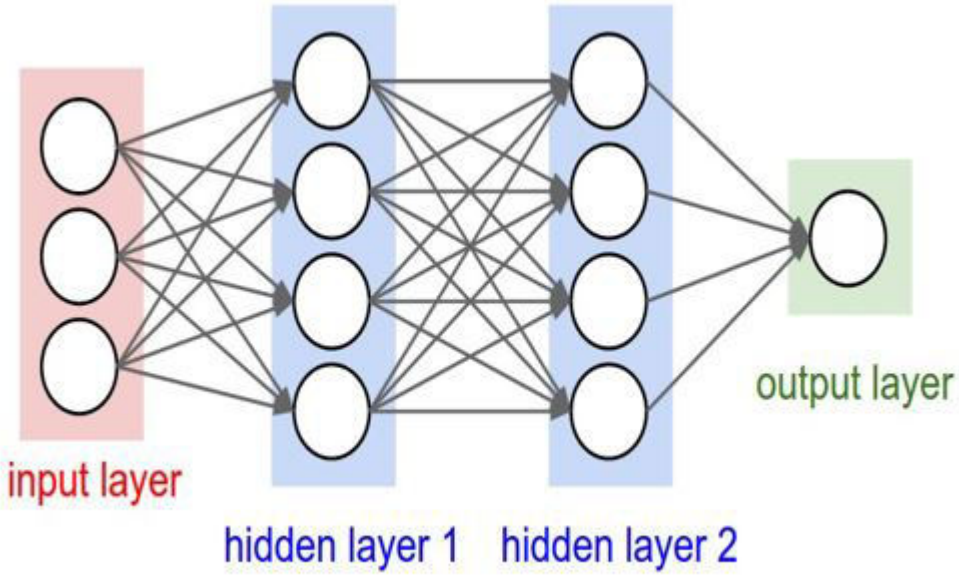

Figure 4.8: Full Connection layer

The next figures present the screenshot of our architecture :

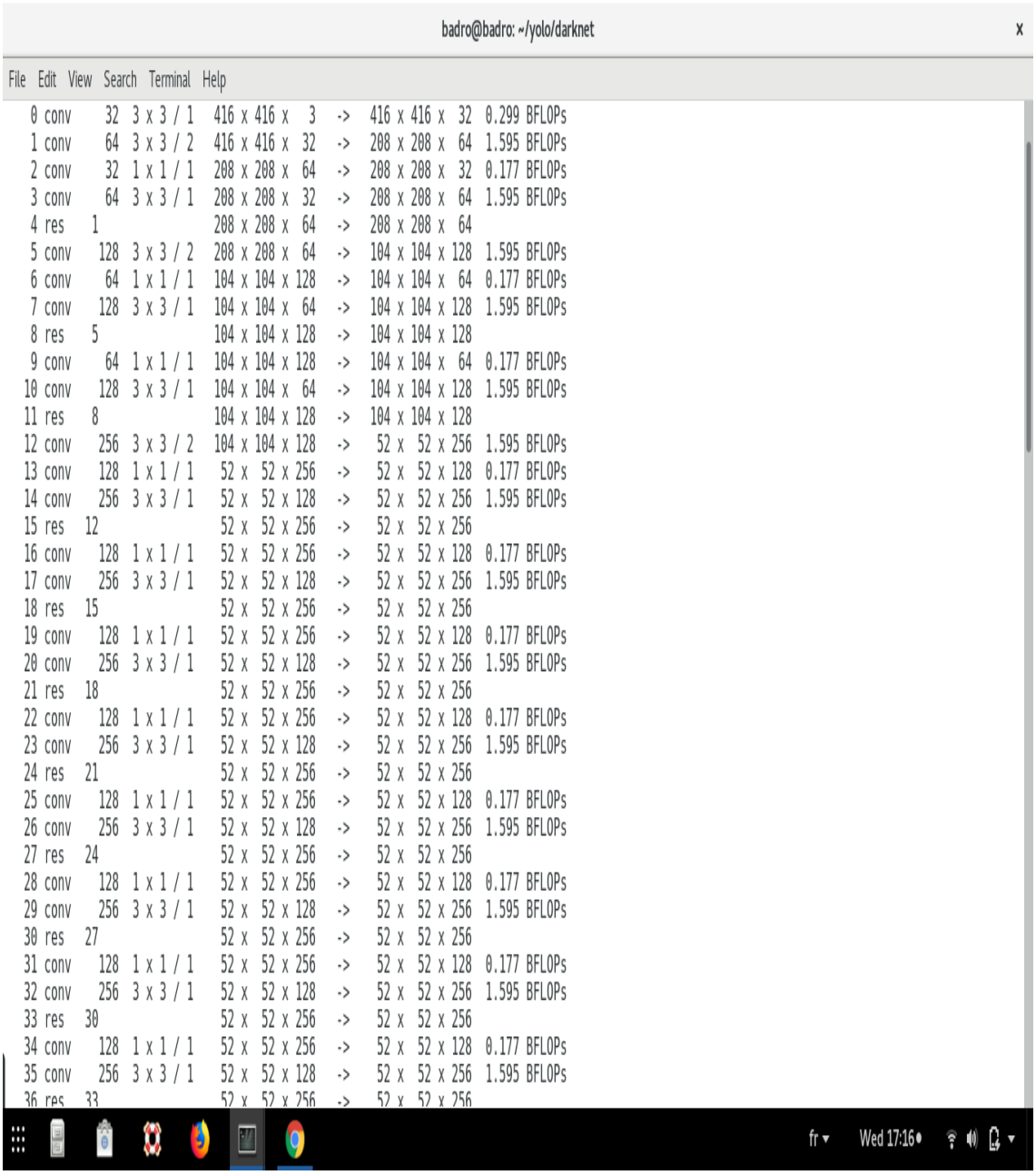

Figure 4.9: Our architecture part1.

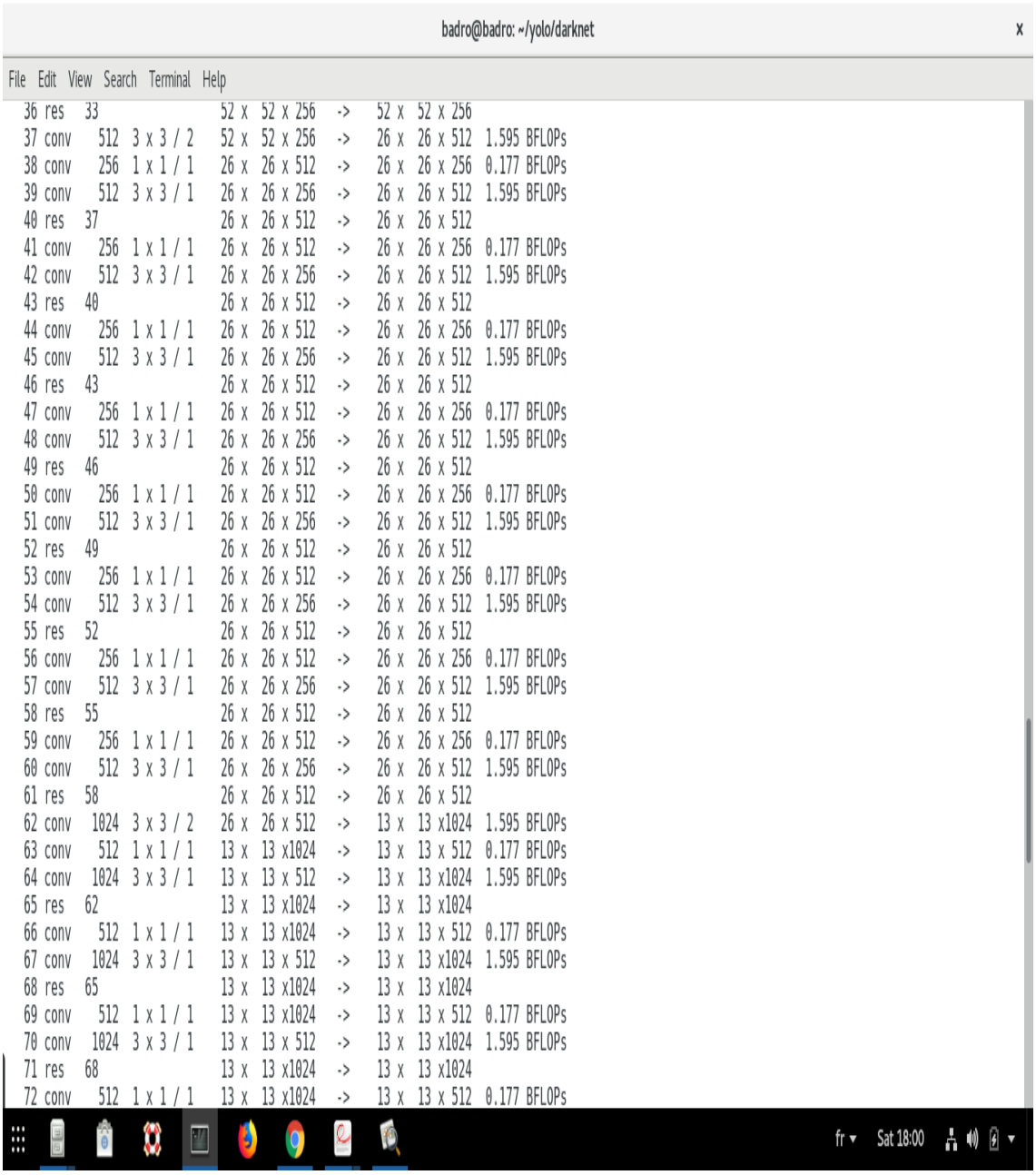

Figure 4.10: Our architecture part2 .

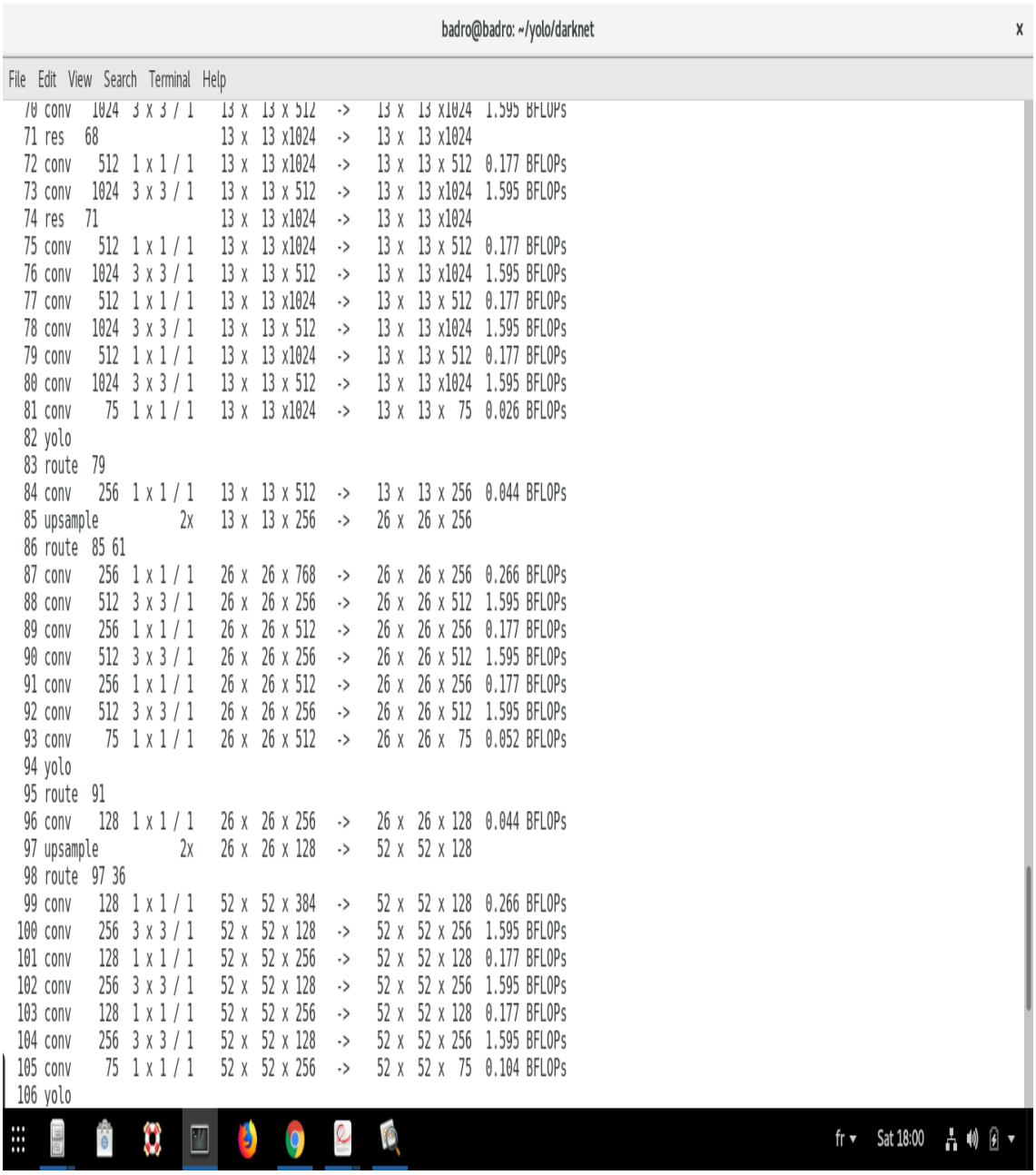

Figure 4.11: Our architecture part3.

4.6 Results

For the results we chose different image with different views .

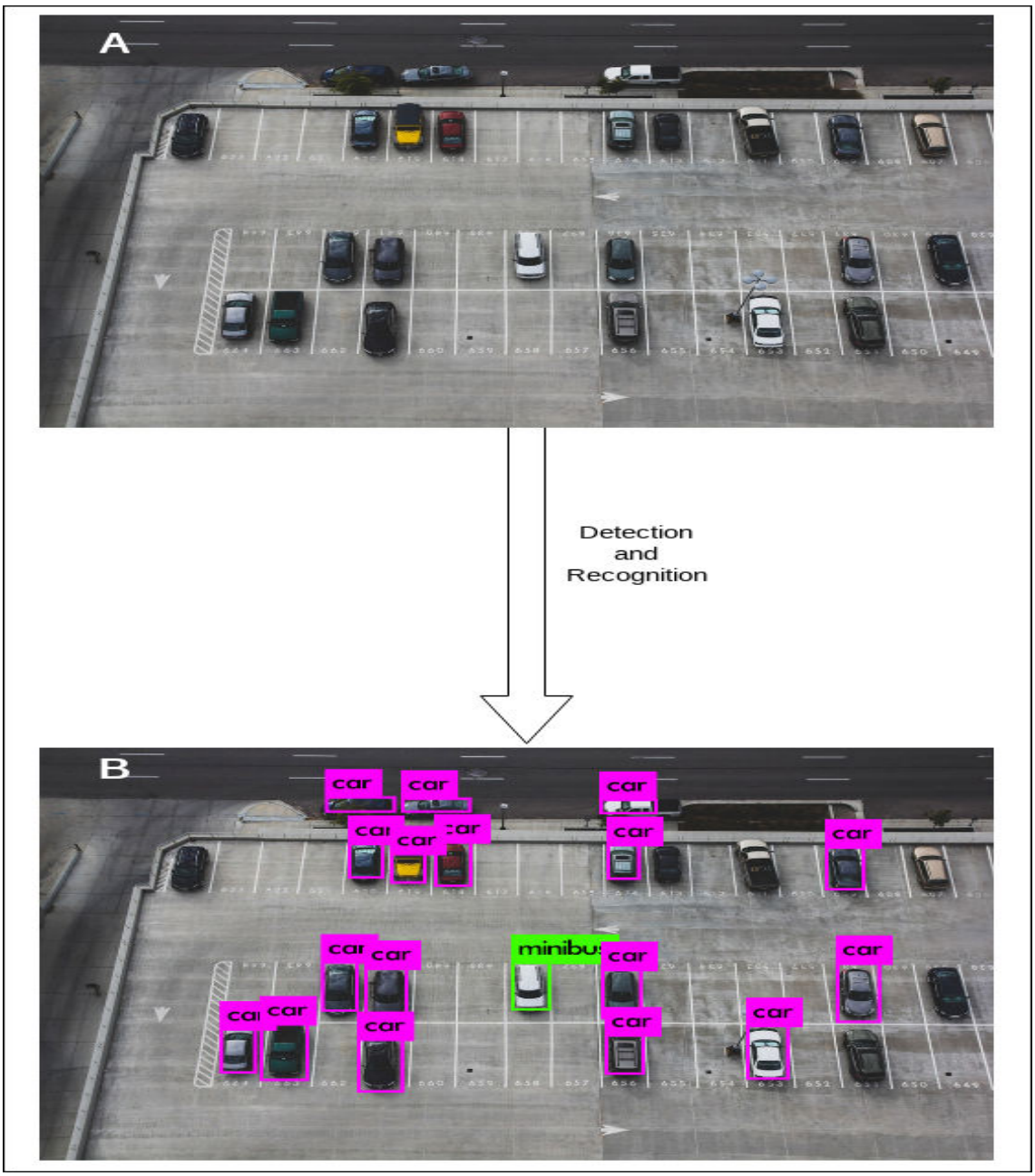

Figure 4.12: (A) Before and (B) after the detection.

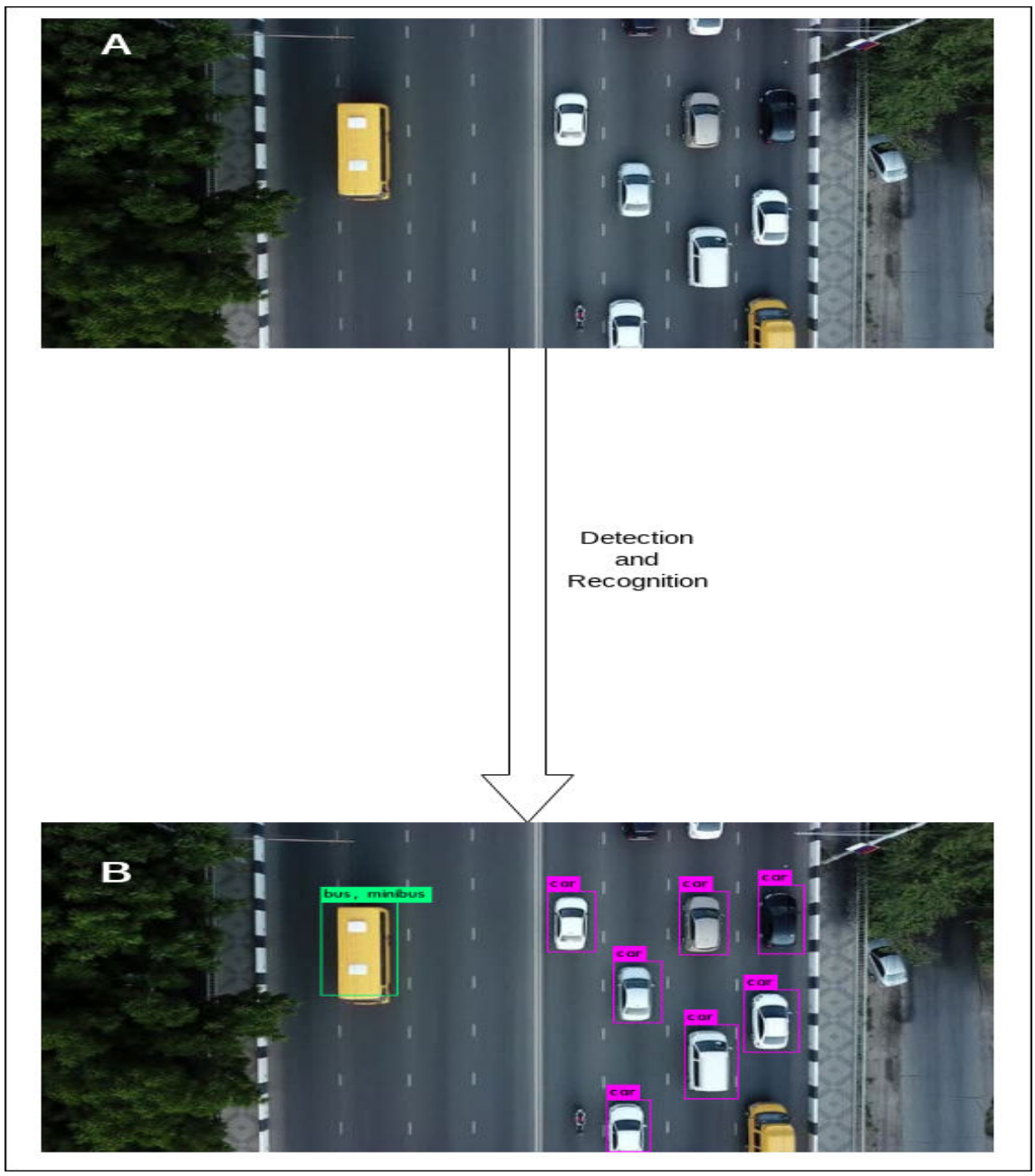

Figure 4.13: Before and after the detection.

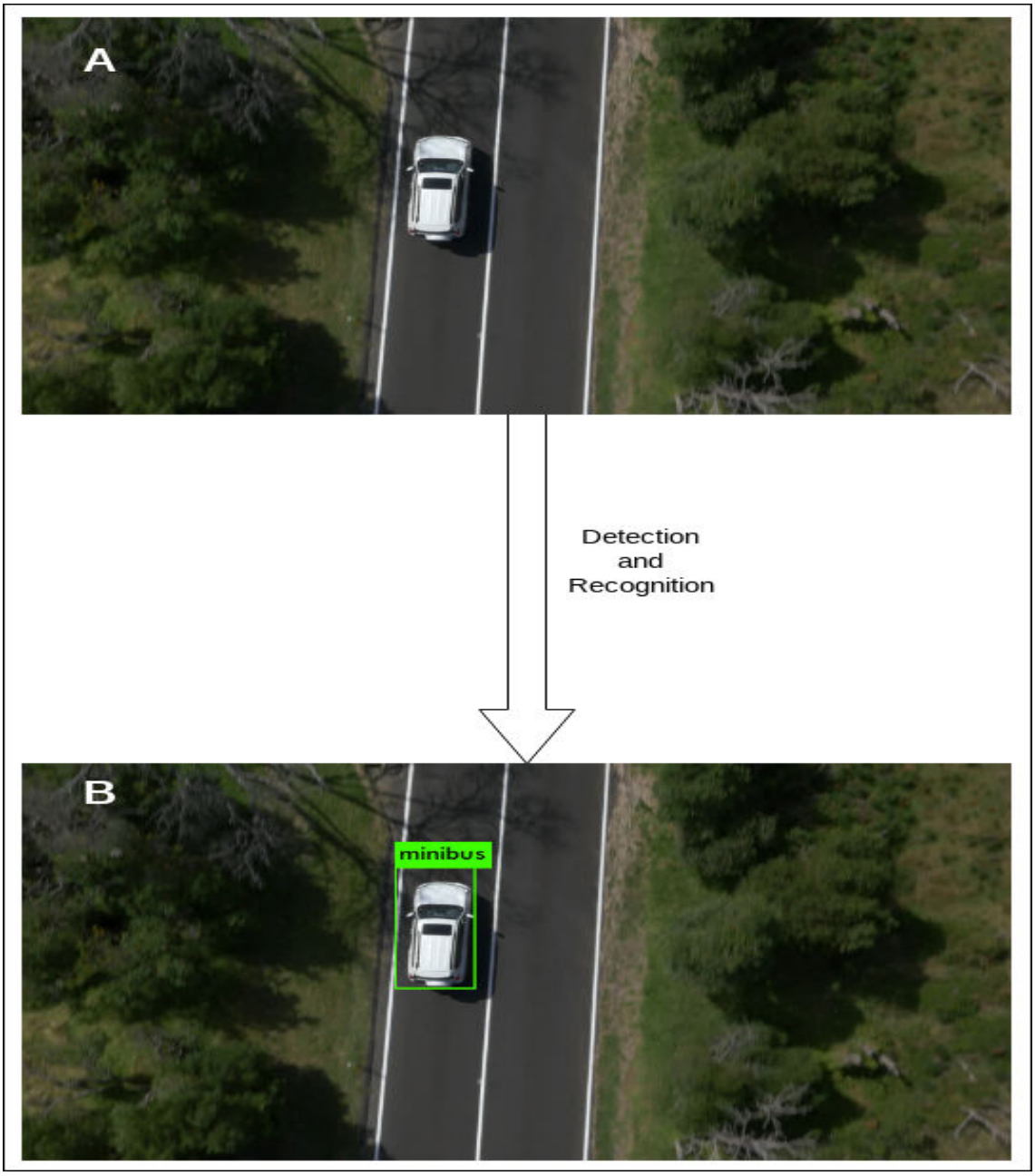

Figure 4.14: Before and after the detection .

This shot was taking by a algerian drone :

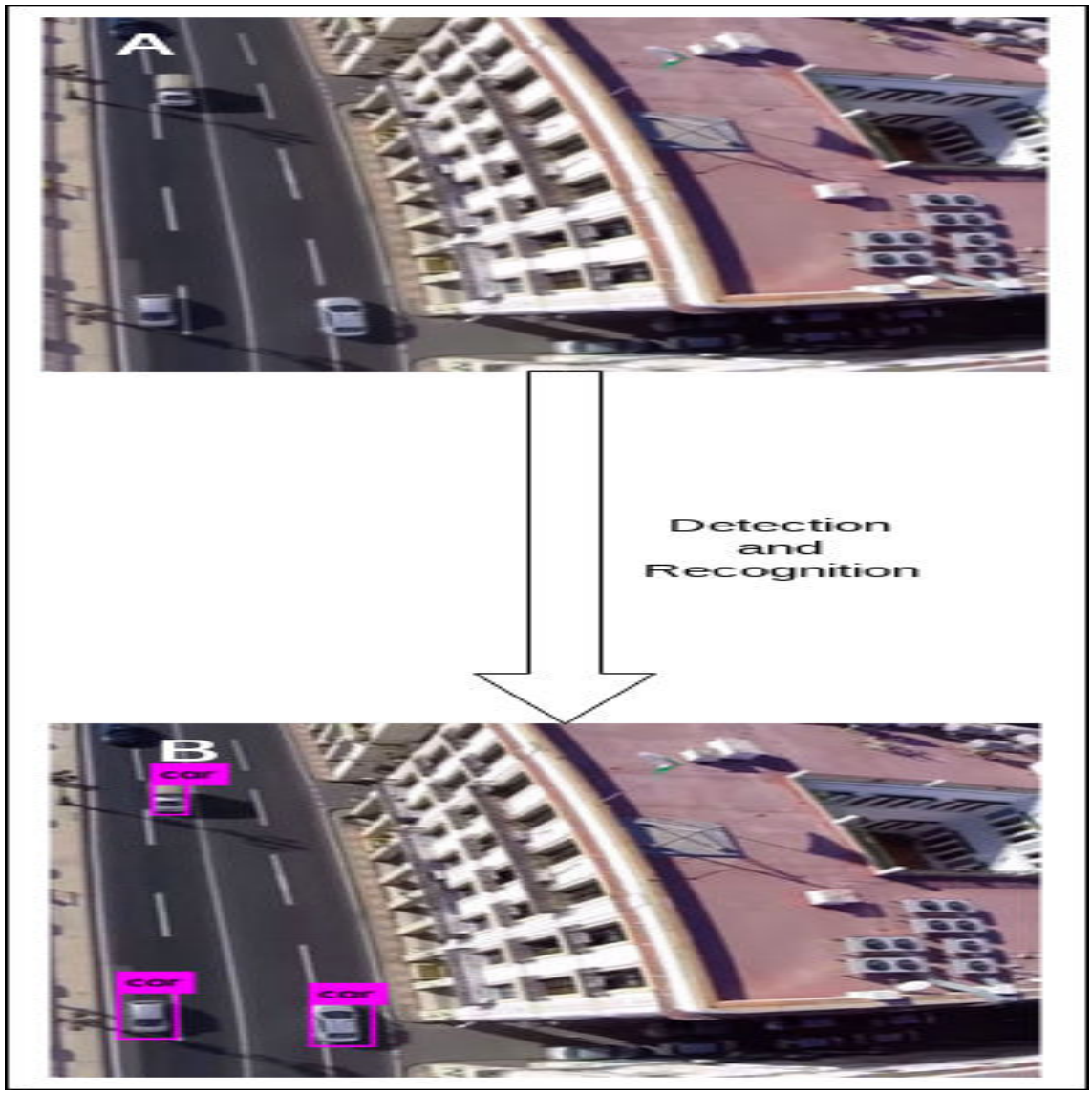

Figure 4.15: Before and after detection .

This is a low light shot from algerian drone :

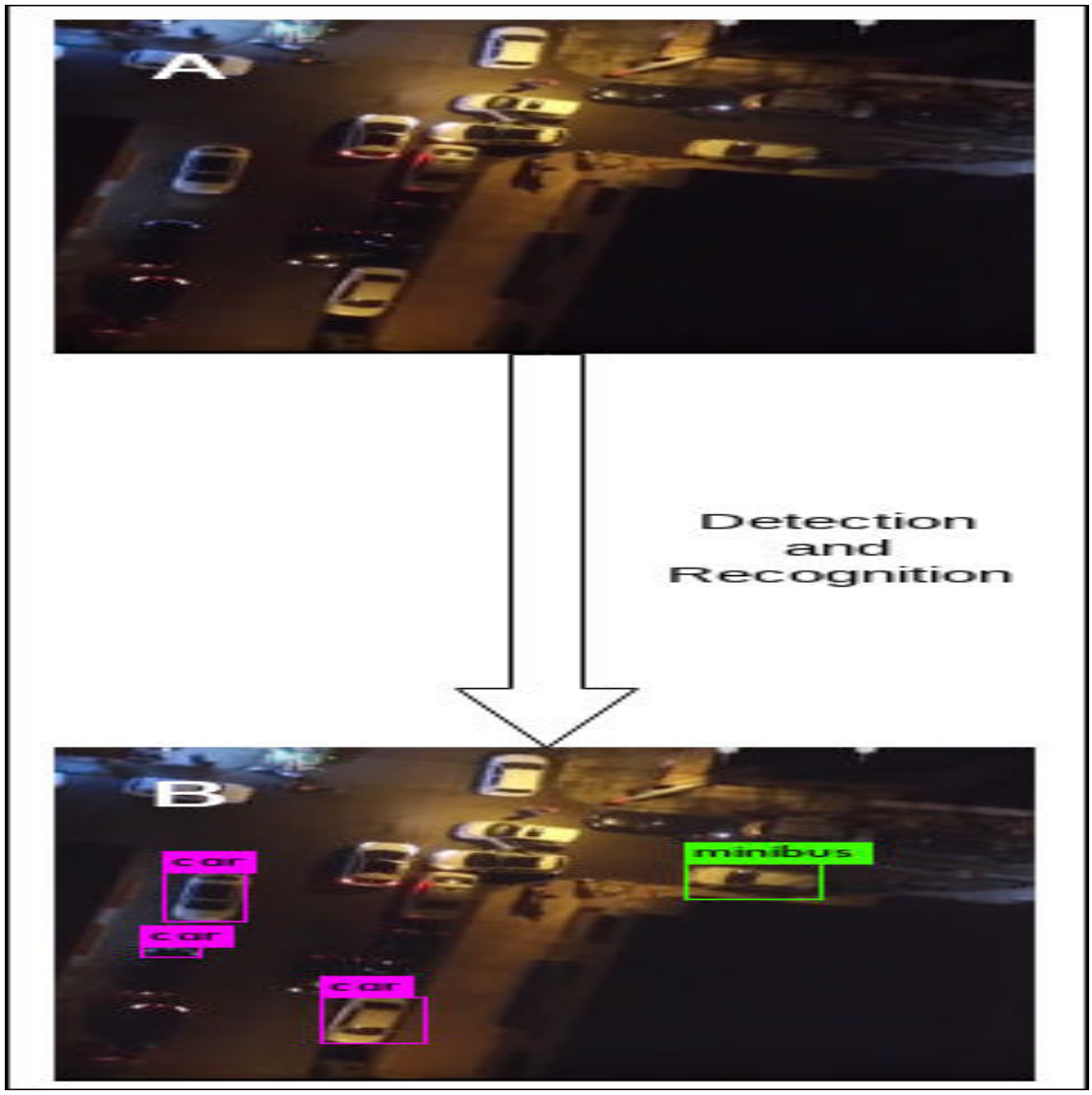

Figure 4.16: Before and after detection .

4.7 Limits

This dataset contain many types of cars but not all of them , there may be some types in our country that are not in this dataset and this may affect the detection and recognition.

The number of class are limited to 3 classes only (car , minibus,bus) ,it can not detect bicycle ,human or other type of objects .

The height of the image and blurred image may affect the detection of the objects and make it very hard .

4.8 Conclusion

In this chapter we have presented our model with it architecture, starting from the orginal image and pre processing traitment to the CNN layers ,after that we presented the results of our model in difference views and shoots from helicopter and drone and it gave us a satisfying results with good accuracy in difference conditions ,high light like figure 4.15 and low light in figure 4.16 .

Conclusion and perspectives

In this project we discussed the main concepts of machine and deep learning . We introduced convolutional neural networks (CNN) by presenting the different types of layers used in the classification .We also talked about concepts and technique of object detection and recognition ,then we presented transfer learning architecture and described his different models .

In this work a vehicle detection method based on YOLO deep learning algorithm for aerial image is presented . this methode aplly convolutional neural network architceture (CNN) for extraction features . We had a good test results especially for small objects, for Images were taken at night, as well and meets the real-time requirements by Transfer learning . As future work the performance could be further improved by taking advantage of the immense computing power of the GPU graphics processors by distributing the calculations on several GPUs.

Bibliography

[1] E. Charniak, D. McDermott , Addison-Wesley [*Introduction to Artificial Intelligence*]. 1985

[2] Stuart J. Russell and Peter Norvig [*Artificial Intelligence A Modern Approach Second Edition*]. 2002

[3] N.J. Nilsson , Morgan Kaufmann [*Artificial Intelligence: A New Synthesis*]. San Francisco, CA, 1998

[4] T. Dean, J. Allen, Y. Aloimonos [*Artificial Intelligence: Theory and Practice*]. New York, 1995.

[5] P.H. Winston , Addison-Wesley [*Artificial Intelligence, second ed*]. 1984

[6] Andriy Burkov [*The Hundred-Page Machine Learning Book*]. 2019

[7] Alex Smola and S.V.N. Vishwanathan [*Introduction to Machine Learning*]. 2010

[8] Trevor Hastie , Robert Tibshirani , Jerome Friedman [*The Elements of Statistical Learning Second Edition*]. 2017

[9] Aurélien Géron [*Hands-On Machine Learning with Scikit-Learn and TensorFlow First Edition*]. March 2017

[10] Andreas C. Müller, Sarah Guido [*Introduction to Machine Learning with Python*]. 2016

[11] Krzysztof Grabczewski [*Meta-Learning in Decision Tree Induction*]. 2013

[12] Jan Rauch [*Observational Calculi and Association Rules*]. 2012

[13] Richard E. Neapolitan [*Learning Bayesian networks*]. 2004

[14] Finn V. Jensen [*An introduction to Bayesian networks*]. 1996

[15] Yoshua Bengio Aaron Courville and Pascal Vincent [*Representation Learning: A Review and New Perspectives*]. 2014

[16] Simon S. Haykin [*Neural networks a comprehensive foundation 2nd edition*]. 1994

[17] David Kriesel [*A Brief Introduction to Neural Networks*]. 2007

[18] Rojas R [*Neural Networks - A Systematic Introduction*]. SpringerVerlag, Berlin, New-York, 1996

[19] Thomas Cambrai [*Comprendre l'intelligence artificielle : Introduction aux réseaux de neurones et au Deep Learning*]. 2019

[20] Y. Bengio, I. J. Goodfellow, and A. Courville [*Deep Learning (Adaptive Computation and Machine Learning series)*]. 2016

[21] Ian Goodfellow ,Yoshua Bengio [*L'apprentissage profond*]. octobre 2018

[22] Yoshua Bengio [*Deep Learning*]. novembre 2016

[23] Chollet F [*Deep Learning with Python*]. Manning Publications Company . 2017

[24] Aurélien Géron [*Deep Learning avec TensorFlow*]. Mise en oeuvre et cas concrets . 2017

[25] Jean-Claude Heudin [*Comprendre le Deep Learning: Une introduction aux réseaux de neurones*]. 2016

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich . [*Going deeper with convolutions*]. 2014

[27] K. Simonyan and A. Zisserman [*Very deep convolutional networks for largescale image recognition*]. 2014

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. [*Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*]. 2015

[29] A. Krizhevsky, I. Sutskever, and G. E. Hinton . [*Imagenet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems*]. Curran Associates, Inc., 2012.

[30] Ren, S., He, K., Girshick, R., and Sun, J . [*Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems*]. 2015

[31] Bishop, C. M [*Pattern Recognition and Machine Learning*]. (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[32] LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. [*Backpropagation applied to handwritten zip code recognition. Neural computation*]. 1989

[33] R. Girshick, J. Donahue, T. Darrell, and J. Malik . [*Rich feature hierarchies for accurate object detection and semantic segmentation*]. in Computer Vision and Pattern Recognition, 2014.

[34] Param Uttarwar [*Object Detection and Recognition Using Deep Learning in OpenCV*]. 2018

[35] Krizhevsky, A., Sutskever, I., and Hinton, G. E [*Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems*]. 2012

[36] N. Dalal and B. Triggs . [*Histograms of oriented gradients for human detection*]. in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on . 2005

[37] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. [*Ssd: Single shot multibox detector. In European Conference on Computer Vision*]. 2016

[38] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. [*The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision*]. Jan 2015

[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich . [*Going deeper with convolutions*]. 2014

[40] M. Lin, Q. Chen, and S. Yan. [*Network in network*]. 2013

[41] Szeliski R . [*Computer vision algorithms and applications*]. 2010

[42] Latharani, T. R., Kurian, M. Z., and M. [*Various Object Recognition Techniques*]. 2011

[43] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei . [*Imagenet large scale visual recognition challenge*]. 2014

[44] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman . [*The pascal visual object classes (voc) challenge*]. Int. J. Comput. Vision June 2010 .

[45] Andrew Gibiansky [*lecture 09 Convolutional Neural Network: Architectures, Convolution/Polling Layers*]. S231n Convolutional Neural Networks for Visual Recognition. 2015

[46] Brinkmann, R. [*The Art and Science of Digital Compositing- Techniques for Visual Effects, Animation and Motion Graphics*]. Second Edition. 2008

[47] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. [*Object Detection with Discriminatively Trained Part Based Models*]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2009

[48] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson [*How transferable are features in deep neural networks*]. in Advances in neural information processing systems . 2014 .

[49] Dipanjan Sarkar , Raghav Bali ,Tamoghna Ghosh [*Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras*]. 2018

[50] Yi Chang , Jianhui Chen , Makoto Yamada [*Transfer Learning: Algorithms and Applications*]. novembre 2018

[51] Indra den Bakker [*Python Deep Learning Cookbook: Over 75 practical recipes on neural network modeling, reinforcement learning, and transfer learning using Python*].

[52] Emma Weber [*Turning Learning into Action: A Proven Methodology for Effective Transfer of Learning*]. 2014

[53] development team [*Python Tutorial - Guido van Rossum and the Python*].

[54] Tim Bailey [*An Introduction to the C Programming Language and Software Design*].

[55] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn, Kurt Smith [*Cython- The best of both worlds*].

[56] [*ANACONDA CHEAT SHEET – ANACONDA DOCUMENT*].

[57] Joe Minichino, Joseph Howse [*Learning OpenCV 3 Computer Vision with Python*].

[58] Tom Hope, Yehezkel S. Resheff, Itay Lieder [*Learning TensorFlow: A guide to building deep learning systems*].