REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEURE ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ DES MATHEMATIQUES ET DE l'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

NACHEF Abd el karim
BOUMEDIENE Noureddine

Sur le thème

Détection d'objet en temps réel en utilisant une approche basée sur l'apprentissage profond

---

Soutenu publiquement le 27 / 06 / 2022 à Tiaret devant le jury composé de :

Mr MOSTEFAOUI Kadda	Grade Université MAA	Président
Mr SAFA Khaled	Grade Université MAA	Encadrant
Mr BEKKI Khadir	Grade Université MAA	Examinateur

2021-2022

TABLE OF CONTENTS:

ABSTRACT:

Moving object detection is a key step in many computer vision algorithms such as video surveillance, human motion analysis, robotics, sports footage analysis and others. Recently, the accuracy of object detection has been improved through the performance of approaches based on deep learning algorithm such as region-based convolutional network (R-CNN), YOLO (You Only Look Once) and others. The objective of this Master's thesis is to identify and evaluate the performance of existing deep learning models that are suitable and highly efficient for real-time object recognition.

Keywords: object detection, deep learning, R-CNN, SSD, YOLO.

ملخص:

يعد اكتشاف الأجسام المتحركة خطوة أساسية في العديد من خوارزميات الرؤية الحاسوبية مثل المراقبة بالفيديو، وتحليل الحركة البشرية، والروبوتات، وتحليل اللقطات الرياضية وغيرها. في الآونة الأخيرة، تم تحسين دقة اكتشاف الأشياء من خلال أداء الأساليب القائمة على خوارزمية التعلم العميق مثل الشبكة التلافيفية القائمة على المنطقة (R-CNN) وYOLO (أنت تنظر مرة واحدة فقط) وغيرها. الهدف من أطروحة الماستر هذه هو تحديد وتقييم أداء نماذج التعلم العميق الحالية المناسبة وذات الكفاءة العالية للتعرف على الأشياء في الوقت الفعلي.

الكلمات الدالة: اكتشاف الكائن، التعلم العميق، R-CNN، SSD، YOLO.

Résumé:

La détection d’objets mobiles est une étape clé de nombreux algorithmes de vision par ordinateur tels que la vidéo surveillance, l’analyse du mouvement humain, la robotique, l’analyse de séquences sportives et autres. Récemment, la précision de la détection d'objets a été améliorée grâce aux performances des approches basées sur un algorithme d'apprentissage profond tel qu'un réseau convolutif basé sur la région (R-CNN), YOLO (You Only Look Once) et d'autres. L'objectif de ce mémoire de Master est d'identifier et d'évaluer les performances des modèles d'apprentissage profond existants qui sont appropriés et très efficaces pour la reconnaissance d'objets en temps réel.

Mots clés : détection d’objets, apprentissage profond, R-CNN, SSD, YOLO.

ACKNOWLEDGMENT:

This thesis was made possible by the support of many people to whom we would like to express our gratitude.

We'd like to thank my supervisor Mr. Safa Khaled for guiding us over these months. He never seemed to tire of responding to our inquiries. This helped us identify our shortcomings and taught us an appropriate scientific research method as we progressed in our investigation.

We thank our teachers for their welcome, support and all the knowledge they shared with us.

We'd like to thank our parents for their unwavering support, as well as our graduating friends “Monis Ibrahim”, “Abdelkarim Chanane”, and “Hocine Bouarara”, who guided us through the writing of their graduation theses.

Finally, we thank all our relatives, friends and colleagues who have supported us in many ways during the development of this thesis.

LIST OF FIGURES:

LIST OF TABLES:

LIST OF ABBREVIATIONS:

AI	Artificial Intelligence
NN	Neural Network
ANN	Artificial Neural Network
FFNN	Feed Forward Neural Networks
FBNN	Feedback Neural Networks
ReLU	Rectified Linear Unit
DL	Deep Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
mAP	Mean average precision
RoI	Regions of Interest
YOLO	You Only Look Once
DNN	Deep Neural Network
JSON	JavaScript Object Notation
API	Application Programming Interface
REST	Representational State Transfer
R-CNN	Region-based Convolutional Neural Network
SVM	Support Vector Machine
RPN	Region Proposal Network
RoI	pooling Region of Interest pooling
XML	Extensible Mark-up Language
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
TP	True Positive
FP	False Positive
TN	True Negative
FN	False Negative
SSD	Single Shot Detector
ADS	Autonomous Drive System
AP	Average Precision
BEV	Bird’s Eye View
BN	Batch Normalization
CM	Convolutional-Maxpooling
DATMO	Detection and Tracking of Moving Objects
EKF	Extended Kalman Filter

FEN	Feature Extractor Network
FPN	Feature Pyramid Network
FPS	Frames Per Second
ICP	Iterative Closest Point
IoU	Intersection over Union
LiDAR	Light Detection and Ranging
MSE	Mean Squared Error
NMS	Non-Maximum Suppression
SLAM	Simultaneous Localization and Mapping
LOAM	Lidar Odometry and Mapping
LReLU	Leaky Rectified Linear Unit

CHAPTER 1: INTRODUCTION

1. INTRODUCTION:

In the areas of artificial intelligence (AI) and machine learning, there are a number of exciting and amazing advances that will ultimately benefit the lives of billions of people with greater impact than ever before. Many previously unsolvable issues have been addressed, and significant funds have been committed in research and development in this field.

In the recent years, computer vision is one of the most important sub-fields of artificial intelligence and machine learning, due to its wide variety of applications and enormous potential. Its major purpose is to mimic human vision's tremendous capabilities.

Organizations, especially security companies, have made computer vision a challenge by striving to build new and more efficient deep learning algorithms and methodologies in order to produce intelligent applications, with one of the aims being real-time object detection.

The process of object classification and localization in a real-time scenario is challenging. However, this did not hinder the growth of this branch. In classical methods it was difficult to keep up with this development, which necessitated the development of more efficient deep learning-based methods, such as two-stage detection learning, which was slow and time-consuming to learn, and single-stage detection learning, which opened a path for real-time object discovery.

Our focus is to identify and evaluate the performance of existing deep learning-based methods that are suitable and highly efficient for real-time object recognition.

1.1. Motivation and problem statement:

Object detection research has opened up new and exciting prospects in practically every business. As a result, many businesses and organizations are considering how to incorporate them into their operations. Object detection is a key ability required by most computer visions, which can be applied to many applications, especially surveillance applications. Pedestrian detection is a key issue in surveillance, with several applications such as person identification, person count and tracking. Object detection's main goal is to help us recognize

and locate object in images and videos. Deep learning methods are the most effective way for object detection in this scenario where a sophisticated Object detector, specifically created for high-end surveillance applications, is required to not only locate the bounding box and label it but also provide their relative locations. In this master thesis, we present a person and cat detection approach from webcam video utilizing an efficient method called yolo (you only look once).

1.2. Scope of the master's thesis:

Since object detection is a new field, it is still being developed. As a result, we decided to create an object detection model in real time. After reading a few publications in this field, we became interested in this topic. As a result, we're extremely motivated to develope a real time object detection algorithm using YOLO, and acronym for "You Only Look Once".

1.3. Outline:

Our topic is a real-time object detection system, In the subsequent chapters we present the following. In Chapter 2, we will discuss about deep learning and object detection, as well as previous work related to it. In Chapter 3, We will state the differences between detection cases, we will evaluate the current fastest detection models, and we select the fastest model to train our data on after it's been gathered and recorded in a certain format. we'll discuss the application we're attempting to create. We will implement a yolo model that's both accurate and relatively fast, and then discuss the results obtained. finally, we address the project's conclusions and future work.

CHAPTER 2: BACKGROUND & RELATED WORK

2. Background and related work:

2.1. Deep learning:

2.1.1.The history of deep Learning:

The history of deep learning goes back to a field that is now called cybernetics. This science started in the 1940s with McCulloch and Pitts, where they came up with the idea that neurons are units with finite values and enabled and disabled states. One can build a Boolean circuit by connecting these cells together and making a logical deduction. The brain is basically a logical reasoning machine because neurons are binary. The neuron calculates a weighted sum of the inputs and compares it to the lower bound: if the value is higher than the lower limit then the neuron starts working, or if it is lower than that, it stops. This is a simplified explanation of how neural networks work [1].

In 1947, Donald Hebb came up with the idea that neurons in the brain learn by changing the strength of the connections between neurons, which is called hyper learning: if two neurons are activated together, the connection between them increases, but if they don't work together, the connection less [1].

Later in 1948, Norbert Wiener proposed cybernetics, which is based on the idea that by having systems with sensors and actuators, you have a feedback loop and a self-regulating system. All Elements The fundamentals of automotive reactions derive from this craft [1].

In 1957, Frank Rosenblatt published the Perceptron, an educational algorithm that changes the weight of very simple neural networks [1].

Generally speaking, the idea of building thinking machines that mimic neurons originated in the 1940s, then took off in the 1950s, and died out completely in the late 1960s. of this field of research in 1960 are:

The researchers used bi-valent neurons. On the other hand, the backscatter operation is to use a continuous activation function. At the time, the researchers had no idea of using connected neurons and did not think to be able to train with gradients because neurons are binary and indistinguishable [1].

The use of connected neurons requires multiplying the cell's activation function by a weight to obtain the contribution of this value to the weighted sum. But before 1980, the process of multiplying two numbers, especially floating-point numbers, was very slow, which prevented the use of cells. Deep learning. Was born in 1985 [1].

It was introduced as a term by (Rina Decher) in 1986, and to artificial neural networks by Igor Eisenberg and colleagues in 2000, in the context of logic neurons [1].

Deep learning restarted in 1985 with the advent of error backpropagation. In 1995, the field stagnated again and the machine learning community abandoned the idea of neural networks. In the early 2010s, researchers started using neural networks for speech recognition and there was a significant improvement in performance, after which neural networks spread widely in the commercial field. In 2013, the field of computer vision began to shift towards the use of neural networks. The same transition happened in natural language processing in 2016. Soon, similar revolutions will occur in robotics, control and many other fields [2].

The first supervised, feed-forward, multi-level deep learning algorithm was published by (Alexei Evakhenko) and Lapa in 1965 [3].

Work on other construction methods for deep learning, particularly those designed for computer vision, began with (Neocognitron), a multi-layered artificial neural network introduced by (Kunihiko Fukushima) in 1980. to a deep neural network in the purpose of recognizing a handwritten postal code by mail. While the algorithm was working, the training required 3 days [3].

In 1991, these systems were used to recognize isolated 2D handwritten numbers, while 3D objects were recognized by manually matching 2D images to a 3D object model. Wong et al suggested that the human brain does not use a federated 3D object model, and in 1992 they published (Cresceptron), a method for performing 3D object recognition in noisy scenes. Since he uses natural images directly, (Cristron) started with general-purpose visual learning of natural 3D worlds. (Cristron) is a series of layers similar to a (neocogonitron). But while Neocognitron requires a human programmer for manual merging features, Cresceptron has learned an open number of features in each layer without supervision, where each feature is represented by the convolution kernel. Cristron

segmented each learned object of a clutter scene through back-end analysis on the network. Maximum pooling, which is now often adopted by deep neural networks (e.g., ImageNet tests), was first used to reduce position accuracy by a factor of (2 x 2) to 1 throughout the sequence to improve generalization. In 1994, André de Carvalho, together with Mike Fairhurst and David Bisset, published the experimental results of a Boolean algebra multilayer neural network, also known as a weightless neural network, consisting of 3 self-organizing layers of the extraction unit neural network (SOFT) followed by a multi-layer classification neural network (GSN) unit, which was trained independently. Each layer of the feature extraction module extracts features of increasing complexity compared to the previous layer [3].

In 1995, (Branden Fry) demonstrated that a network containing six fully connected layers and several hundred hidden units could be trained using the sleep-wake algorithm (over two days), developed jointly with Peter Dean and (Hinton). Many factors contribute to the slow speed, including the vanishing gradient problem analyzed in 1991 by Sepp (Hockerter). Simpler models that use task-specific manual features such as Gabor filters and support vector machines (SVMs) were a popular choice in the 1990s and 2000s, but because of artificial neural networks (ANN), we became computationally expensive and did not understand how to connect the brain to its biological networks [3].

2.1.2.Types of learning:

There are three types of learning: supervised, unsupervised and semi-supervised.

2.1.2.1. Supervised Learning:

Supervised learning is considered to be the most elementary class of machine learning algorithms [4]. As the name suggests, these algorithms require direct supervision. supervised learning is based on training. models are trained using labelled dataset, where the model learns about each type of data. Once the training process is completed, the model is tested on the basis of test data (a subset of the training set), and then it predicts the output.

Supervised learning problems can be further grouped into regression and classification problems.

When the output variable is a category, such as "spam " and "not spam," or "disease" and "no disease," it is referred to as a classification problem. When the output variable is a real value, such as "salary" or "weight," it is referred to as a regression problem.

The following are some well-known supervised machine learning algorithms:

For regression issues, linear regression is used.

For classification and regression issues, random forest is used.

For classification tasks, support vector machines are useful.

2.1.2.2. Unsupervised Learning:

In unsupervised learning the goal is to learn relationships among elements in a data set D = {x1, x2, ..., xn} and classify the raw data without relying on a ground truth. Since it is not clear which patterns should be learned there is no obvious error metric which leads to search indirect hidden structures, patterns or features in the data [5].

2.1.2.3. Semi-supervised Learning:

Semi-supervised learning combines the advantages of both methods by combining a little quantity of labeled data with a large amount of unlabeled data [6].

2.1.3.Artificial neural network:

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process a supervised learning, called deep learning, that uses interconnected nodes or neurons (perceptron) in a layered structure that resembles the human brain [7]. A perceptron is a function that maps the dot product of a weight vector $w \in R^L$ and its corresponding input vector $x \in R^L$ plus a bias to an output value y_j:

$$y_j = f\left(\sum_{i=1}^{L} w_{ij} x_i + b_j\right), j = \{1, 2, \dots, M\} \qquad (1)$$

where f: R → R is an activation function.

Figure 1 represents a perception, it is a neural network unit (an artificial neuron) that does certain computations to detect features in the input data.

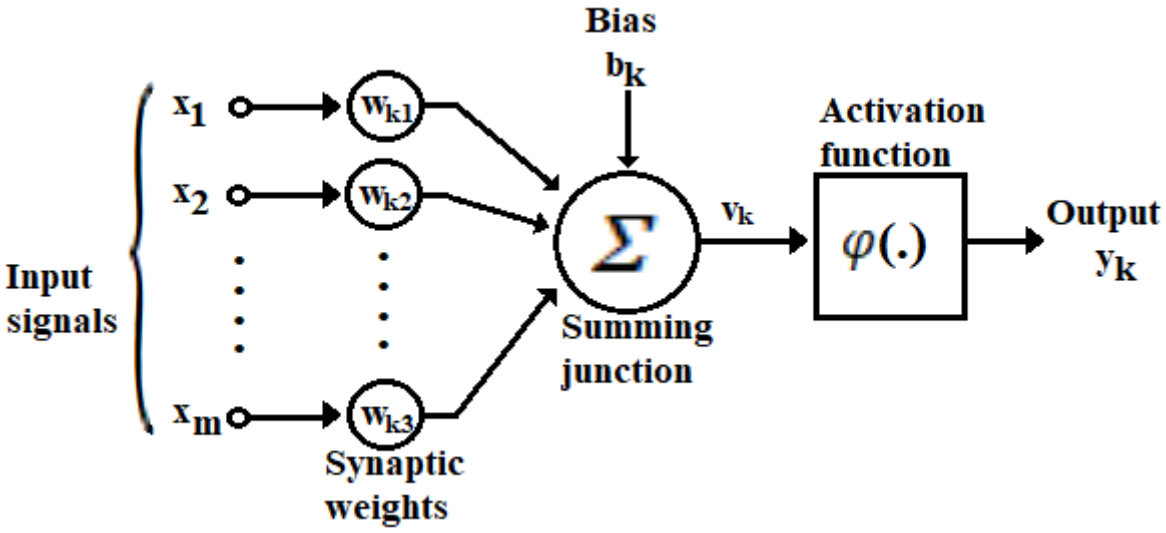


Figure 1: Structure of a single perceptron or neuron

There are various types of neural networks such as Hopfield network, the multilayer perceptron, the Boltzmann machine, and the Kohonen network. The most commonly used and successful neural network is the multilayer perceptron and will be discussed in the following section.

2.1.3.1. Multilayer perceptron(mlp):

There are three kinds of layers in mlp network. The input layer is the raw data of the neural network, this layer transmits information (features) to the hidden layers. The hidden layers are the intermediate layers between the input layer and the output layer and is where all the calculations are performed [7]. The output layer is the layer that gives the results of a given input.

Figure 2 represents a multi-layer neural network architecture where each layer consists of a certain number of perceptions. In contrast, the number of perceptions in the last layer represents the number of classifications.

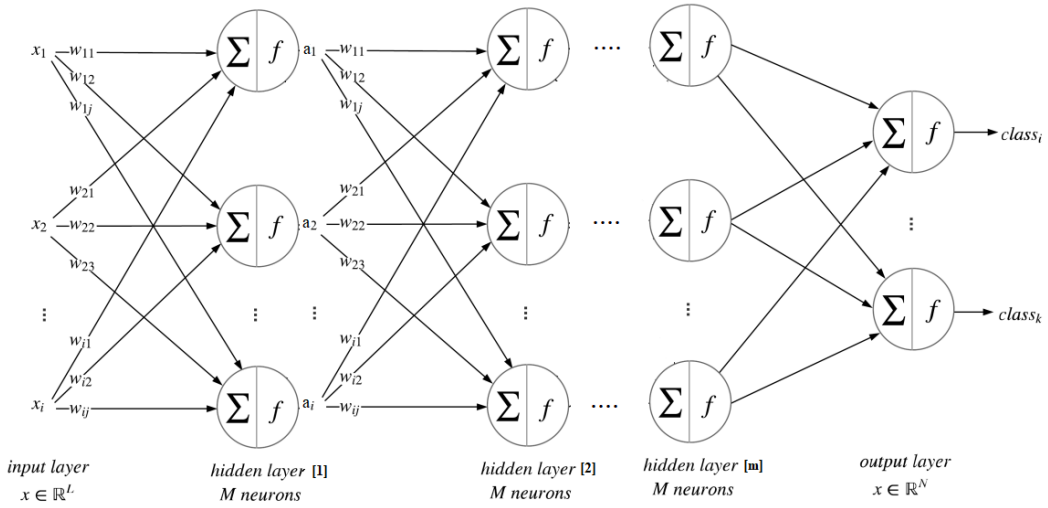


Figure 2: Multilayer perceptron of neural networks

The output $O_l \epsilon R^M$ of an arbitrary layer l is computed as:

$$O_l = f^1(w^l x + b^l) \qquad (2)$$

The output of the first layer becomes the input to the second layer, the second to the third and so on successively [8]. A hidden layer l with N perceptrons and M input values can be defined as a function $R^M \rightarrow R^N$ where N is the number of perceptrons in the layer and M is the number of inputs. A neural network with n layers can be seen as a series of nested functions where the output of the first layer becomes the input to the second, the second to the third and so on successively. This can be described mathematically as

$$0 = f^n\left(w^n \dots f^2\left(w^2 f^1(w^1 x + b^1)\right) + b^2) \dots\right) + b^n \qquad (3)$$

In the equation y, 'x' is the input value given at the neuron, 'w' is the weighted value of the synapse, 'n' is the number of neurons, 'b' is the bias and 'y' is the output of the network [9]. Therefore, according to the equation, the value of output 'y' is equal to the summation of the product of the values of 'x' with their corresponding weights and bias 'b'.

$$y = \sum_{i=1}^{n} (w_n \times x_n) + b \qquad (4)$$

A neural network can perform binary classification with just one perceptron in its simplest form, but by increasing the number of perceptrons and building the network in specific architectures, they can be universal approximators to almost any continuous set function, making them suitable for a variety of machine learning tasks. Even though there are multiple different types of neural networks,

the word ANN is used to refer to all of them: Modular Neural Networks, Convolutional Neural Networks, Recurrent Neural Networks, and so on [10].

Recall that in order for a neural network to learn, weights associated with neuron connections must be updated after forward passes of data through the network. These weights are adjusted to help reconcile the differences between the actual and predicted outcomes for subsequent forward passes. Since we are talking about the difference between actual and predicted values, the error would be a useful measure here, and so each neuron will require that their respective error be sent backward through the network to them in order to facilitate the update process. hence, backpropagation of error. But how, exactly, do the weights get adjusted?

2.1.3.2. Back-propagation:

Backward Propagation is the preferable method of adjusting or correcting the weights to reach the minimized loss function [11].

Rumelhart et al. were the first to propose the approach in 1986. If the result differs from the predicted output, networks can use this strategy to change the weights of hidden layers. The error is calculated and backpropagated to all the layers of the network to adjust the weights according to the requirement [12].

A loss function calculates the difference between a predicted output $\hat{y}_i$ and its actual value y_i. A classic error function for back-propagation is the mean squared error [13].

$$L(X, w) = \frac{1}{2N} \sum_{i=1}^{N} (\hat{y}_i(X, w) - y_i)^2 \qquad (5)$$

where y_i is the target value for an input pair (x_i, y_j) and $\hat{y}_i$ is the computed output of the network on input x_i. Other error functions can be used but its convenient mathematical properties make it a good choice for generalized learning methods [13].

Back-propagation refers to the process of calculating the gradient backwards across the network, starting with the gradient of the weight in the last layer, then the penultimate, and so on. When opposed to the naive approach of calculating each layer separately, the gradient computations from one layer are reused in the computations of the preceding layer, allowing for efficient gradient computation at each layer [13].

As a result, back-propagation tries to minimize the loss function L with respect to the weights of the neural network by computing the value of $\frac{\delta L}{\delta w_{ij}^k}$ for each weight w_{ij}^k. This derivative can be calculated separately for each input-output pair and then combined at the end [13].

$$\frac{\delta L(X,w)}{\delta w_{ij}^k} = \frac{1}{N}\sum_{d=1}^{N}\frac{\partial}{\partial w_{ij}^k}\left(\frac{1}{2}(\hat{y}_d - y_d)^2\right) = 1/N\sum_{d=1}^{N}\frac{\partial E_d}{\partial w_{ij}^k} \qquad (6)$$

Finally, the weights can be updated according to the learning rate α and the total gradient:

$$\Delta w_{ij}^k = -\alpha\frac{\delta L(X,w)}{\delta w_{ij}^k} \qquad (7)$$

Figure 3 represents the complete cycle of a multi-layer neural network, where. (1): Forward propagation refers to the calculation and storage of intermediate variables (including outputs) for a neural network in order from the input layer to the output layer. (2): It represents the difference between the truth value and the expected value. (3): shows the Backpropagation, which updates the weights based on the value of the extracted error.

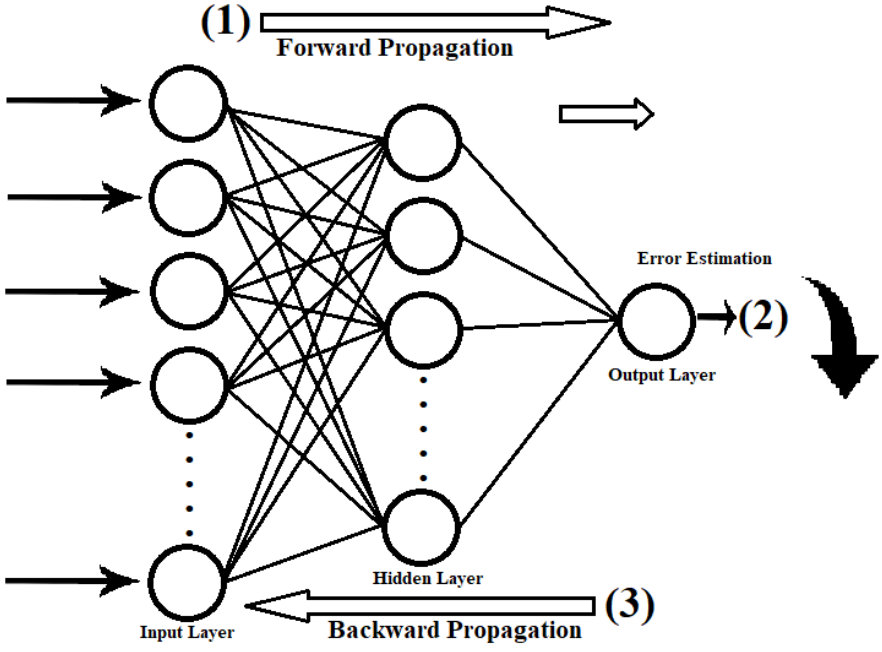


Figure 3: forward / backward Propagation

2.1.3.3. Activation Function:

Activation functions are used in ANNs to convert the input signal into an output signal which in turn is fed as input to the next layer. It plays an important role on the accuracy of the ANN prediction thus its selection must be careful. If an ANN does not have an activation function, the output signal would be a linear function and the network acts as a Linear Regression Model. The result of this is a network with limited performance [14]. An important feature that needs to be considered is that it must be differentiable to be able to perform back-propagation optimization for gradient error calculations. There are several activation functions but the use of each of them depend heavily on the goal of each layer in the ANN as they have different properties. Some common activation functions are presented below [14].

- **Sigmoid:** $\boldsymbol{f(x) = \sigma(x) = \frac{1}{1+e^{-x}}}$ Sigmoid function is a mathematical function which has a characteristic S-shaped curve. There are a number of common sigmoid functions, such as the logistic function, the hyperbolic tangent, and the arctangent. All sigmoid functions have the property that they map the entire number line into a small range such as between 0 and 1, or -1 and 1, so one use of a sigmoid function is to convert a real value into one that can be interpreted as a probability.

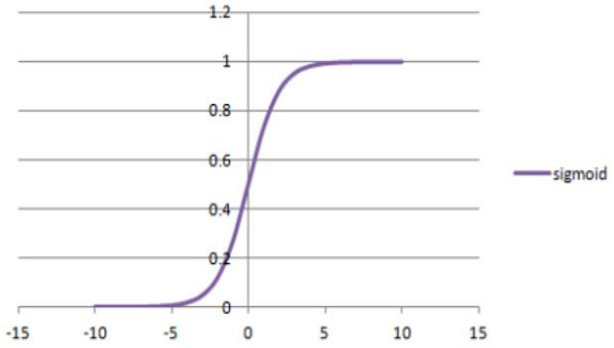


Figure 4: Sigmoid Functions **[15]**.

- **Tanh:** $\boldsymbol{f(x) = \tanh(x) = \frac{1-e^{-2x}}{1+e^{-2x}}}$ Because the non-linear activation function's values span between -1 and 1, it's comparable to the exponential function in the negative inputs give negative results, and only zero-valued inputs are allocated to outputs close to zero. The distinctions are in the symmetry (Tanh is symmetric about the origin) and the gradient (Tanh has a steeper gradient). As a result, the outputs

from previous layers will have various signs when fed as input to the following layer.

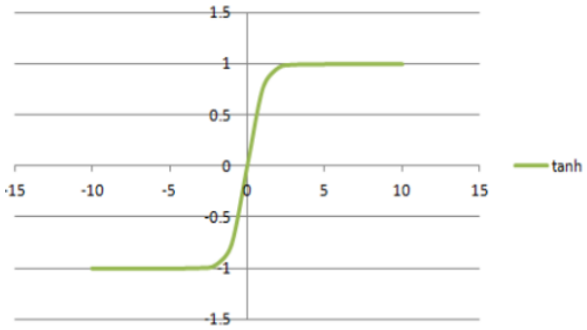


Figure 5: Tanch Functions **[15]**.

- **ReLU:** $f(x) = ReLU(x) = max(0, x)$ The fundamental advantage of the rectified linear activation function, also known as ReLU (Rectified Linear Unit), is that it does not activate all perceptrons at the same time because it returns zero for negative inputs. As a result, the network becomes sparse and efficient. However, because the gradient is 0 for negative values, the vanishing problem still exists, preventing the network from being updated during back propagation.

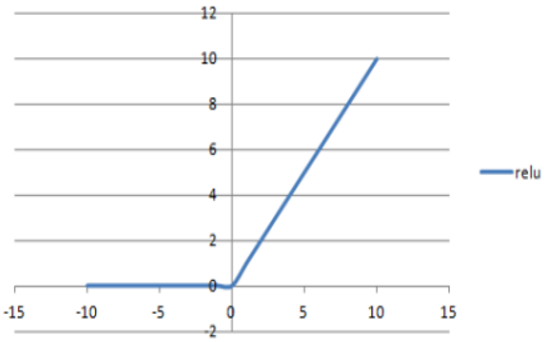


Figure 6: ReLU Functions **[15]**.

- **Leaky ReLU:** $f(x) = LReLU(x) = max(\alpha x, x), \alpha \leq 1$, is an improved version of ReLU. It solves the vanishing gradient problem by inserting a small linear component for negative values. to make the derivation a slightly positive value instead of 0.

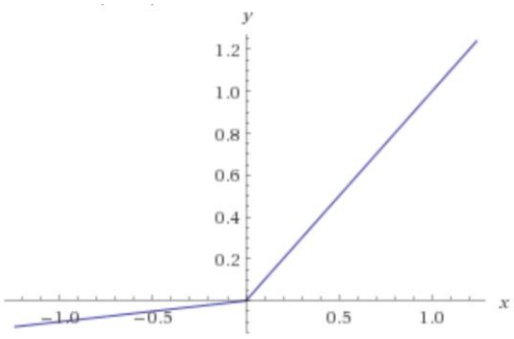


Figure 7: Leaky Functions **[15]**.

- **Softmax:** $f(x) = \sigma(x)_j = \frac{e^{xj}}{\sum_{k=1}^{M} e^{xk}}$, $for\ j = 1, 2, \dots, M\ and\ x = (x_1, \dots, x_M) \epsilon R^M$,It is a classification of more than one variable. It is used in place of the sigmoid function as a classification. It squeezes the outputs for each variable in the feature vector x between 0 and 1 dividing it by the sum of all variables. This makes it so that the sum of all variables in x will result in 1 after being run through the softmax activation function. This activation function is best employed at the classifier's output layer to obtain probabilities that may be used to classify each input.

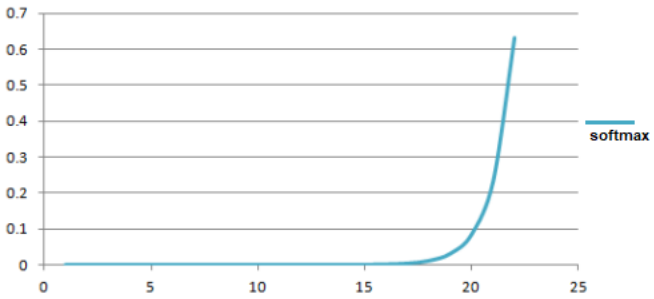


Figure 8: SoftMax Functions **[15]**.

2.1.3.4. Batch Normalization:

Deep neural networks are difficult to Learn. Batch normalization (also known as batch norm) is a method used to make training of artificial neural networks faster and more stable through normalization of the layers' inputs by re-centering and re-scaling. [16].

Batch normalization is a routine operation in the construction of neural network models, which not only speeds up the convergence of the model to some extent but also plays the most important role in solving the gradient dispersion problem in the network, i.e., the phenomenon of unstable gradient variation. If normalization is not used, the distribution of data obtained by the model after training at each layer has differences, and the computational volume of the model

network will increase for different data processing, causing the network model to be more complex, and thus prone to overfitting and slower convergence of the model. that is, the essence of normalization is to make the model acquired picture feature data more reasonable, optimize the learning ability of the model, improve the generalization ability of the model, and improve the counting ability [16].

2.1.4.Convolutional neural network:

A CNN is a subset of the neural networks mentioned above. A convolutional neural network is a sort of supervised deep learning method that is an extension of artificial neural networks (ANN) [17]. It is primarily used for jobs involving picture recognition. Image/video recognition [18], semantic parsing, natural language processing, and paraphrase detection are among applications where convolutional neural networks (CNNs) are useful [19]. The Convolutional layer, Pooling layer, and Fully-connected layer are the three layers that make up a convolutional neural network. Convolution's main goal is to extract features from input photos while keeping the spatial relationship between pixels intact [20]. It accomplishes this by learning visual attributes from small squares of input data.

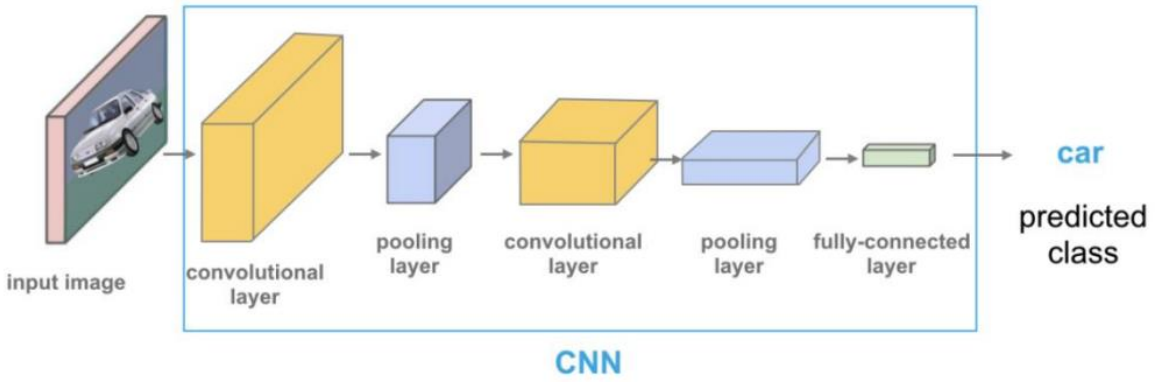


Figure 9: Representation of CNN (Camacho et al., 2018). **[92]**

2.1.4.1. The Convolutional Layer:

A series of filters with learnable parameters are used to extract features from input data in this sort of layer. They can be compared to CNN's weights and biases. The layers are constructed in such a way that the first detects a collection of low-dimensional patterns in the input, such as edges, blobs of color, and so on [21]. The second detects patterns of patterns, and so on. The convolutional layer learns features by back-propagation in the same way as a multi-layer perceptron network (or ANN) does.

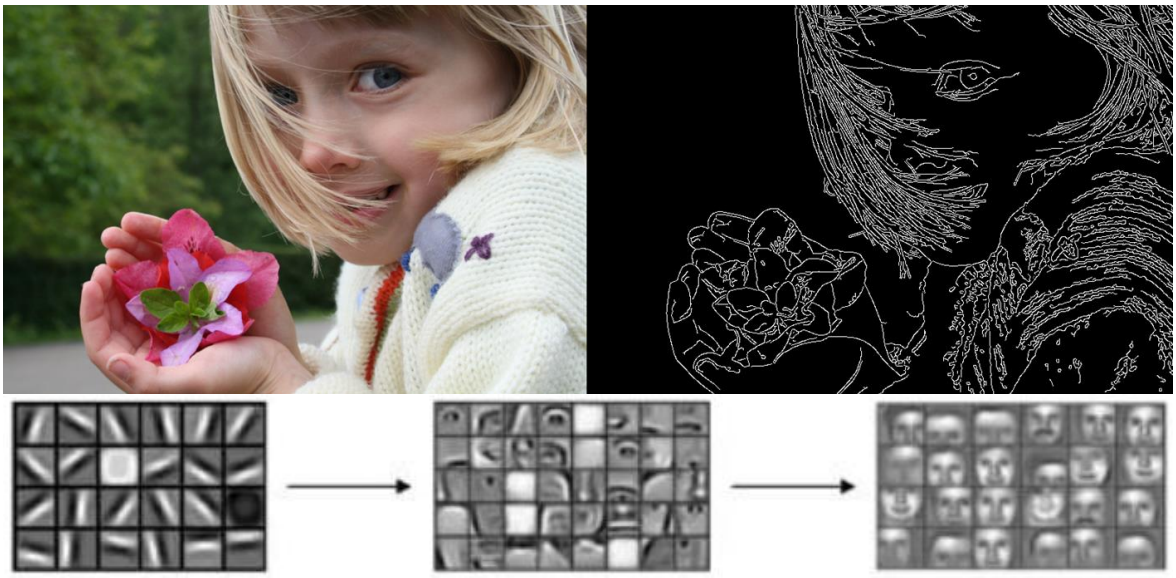


Figure 10: Feature Filters of Front, Middle and Rear-End Layers in a CNN **[96]**

Convolution is performed by sliding a fixed-size kernel over the input matrix. At each stage, the elements that fall inside the kernel are combined using matrix multiplication of the kernel and the region in the input matrix where the kernel overlaps. Other parameters are zero-padding, which adds zeros around the input matrix to preserve the size of the input matrix (because a convolution reduces the dimension of the input matrix), and stride, which specifies how many elements the kernel should bounce over between steps. In terms of output volume, the larger the stride, the smaller the output volume. An important parameter to specify for a convolutional layer is the number of filters, which determines the depth of the convolutional layer. Each filter learns to look for different visual features in the input. The convolutional layer accepts an input of size $W_1 \times H_1 \times D_1$. It requires four parameters: the number of filters **K**, the kernel size **F**, the stride S, and the zero-padding **P**. The layer produces an output of size $W_2 \times H_2 \times D_2$ where [22] see Figure 11:

$$W_2 = (W_1 - F + 2P)/S + 1$$
$$H_2 = (H_1 - F + 2P)/S + 1, \qquad (8)$$
$$D2 = K.$$

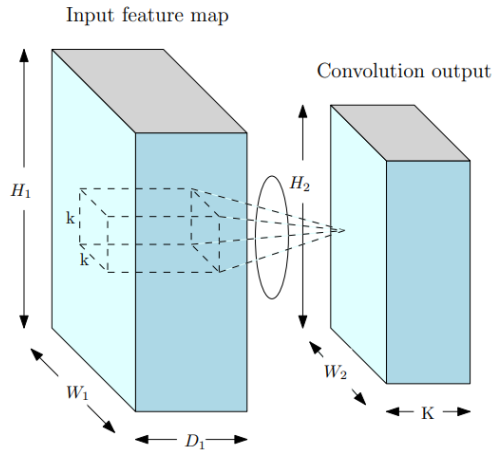


Figure 11: The input volume of size $[\boldsymbol{W_1 \times H_1 \times D_1}]$ is convolved with a $\boldsymbol{K \times K \times K}$kernel obtaining an output volume $[\boldsymbol{W_2 \times H_2 \times K}]$ **[22].**

A feature filter, when slid on the input layer of a neural network, performs the convolution process, resulting in the generation of a feature map. A convolutional layer is the layer that performs the convolution process. convolutional neural networks are networks made up of convolutional layers. The filter searches the input layer for any given pattern in the beginning. The filter searches for the purpose of learning to detect a pattern during the training of the algorithm, which finally becomes a search to validate the existence of a specific pattern. Each of the filters in each convolutional layer with its respective number of kernels produce a separate activation map. Stacking these activation maps along the depth dimension lead to that deeper layers in the network can perform more complex associations. There are two types of convolutions [22]:

- **2D Convolution:** Convolution is used in 2D CNNs to extract features from only 2D space. The value of a unit at (x, y) in the i-th layer of the j-th feature map, written as v_{ij}^{xy} in formal terms, is given by:

$$\boldsymbol{V_{ij}^{xy} = f(b_{ij} + \left(\sum_m \sum_{p=0}^{P_i=1} \sum_{q=0}^{Q_i=1} w_{ijm}^{pq} \, v_{(i=1)m}^{(x+p)(y+q)}\right)} \qquad (9)$$

where f is an activation function, $\boldsymbol{b_{ij}}$is the bias for the feature map, $\boldsymbol{m}$ is the number of filters in the $(i-1)$th layer, W_{ijm}^{pq} is the value at the position (p, q) of the kernel connected to the kth feature map, and p_i and Q_i are the height and width of the kernel, respectively.

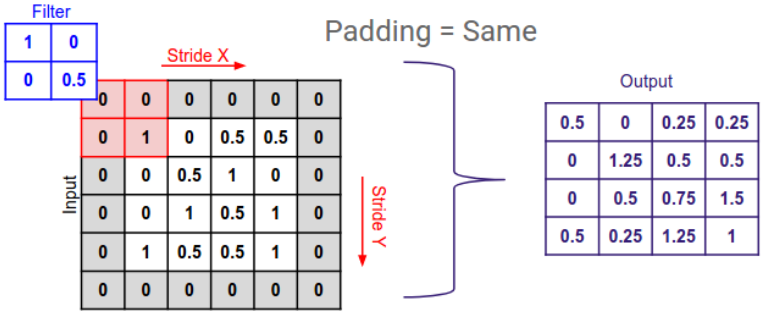


Figure 12: Convolution 2d with stride and padding **[103]**.

- **3D Convolution:** Used either for spatial feature extraction of 3D images, or spatio-temporal feature extraction of 2D images [23]. The above equation can be extended as follows:

$$V_{ij}^{xy} = f\left(b_{ij} + \left(\sum_m \sum_{p=0}^{P_i=1} \sum_{q=0}^{Q_i=1} \sum_{r=0}^{R_i=1} w_{ijm}^{pqr} v_{(i=1)m}^{(x+p)(y+q)(z+r)}\right)\right) \quad (10)$$

where R_i is the size of the 3D kernel along the third spatial dimension and w_{ijm}^{pqr} is the (p, q, r) th value of the kernel connected to the mth feature map in the previous layer.

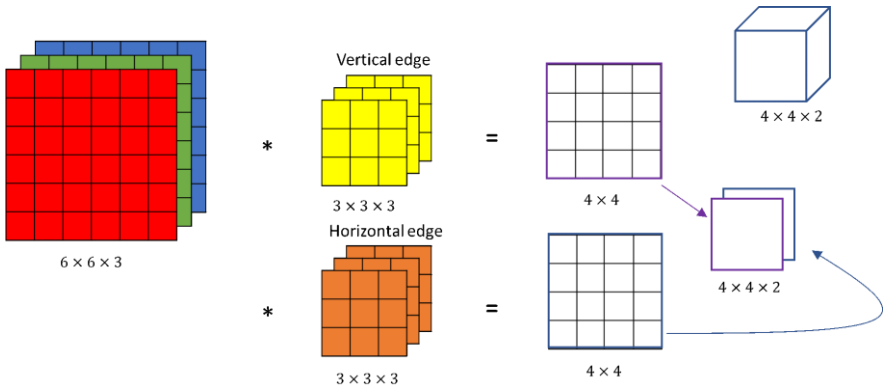


Figure 13: convolutions 3D to use multiple filters **[104]**

2.1.4.2. Pooling Layer:

After convolution, the procedure of pooling is conducted [24]. Its key characteristic is that it compresses the input into a patch. The following are two common pooling functions:

- **Average Pooling:** Calculate the average value for each patch on the feature map Figure 15.
- **Maximum Pooling (or Max Pooling):** Calculate the maximum value for each patch of the feature map Figure 14.

The network gains two things:

- Controls overfitting by reducing the amount of training parameters and the cost of computation.
- Makes the network invariant to certain distortion.

[24]

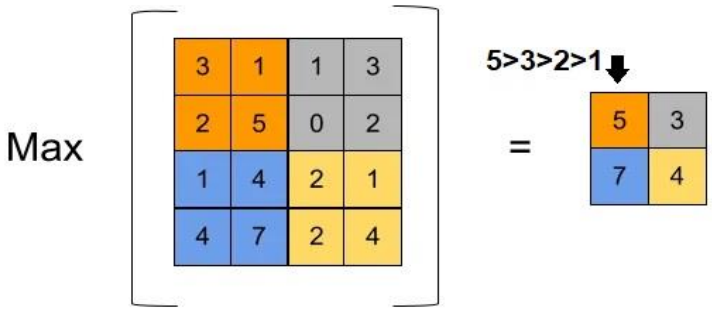


Figure 14: a portrait showing max pooling **[97]**.

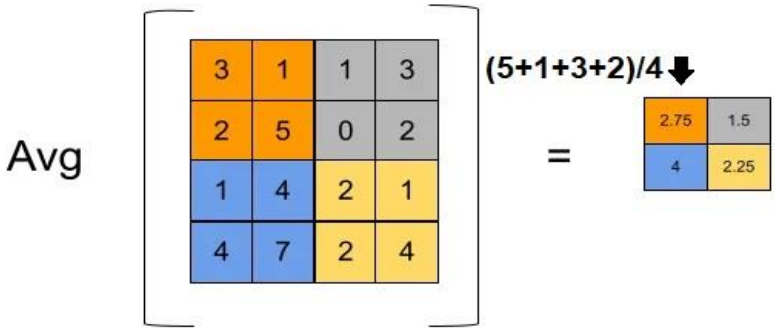


Figure 15: a portrait showing average pooling **[97]**.

2.1.4.3. Fully Connected Layer:

The final convolution or pooling layer's output feature maps are converted into a 1-Dimension array of numbers and connected to one or more fully connected layers, in which a weighted connection is made between every input and every output by a weight. Using a fully connected layer to learn non-linear combinations of these features is a good approach to start. As a result, fully connected layers are frequently used as the CNN's final layers. They add the weighted total of the previous layer's features, showing the precise parameter inputs, to arrive at a certain output goal result [25].

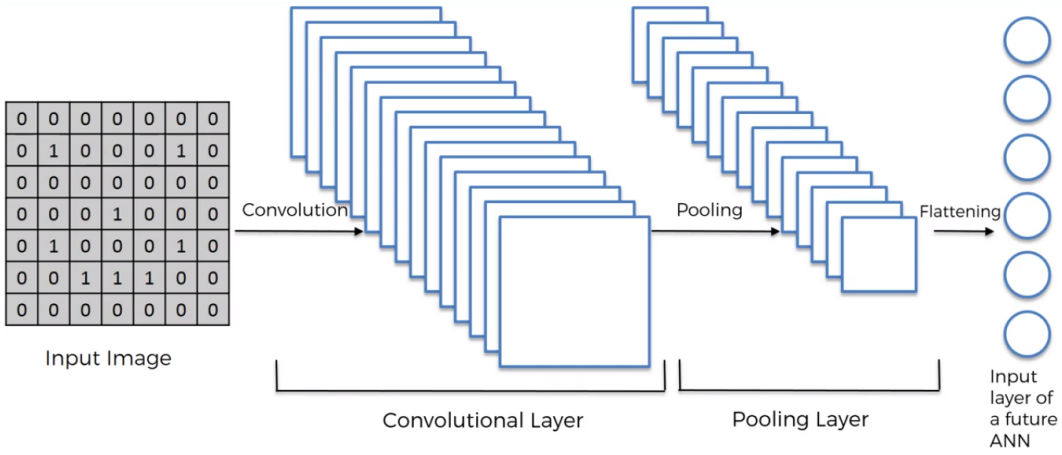


Figure 16: A picture showing the flattening of the output **[105]**.

2.1.4.4. Why Convolutional Neural Networks?

While neural networks have been present for over 50 years, there are a few reasons why CNNs have become the industry standard for object recognition and categorization [26]. The following are some of their primary advantages [21]:

- **CNNs have fewer memory requirements:** CNNs use the fact that the input data can be viewed as a multi-channel image to reduce the dimensionality of the input while keeping characteristics that can be recovered for classification within the input image.
- **They are easier and better to train:** A CNN's training time is proportionately less than that of a normal neural network due to its lower architectural complexity. Moreover, because of a lower number of parameters, the susceptibility to noise is lower during the training process. Hence, the performance of a standard neural network will always be poorer than a CNN for image classification purposes**.**
- **They are rugged to shifts and distortion in the input:** Because the same weight configuration is employed across space, CNNs are shift invariant. Although a typical neural network may achieve this, many units with similar weight values at different regions of the input would be required, adding memory and training time loads. CNNs can withstand a variety of distortions, including shape shifts, partial occlusions, horizontal and vertical shifts, and so on.

However, because the specific spatial correlations between higher-level data are lost in the subsequent down-sampling step, CNNs are only effective for broad object detection tasks [27].

2.2. Object detection:

Computer vision is an interdisciplinary field that has sparked a lot of interest in recent years [28].

Object detection is a supervised machine learning process that determines the instance of the class to which the object belongs while also estimating the object's location by reporting the bounding box around it. Each image in the training dataset must be accompanied with a file containing the object's boundaries and classes. Object localization and categorization are the first two stages of the process.

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.

Object detection can be used in a variety of situations, including human-computer interaction, defense, robotics, and transportation (auto-pilot) [29].

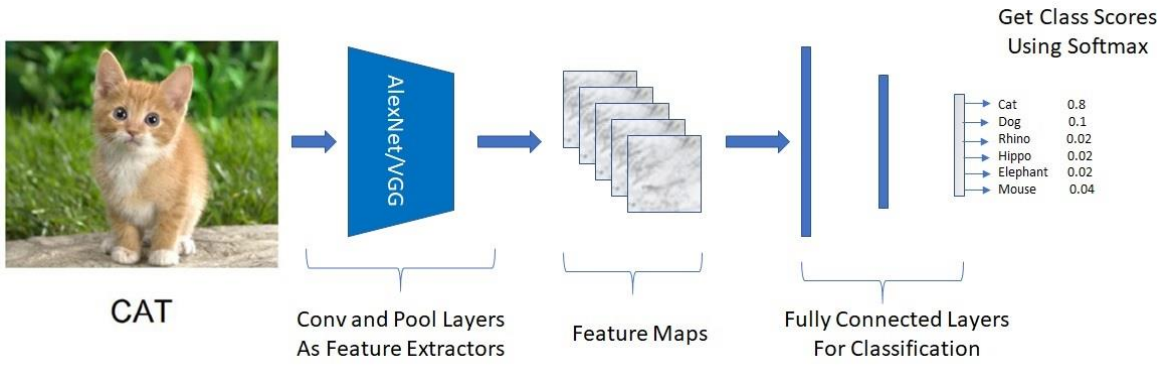


Figure 17: classification with localization **[106]**.

2.2.1.History of detection algorithms:

In this section will simply provide a cursory review of current developments.

A good means of judging how close the computer vision community has come to solving the problems of object detection is to look at the results of challenges like the PASCAL Visual Object Challenge (VOC), and later the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [30].

Pascal VOC began in 2005 and was hosted every year till 2012. It usually included classification as well as localisation tasks, among other things. The organizers published a report in 2009 that went into greater detail about the challenge and this year's contributions [31]. Most of the entries at the time used a bag-of-visual-words technique based on hand-crafted features like SIFT [32] and HOG [33], in which feature vectors are computed at key point locations in the image and then a histogram over the feature vectors is used to identify the image content.

When the PASCAL VOC ended in 2012, the ILSVRC was held annually from 2010 onwards and became the key benchmark for object detection. It uses images from the massive ImageNet dataset (over 14 million images) to perform classification and detection tasks, with up to 1000 possible classes for classification and 200 classes for detection.

When Krizhevsky et al. [34] used a deep convolutional neural network (CNN) for the first time in 2012, the challenge experienced a significant boost in performance. They reduced the top-5 classification error1 from 25.2 percent (for the second-best entry) to 15.3 percent, as well as the localization error from 50 percent to 34.3 percent. Since then, convolutional neural networks of increasing depth have dominated the ILSVRC. [35] The winning entry had a top-5 error of 6.7 percent in 2014, indicating that the categorization process has become reasonably simple. The best detection mean average precision (mAP) has increased from 22.58 percent in 2013 to 43.93 percent in 2014. The localisation error has decreased to 25% [36].

The requirement to develop discovery methods in a convolutional neural network has led to the development of many methods that fall into two categories: two-stage detection A selective search algorithm or a region proposal network first proposes a set of regions of interest in two-stage object detectors [37][42]. The region candidates are then processed by a classifier. The R-CNN family is an

example of this type of 2D object detector [38]. One-stage detectors, on the other hand, bypass the region proposal stage and conduct detection across a large number of potential locations. It's faster and easier to use than two-stage detectors, but there's a trade-off in terms of accuracy. To yet, achieving real-time capabilities has necessitated an unavoidable trade-off [39].

2.2.2.Two-stage detection:

The following methods are examples of older two-stage detection methods:

2.2.2.1. R-CNN:

RCNN is introduced after the success of CNN on image categorization with AlexNet [40]. R-CNN [41] is a hybrid method that uses both traditional and learning-based techniques. Selective Search is used to generate proposals in this method [42]. The feature extraction for each region proposal is done with CNN, and the feature vectors are classed with SVM, while the bounding box regression is done with a fully connected neural network Figure 18.

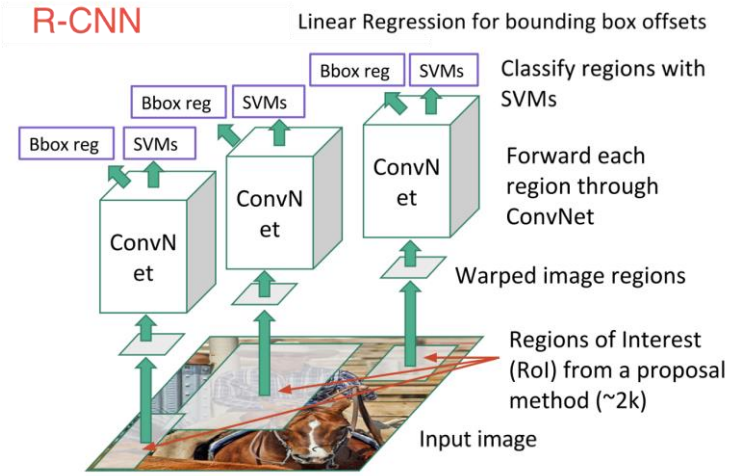


Figure 18: R-CNN **[91]**.

The main drawback of this RCNN is that it takes a long time to train. Furthermore, because the detection takes a lengthy time, the algorithm cannot be done in real-time.

2.2.2.2. Spp-net (Spatial Pyramid Pooling):

SPP is a technique for extracting characteristics from regions of various sizes. SPP-Net As a result, it takes advantage of parameter sharing at the CNN layer and can adapt to a variety of input sizes (since input size is limited by the fc layers not conv layers, in R-CNN the crop has to be warped to fulfil certain size; SPP in SPP-Net elevates this size constraint) [43]. in Figure 19

SPP Net replaces the last pooling layer of traditional convolutional neural network with a spatial pyramid pooling layer. It can diversify the size of the input image, avoiding the information loss and distortion caused by the image distortion, and improving the detection accuracy [44].

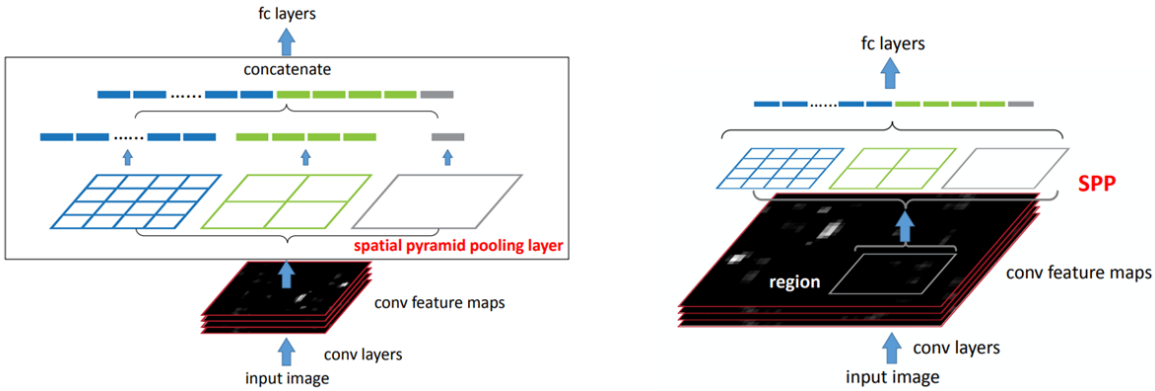


Figure 19: Spp-net (SPP illustration)**.**

2.2.2.3. Fast R-CNN:

Fast R-CNN was created to boost the detection speed of its R-CNN processor [45]. To lower the computational cost, this approach added Roi Pooling layers. Roi Pooling is a type of pooling that consists of suggestions for fixed-sized forms. As a result, instead of processing feature extraction for each proposal, it is done once for each image. They also introduced a deep learning-based method for classification, and a regression network for finding bounding boxes, which is integrated with the feature extraction network. The most notable modifications are that the feature map is generated over the image before the region suggestions are created, and the SVM is replaced with a SoftMax layer [46]. The training and testing time has been significantly shortened as a result of these enhancements. Instead of repeating the convolution process for each of the 2000 regions, the operation is completed once. In Figure 20.

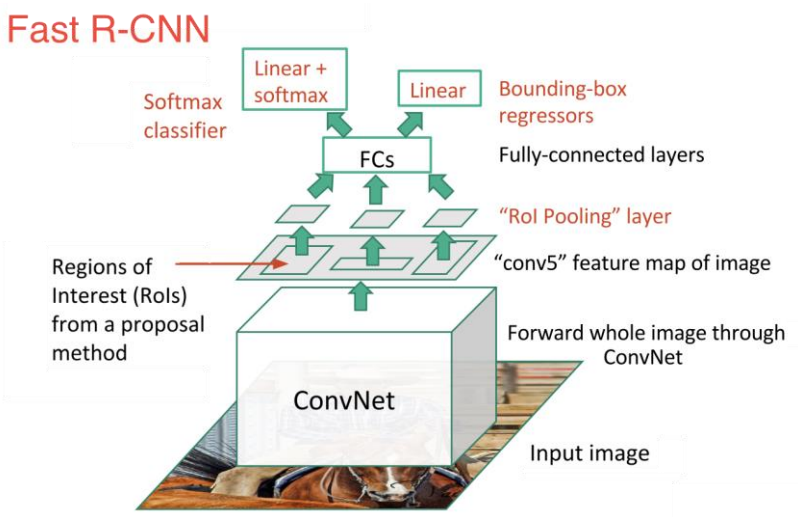


Figure 20: Fast R-CNN **[91]**.

Nonetheless, despite the improvements, the selective search algorithm for generating region proposals had to be modified.

2.2.2.4. Faster R-CNN:

The fundamental idea behind Faster R-CNN was to use a fast neural net to replace the slow selective search algorithm.

Specifically. Ren et al. later introduced Faster R-CNN [47]. The proposal generating step is likewise accomplished by CNN layers (Region Proposal Network - RPN) instead of additional proposal generators, which is the key distinction from Fast R-CNN. Fast R-CNN is utilized for the rest, and the two structures are integrated into one network for the first time, making the architecture trainable from start to finish.in Figure 21.

To anticipate the regions, a separate Region Proposal Network (RPN) was deployed. The predictions are then made and filtered with a Roi pooling kernel before being fed into a Fast R-CNN [47]. The projected object's name and bounding boxes are the outcomes.

As a result, the Faster R-CNN is the fastest CNN and can-do real-time detections.

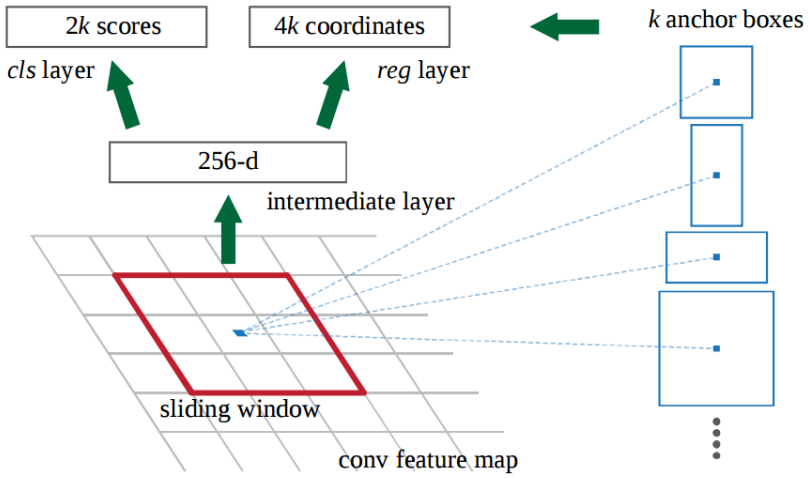


Figure 21 : Faster R-CNN **[109]**.

2.2.2.5. Mask R-CNN:

Mask R-CNN was created as a modified version of Faster R-CNN by He et al [48]. to develop a framework for object instance segmentation. ResNet-FPN [49](feature pyramid network) is used after the feature extraction backbone in the Mask R-CNN method to connect deeper layers with the preceding ones. This approach improves algorithm accuracy while cutting down on computing time. The FPN architecture. in Figure 22.

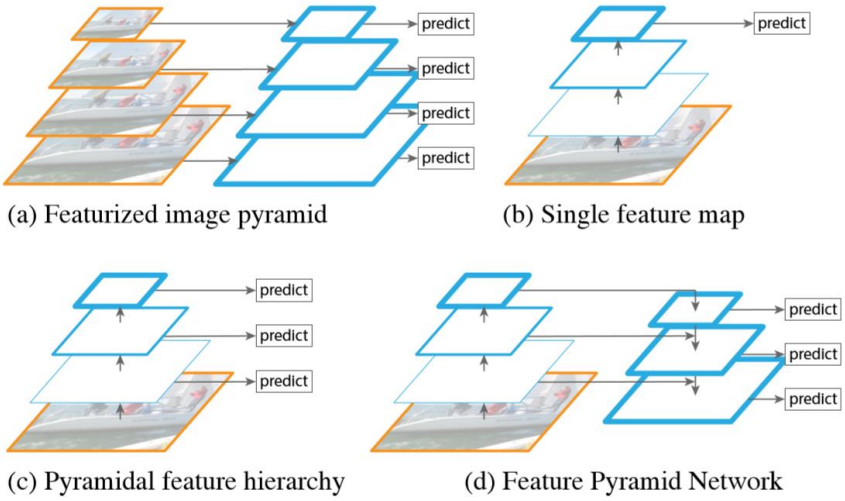


Figure 22: (FPN architecture) pyramid alternatives Mask R-CNN **[101]**

So, the overall structure can be illustrated by the following figure 23.

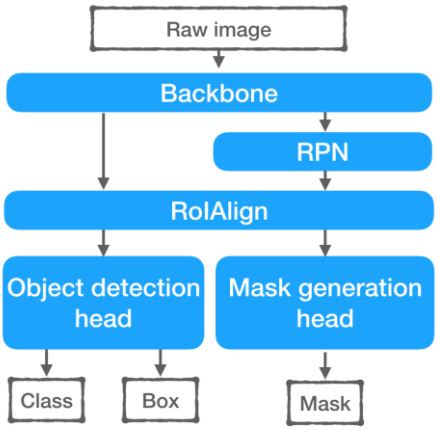


Figure 23: Mask R-CNN **[101]**.

2.2.3.One-stage detection:

One-stage detection methods are one of the most recent advancements in the field of speedy and precise detection, and they are summarized as follows:

2.2.3.1. Yolo (you only look one):

YOLO (You Only Look Once) is a real-time detector created by Redmon et al [50]. Its distinguishing feature is that it approaches detection as a regression problem. It is a one-stage detector that does both evaluation and detection at the same time. It's a single neural network that predicts multiple bounding boxes and class probabilities for each box at the same time. The YOLOv1 uses Darknet, which is an open-source neural network framework written in C and CUDA [51]. The main idea of how this network works is the algorithm divides the image into grids and runs the image classification and localization algorithm on each of the grid cells [52]. For example, we have an input image of size (256 × 256). We place a (3× 3) grid on the image (see Fig. 24).

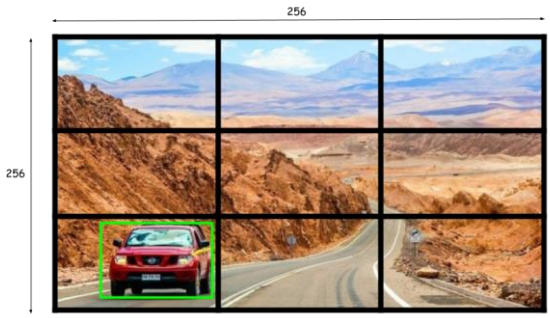


Figure 24: Grid (3 x 3) representation of the image

In Figure 25, the image is divided into 9 squares, and therefore the output matrix is 9 matrices, each with a depth of (c+5) which describe the center coordinates, the dimensions, the objectness score and C class confidences for each bounding box.

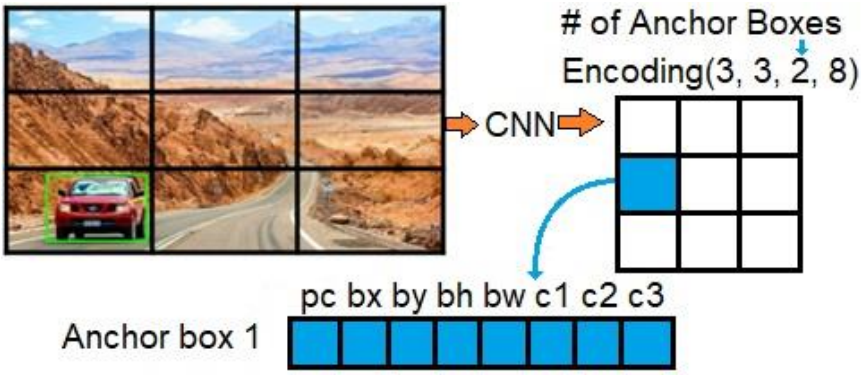


Figure 25: Output structure in YOLO

The YOLO [52] model is the first attempt to construct a rapid real-time object detector (see Table 1 for architecture details).

However, YOLO had certain limitations: it could only detect up to two things at a time, making it difficult to detect small or cluttered objects [50]. only the last feature map was used for prediction, which was not suitable for predicting objects at multiple scales and aspect ratios.

Type	Filters	Size/Stride	Activation	Output	
Convolutional	64	$7 \times 7/2$	LReLU & BN	$224 \times 224 \times 64$	
Maxpool		$2 \times 2/2$		$112 \times 112 \times 64$	
Convolutional	192	$3 \times 3/2$	LReLU & BN	$112 \times 112 \times 192$	
Maxpool		$2 \times 2/2$		$56 \times 56 \times 192$	
Convolutional	128	1×1	LReLU & BN	$56 \times 56 \times 128$	
Convolutional	256	3×3	LReLU & BN	$56 \times 56 \times 256$	
Convolutional	256	1×1	LReLU & BN	$56 \times 56 \times 256$	
Convolutional	512	3×3	LReLU & BN	$56 \times 56 \times 512$	
Maxpool		$2 \times 2/2$		$28 \times 28 \times 512$	
Convolutional	256	1×1	LReLU & BN	$28 \times 28 \times 256$	$\times 4$
Convolutional	512	3×3	LReLU & BN	$28 \times 28 \times 512$	
Convolutional	512	1×1	LReLU & BN	$28 \times 28 \times 512$	
Convolutional	1024	3×3	LReLU & BN	$28 \times 28 \times 1024$	
Maxpool		$2 \times 2/2$		$14 \times 14 \times 1024$	
Convolutional	512	1×1	LReLU & BN	$14 \times 14 \times 512$	$\times 2$
Convolutional	1024	3×3	LReLU & BN	$14 \times 14 \times 1024$	
Convolutional	1024	3×3	LReLU & BN	$14 \times 14 \times 1024$	
Convolutional	1024	$3 \times 3/2$	LReLU & BN	$7 \times 7 \times 1024$	
Convolutional	1024	3×3	LReLU & BN	$7 \times 7 \times 1024$	
Convolutional	1024	3×3	LReLU & BN	$7 \times 7 \times 1024$	
Connected				$1 \times 1 \times 4096$	
Connected			$7 \times 7 \times 30$		

Table 1: YOLOv1 model.

2.2.3.1.1. IOU (Intersection over union):

Is a metric for measuring the accuracy of an object detection model. used to describe the extent of overlap of two boxes. where we train a model to output a box that fits perfectly around an object. For example, in the image below, we have a green box, and a blue box. The green box represents the correct box, and the blue box represents the prediction from our model. If the prediction is completely correct, (IOU = 1). The lower the (IOU), the worse the prediction result [53].

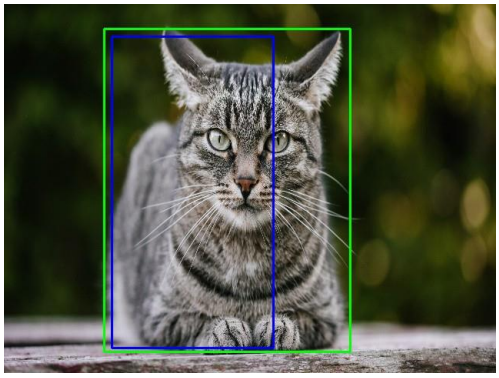
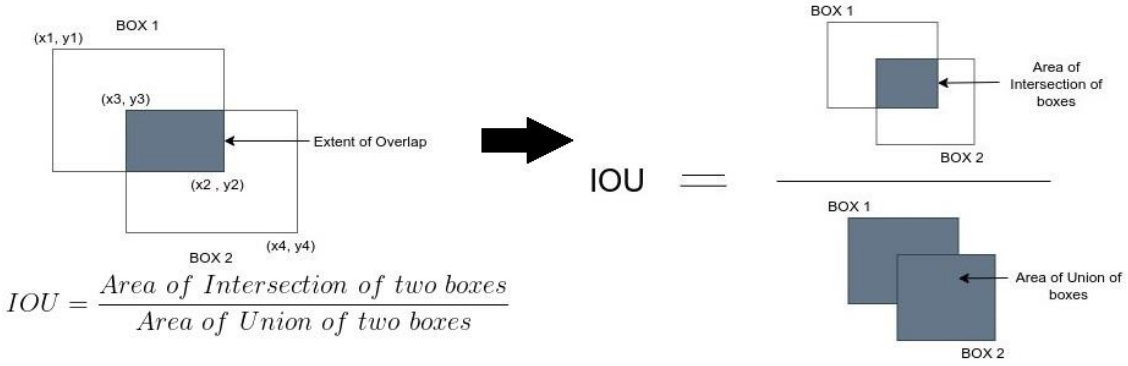


Figure 26: Intersection over union.

Let us assume that box 1 is represented **by [x1, y1, x2, y2]**, and box 2 is represented **by [x3, y3, x4, y4]** [53]. (**Figure 27** shows the calculation.)



$$IOU = \frac{Area\ of\ Intersection\ of\ two\ boxes}{Area\ of\ Union\ of\ two\ boxes}$$

Figure 27: Calculation (Intersection over union) **[53]**.

2.2.3.1.2. Loss Function Explanations:

The location x, y, and size w, h of bounding boxes, as well as the objectness $p(C)$ (or confidence) and class probabilities C, determine the multi-part loss function. Two gain factors $(\lambda_{coord}\ and\ \lambda_{noobj})$ are used to control the contribution of each part to the total loss. During the training, the function utilized to optimize is [52].

$$\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right]$$
$$+\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}\left[(\sqrt{w_i}-\sqrt{\hat{w}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2\right]$$
$$+\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{obj}\left(C_{ij}-\hat{C}_{ij}\right)^2+\lambda_{noobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathbb{1}_{ij}^{noobj}\left(C_{ij}-\hat{C}_{ij}\right)^2$$
$$+\sum_{i=0}^{s^2}\sum_{c\in classes}\mathbb{1}_{i}^{obj}\left(p_i(c)-\hat{p}_i(c)\right)^2, \quad (11)$$

It calculates the difference between true and forecasted parameters' Mean Squared Error (MSE) [52]. 1_i^{obj} indicates whether cell i includes an item, and 1_{ij}^{obj} indicates whether the grid cell i's j-th bounding box predictor is a contender for the prediction. The λ parameters are used to increase the loss from bounding box coordinate and to decrease the loss from confidence predictions in boxes that don't contain objects [54].

2.2.3.1.3. Non-max suppression:

This method is used to "suppress" the less likely bounding boxes and only maintain the best [55]. is a technique used primarily in object detection that aims to select the best bounding box out of a set of overlapping boxes, ensuring that the algorithm will recognize the required part once and not more than once, then the algorithm will determine the efficiency of each square if it covers the object in the image well, and the efficiency is determined by the IOU mentioned above, and it deletes the minimum values of the non-maximum suppression, i.e. it keeps only the highest values of the non-maxim [55].

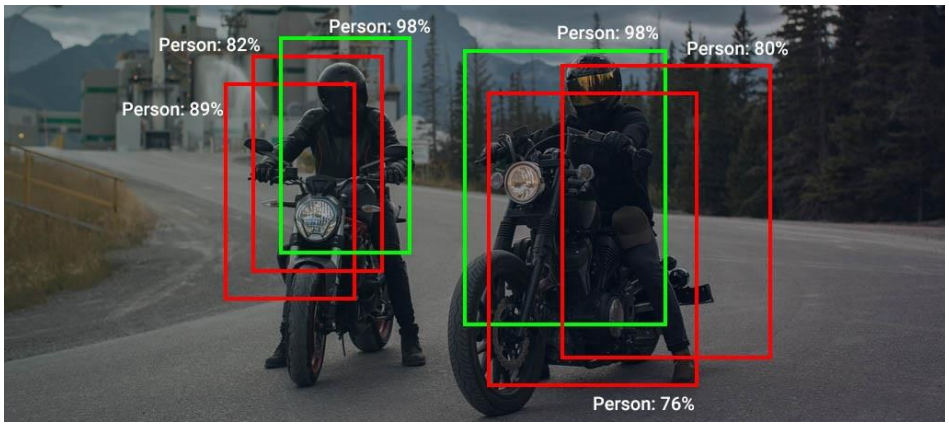


Figure 28: Discovered more than once (human) **[55]**.

2.2.3.1.4. Anchor boxes:

A set of predetermined bounding boxes of a specific height and breadth are known as anchor boxes. These boxes are often chosen depending on object sizes in your training datasets to capture the scale and aspect ratio of various object classes you want to detect. The number of tiled anchor boxes equals the number of network outputs. The network produces predictions for all outputs [56].

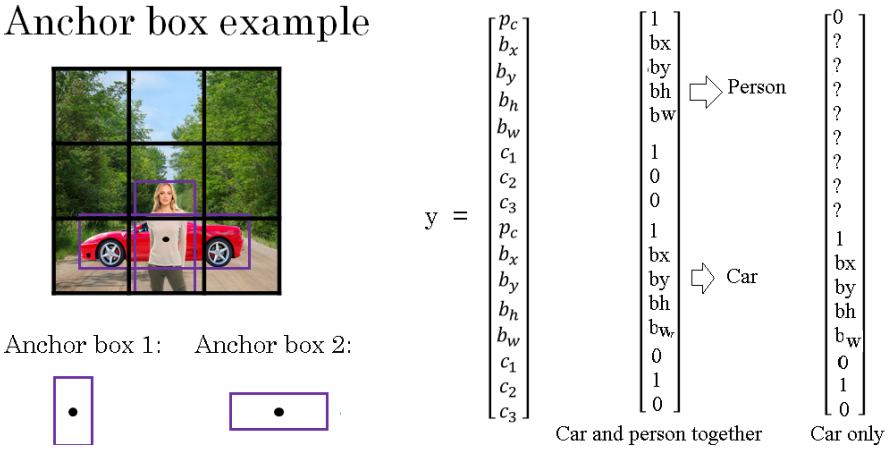


Figure 29: Example Anchor box

Note: Because there are instances when an object does not fit in the Anchor boxes, it is vital to select appropriate boxes because this error may damage the output results and accuracy (IOU) [56].

2.2.3.2. SSD (single shot detector):

SSD [57] is one of the earliest attempts at using convolutional neural networks in pyramidal feature hierarchies for effective identification of objects of varied sizes, proposed by Liu et al. shortly after the YOLO technique is introduced [58]. functions as a single-stage multiple-class object detector that regresses class confidences and bounding boxes from a fixed collection of bounding boxes of various sizes and scales. SSD incorporates concepts from RPN of Faster R-CNN and YOLO, as well as multiscale convolutional layers for feature extraction, to improve detection speed while maintaining accuracy. Per feature map location, SSD discretizes the output space of bounding boxes into a set of default boxes at various aspect ratios and scales. The network generates scores for the existence of each object type in each default box at the moment of prediction, as well as modifications to the box to better reflect the object shape. In addition, to handle objects of varied sizes naturally, the network combines predictions from many

feature maps with different resolutions. SSD achieved comparable detection accuracy with Faster R-CNN but enjoyed the ability to do real-time inference [57].

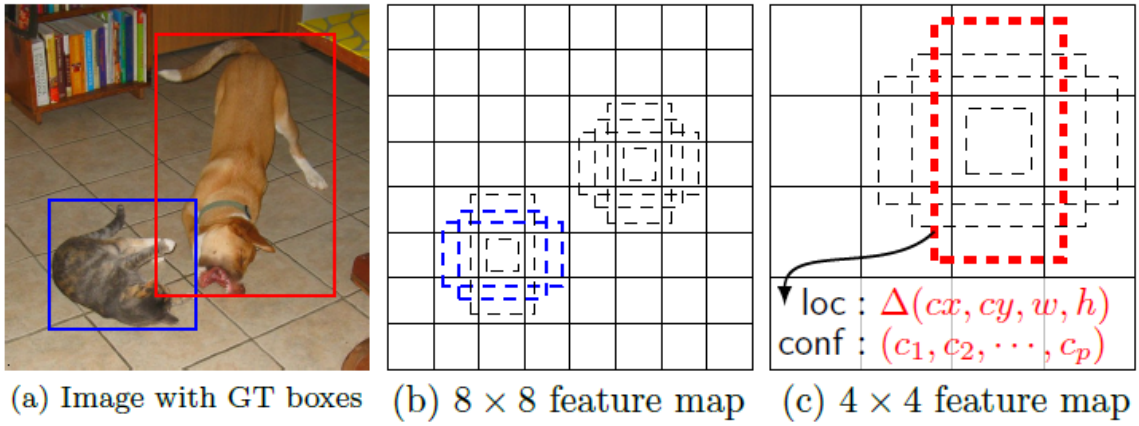


Figure 30: SSD framework **[99]**.(a): The photos with their respective bounding boxes are the input to SDD. (b): Default boxes with various aspect ratios correlate to a smaller region in fine-grained feature maps. (c): For coarse-grained feature maps these boxes are bigger and thus more suitable for larger objects.

The anchor boxes on various levels are rescaled such that a single feature map is solely responsible for objects at a single scale. An anchor box's width, height, and center placement are all normalized to be (0, 1). Every position (i, j) of the l-th feature layer of size $m \times h$ has a linear scale value proportionate to its layer level and 6 possible width-to-height ratios associated with it [58]. There are a total of 6 anchor boxes per feature cell, where the scale at each level is

$$\boldsymbol{s_l = s_{min} + \frac{s_{max} - s_{min}}{L - 1}(l - 1)} \qquad (12)$$

Where the level index $l = 1, \ldots, L$, the aspect ratios r ∈ {1,2,3,1/2,1/3}, with an additional scale $s_l' = \sqrt{S_l S_l + 1}$ when r = 1. Each box's width and height can therefore be calculated as $w_l^r = s_l\sqrt{r}$ and $h_l^r = s_l/\sqrt{r}$, with the center location $\left(x_l^i, y_l^i\right) = (\frac{\text{i+0,5}}{m}, \frac{j+0.5}{n})$. The model generates four anchor box offsets and C class probabilities for each of k anchor boxes at each location of each feature map, yielding $k.m.n(c + 4)$ outputs [57].

The loss function seems a lot like the one in YOLO. With some minor alterations, it's defined as the sum of a localization loss and a classification loss.

TYPE / NAME	GRID SIZE	KERNEL SIZE
Conv 6	19×19	3x3x1024
Conv 7	19×19	1x1x1024
Conv 8_2	10×10	1x1x256 3x3x512-s2
Conv 9_2	5×5	1x1x128 3x3x256-s2
Conv 10_2	3×3	1x1x128 3x3x256-s1
Conv 11_2	1×1	1x1x128 3x3x256-s1

Table 2: SSD model.

2.2.3.3. Retina NET:

(Lin et al). presented RetinaNet as another popular one-stage object detection in 2018 [59]. The key innovation of the RetinaNet method is the addition of a new loss function, called focal loss, to provide robustness against class imbalance. RetinaNet addresses the great disparity between the background, which includes no items, and the foreground, which contains things of interest, by modifying the usual cross entropy loss function such that it down-weights the loss assigned to well-classified examples [60].

The cross entropy (CE) function is used over the confidence scores as below: $$CE(p) = -log(p) \quad (13)$$

The following relationship describes focal loss as a function:

$$FL(p) = -\alpha(1-p)^{\gamma}\log(p) \quad (14)$$

where γ is a focusing parameter and α is a correction factor. Hard sample losses are more important than simple sample losses when the parameter γ is greater than zero [60]. (See Figure 31).

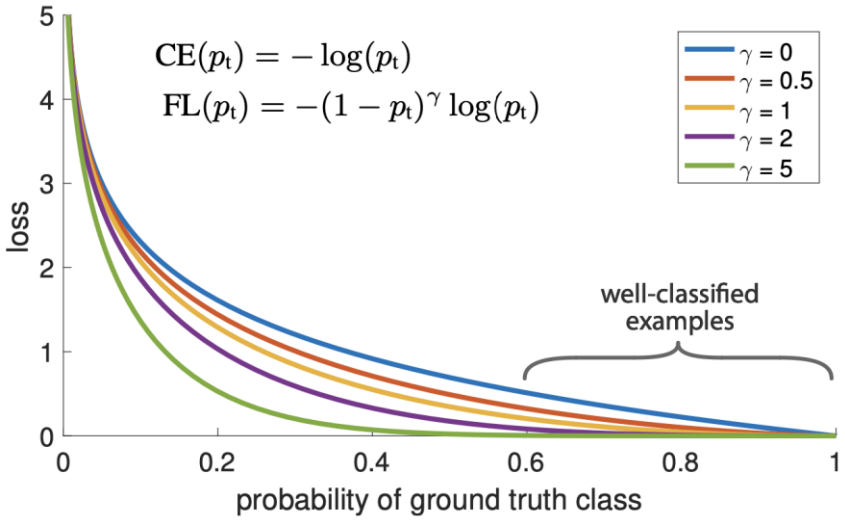


Figure 31: Model performance in terms of loss values with various focusing parameter values, While α=1 **[60]**.

Furthermore, feature pyramid networks were employed to detect multi-scale objects at various layers of feature maps [59]. (See Figure 32).

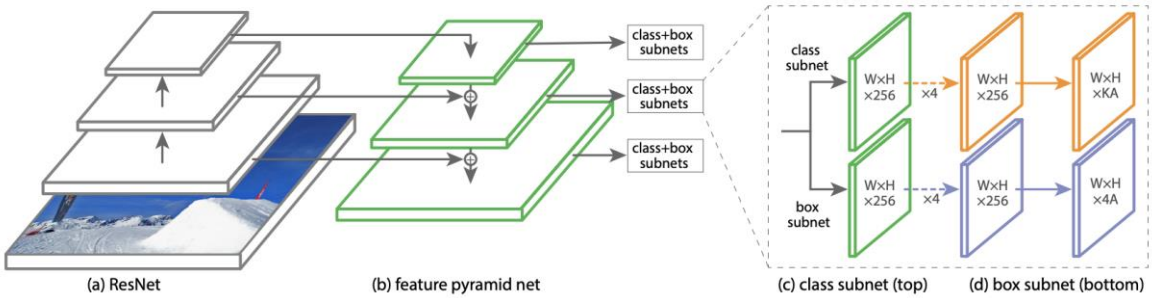


Figure 32: The RetinaNet network architecture used a Feature Pyramid Network on top of the feed-forward ResNet architecture.

2.2.3.4. Yolo v2:

Later, Redmon et al [61]. refined their work YOLO by creating YOLOv2, which uses an entirely new feature extractor backbone called Darknet19, which has 19 convolutional layers. To predict bounding boxes in YOLOv2, fully connected layers are deleted and only convolutional layers are used [61]. Batch normalization is applied to all convolutional layers, resulting in a considerable speedup in the learning process and an increase in the mAP [62]. which improved detection performance while keeping inference speed in real time. The weights obtained by YOLOv2 were more sensitive to capturing fine-grained information because it used a more powerful deep convolutional backbone architecture that

was pretrained on higher resolution photos from ImageNet (from 224 224 to 448 448). In addition, the anchor method utilized in SSD was a source of inspiration.

The improvement aspects are [63]:

- Batch normalization.
- High resolution classifier.
- Anchor Boxes.
- Fine-grained features.
- Multi-scale training.
- Darknet-19.

Given an anchor size (p_w, p_h)at a certain grid cell with its left corner at (C_x, C_y) the model predicts the offset scale, (t_x, t_y, t_w, t_h) and a confidence prediction representing the IoU between the predicted box and any ground truth box. The corresponding predicted bounding box b has center (b_x, b_y) and size (b_w, b_h) [62]. (See Figure 33).

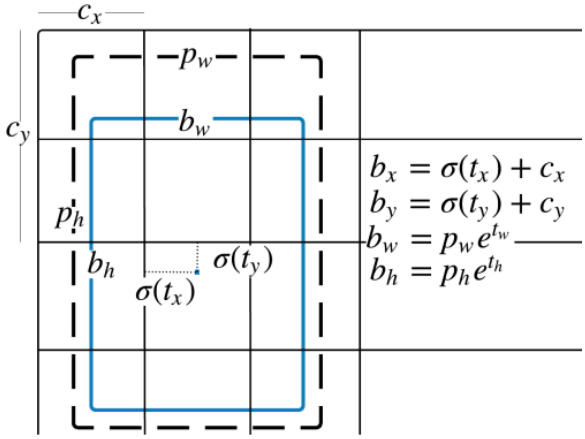


Figure 33: Dimension priors and position prediction for bounding boxes **[61]**.

Although fine-grained characteristics from a previous layer are passed to the output detection layer, the detection is still done at the final coarse-grained layer, which misses many of the smaller items [62]. (See Table 3).

Type	Filters	Size/Stride	Activation	Output
Convolutional	32	$3 \times 3/1$	LReLU & BN	$224 \times 224 \times 32$
Maxpool		$2 \times 2/2$		$112 \times 112 \times 32$
Convolutional	64	3×3	LReLU & BN	$112 \times 112 \times 64$
Maxpool		$2 \times 2/2$		$56 \times 56 \times 64$
Convolutional	128	3×3	LReLU & BN	$56 \times 56 \times 128$
Convolutional	64	1×1	LReLU & BN	$56 \times 56 \times 64$
Convolutional	128	3×3	LReLU & BN	$56 \times 56 \times 128$
Maxpool		$2 \times 2/2$		$28 \times 28 \times 128$
Convolutional	256	3×3	LReLU & BN	$28 \times 28 \times 256$
Convolutional	128	1×1	LReLU & BN	$28 \times 28 \times 128$
Convolutional	256	3×3	LReLU & BN	$28 \times 28 \times 256$
Maxpool		$2 \times 2/2$		$14 \times 14 \times 256$
Convolutional	512	3×3	LReLU & BN	$14 \times 14 \times 512$
Convolutional	256	1×1	LReLU & BN	$14 \times 14 \times 256$
Convolutional	512	3×3	LReLU & BN	$14 \times 14 \times 512$
Convolutional	256	1×1	LReLU & BN	$14 \times 14 \times 256$
Convolutional	512	3×3	LReLU & BN	$14 \times 14 \times 512$
Maxpool		$2 \times 2/2$		$7 \times 7 \times 512$
Convolutional	1024	3×3	LReLU & BN	$7 \times 7 \times 1024$
Convolutional	512	1×1	LReLU & BN	$7 \times 7 \times 512$
Convolutional	1024	3×3	LReLU & BN	$7 \times 7 \times 1024$
Convolutional	512	1×1	LReLU & BN	$7 \times 7 \times 512$
Convolutional	1024	3×3	LReLU & BN	$7 \times 7 \times 1024$
Convolutional	1000	1×1		$7 \times 7 \times 1000$
Avgpool		Global		1000
Softmax				

Table 3: YOLOv2 model.

2.2.3.5. Yolo v3:

YOLOv3 is a new version of the YOLO baseline algorithm developed by Redmon et al [64]. Because there may be occasions where a cell contains more than one class, YOLOv3 uses the logistic loss function instead of the softmax layer to enable multi-class detection.

In terms of detection performance, YOLOv3 surpasses SSD and R-FCN, as well as Faster R-CNN and RetinaNet in terms of detection time, as shown in Figure 34. There are a few key differences between YOLOv3 and previous algorithms in terms of detection performance and/or detection time [64].

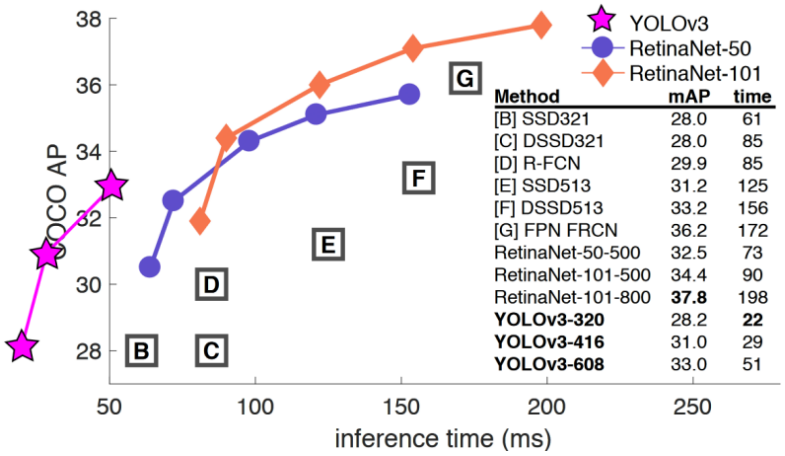


Figure 34: Comparison of YOLOv3 and the other state-of-the-art algorithms **[64]**.

The most significant modifications are as follows [64]:

- Bounding Box Prediction.
- Class Predictions.
- Predictions across scales.
- Feature Extractor.
- Darknet53.

2.2.3.5.1. Backbone:

The Darknet backbone has been upgraded to Darknet53, which has 53 convolutional layers with batch normalization and Leaky-ReLU activation after each [65]. Figure 35.

2.2.3.5.2. Feature Pyramids:

For diverse sized objects, YOLOv3 detects at three different scales. Each of these layers is connected to the other utilizing FPN architecture in a top-down way. Shallower layers can use the semantic information gathered in deeper layers because to this connection [66].

2.2.3.5.3. Loss function:

Calculates the objectness score with logistic regression for each bounding box [67].

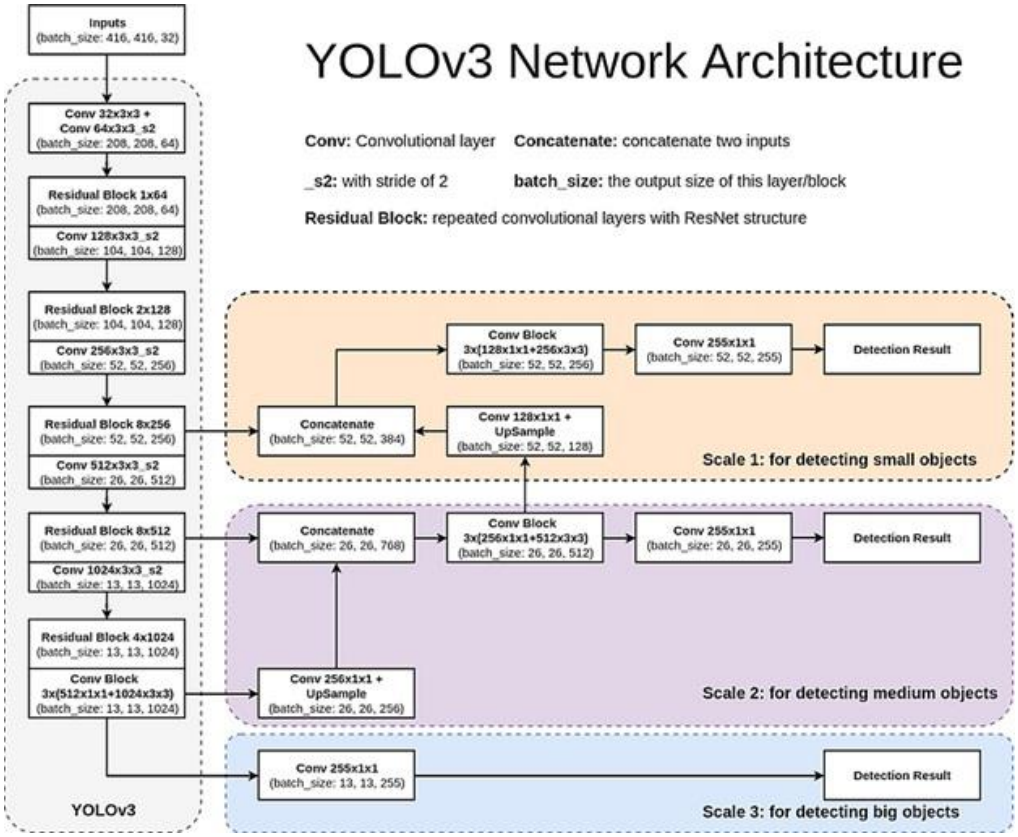


Figure 35: YOLOv3 model **[98]**.

2.2.3.6. Tiny-yolov3:

Tiny-YOLOv3 is a reduced and scaled-down version of YOLOv3. Despite the fact that Tiny-YOLOv3 has fewer layers than YOLOv3, the model's accuracy is nearly identical to that of its larger counterpart when high frame rates are considered. Tiny-YOLOv3 has only 13 convolutional layers and 8 max-pool layers, hence it takes far less memory to execute than YOLOv3. The main distinction between YOLOv3 and TinyYOLOv3 is that the former can detect items at three different scales, whereas the latter can only detect objects at two scales. Apart from these differences, both variations work in the same way [68].

Tiny-YOLOv3 has a significantly lower number of convolutional layers than YOLOv3. Tiny-YOLOv3 features only 13 convolutional layers in its basic structure, with a total of 23 layers Table 4. Tiny-YOLOv3 uses a restricted number of 1 x 1 and 3 x 3 kernels to extract the characteristics. Unlike YOLOv3, which uses stride 2 convolutional layers for down sampling, Tiny-YOLOv3 employs the pooling layer [68]. TinyYOLOv3's convolutional layer structure is comparable to that of YOLOv3. Figure 36.

Layer	Type	Filters	Size/Stride	Input	Output
0	Convolutional	16	3 × 3/1	416 × 416 × 3	416 × 416 × 16
1	Maxpool		2 × 2/2	416 × 416 × 16	208 × 208 × 16
2	Convolutional	32	3 × 3/1	208 × 208 × 16	208 × 208 × 32
3	Maxpool		2 × 2/2	208 × 208 × 32	104 × 104 × 32
4	Convolutional	64	3 × 3/1	104 × 104 × 32	104 × 104 × 64
5	Maxpool		2 × 2/2	104 × 104 × 64	52 × 52 × 64
6	Convolutional	128	3 × 3/1	52 × 52 × 64	52 × 52 × 128
7	Maxpool		2 × 2/2	52 × 52 × 128	26 × 26 × 128
8	Convolutional	256	3 × 3/1	26 × 26 × 128	26 × 26 × 256
9	Maxpool		2 × 2/2	26 × 26 × 256	13 × 13 × 256
10	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
11	Maxpool		2 × 2/1	13 × 13 × 512	13 × 13 × 512
12	Convolutional	1024	3 × 3/1	13 × 13 × 512	13 × 13 × 1024
13	Convolutional	256	1 × 1/1	13 × 13 × 1024	13 × 13 × 256
14	Convolutional	512	3 × 3/1	13 × 13 × 256	13 × 13 × 512
15	Convolutional	255	1 × 1/1	13 × 13 × 512	13 × 13 × 255
16	YOLO				
17	**Route 13**				
18	Convolutional	128	1 × 1/1	13 × 13 × 256	13 × 13 × 128
19	Up-sampling		2 × 2/1	13 × 13 × 128	26 × 26 × 128
20	**Route 19 8**				
21	Convolutional	256	3 × 3/1	13 × 13 × 384	13 × 13 × 256
22	Convolutional	255	1 × 1/1	13 × 13 × 256	13 × 13 × 256
23	YOLO				

Table 4: Tiny-YOLOv3 model.

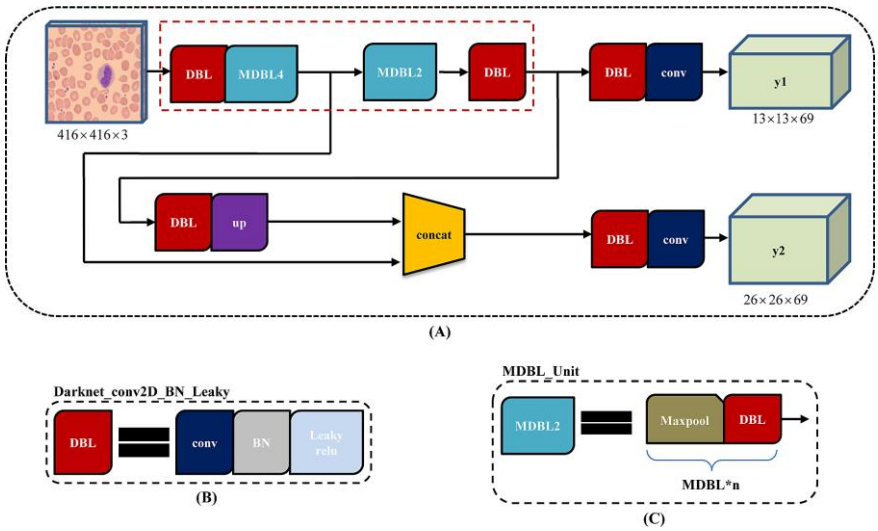


Figure 36: Architecture of Tiny-YOLOv3 **[102]**.

2.2.3.7. Yolo v4:

Yolov4 was announced in 2020 by Bochkovskiy et al [69]., and it is an upgraded version of the YOLOv3 algorithm, with a mAP improvement of up to 10% and a 12 percent increase in the number of frames per second. Darknet, the same framework utilized for its predecessors, is also employed to construct this new structure [70]. Figure 37 shows a detailed performance comparison.

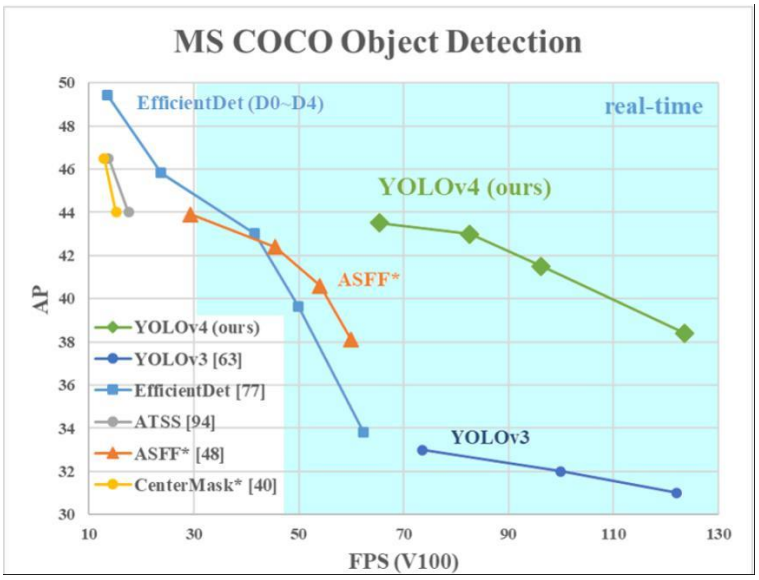


Figure 37: Comparison of YOLOv4 and the other state-of-the-art algorithms **[111]**.

Figure 37 shows that YOLOv4 outperforms YOLOv3 and EfficientDet in terms of inference time. In their publication, the authors of YOLOv4 present a series of contributions dubbed a "bag of freebies." There are a number of things that may be done to increase the model's performance without adding to the inference time lag [70].

2.2.3.7.1. Backbone:

YOLOv4 employs CSP-Darknet-53, a variation of the Darknet framework's Cross Stage Partial Network (CSPNet). By partially concatenating the top and bottom layers of the network, the suggested topology merely connects feature maps. On the ImageNet dataset [71], this method reduces computation costs by 20% while maintaining at least the same accuracy [72].

2.2.3.7.2. SPP in YOLOv4:

The backbone of YOLOv4 is Spatial Pyramid Pooling (SPP) [73]. SPP is another option for dealing with objects of various sizes. The SPP block takes input feature maps and feeds them into three parallel max pooling layers with variable scales and strides. Originally, the approach was employed to generate fixed size output for different sized inputs to feed the output to a fully connected layer [74]. The three layers are then joined together to create multi-scaled input features.

2.2.3.7.3. Activation function:

In the feature extraction backbone, YOLOv4 employs Mish activation on several convolutional layers, in addition to Leaky-ReLU activation, which was originally utilized in YOLOv3 [75]. $f(x) = x\,tanch\big(softplus(x)\big)$ can define Mish, which is a continuously differentiable activation function. On AP with CSP-Darknet-53 backbone, it outperforms Leaky-ReLU by 2.1 percent in the MS-COCO dataset [70].

2.2.3.7.4. Feature Pyramids:

In YOLOv4, a Path Aggregation Network (PANet) is added to the generic Feature Pyramid Network, allowing for the use of deeper features for the preceding outputs. PANet merely adds a bottom-up path that improves communication between the lower and upper layers [76]. Figure 38.

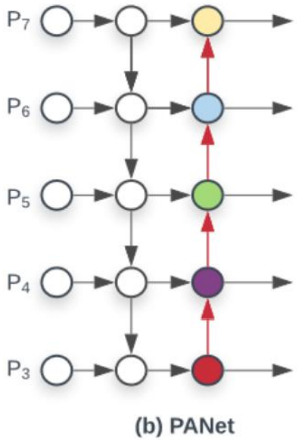


Figure 38: PANet advances this approach with an additional bottom-up connection **[76]**.

2.2.3.7.5. Data Augmentation:

YOLOv4 is educated in a unique way. Rather of using the same photos at each epoch, it crops four separate images and merges them into a single network-sized image, so the network encounters a fresh input every time it runs. Furthermore, four separate scenes can be used to learn batch normalization parameters in each image. As a result, there is less need for bigger mini-batch sizes. Mosaic is the name given to this procedure [77]. Figure 39.

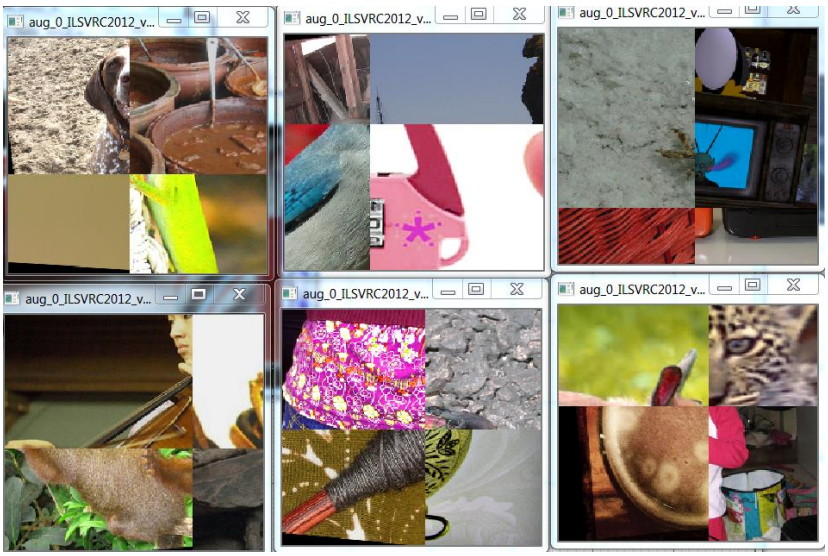


Figure 39: Mosaic data augmentation **[111]**.

2.2.3.8. Yolo v5:

YOLOv5 was only released on GitHub in 2020, with no accompanying paper. It differs from all previous releases in that it is a PyTorch implementation rather than a fork of Darknet [78]. According to their repository, YOLOv5 outperforms EfficientDet, another state-of-the-art algorithm, by roughly 10% AP in MS-COCO [70] dataset in similar depth networks, resulting in a similar level of FPS. Figure 40 shows a detailed comparison of various algorithms.

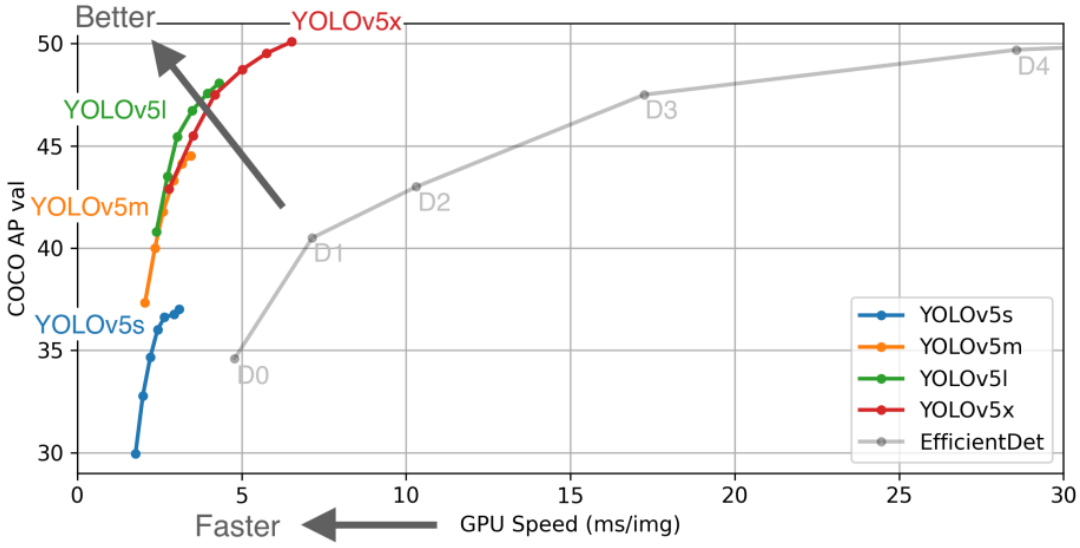


Figure 40: Comparison of YOLOv5 and EfficientDet with different network sizes **[112]**.

Mosaic data augmentation and autolearning bounding box anchors are two of the most significant enhancements. YOLOv5 is significantly faster and lighter

than YOLOv4, with accuracy comparable to the YOLOv4 test. YOLOv5 feeds training data through a data loader with each training batch, which augments data online. Scaling, color space changes, and mosaic augmentation are the three types of augmentations performed by the data loader. Mosaic data augmentation is the most novel of all, as it mixes four photos into four random-ratio tiles [78].

2.2.3.8.1. Backbone: YOLOv5 uses CSPNet like YOLOv4.

2.2.3.8.2. SPP: YOLOv5 uses SPP as in YOLOv4.

2.2.3.8.3. Activation function: Unlike YOLOv4, YOLOv5 utilizes the Leaky-ReLU function after all convolutional layers, similar to YOLOv3.

2.2.3.8.4. Feature Pyramids: Just like YOLOv4, YOLOv5 adds a Path Aggregation Network (PANet) to the generic Feature Pyramid Network [78].

2.2.3.8.5. Focus (also called by DepthToSpace): Focus is a basic strategy that seeks to speed up the process by lowering the input resolution and convolution operation cost. This method simply reduces a tensor's width and height while increasing the number of channels [79].

Note: Because YOLOv5 is open source, there is no paper or documentation that explains the algorithm.

2.2.4. Comparison of Faster-RCNN, YOLO, and SSD for Real-Time:

The following experiment compared the speed of YOLO, SSD, and Faster-RCNN when they were trained on the same data set. Regional proposal and classification are two steps of the R-CNN detector. The detector first finds 2,000 boxes that represent the target object's region using Selective Search. Following that, all Bounding Boxes are classified using CNN. The processing speed slows as the quantity of calculation cost rises. Faster RCNN performs object detection once in the output feature map after going through CNN to adjust for the processing speed. The Region Proposal Network is utilized by Faster RCNN to address the bottleneck produced by the selective search algorithm. Faster R-CNN is 200 times faster than R-CNN in terms of processing performance [80]. Object detection with RCNN family has the drawback of sluggish processing speed,

making it unsuitable for real-time applications, unlike Yolo and SSD which are faster because they both use one-stage detection algorithms [81].

Pascal VOC2007 has 20 categories with 5k images in the train Val set and 5k images in the test set.

#	Algorithm	Ref	Detection Model	Train data	Test Data	FPS	mAP %
1	Faster RCNN	(Shaoqing R., et al., 2015)	VGG16	VOC 07++12 + COCO	VOC12	-	75.9
2	Faster RCNN	(Shaoqing R., et al., 2015)	VGG16	VOC 07 +12 + COCO	VOC07	-	78.8
3	Faster RCNN	(Joseph R., et al., 2016)	VGG16	VOC 07+12	VOC07	7	73.2
4	Faster RCNN	(Joseph R., et al., 2016)	ZF	VOC 07+12	VOC07	18	62.1
5	Faster RCNN	(Wei Liu, et al., 2016)	-	VOC 07++12	VOC12	-	70.4
6	Faster RCNN	(Wei Liu, et al., 2016)	-	VOC 07++12 + COCO	VOC12	-	75.9
7	Faster RCNN	(Dai, J., et al., 2016)	Resnet101	VOC 07+12	VOC12	2	76.4
8	Faster RCNN	(Dai, J., et al., 2016)	Resnet101	VOC 07+12 + COCO	VOC12	0.3	85.6
9	Faster RCNN	(Dai, J., et al., 2016)	Resnet101	VOC 07++12	VOC12	2	73.8
10	Faster RCNN	(Dai, J., et al., 2016)	Resnet101	VOC 07++12 + COCO	VOC12	0.3	83.8
11	Faster RCNN	(Jonathan, H.., et al., 2017)	Resnet101	ImageNet	ImageNet	-	32
12	YOLO1	(Joseph R., et al., 2016)	YOLO	VOC 07+12	VOC07	45	63.4
13	YOLO1	(Joseph R., et al., 2016)	VGG16	VOC 07+12	VOC07	21	66.4
14	YOLO1	(Joseph R., et al., 2016)	YOLO	VOC 07+12	VOC12	-	57.9
15	YOLO2	(Joseph R., et al., 2016)	Darknet19	VOC 07+12	VOC07	40	78.6
16	YOLO2	(Joseph R., et al., 2016)	Darknet19	VOC 07++12	VOC12	-	73.4
17	YOLO3	(Redmon and Farhadi, 2018)	Darknet53	COCO	COCO	19	57.9
18	SSD300	(Wei Liu, et al., 2016)	VGG16	VOC 07++12	VOC12	-	72.4
19	SSD300	(Wei Liu, et al., 2016)	VGG16	VOC 07++12 + COCO	VOC12	-	77.5
20	SSD300	(Wei Liu, et al., 2016)	VGG16	VOC 07+12	VOC07	46	74.3
21	SSD300*	(Wei Liu, et al., 2016)	VGG16	VOC 07++12	VOC12	-	75.8
22	SSD512	(Wei Liu, et al., 2016)	VGG16	VOC 07++12	VOC12	-	74.9
23	SSD512	(Wei Liu, et al., 2016)	VGG16	VOC 07++12 + COCO	VOC12	-	80
24	SSD512	(Wei Liu, et al., 2016)	VGG16	VOC 07+12	VOC07	19	76.8
25	SSD512*	(Wei Liu, et al., 2016)	VGG16	VOC 07++12	VOC12	-	78.5
26	SSD	(Jonathan, H., et al., 2017)	Inception V2	ImageNet	ImageNet	-	22

Table 5: comparison of Faster-RCNN, YOLO, and SSD for object detection **[114]**.

Table 5. represents previous studies' results for the three algorithms: Faster RCNN, YOLO, and SSD. The first column contains the algorithm name; the second column contains the reference number. The detection model is the used network structure. Train data and Test data are the used train set and test set [80]. FPS is the number of processed frames per second; mAP is the mean average precession mentioned in section 3.1.3. The (-) was used for values that were not mentioned in the reference. VOC07 and VOC12 are Pascal VOC 2007 and Pascal VOC 2012. VOC07++12 means using both train and evaluation sets in Pascal

VOC 2007 with the train set of Pascal VOC 2012 for training [81]. The number after algorithm name (300, 512) represents the dimensions of network input, (*) after algorithm name mean using data augmentation (generating more data from the original by applying some image transformations like rotating, scaling, adding noise, and other methods) for training data, the increase in training data usually requires more training steps too [82].

2.3. Conclusion:

In this chapter we discussed deep learning, as well as the neural network that relies on it. Then there was the object detection in the image. We concentrated on the convolutional neural network, which is the most important neural network in this sector. After that, the most significant ancient and modern techniques that resulted from this idea were then covered. then we analyzed the results of training conducted by researchers on the best models using.

CHAPTER 3: PROJECT DEVELOPMENT:

3. Project development:

In this chapter, we begin discussing the experimental part of the thesis. First, we will discuss selection criteria for methods and datasets. Then we will describe the selected methods, their parameters and the selected datasets. Finally, we will discuss postprocessing and evaluation.

3.1. Performance metrics:

3.1.1.Detection cases:

When the detector is used with the image to search for **people**, it can produce four different cases:

3.1.1.1. True Positive (TP):

Detecting the object while it is in the photo frame**.**

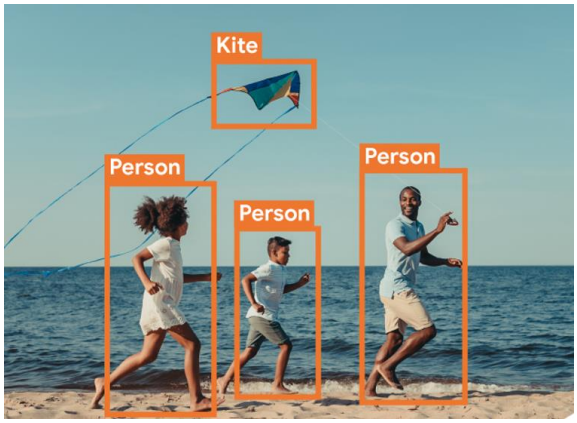


Figure 41: True Positive example **[113]**.

3.1.1.2. True Negative (TN):

The object is not detected in the empty image of the requested object**.**

Figure 42: True Negative example **[107]**.

3.1.1.3. False Positive (FP):

Detection of another object instead of the requested object.

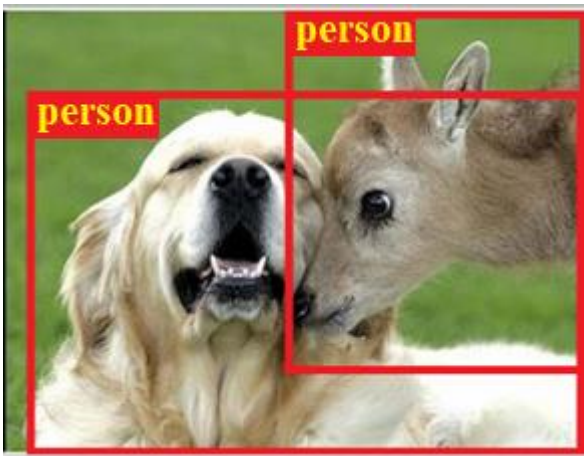


Figure 43: False Positive example.

3.1.1.4. False Negative (FN):

No detection of objects in an image with objects.

Figure 44: False Negative example **[108]**.

3.1.2.Average precision (AP):

It is calculated using area under the curve (AUC) of the Precision x Recall curve. by averaging the precision of all recall values between 0 and 1.

3.1.3.Mean Average Precision (MAP):

The compares the real box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

3.1.4.Recall:

Measures how good you find all the positives. For example, we can find 80% of the possible positive cases in our top K predictions.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths} \quad (15)$$

3.1.5.Precision:

The percentage of your predictions are correct.

$$Precision = \frac{TP}{\mathrm{TP+FP}} \quad (16)$$

3.2. Implementation:

3.2.1.Software Environment:

Python was chosen as the programming language because it is a high-level programming language that is simple to learn and code, making it a popular choice for constructing machine learning and deep learning algorithms [83].

CUDA and cuDNN were installed because they allow the algorithm to be trained on a GPU, which is faster and more efficient than training on a CPU [84].

3.2.2.Hardware Environment:

Table10 shows the hardware specs of the system on which the algorithm was trained and implemented.

System	Windows 10/2021
GPU	NVIDIA GT 630M 2GB
CPU	Intel Core I7 2generation
RAM	10GB

Table 6: Hardware pramétre.

3.2.3.Virtual environment:

It is the environment for running Python codes.

- PyCharm Community Edition 2021.3.2. [85].
- Jupyter Notebook (anaconda3). [86]
- Activate the environment using the activation codes in Python on CMD “python -m venv name-virtual “.

3.2.4.Preparation of the data:

- **Data set:** The program in this project was created to detect two different objects: a person and cat. The photos of the two categories were collected from various perspectives to form a dataset. which are divided into three groups: Training set (80%) and test set (10%) and Val (10%). After that, in order to train the model, the photos must be labeled and annotations must be created.
- **Labelling:** The labeling procedure entails creating an XML or CSV file for each frame that contains the names of the items displayed as well as the coordinates (in pixels) of their placements. Labeling is an opensource program that makes the task easier by allowing you to visualize the process graphically [87]. In Figure 45.

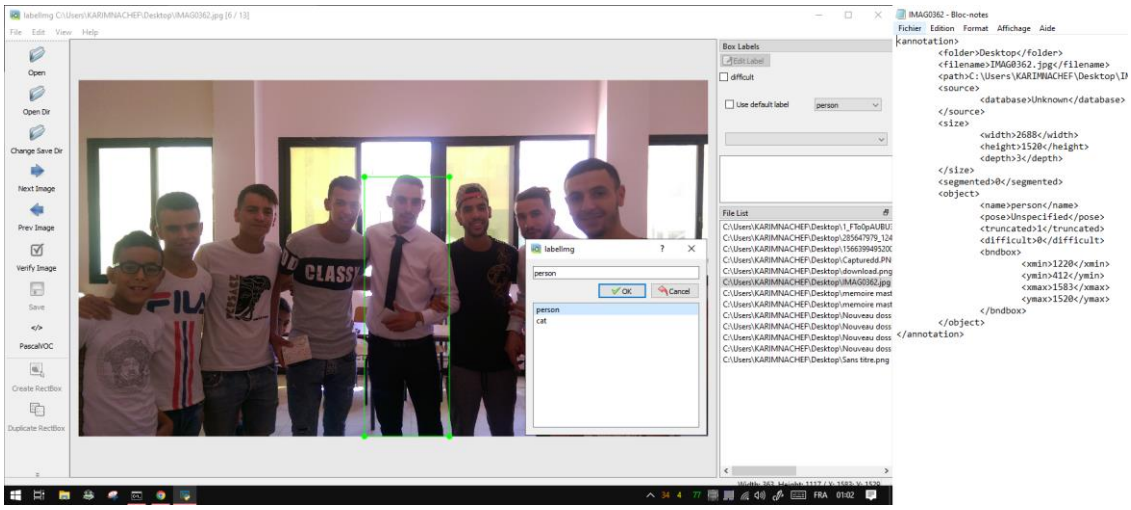


Figure 45: Process of labelling and an XML file example.

3.2.5.Object Detector:

A Convolutional Neural Network derived from the TensorFlow deep learning framework was utilized as the classifier. The model utilized is yolov3 [88], which was chosen after studying the characteristics of various algorithms. It has a simple and adaptable structure. It performs high-accuracy real-time detections as a result of its features. Furthermore, it is built using TensorFlow Object Detection [89], which is simple to use and understandable.

3.2.6. Yolov3 architecture:

We will discuss our model we're using in this project (i.e., Yolo). We retrain the yolo models to detect a person or cat by fine tuning the parameters of only the last three layers, as well as hyperparameters like learning rate and number of iterations used in the model. and by freezing all the parameters of the first 249 layers, in our model the first 249 layer are considered as feature extraction, while the last three layers are considered as classification and localization. The last three layers are layers responsible for outputting the object class to be detected, so the weights in these layers are updated during training. YOLOv3 has three final layers, the first has a dimension divided by 31 compared to the initial image, the second by 16 and the third by 8. Thus, starting from an image of size 480×640 pixels, the three features' maps output from the network will have respective sizes of 13×13, 26×26 and 52×52 pixels. It is in this sense that YOLOv3 predicts three levels of detail, to detect large, medium and small respectively. Figure 46.

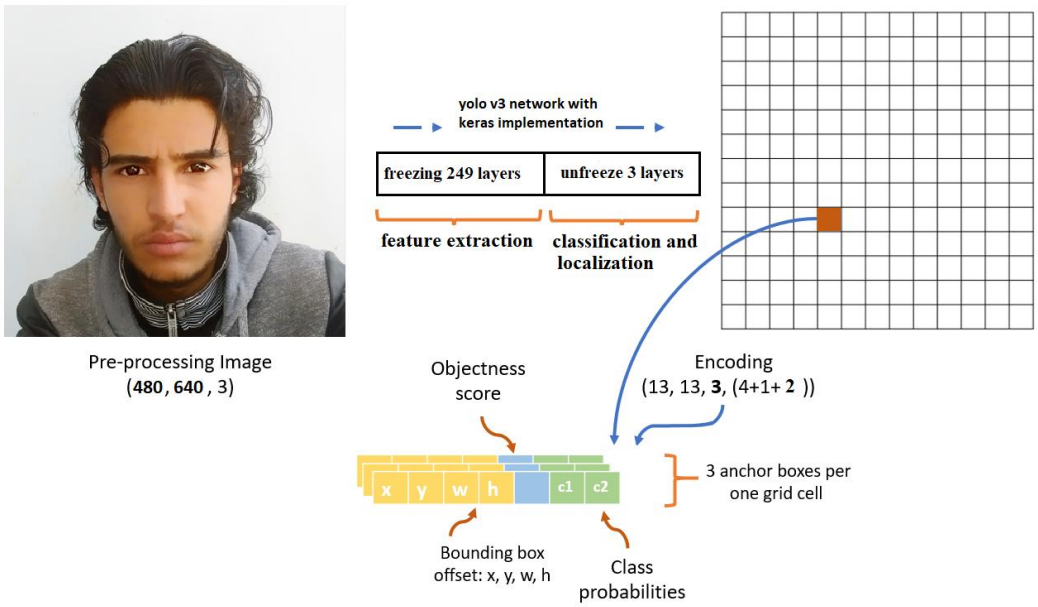


Figure 46: Model of YOLOv3.

figure 47 shows the detailed structure of our yolov3 model using keras.

```
Layer (type)                 Output Shape              Param #     Connected to
==================================================================================
input_1 (InputLayer)         (None, 480, 640, 3) 0
__________________________________________________________________________________
conv2d_1 (Conv2D)            (None, None, None, 3 864         input_1[0][0]
__________________________________________________________________________________
batch_normalization_1 (BatchNor (None, None, None, 3 128         conv2d_1[0][0]
__________________________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, None, None, 3 0           batch_normalization_1[0][0]
__________________________________________________________________________________
zero_padding2d_1 (ZeroPadding2D (None, None, None, 3 0           leaky_re_lu_1[0][0]
__________________________________________________________________________________
conv2d_2 (Conv2D)            (None, None, None, 6 18432       zero_padding2d_1[0][0]
__________________________________________________________________________________
batch_normalization_2 (BatchNor (None, None, None, 6 256         conv2d_2[0][0]
__________________________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, None, None, 6 0           batch_normalization_2[0][0]
__________________________________________________________________________________
conv2d_3 (Conv2D)            (None, None, None, 3 2048        leaky_re_lu_2[0][0]
__________________________________________________________________________________
batch_normalization_3 (BatchNor (None, None, None, 3 128         conv2d_3[0][0]
__________________________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, None, None, 3 0           batch_normalization_3[0][0]
__________________________________________________________________________________
conv2d_4 (Conv2D)            (None, None, None, 6 18432       leaky_re_lu_3[0][0]
__________________________________________________________________________________
batch_normalization_4 (BatchNor (None, None, None, 6 256         conv2d_4[0][0]
__________________________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, None, None, 6 0           batch_normalization_4[0][0]
__________________________________________________________________________________
add_1 (Add)                  (None, None, None, 6 0           leaky_re_lu_2[0][0]
                                                              leaky_re_lu_4[0][0]
__________________________________________________________________________________
zero_padding2d_2 (ZeroPadding2D (None, None, None, 6 0           add_1[0][0]
__________________________________________________________________________________
conv2d_5 (Conv2D)            (None, None, None, 1 73728       zero_padding2d_2[0][0]
__________________________________________________________________________________
batch_normalization_5 (BatchNor (None, None, None, 1 512         conv2d_5[0][0]
__________________________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, None, None, 1 0           batch_normalization_5[0][0]
__________________________________________________________________________________
conv2d_6 (Conv2D)            (None, None, None, 6 8192        leaky_re_lu_5[0][0]
__________________________________________________________________________________
batch_normalization_6 (BatchNor (None, None, None, 6 256         conv2d_6[0][0]
__________________________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, None, None, 6 0           batch_normalization_6[0][0]
__________________________________________________________________________________
conv2d_7 (Conv2D)            (None, None, None, 1 73728       leaky_re_lu_6[0][0]
__________________________________________________________________________________
batch_normalization_7 (BatchNor (None, None, None, 1 512         conv2d_7[0][0]
__________________________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, None, None, 1 0           batch_normalization_7[0][0]
__________________________________________________________________________________
add_2 (Add)                  (None, None, None, 1 0           leaky_re_lu_5[0][0]
                                                              leaky_re_lu_7[0][0]
__________________________________________________________________________________
                                      ⋮
__________________________________________________________________________________
yolo_loss (Lambda)           (None, 1)           0           conv2d_59[0][0]
                                                             conv2d_67[0][0]
                                                             conv2d_75[0][0]
                                                             input_2[0][0]
                                                             input_3[0][0]
                                                             input_4[0][0]
==================================================================================
======
Total params: 61,581,727
Trainable params: 37,695
Non-trainable params: 61,544,032
__________________________________________________________________________________
__________________________________________________
```

Figure 47: Architecture of YOLOv3.

3.2.7.Training:

The training procedure is started after you've completed all of the steps above and made any necessary adjustments to the configuration file. Figure 48 shows the step count and classification loss for each step on the screen. It's worth noting that the classification loss begins at a very high number and steadily reduces as the algorithm learns over time.

Following training, a set of data representing the trainer's weights and the value of the loss she has achieved is prepared.

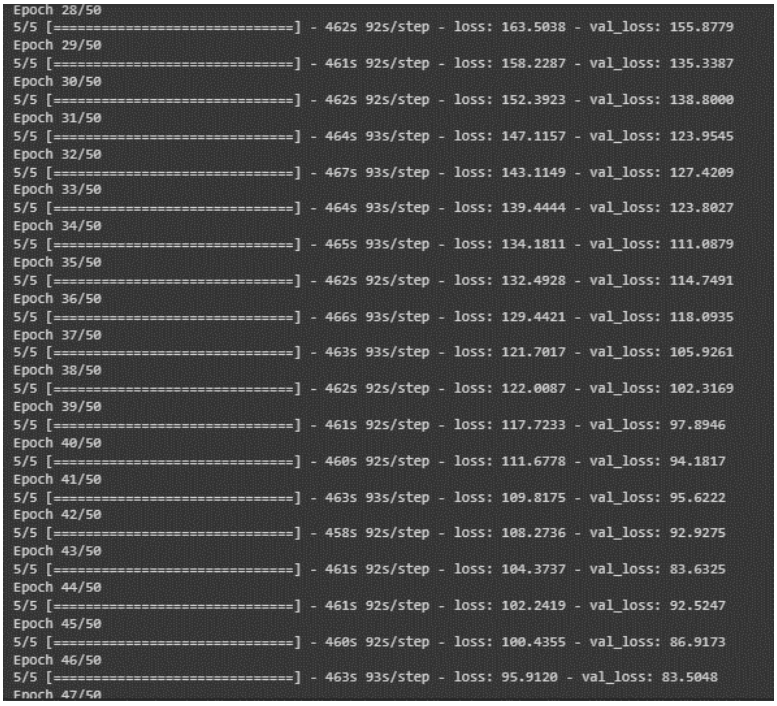


Figure 48: showing the steps of learning and reduction of the loss function.

3.2.8.Results:

The result is a square box that surrounds the detected object and displays the object's name as well as the detective's certainty [90].

Following the completion of the algorithm's training process, data a person and a cat was utilized to evaluate the yolov3 algorithm. The following are the results of the test:

Figure 49 and 50 represent the result of applying precision and Recall to the training data for the cat and person groups, respectively. When a model has high

recall but low precision, then the model classifies most of the positive samples correctly but it has many false positives (i.e., classifies many Negative samples as Positive). When a model has high precision but low recall, then the model is accurate when it classifies a sample as Positive but it may classify only some of the positive samples. The precision-recall curve encapsulates the tradeoff of both metrics and maximizes the effect of both metrics. It gives us a better idea of the overall accuracy of the model.

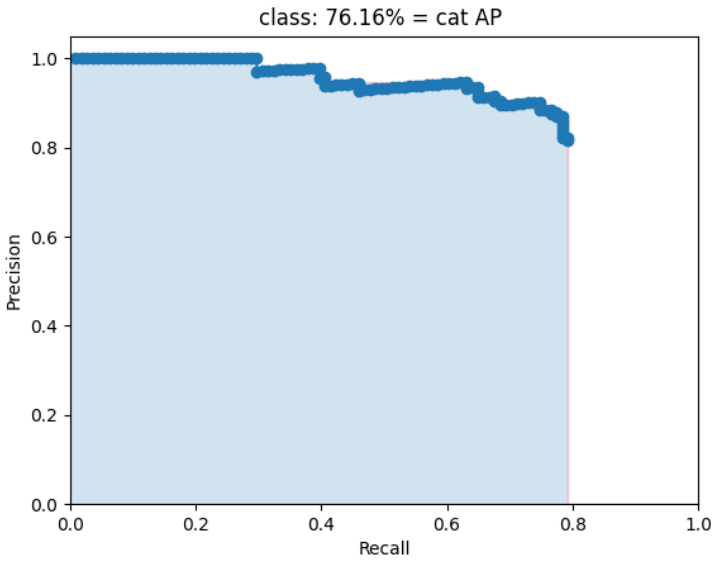


Figure 49: Precision-Recall Curve for Yolov3 (category cat).

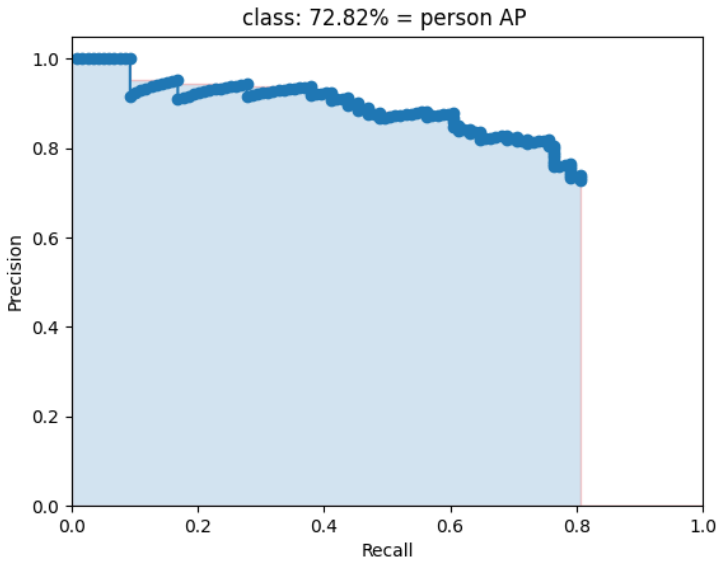


Figure 50: Precision-Recall Curve for Yolov3 (category person).

We have implemented our program code responsible for calculating both the Average Precision and Mean Average Precision, the results of the calculation appeared in Figure 51.

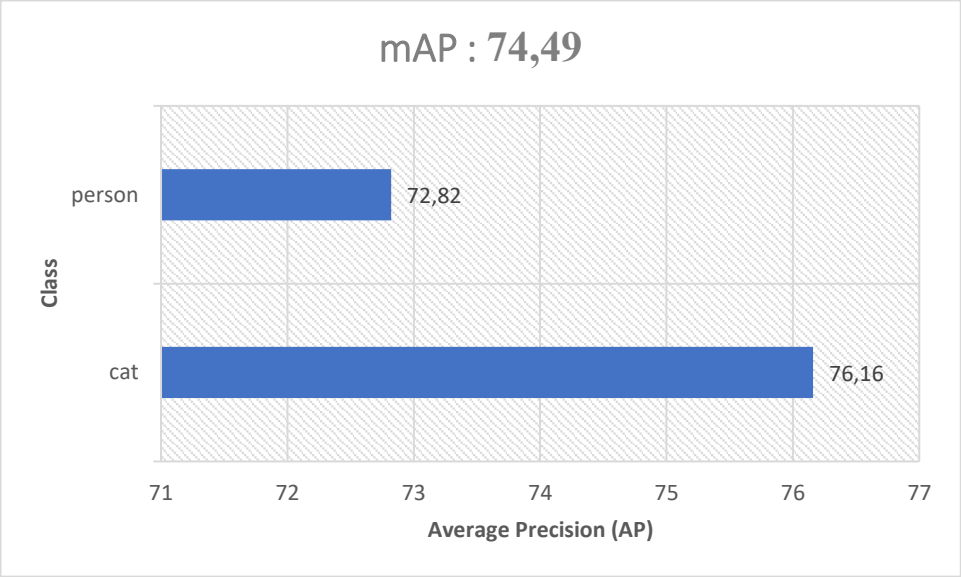


Figure 51: Mean Average Precision (mAP) of testing Yolov3 on our dataset.

The following results (figure 52 and figure 53) were obtained after running the code to distinguish between two categories (cat and person).

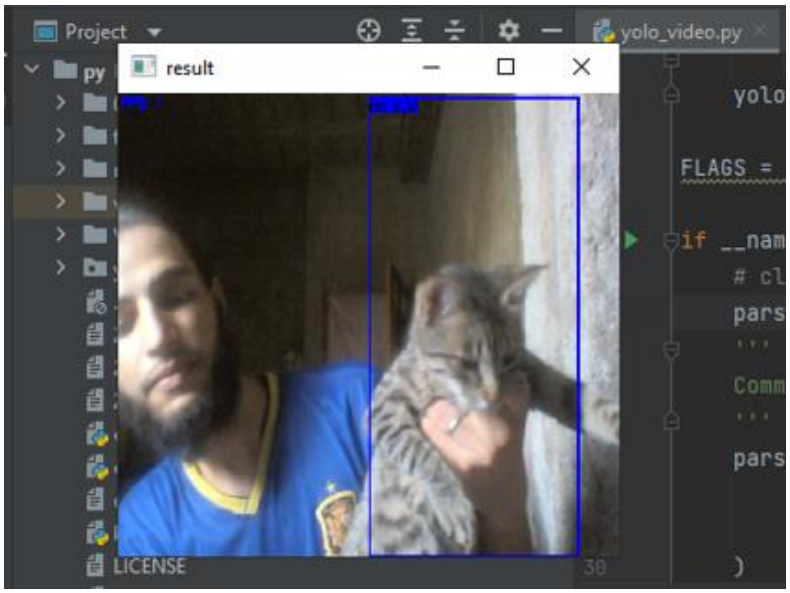


Figure 52: YOLOv3 for detecting cat on test video.

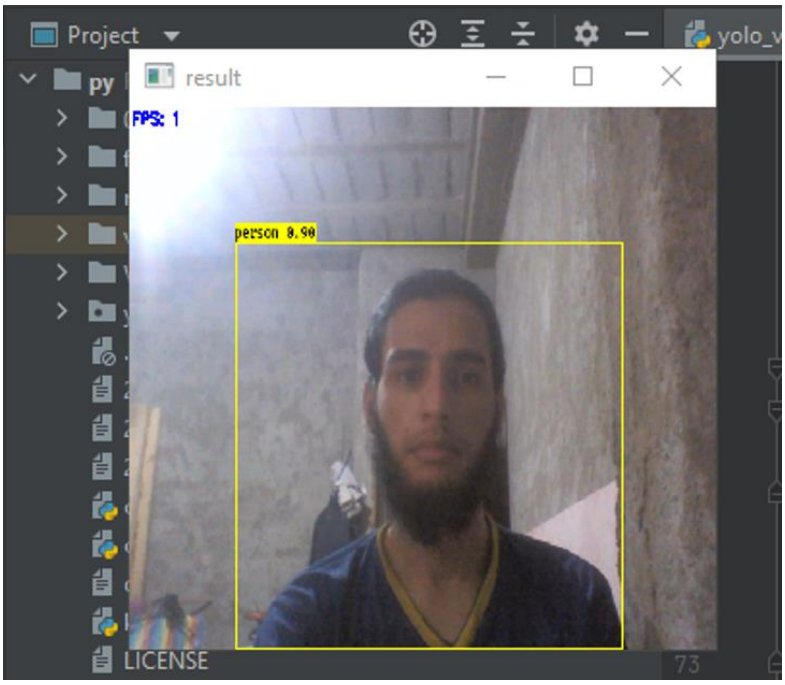


Figure 53: YOLOv3 for detecting person on test video.

3.2.9.Conclusion:

By studying the performance of these algorithms on a standard dataset, YOLOv3 have been identified as the most suitable and efficient deep-learning models to perform real-time object detection and recognition. An experiment has been carried out to evaluate the classification performance of these deep-learning algorithm. After the preparation of dataset, the algorithm has been trained on the (person and cat) dataset. The trained model has been evaluated on the collected test image, from which the number of true positives, true negatives, false positives and false negatives have been identified for each frame of the detections made by our deep-learning model. Using these results, the Accuracy, Precision, Recall and Average precision the of the model have been calculated and the performance of the YOLOv3, model have been evaluated.

GENERAL CONCLUSION:

This thesis report discusses about the most suitable deep-learning models for real-time object detection and recognition and evaluates the performance of these algorithms on the detection and recognition of two classes (person and cat). The results of this research are discussed in chapter 3. The main objective of our work was to find the best answer to the following question: what are the most suitable and efficient Deep Learning models for real-time object recognition? to do this, a literature review has been performed to obtain knowledge about various deep learning models that are capable of performing real-time object detection and recognition. This project started with the aim of creating an application capable of identifying and classifying person from a camera. After months of intensive investigation, we have come to the conclusion that Yolo (you only look once) is the best model in terms of accuracy and speed that can be used in real-time object detection. Supervised learning-based object detection models are data-hungry that require large amounts of annotated data in order to achieve high performance. Data annotation process is a costly work which requires lots of time. we were partially successful, but not totally, especially given the difficulty of training yolo network while utilizing a relatively weak device. After the model was trained, the process of detecting a person or other object from webcam video capture was fine. However, the accuracy was low in comparison to what could be achieved with certain adjustments.

In terms of possible enhancements, we'll aim to make the model's structure more balanced in order to improve the precision of the object's detection and localization. Attempting to extend the data and select more powerful training devices. Finally, highlight that the technologies employed in this project have a potential future as well as a successful present. Though this project has an impact on the security sector, by modifying the data set, this program might be used in any field where object detection is required.

References:

[1] H. Geoffrey , Deep Learning, Canadian Institute for Advanced Research : Department of Computer Science University of Toronto , 2012.

[2] C. Alfredo , «Eigen-stuff,» 23 Aug 2021. [En ligne]. Available: https://atcold.github.io.

[3] A. Courville, «Deep learning,» 2016. [En ligne]. Available: https://www.marefa.org/%D8%AA%D8%B9%D9%84%D9%85_%D8%B9%D9%85%D9%8A%D9%82.

[4] C. Sai Krishna et V. B, «Machine Learning Algorithms,» 2018. [En ligne]. Available: https://www.analyticsvidhya.com/blog/category/intermediate/.

[5] M. Usama, «Techniques, Applications and Research Challenges,» 2019. [En ligne]. Available: https://ieeexplore.ieee.org/document/8713992.

[6] M. Batta, «Machine Learning Algorithms. International Journal of Science and Research,» 2020. [En ligne]. Available: https://doi.org/10.21275/ART20203995.

[7] Stanford University, «Convolutional neural networks for visual recognition,» 2019. [En ligne]. Available: http://cs231n.stanford.edu/, 2019. Online; accessed 11 February 2019.

[8] V. V. FJODOR , «Multilayer feedforward networks areuniversal,» 31 MARCH 2017. [En ligne]. Available: https://www.asimovinstitute.org/author/fjodorvanveen/.

[9] B. C.M, «Pattern recognition and machine learning,» *springer,* 2006.

[10] V. V. F, «Neural network zoo prequel,» 10 May 2019. [En ligne]. Available: https://www.asimovinstitute.org/author/fjodorvanveen/.

[11] R. D.E et H. G.E, «Learning representations by back-propagating errors,» *nature,* p. 533–536, 1986.

[12] L. D, «What Is an Artificial Neural Network?,» 2019. [En ligne]. Available: www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/.

[13] Brilliant, «Backpropagation,» 10 May 2019. [En ligne]. Available: https://brilliant.org/wiki/backpropagation/.

[14] U. Stanford, «Deep Learning for Computer Vision,» 2022. [En ligne]. Available: http://cs231n.stanford.edu/.

[15] S. Sharma et A. Athaiya, «Activation Functions in Neural Networks.,» *International Journal of Engineering Applied Sciences and Technology.,* 2020.

[16] S. Christian , «Accelerating Deep Network Training by Reducing Internal Covariate Shift,» 11 Feb 2015. [En ligne]. Available: https://arxiv.org/abs/1502.03167.

[17] K. R, H. S et R. C, «Using convolutional neural networks for image,» 11 February 2019.

[18] C. Z, F. Q, F. R.S et V. N, «A unified multi-scale deep convolutional neural network for fast object detection,» *in European conference on computer vision,* pp. 354-370, 2016.

[19] A. M, «A Comprehensive Guide to Types of Neural Networks,» 25 Jan 2019. [En ligne]. Available: https://www.digitalvidya.com/blog/types-of-neural-networks/.

[20] H. t. H. Dang , «The Modern History of Object Recognition – Infographic,» 28 Apr 2017. [En ligne]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148.

[21] K. R, H. S and R. C, "Using convolutional neural networks for image recognition," 11 February 2019.

[22] Stanford, University, «Convolutional neural networks for visual recognition.,» 11 February 2019. [En ligne]. Available: http://cs231n.stanford.edu/. [Accès le 11 February 2019].

[23] T. D, B. L, F. R, T. L et P. M, «Learning spa-tiotemporal features with 3d convolutional networks,» chez *Proceedings of IEEE International Conference on Computer Vision*, 2015, p. 4489–4497.

[24] B. Jason , «A Gentle Introduction to Pooling Layers for Convolutional Neural Networks,» 22 April 2019. [En ligne]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[25] Stanford, University, «Convolutional neural networks for visual recognition.,» 11 February 2019. [En ligne]. Available: http://cs231n.stanford.edu/.

[26] L. Samer, Hijazi, K. Rishi et R. Chris , «Using convolutional neural networks for image recognition,» 2015.

[27] J. N, T. T, H. G, K. A et T. Y, «Does the brain do inverse graphics?,» *Brain and Cognitive Sciences Fall Colloquium*, 2012.

[28] F. L, «Mit autonomous vehicle technology study,» *Large-scale deep learning based analysis of driver behavior and interaction with automation.*, p. arXiv:1711.06976, Nov 2017.

[29] M. Kumar et R. Choudhary, «Artificial neural network related to biological neuron network: a review.,» *Emerging Trends in Big Data , IoT and Cyber Security*, pp. 55-62, 2017.

[30] R. Olga , D. Jia , S. Hao et K. Jonathan, ImageNet Large Scale Visual Recognition Challenge, International Journal of Computer Vision 115(3), 2014.

[31] E. M, G. L, W. C.K, W. J et Z. A, «The Pascal Visual Object Classes (VOC) Challenge.,» *International Journal of Computer Vision*, pp. 303-338, 2010.

[32] L. David, «Distinctive image features from scale-invariant keypoints,» *International Journal of Computer Vision*, pp. 91-110, November 2004.

[33] D. N et T. B, «Histograms of oriented gradients for human detection,» *in Computer Vision and Pattern Recognition*, pp. 886-893, 20-25 June 2005.

[34] K. A, S. I et H. G.E, «ImageNet Classification with Deep Convolutional Neural Networks,» *in Advances in Neural Information Processing Systems 25*, 3 December 2012.

[35] S. C, L. W, j. Y, S. P, R. S, A. D, E. D, V. V et R. A, «Computer Vision and Pattern Recognition,» *Going deeper with convolutions*, 14 Sep 2014.

[36] S. Karen et Z. Andrew , «Very Deep Convolutional Networks for Large-Scale Image Recognition,» *Computer Vision and Pattern Recognition ,* 4 Sep 2014.

[37] U. J.R., v. d. S. K.E.A., G. T et S. A.W.M., «Selective search for object recognition,» *International Journal of Computer Vision*, pp. 154-171, Sep 2013.

[38] G. Ross, D. Jeff, D. Trevor et M. Jitendra, «Rich feature hierarchies for accurate object detection and semantic segmentation,» *Computer Vision and Pattern Recognition*, 22 Oct 2014.

[39] H. Jonathan , R. V. , S. Chen , Z. Menglong , B. A. , F. A, F. Ian S, W. Z, S. Yang , G. S. et M. K. , «Speed/accuracy trade-offs for modern convolutional object detectors,» *In IEEE Conference on Computer Vision and Pattern Recognition,* 2017.

[40] A, Krizhevsky; I, Sutskever; G.E, Hinton, «"Imagenet classification with deep convolutional neural networks,» *Advances in neural information processing systems,* pp. 1097-1105, 3 December 2012.

[41] G. R.B., D. J, D. T et M. J, «Rich feature hierarchies for accurate object detection and semantic segmentation,» *Computer Vision and Pattern Recognition,* 11 Nov 2013.

[42] U. J.R., V. D. S. K.E., G. T et S. A.W., «Selective Search for Object Recognition,» *International Journal of Computer Vision,* pp. 104, 154-171, 2013.

[43] . L. Lingling, Y. Zhengyan , J. Licheng , L. Fang et L. Xu , «High-Resolution SAR Change Detection Based on ROI and SPP Net,» *Journals & Magazines,* pp. 177009 - 177022, 2 December 2019.

[44] P. Pulak , Z. Cheng et Z. Christopher , «SPP-Net: Deep Absolute Pose Regression with Synthetic Views,» *Computer Vision and Pattern Recognition,* 9 Dec 2017.

[45] G. Ross , «Fast R-CNN,» *Computer Vision and Pattern Recognition,* 30 Apr 2015.

[46] G. R. , «Fast R-CNN,» *EEE Conference on Computer Vision (ICCV),* pp. 1440-1448., September 2015.

[47] R. Shaoqing, H. Kaiming, G. Ross et S. Jian, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 1137-1149, 6 Jun 2016.

[48] H. K, G. G, D. P et G. R.B., «Mask R-CNN,» *IEEE Trans Pattern Anal Mach Intell,* 5 Jun 2018.

[49] L. Tsung-Yi, D. Piotr , G. Ross , H. Kaiming, H. Bharath et B. Serge , «Feature Pyramid Networks for Object Detection,» *Computer Vision and Pattern Recognition,* 19 Apr 2017.

[50] R. I. H. M et L. J, «Exploiting temporal information for 3d human pose estimation,» 23 November 2017.

[51] R. J, D. S, G. R et F. A, «You only look once: Unified ,real time object detection,» *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* pp. 779-788, Decem 2016.

[52] R. Joseph , D. Santosh, G. Ross et F. Ali , «You Only Look Once: Unified, Real-Time Object Detection,» *Computer Vision and Pattern Recognition,* 8 Jun 2015.

[53] R. Hamid , T. Nathan , G. JunYoung , S. Amir , R. Ian et S. Silvio , «Proceedings of the IEEE/CVF,» *Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 658-666, 2019.

[54] R. Joseph et F. Ali, «Yolov3: An incremental improvement.,» *Computer Vision and Pattern Recognition,* 8 Apr 2018.

[55] V. M. Lars et O. Roland , «Improving the Canny Edge Detector Using Automatic Programming,» *Improving Non-Max Suppression,* pp. 461-468, 20 July 2016.

[56] Z. Yuanyi, W. Jianfeng , P. Jian et Z. Lei , «Proceedings of the IEEE/CVF Winter,» *Conference on Applications of Computer Vision (WACV),* pp. 1286-1294, 2020.

[57] L. Wei , . A. Dragomir, E. Dumitru , S. Christian , R. Scott , F. Cheng-Yang et C. B. Alexander , «SSD: Single Shot MultiBox Detector,» *Computer Vision and Pattern Recognition,* 29 Dec 2016.

[58] L. Tsung-Yi, D. Piotr, G. Ross, H. Kaiming, H. Bharath et B. Serge, «Feature Pyramid Networks for Object Detection.,» *Computer Vision and Pattern Recognition.*, 9 Dec 2016.

[59] L. T, G. P, G. R.B, H. K et D. P, «Focal Loss for Dense Object Detection,» *Computer Vision and Pattern Recognition*, 7 Aug 2017.

[60] Y. W, C. W, H. Z, Y. D et S. Wei, «Automatic Ship Detection Based on RetinaNet Using Multi-Resolution Gaofen-3 Imagery,» 19 January 2019.

[61] R. Joseph et F. Ali, «YOLO9000: Better, Faster, Stronger.,» *Computer Vision and Pattern Recognition.*, 25 Dec 2016.

[62] L. Wei , A. Dragomir , E. Dumitru , S. Christian , R. Scott , F. ChengYang et C. Alexander, «SSD: Single Shot MultiBox Detector,» *Computer Vision2016*, pp. 21-37, 29 Dec 2015.

[63] R. J et F. Ali , «YOLOv3: An Incremental Improvement,» *Computer Vision and Pattern Recognition*, 8 Apr 2018.

[64] R. J et F. A, «YOLOv3: An Incremental Improvement,» *Computer Science, Computer Vision and Pattern Recognition*, 8 Apr 2018.

[65] A. Karl, G. Horst-Michael, S. Martin et M. Stefan, «Complex-YOLO:Real-time 3d object detection on point clouds.,» *European Conference on Computer Vision.*, 24 Sep 2018.

[66] I. Google , «Une plate-forme Open Source de bout en bout dédiée au machine learning,» 13 june 2019. [En ligne]. Available: https://www.tensorflow.org/. [Accès le 13 june 2019].

[67] R. J, D. S, G. R et F. A, «You Only Look Once: Unified, Real-Time Object Detection,» *in Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779-788, 8 Jun 2015.

[68] X. Dong, S. Feng, L. Zesen , L. Ba.Tuan , L. Xiwen et L. Xuera, «A Target Detection Model Based on Improved Tiny-Yolov3 Under the Environment of Mining Truck,» *Tiny-Yolov3*, 15 July 2019.

[69] B. Alexey , W. Chien-Yao et M. L. Hong-Yuan , «YOLOv4: Optimal Speed and Accuracy of Object Detection,» *Computer Vision and Pattern Recognition*, 23 Apr 2020.

[70] L. T, M. M, B. S.J., B. L.D., G. R.B., H. J, P. P, R. D, D. P et Z. C.L., «Microsoft COCO: Common Objects in Context,» *Computer Vision and Pattern Recognition*, 1 May 2014.

[71] R. O, D. J, S. H, K. J, S. S, M. S, H. Z, K. A, K. A, B. M, B. A.C. et F.-F. L, «ImageNet Large Scale Visual Recognition Challenge,» *Computer Science International Journal of Computer Vision*, 1 Sep 2014.

[72] W. C.Y, L. H.Y.M., W. Y.H, C. P.Y., H. J.W. et Y. I.H., «Cspnet: A new backbone that can enhance learning capability of cnn,» *in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 390-391, 2020.

[73] k. He, W. Zhang, S. Ren et J. Sun, «Spatial pyramid pooling in deep convolutional networks for visual recognition,» *IEEE transactions on pattern analysis and machine intelligence,*, pp. 1904-1916, 15 June 2014.

[74] H. Z et W. J, «DC-SPP-YOLO: Dense connection and spatial pyramid pooling based YOLO for object detection,» *Information Sciences*, pp. 241-258, June 2020.

[75] M. D, «Mish: A Self Regularized Non-Monotonic Activation Function,» *Computer Science > Machine Learning,* 13 Aug 2020.

[76] S. Liu, L. Qi, H. Qin, J. Shi et J. Jia, «Path Aggregation Network for Instance Segmentation,» *Computer Science IEEE/CVF Conference on Computer Vision and Pattern Recognition,* p. 8759–8768, 5 March 2018.

[77] K. Taghi M, «A survey on Image Data Augmentation for Deep Learning,» *Journal of Big Data,* 06 July 2019.

[78] J. G, S. A, B. J, NanoCode012, C. A, TaoXie, C. L, L. A.V., tkianai, yxNONG, H. A, lorenzomammana, H. J, D. L, Marc et K. Y, «ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration,» *semanticscholar,* 5 January 2021.

[79] R. T, L. H, N. A, B. B. E, S. G et F. I, «Tresnet: High performance gpu-dedicated architecture,» *in Proceedings of the IEEE/CVF Winter Conference on Applications of Computer,* pp. 1440-1409, 2021.

[80] B. Alexey, W. Chien-Yao et M. L. Hong-Yuan, «YOLOv4: Optimal Speed and Accuracy of Object Detection.,» *Computer Vision and Pattern Recognition.,* 23 Apr 2020.

[81] K. Jeong-ah, S. Ju-Yeong et P. Se-ho, «Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition,» *Authorized licensed use limited to: University of Gothenburg,* pp. 1-4, 20 December 2020.

[82] R. S, H. K, G. R et S. J, «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.,» *IEEE Transactions on Pattern Analysis and Machine Intelligence,* pp. 91-99, 2015.

[83] H. William F, A. Robert W, L. Antonio T et S. Joshua S, «PVLIB Python 2015,» *2015 IEEE 42nd Photovoltaic S...,* 2015.

[84] J. Marc , V.-L. Pedro et P. Antonio J, «Performance Evaluation of cuDNN Convolution Algorithms on NVIDIA Volta GPUs».*Journals & Magazines* .

[85] X. Frank F, V. Bogdan et N. Graham , «In-IDE Code Generation from Natural Language: Promise and Challenges,» pp. 1-47, 31 Apr 2022.

[86] R. Derek , E. Neil A, L. V. Enrique et S. Margaret-Anne D, «Error Identification Strategies for Python Jupyter Notebooks,» *Software Engineering,* 7 Apr 2022.

[87] P. Sakrapee , S. Jamie , J. Pranam et V.-D. H. Anton , «Effective Semantic Pixel Labelling With Convolutional Networks and Conditional Random Fields,» *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops,* pp. 36-46, 2015.

[88] Pkulzc, «Faster R-CNN Inception Resnet v2,» 2019. [En ligne]. Available: https://github.com/tensorflow/models/blob/master/research/object_detection/models/faster_rcnn_inceptio. [Accès le April 2020].

[89] TensorFlow, «TensorFlow Object Detection API,» 2020. [En ligne]. Available: https://www.tensorflow.org/lite/models/object_detection/overview. [Accès le April 2020].

[90] J. Moran , L. Haibo , W. Zhongbo , H. Bin et C. Zheng , «The Application of Improved YOLO V3 in Multi-Scale Target Detection,» 8 August 2019.

[91] X. Joyce, «Deep Learning for Object Detection: A Comprehensive Review Fast R-CNN,» 11 Sep 2017. [En ligne]. Available: https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9.

[92] D. M. Camacho, K. M. Collins, R. K. Powers, . J. C. Costello et J. J. Collins, «Next-Generation Machine Learning for Biological Networks,» *CNN*, 7 Jun 2018.

[93] S. Sharma et A. Athaiya, «Activation Functions in Neural Networks.,» *International Journal of Engineering Applied Sciences and Technology*, pp. 310-316, 2020.

[94] S. S. et A. A., «Activation Functions in Neural Networks.,» *International Journal of Engineering Applied Sciences and Technology*, pp. 310-316, 2020.

[95] t. re, «yuk fb,» *n,h,ghgh,h,*, 2012.

[96] M. Zeiler et R. Fergus, «Visualizing and understanding convolutional networks,» *In European conference on computer vision*, pp. 818-833, 2014.

[97] F. P., «Pictorial example of max-pooling,» *Max-pooling / Pooling*, 2018.

[98] C. «A Closer Look at YOLOv3,» 2018. [En ligne]. Available: https://www.cyberailab.com/home/a-closer-look-at-yolov3.

[99] L. Wei , A. Dragomir , E. Dumitru , S. Christian , R. Scott , F. ChengYang et B. d Alexander C. , «Single shot multibox detector.,» Dec 2015.

[100] S. Liu, Q. L., Q. H, S. J. et J. J, «Path aggregation network for instance segmentation.,» *in Proceedings of the IEEE conference on computer vision and pattern recognition*, p. 8759–8768, 2018.

[101] A. E.H., A. C.H., B. J.R., B. P.J. et O. J.M., «Pyramid methods in image processing,» *RCA engineer, vol. 29, no. 6.*, pp. 33-41, 1984.

[102] . T. T. Alireza, R. Abbas et M. Muharram , «A lightweight Tiny-YOLOv3 vehicle detection approach,» *Journal of Real-Time Image Processing*, p. 2389–2401, 21 May 2021.

[103] D. Vincent et V. Francesco , «A guide to convolution arithmetic for deep learning,» *Machine Learning*, 23 Mar 2016.

[104] V. Nitika, B. Edmond et V. Jakob , «FeaStNet: Feature-Steered Graph Convolutions for 3D Shape Analysis,» pp. 2598-2606, 2018.

[105] P. Dan C. , H. Mark et S. Thuraiappah , «A manifold flattening approach for anchorless localization,» *springer*, p. 319–333, 04 December 2011.

[106] L. Ang , Y. Xue et Z. Chongyang , «Rethinking Classification and Localization for Cascade R-CNN,» *Computer Vision and Pattern Recognition*, 27 Jul 2019.

[107] «This is how streets and places around the world looked because of Corona (watch),» 18 Mars 2020. [En ligne]. Available: https://arabi21.com/story/1253695.

[108] P. Mike , «HERE & NOW,» 5 SEPTEMBER 2018. [En ligne]. Available: https://pbswisconsin.org/watch/here-and-now/noon-wednesday-guest-mike-parson-tokopa.

[109] C. Nguyen, G. Tran, T. Nghiem, N. Doan, D. Gratadour, J. Burie et C. Luong, «Towards real-time smile detection based on faster region convolutional neural network,» *1st International Conference on Multimedia Analysis and Pattern Recognition*, pp. 1-6, 2018.

[110] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu et A. Berg, «Ssd: Single shot multibox detector.,» *In European conference on computer vision*, pp. 21-37, October 2016.

[111] A. Bochkovskiy, C. Wang et H. Liao, «Optimal speed and accuracy of object detection,» *Yolov4*, 2020.

[112] N. T. t. y. l. G. Jocher, A. Stoken, J. Borovec, A. Chaurasia, L. Changyu, A. Laughing et A. Hogan, «ultralytics/yolov5,» *yolov5*, Apr 2021.

[113] R. Vivek et H. Jonathan , «tensorflow,» *Google Research,* Sep 2020.

[114] S. alkentar et a. , «Practical comparation of the accuracy and speed of YOLO, SSD and Faster RCNN for drone detection,» *Journal of Engineering,* vol. 27 , August 2021.