PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA

University of Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculity of science and rechnology

Electronic Department

Memory

Presented to get

The master's diploma

Faculty : **Electronic**

Specialty : industrial electronics

By

- **Kerraiou Fath-allah**
- **Belaiboud Abdelghani**

Entitled

Object Detection with Convolutional Neural Networks

Supported on : ……

Before a jury composed of :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
Dr. BELHADAD Yahya	*Dr.*	*Président*	*Univ-BBA*
Dr. SIDAHMED Soumia	*Dr.*	*Encadreur*	*Univ-BBA*
Dr. AZOUG Seif Eddine	*Dr.*	*Examinateur*	*Univ-BBA*

College year 2021/2022

Acknowledgements

First and foremost, we would like to praise Allah the Almighty, the Most Gracious, and the Most Merciful for His blessing given to me during my study and in completing this thesis.

We would like to express my gratitude and sincere thanks to my principal supervisor **Dr.SID AHMED Soumia** and co-supervisor **Dr. BOUDECHICHE DjamelEddine,** for their guidance, advice, feedback and continuous support throughout the whole thing. I would also like to thank my associate supervisor **Prof MESSALI Zoubeida**, for her guidance, and feedback on improving my thesis.

We are very much grateful to all teachers of the University of Mohamed Elbachir El Ibrahimi, who provided us with the tools necessary for the success of our university studies during my course.

Last but not least, we would like to thank our families for supporting us throughout our life. Without their love and encouragement, we would not have finished this thesis.

Table of Contents

Chapter 1: Deep learning

Chapter 2: Object detection

Chapter 3: Model implementation and experimental results

List of Figures

List of Equations

List of Acronyms

AI : Artificial Intelligence

ANN : Artificial Neural Network

ConvNets : Convolutional Networks

DL : Deep Learning

DNN : Deep Neural Network

FC : Fully Connected

GPU : Graphic Processor Unit

IoU : Intersection over Union

ML : Machine Learning

MLP : Multi-Layer Perceptrons

MSE: Mean Squared Error

NN : Neural Network

R-CNN: regios-based convolutional neural netwok

Relu : Rectified Linear Unit

RGB : Red Green Blue

RPN : Region Proposal Network

SVM: Support Vector Machine

VGG : Visual Geometry Group

XML : Extensible Markup Language

Yolo: You Only Look Once

GENERAL INTRODUCTION

In recent years, with the rapid development of Deep Learning, a number of research areas have achieved good results, and accompanied by the continuous improvement of convolution neural networks, Computer Vision has arrived at a new peak. the architecture of convolution neural network is constantly improving. In addition, the return of the convolution neural network also makes the application of Computer Vision greatly improve, such as face recognition, object detection, object tracking, semantic segmentation, and so on. Object detection as one of the important applications in the field of Computer Vision has been the focus of research, and convolution neural network has made great progress in object detection [1].

So, we are going to implement deep learning and object detection concepts for brain tumor classification and regression. nowadays patient diagnosis relies on a doctor's manual evaluation of a patient and his or her test results. With no automated tools to help with a doctor's diagnosis and limited number of available doctors, not only is there a higher risk of misdiagnosis but also an increase in wait time for patients to be seen. Doctors must take the time to manually review test results and images rather than spending time with the patient. In order to improve patient care, enhanced medical technology in the form of automated tools is necessary to increase doctor efficiency and decrease patient time in hospitals and time toward recovery. The purpose of this of thesis is to develop automated methods to aid doctors in diagnosis in order to prevent misdiagnosis and decrease patient wait time. In particular, this thesis achieves this automation through the classification and regression (object detection) of brain tumor from patient brain images.

The structure of this thesis:

- On Chapter 1, focuses on the tools of deep learning, and its notions and terminology.
- On Chapter 2, present fundamentals of object detection, and its architectures and impact of deep learning in object detection.
- On Chapter 3, implementation of deep learning and CNN to construct our model. Then, perform a custom data training for a pre-trained model of object detection.

Chapter 1: Deep learning

Abstract

In this chapter, we did cover the key principles of deep learning, as well as basic and advanced deep learning concepts, and took a deep dive into neural networks, how they adapt and learn from any datasets, also we addressed convolutional neural network.

1.1 Introduction

In this chapter, we will learn the core principles behind deep learning and the basic deep learning concepts as well advanced. Deep learning is one of the widely common machine learning subfield, so for the sake of better understanding we are going to start this chapter with approach on machine learning and we'll take a look into common machine learning methods of supervised and unsupervised learning, and specifically supervised learning because this is what we are going to use in our studies for this thesis. After that we are going in-depth into neural networks, how they work and learn and cover up everything about them so this what gives us the foundation of deep learning and makes things easier for us in next section, we will present most common deep neural network architecture for image datasets. We will precise the neural network architecture considered in our study to realize our project and provide solutions for our problems, once we picked our DNN architecture, we'll go in detail and touch everything related to it.

1.2 Machine learning

Machine learning is a branch within artificial intelligence that has been a fundamental part in modern digital solutions [2]. Machine learning refers to algorithms that computers use to learn from data, allowing it to make predictions on future [3]. The purpose of machine learning algorithms is to over time, learn during its execution from different pattern recognition methods. These algorithms are able to extract features from input data that is used to make decisions based on data of similar kind. Machine learning solutions are used in a large array of areas including robotics, traffic prediction and product recommendation [1, 4, 5]

For the sake of clarity, we need to understand how machine-learning algorithms work. Machine learning algorithms can be broadly classified into two categories:

1.2.1 Supervised learning

Is the machine learning approach defined by its use of labeled datasets to train algorithms to classify data and predict outcomes. Using labeled training data, the algorithm learns the rule for mapping the input variables into the target variable. For example, a supervised learning algorithm learns to predict whether there will be rain (the target variable) from input variables such as the

temperature, time, season, atmospheric pressure, and so on [3]. Figure 1.1 illustrates a supervised learning process.

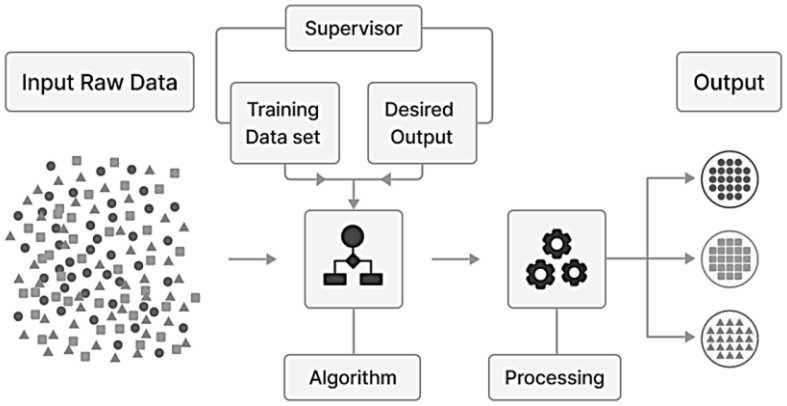

Figure 1.1: Supervised Learning process

1.2.2 Unsupervised learning

Using unlabeled training data, the algorithm learns associative rules for the data. The most common use case for unsupervised learning algorithms is in clustering analysis, where the algorithm learns hidden patterns and groups in data that are not explicitly labeled [3].

1.2.3 Supervised learning methods

There are two main areas where supervised machine learning comes in handy: classification problems and regression problems:

- **Classification:** To predict the outcome of a given sample where the output variable is in the form of categories. Examples include labels such as male and female, sick and healthy, etc.
- **Regression:** To predict the outcome of a given sample where the output variable is in the form of real values. Examples include real-valued labels denoting the amount of rainfall, height of a person, etc.

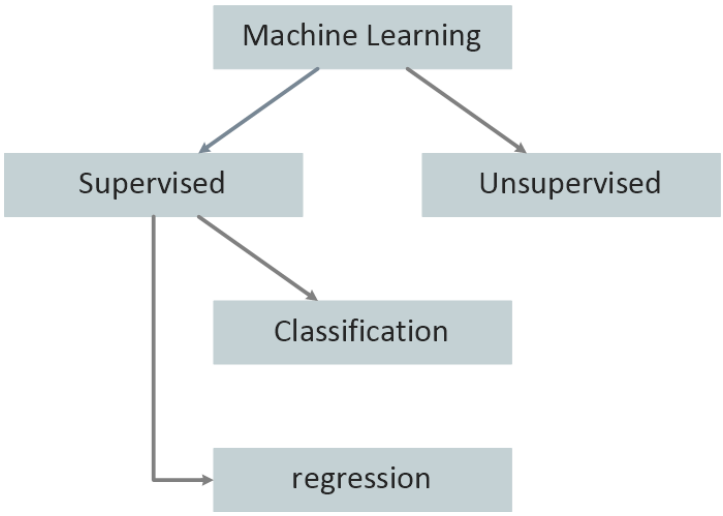

Figure 1.2: Hierarchy Machine Learning algorithms

1.3 Datasets

A dataset can be defined as a collection of related data represented in columns and rows. If D = {X, Y} is a dataset, it can be broken down into a set of input data (x, y). Where x ∈ X is a vector containing features from the dataset D, and y ∈ Y is the correlated values from these features. In the field of object detection, datasets that are appropriate and fit the recognition tasks at hand are important. When developing an object detection system, datasets are a necessary component for training as well as evaluating developed object detection model [6].

There are different types of data we can use to train prediction models, of course data type selection depends on the model task:

Quantitative Data: Numerical. E.g. height, weight.

Categorical Data: Data that can be labeled or divided into groups. E.g. race, gender, hair color.

1.4 Deep learning

Deep learning attempts to imitate how the human brain can process light and sound stimuli into vision and hearing. A deep learning architecture is inspired by biological neural networks and consists of multiple layers in an artificial neural network, it has proven its usefulness in almost all areas of science and engineering.

Deep learning uses a cascade of nonlinear processing unit layers in order to extract or transform features (or representations) of the data. The output of one layer serves as the input of the successive layer [7].

Deep Learning was first in the 1980s, but it has only become useful recently because:

- It requires large amounts of labeled data
- It requires significant computational power (high performing GPUs)
- Supervised Learning process the deep learning model maps the input and the output to find a correlation between them. This correlation can be then used to cluster, predict, classify, and even generate new samples of data.
- One needs to train a deep learning model to make it learn and produce accurate results.

1.4.1 Automate Feature Extraction using DL

Deep Learning can essentially do everything that machine learning does, but not the other way around. For instance, machine learning is useful when the dataset is small and well-curated, which means that the data is carefully preprocessed [8].

Data preprocessing requires human intervention. It also means that when the dataset is large and complex, machine learning algorithms will fail to extract information, and it will underfit. Generally, machine learning is alternatively termed shallow learning because it is very effective for smaller datasets [8]. Figure 1.3 show the accuracy of Deep learning versus other ML algorithms.

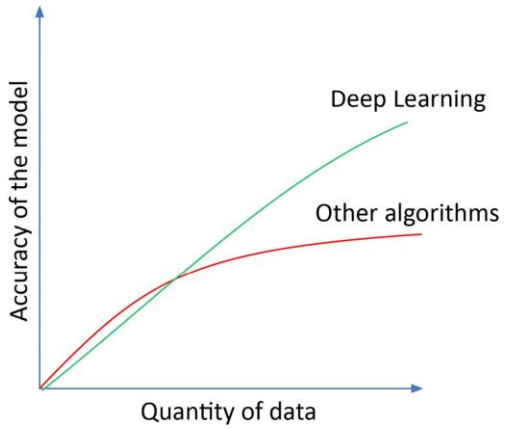

Figure 1.3: The performance of deep learning vs. other learning algorithms

Deep learning, on the other hand, is extremely powerful when the dataset is large. It can learn any complex patterns from the data and can draw accurate conclusions on its own. In fact, deep learning is so powerful that it can even process unstructured data – data that is not adequately arranged like text corpus, social media activity, etc. Furthermore, it can also generate new data samples and find anomalies that machine learning algorithms and human eyes can miss [8].

On the downside, deep learning is computationally expensive compared to machine learning, which also means that it requires a lot of time to process. Deep Learning and Machine Learning are both capable of different types of learning: Supervised Learning (labeled data), Unsupervised Learning (unlabeled data), But their usefulness is usually determined by the size and complexity of the data [8].

1.4.2 Neural networks

Neural networks are the functional unit of deep learning and are known to mimic the behavior of the human brain to solve complex data problems. So neural network is a tool that allow us to do information processing paradigms inspired by the way biological neural systems process data. Machine learning and deep learning try to simulate some properties of biological neural networks. The input data is processed through different layers of artificial neurons stacked together to produce the desired output as shown in Figure 1.4. From speech recognition and person recognition to healthcare and marketing, Neural Networks have been used in a varied set of domains.

In more practical terms neural networks are non-linear statistical data modeling or decision-making tools. They can be used to model complex relationships between inputs and outputs or to find patterns in data.

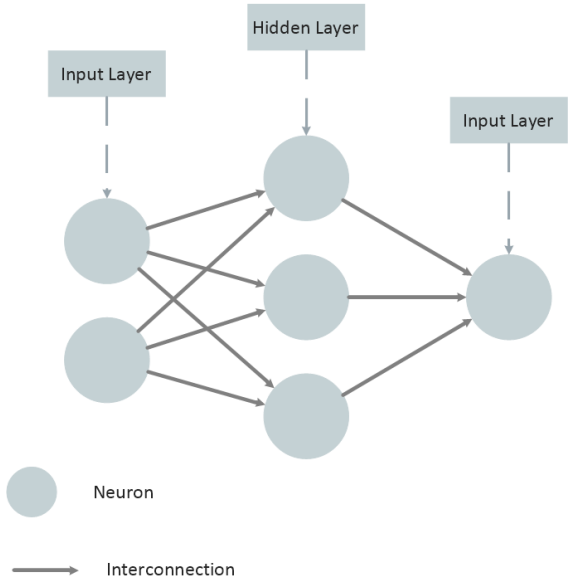

Figure 1.4: Simple Neural Network Architecture

1.4.3 The architecture of neural networks

Let's explain some terminology that lets us name different parts of a network. Suppose we have the network in below (Figure 1.5):

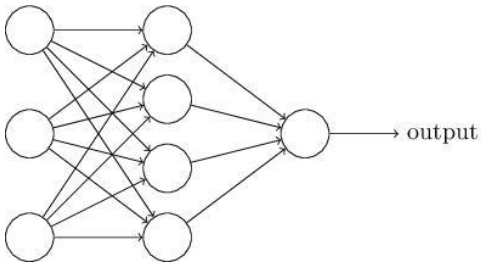

Figure 1.5: Illustration of neural network architecture

So, the leftmost layer in this network is called the input layer, and the neurons within the layer are called input neurons. The rightmost or output layer contains the output neurons, or, as in this case, a single output neuron. The middle layer is called a hidden layer. The network above has just a single hidden layer, but some networks have multiple hidden layers. For example, in Figure 1.6 four-layer network has two hidden layers:

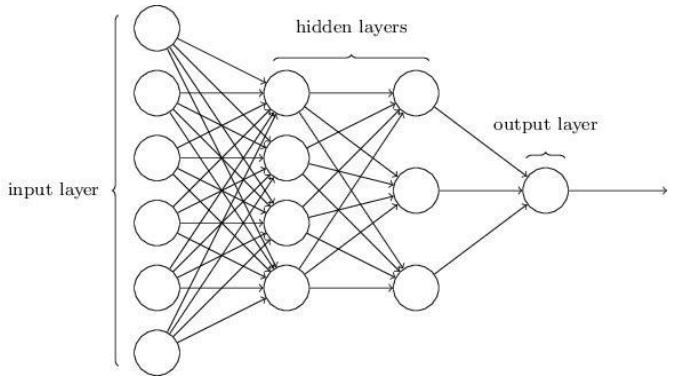

Figure 1.6: multi-layer networks

Such multiple layer networks are sometimes called multilayer perceptrons or MLPs, despite being made up of neurons, sigmoid neurons, ReLu neurons, not perceptrons.

We will begin our discussion of NNs via one of the fundamental building block of deep learning.

1.4.4 Basic Principles of Neural Networks

1.4.4.1 Perceptron:

A perceptron takes several binary inputs, x1, x2, …, and produces a single binary output as shown in Figure 1.7 [9]:

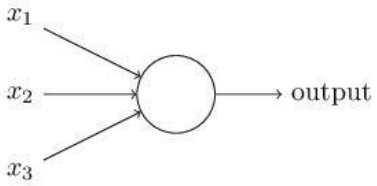

Figure 1.7: illustration of a simple way to describe perceptron

In Figure 1.7 the perceptron has three inputs, x_1, x_2, x_3. In general, it could have more or fewer inputs. Rosenblatt proposed a simple rule to compute the output. He introduced weights, w_1, w_2, . . ., real numbers expressing the importance of the respective inputs to the output. The neuron's output, 0 or 1, is determined by whether the weighted sum $\sum_j w_j x_j$ is less than or greater than some threshold value. Just like the weights, the threshold is a real number which is a parameter of the neuron [9]. To put it in more precise algebraic terms:

$$output = \begin{cases} o & if \quad \sum_j w_j x_j \leq threshold \\ 1 & if \quad \sum_j w_j x_j > threshold \end{cases} \quad 1.1$$

Furthermore, the output model is expressed by:

$$output = \begin{cases} o & if \quad w \cdot x + b \leq 0 \\ 1 & if \quad w \cdot x + b > 1 \end{cases} \quad 1.2$$

Where: $w \cdot x$ denote $\sum_j w_j x_j$ and b denote the bias, $b \equiv -threshold$

the bias is a measure of how easy it is to get the perceptron to fire. For a perceptron with a really big bias, it's extremely easy for the perceptron to output a 1. But if the bias is very negative, then it's difficult for the perceptron to output a 1 [9].

1.4.4.2 Sigmoid neuron

Just like a perceptron, the sigmoid neuron has inputs, x1, x2, ..., But instead of being just 0 or 1, these inputs can also take on any values between 0 and 1. So, for instance, 0.638 is a valid input for a sigmoid neuron. Also just like a perceptron, the sigmoid neuron has weights for each input, w1, w2, ..., and an overall bias, b. But the output is not 0 or 1. Instead, it's $\sigma(wx + b)$, where σ is called the sigmoid function [9], and is defined by:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \quad 1.3$$

$$z = \sum_j w_j x_j + b \quad 1.4$$

So, what makes perceptron and sigmoid neuron dissimilar is the activation function of the weighted input plus the bias, we going to understand more about activation function in next section. To generalize, everything we going to use the word neuron to define a model that takes several input data and produce an output, using weight, transfer function and any type of activation function.

1.4.4.3 Neuron Components

As we said The Neural Network architecture is made of individual units called neurons that mimic the biological behavior of the brain.

Here are the various components of a neuron:

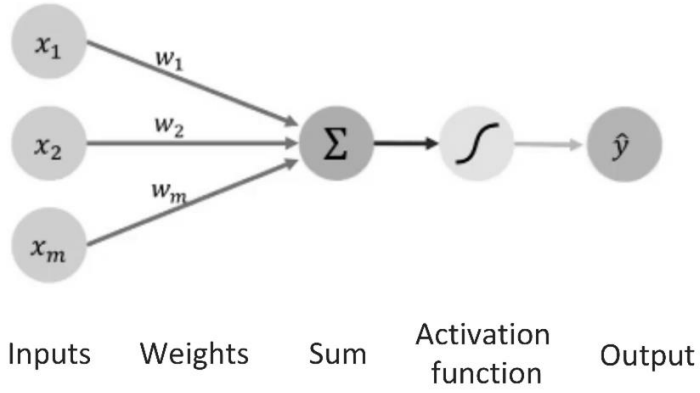

Figure 1.8: schematic architecture of perceptron components

1.4.4.3.1 Input

It is the set of features that are fed into the model for the learning process[10].

1.4.4.3.2 Weight

Its main function is to give importance to those features that contribute more towards the learning [11].

1.4.4.3.3 Transfer function

The job of the transfer function is to combine multiple inputs into one output value so that the activation function can be applied. It is done by a simple summation of all the inputs to the transfer function [11].

1.4.4.3.4 Activation function

Activation functions lie at the core of deep neural networks allowing them to learn arbitrarily complex mappings. The functions are mathematical equations that determine the output of a neural network. The function is attached to each neuron in the network, and determines whether it should be activated fired or not.

So, we know what Activation Function is and what it does. But, why do Neural Networks need it? Well, the purpose of an activation function is to add non-linearity to the neural network.

Linear activation functions produce linear decisions no matter the network size, on the other hand, non-linearities allow us to approximate arbitrarily complex function.

1.4.4.4 Type of activation function:

1.4.4.4.1 ReLu function

The rectified linear activation function is a linear function that will output the input directly if it is positive, otherwise, it will output zero [12].

It has the following properties:

- It does not Saturate.
- It converges faster than some other activation functions [13].

$$R(z) = \max\{0, z\} \quad (1.5)$$

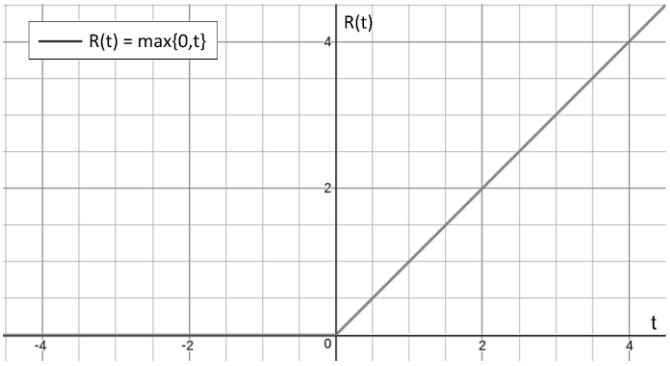

Figure 1.9: ReLu function graph

1.4.4.4.2 Sigmoid function

It's non-linear function, its ranges from 0 to 1 having an S shape. Also known by the name of the logistic or squashing function in some literature. The sigmoid function is used in output layers of the DNN and is used for probability-based output.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

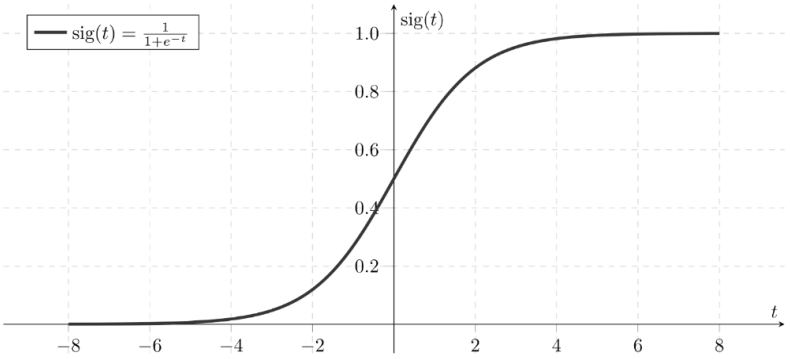

Figure 1.10: sigmoid function graph

When multiple neurons are stacked together in a row, they constitute a layer, and multiple layers piled next to each other are called a neural network or multi-layer neural network. We've described the main components of this type of structure in Figure 1.11.

We'll use w_{jk}^{l} to denote the weight for the connection from the k^{th} neuron in the $(l-1)^{th}$ layer to the j^{th} neuron in the l^{th} layer. So, for example, the diagram of Figure 1.11 below shows the weight on a connection from the fourth neuron in the second layer to the second neuron in the third layer of a network:

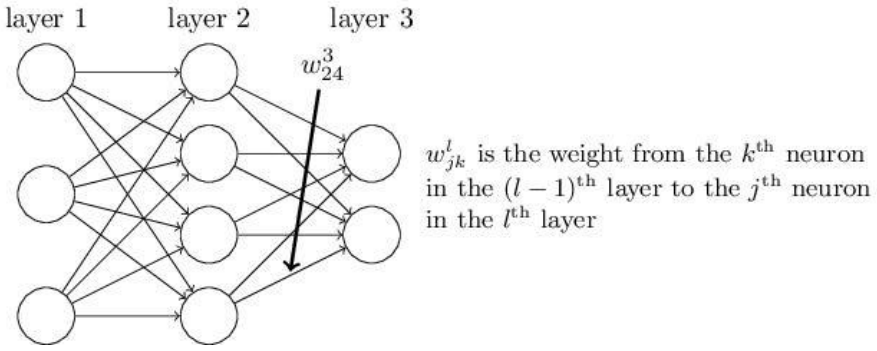

Figure 1.11: Illustration of notation for the network's weights

We use a similar notation for the network's biases and activations. Explicitly, we use b_j^l for the bias of the j^{th} neuron in the l^{th} layer. And we use al j for the activation of the j^{th} neuron in the l^{th} layer. Figure 1.12 shows examples of these notations in use:

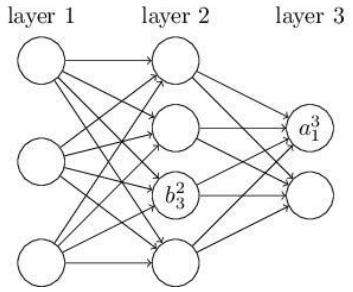

Figure 1.12: Illustration of notation for the network's biases and activations

Suppose we have sigmoid function as activation function. So, with these notations the activation a_j^l of the j^{th} neuron in the l^{th} layer is related to the activations in the $(l-1)^{th}$ layer by the equation:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$z_j^l = \sum_k w_{jk}^l a_k^{l-1} + b_j^l \quad (1.6)$$

$$a_{jk}^l = \sigma\left(z_j^l\right) \quad (1.7)$$

These equations can be rewritten in compact vectorized form:

$$a^l = \sigma(w^l a^{l-1} + b^l) \quad (1.8)$$

$$z^l = w^l a^{l-1} + b^l \quad (1.9)$$

$$a^l = \sigma(z^l) \quad (1.10)$$

1.4.5 Forward propagation

So, to understand the forward propagation mechanism or process, let's explain just mechanism between two layers of neural network and we can apply the same procedure to previous layers, we going to use sigmoid as activation function, as we can see in Figure 1.13, each of these connections has a unique weight and every single neuron has a unique bias:

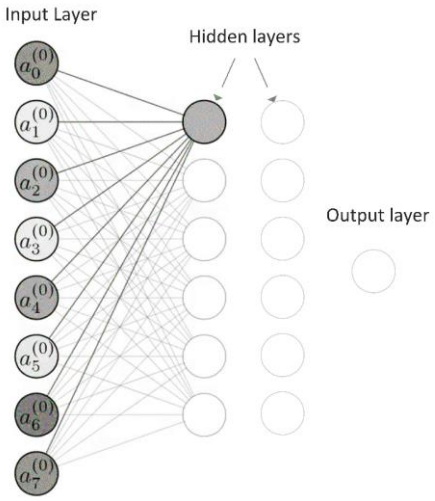

Figure 1.13: Illustration of neural network [8,6,6,1] shape

The actual function to get one neuron's activation in terms of the activations in the first hidden layer or any previous layer, for example let's choose neuron $a_0^{(1)}$ just so we can demonstrate how we can calculate neuron output value:

$$a_0^{(1)} = \sigma(w_{00}^1\, a_0^{(0)} + w_{01}^1\, a_1^{(0)} + \cdots + w_{07}^1\, a_7^{(0)} + b_0^1 \qquad 1.\ 11$$

we'll use matrix multiplication to compute the activations of all the neurons in the next layer simultaneously. you can communicate the full transition of activations from one layer to the next in matrix form, we will have:

$$\sigma\left(\begin{bmatrix} w_{00}^1 & w_{01}^1 & \cdots & w_{07}^1 \\ w_{10}^1 & w_{11}^1 & \cdots & w_{17}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{50}^1 & w_{51}^1 & \cdots & w_{57}^1 \end{bmatrix}\begin{bmatrix} a_0^{(0)} \\ a_1^{(0)} \\ \vdots \\ a_7^{(0)} \end{bmatrix} + \begin{bmatrix} b_0^1 \\ b_1^1 \\ \vdots \\ b_5^1 \end{bmatrix}\right) = \begin{bmatrix} a_0^{(1)} \\ a_1^{(1)} \\ \vdots \\ a_5^{(1)} \end{bmatrix}$$

We repeat the same process until we reach the output layer, just like this we going to get our predicted value $a^L(x) = y(x)$ for one data example x,

1.4.6 Loss function

basically, the cost function measures how bad or good is the prediction. The final output predictions compare the predicted values with the desired output:

$$Loss\big(y(x), a^L(x)\big)$$

1.4.6.1 Types of Loss function

1.4.6.1.1 Regression Loss Function

Regression models deals with predicting a continuous value for example given floor area, number of rooms, size of rooms, predict the price of the room. One of the mostly used loss functions in the regression problem is called **"**Mean Squared Error" and also known as **"**Quadratic Cost":

$$C = \frac{1}{2n}\sum_x \|y(x) - a^L(x)\|^2 \qquad 1.\ 12$$

where: n is the total number of training examples, the sum is over individual training examples x, $y = y(x)$ is the corresponding desired output, L denotes the number of layers in the network, and $a^L = a^L(x)$ is the vector of activations output from the network when x is input.

Also, the quadratic cost function can be written as average $C = \frac{1}{n}\sum_x C_x$ over cost functions C_x for individual training examples, x. where the cost for a single training example is:

$$C_x = \frac{1}{2}\,\|y(x) - a^L\|^2 \qquad 1.\ 13$$

1.4.6.1.2 Classification loss function

Classification problems involve predicting a discrete class output. It involves dividing the dataset into different and unique classes based on different parameters so that a new and unseen record can be put into one of the classes. The most widely used function for this type of learning method is "Binary Cross Entropy" loss function:

$$C = -\frac{1}{n}\sum_x [y \ln a^L + (1-y)\ln(1-a^L)] \qquad 1.\ 14$$

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

1.4.7 Training the neural network

Just telling the neural network model what a terrible job it's doing isn't very helpful. You want to tell it how it should change those weights and biases so as to improve, and we have to make sure that those changes should minimize the loss function. How do you find an input that minimizes the value of this cost function? This is why we need to apply the gradient descent over the cost function.

1.4.7.1 Gradient descent

How can we apply gradient descent to learn in a neural network? The idea is to use gradient descent to find the weights w^k and biases b^l which minimize the cost. To see how this works, let's introduce the gradient descent update rule for weights and biases:

$$w \to w' - \alpha \frac{\partial C}{\partial w} \tag{1. 15}$$

$$b \to b' - \alpha \frac{\partial C}{\partial b} \tag{1. 16}$$

where α is a small, positive parameter (known as the learning rate), $\frac{\partial C}{\partial w}$ partial derivative with respect to weight, $\frac{\partial C}{\partial b}$ partial derivative with respect to bias. Also w' and b'represent old weight and bias, w and b indicate updated weight and bias.

there are many possible local minima you might land in. It depends on which random input you start at, and there's no guarantee that the local minimum you land in will be the smallest possible value for the cost function.

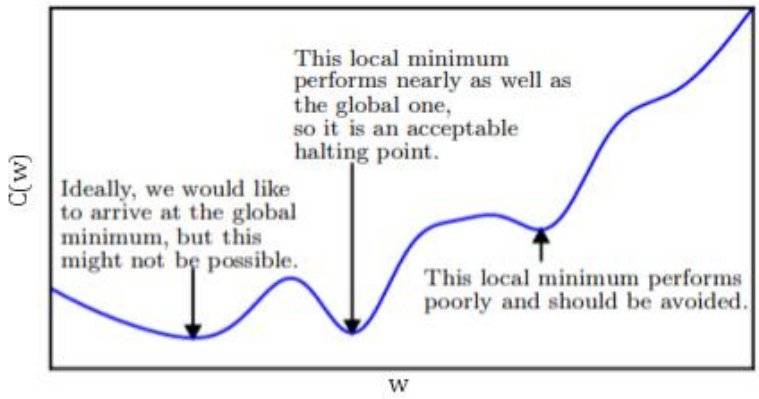

Figure 1.14: illustration of multiple minima for cost function

Gradient descent algorithm:

1. Initialize weights and biases randomly
2. Loop until convergence:
3. Compute gradient $\frac{\partial C}{\partial w}, \frac{\partial C}{\partial b}$
4. Update weights and biases $w \leftarrow w' - \alpha \frac{\partial C}{\partial w}, b \rightarrow b' - \alpha \frac{\partial C}{\partial b}$
5. Return weights and biases

1.4.7.2 Learning rate

the learning rate is a tuning parameter in an optimization algorithm that determines the step size at each iteration while moving toward a minimum of a loss function [14].

In setting a learning rate, there is a trade-off between the rate of convergence and overshooting. A too high learning rate will make the learning jump over minima but a too low learning rate will either take too long to converge or get stuck in an undesirable local minimum [15].

1.4.7.3 Back-propagation

Backpropagation, is an algorithm for computing that derivative of the loss function with respect to the neuron weights $\frac{\partial C}{\partial w}$ and for biases $\frac{\partial C}{\partial b}$.

When training a neural net, the goal is to find neuron parameters (weights) that cause the output of the NN to best fit the data, the back-propagation is the way the NN can "connect" the loss function and outputs with the weight parametrization.

- If the loss function is less than the previous value using the current weights, then the gradient is in a good direction.
- If the loss function is more than the previous, it goes in the opposite direction.
- Repeat until the loss function is zero or cannot make it lower (*convergence*).

When the Neural Network converged, it found a spot in the loss function that increasing or decreasing the weight values makes the loss function increasing.

1.5 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN) is a specialized type of deep neural network which includes many different deep learning algorithms. CNNs are regularized versions of multilayer perceptron with some extra layer to minimize data size and parameters number. CNNs are widely used in areas such as object detection, image classification and regression for feature extraction. CNNs have the advantage of being able to convolve over large datasets with high speed as part of the training process [16].

In the next elements we'll describe convolutional neural networks. Also, interpret every component relate to CNN structure like convolutional layer, pooling and fully connected. These networks use a special architecture which is particularly well-adapted to classify images. Using this architecture makes convolutional networks fast to train. This, in turn, helps us train deep, many-layer networks, which are very good at classifying images.

1.5.1 Architecture

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix, and its activation function is commonly ReLu or Sigmoid. As the convolution kernel slides along the input matrix

for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers [17].

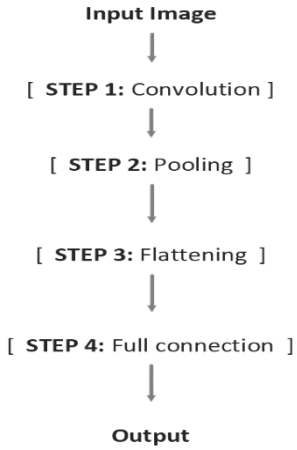

Figure 1.15: Structure of Convolutional Neural Network.

1.5.2 CNN Components

1.5.2.1 Kernels and Filters

A grid of discrete numbers or values describes the kernel. Each value is called the kernel weight. Random numbers are assigned to act as the weights of the kernel at the beginning of the CNN training process. In addition, there are several different methods used to initialize the weights. Next, these weights are adjusted at each training iteration. thus, the kernel learns to extract significant features [18].

The local receptive field: is a defined segmented area that is occupied by the content of input data that a neuron within a convolutional layer is exposed to during the process of convolution [18].

Shared weights and bias: In CNNs, each filter is replicated across the entire visual field. These replicated units share the same parameterization (weight vector and bias) and form a feature map [18]. The shared weights and bias are often said to define a kernel or filter.

1.5.2.2 Convolutional Layer

One typical layer of CNN is a convolutional layer that uses mathematical convolution to process the data. The main purpose for a convolutional layer is to detect features such as edges, lines, blobs of color and other visual elements. To detect these features, the convolutional layer uses filters that are square-shaped objects that scan over the image. The more filters we give to a convolutional layer, the more features it can detect.

The hyperparameters are as follows:

- **The number of filters or depth:** It has an impact on the output's depth. Three distinct filters, for example, would result in three different feature maps, resulting in a depth of three [19].
- **Stride:** It refers to the distance, or the number of pixels, that the kernel travels across the input matrix, a bigger stride results in a lesser output [19].
- **Zero-padding:** When the filters don't fit the input image, it's frequently utilized. All parameters outside of the input matrix are set to zero, resulting in a larger or equal-sized output [19].

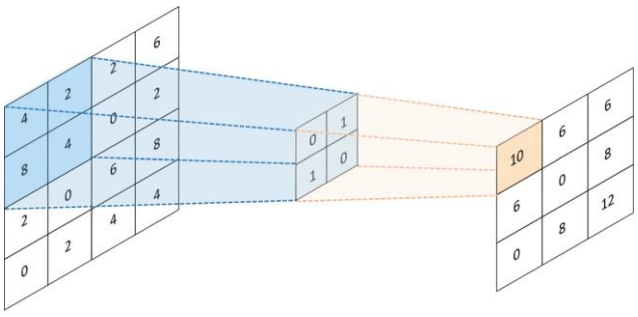

Figure 1.16: An illustration of the convolution of a 2 x 2 kernel moving over a 4 x 4 image, resulting in a 3 x 3 feature map

Convolutional Operation: Initially, the CNN input format is described. The vector format is the input of the traditional neural network, while the multi-channeled image is the input of the CNN. For instance, single channel is the format of the gray-scale image, while the RGB image format is three-channeled. To understand the convolutional operation, let us take an example of a 4x4 gray-scale image with a 2x2 random weight-initialize kernel (Figure 1.16). First, the kernel slides over the whole image horizontally and vertically. In addition, the dot product between the

input image and the kernel is determined, where their corresponding values are multiplied and then summed up to create a single scalar value, calculated concurrently. The whole process is then repeated until no further sliding is possible. Note that the calculated dot product values represent the feature map of the output, we can calculate the convolution layer output by using equation below:

$$output\ size = \left[\frac{(w - k + 2p)}{s}\right] + 1 \qquad 1.\ 17$$

Where: w is the input volume, k is the kernel size, p is the padding, s is the stride

The feature map: is the output of one filter applied to the image data. Basically, the feature map represents the result of convolution operation between filter and image data.

Activation function: is the last component of the convolutional layer to increase the non-linearity in the output. Generally, ReLu function is used as an activation function in a convolution layer. ReLu function allows us to replace every negative value in feature map from the convolutional layer with zero, this is will help us to prevent the values from summing up to zero.

1.5.2.3 Pool layer

The main task of the pooling layer is the sub-sampling of the feature maps. These maps are generated by following the convolutional layer. In other words, this approach shrinks large-size feature maps to create smaller feature maps. It maintains the majority of the dominant information (or features) in every step of the pooling stage. The most familiar and frequently pooling types utilized pooling methods are the max and average pooling.

Max-pool: Calculate the maximum value for each patch of the feature map. Thus, the output after max-pooling layer would be a feature map containing the most prominent features of the previous feature map.

Average pooling: computes the average of the elements present in the region of feature map covered by the filter.

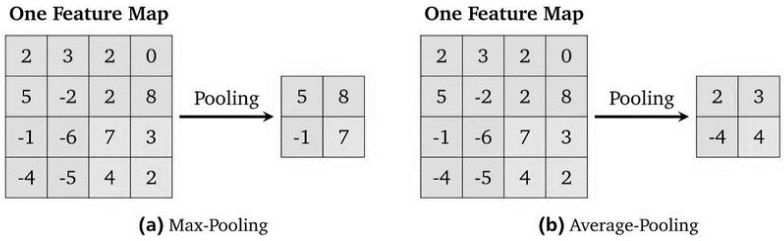

Figure 1.17: Example for the max-pooling and the average-pooling with a filter size of 2×2 and a stride of 2×2

1.5.2.4 Flattening

When we have the pooling layers with many pooled feature maps and then you flatten them. So, you put them into this one long column sequentially one after the other. And you get one huge vector of inputs for an artificial neural network or FC.

So, to sum all this up, we've got an input image. We apply a convolution layer, then we apply pooling, and then we flatten everything into a long vector which will be our input layer for the fully connected layer or ANN.

1.5.2.5 Fully Connected Layer

So, the fully connected layer collects the resulting data from the convolutional and pooling layer, analyzes the data from the layers independently and makes the final classification or localization decision.

1.5.3 Principle working of Convolutional Neural Networks

A CNN has hidden layers of convolution layers that form the base of ConvNets. Like any other layer, a convolutional layer receives input volume, performs mathematical scalar product with the feature matrix (filter), and outputs the feature maps. Features refer to details in the image data like edges, shapes, textures, objects, circles, etc [20].

At a higher level, convolutional layers detect these patterns in the image data with the help of filters. The higher-level details are taken care of by the first few convolutional layers. The deeper the network goes, the more sophisticated the pattern searching becomes. For example, in later layers rather than edges and simple shapes, filters may detect specific objects like eyes or ears, and eventually a cat and a dog [20].

The first hidden layer in the network dealing with images is usually a convolutional layer. When adding a convolutional layer to a network, we need to specify the number of filters we want the layer to have. A filter can be thought of as a relatively small matrix for which we decide the number of rows and columns this matrix has. The value of this feature matrix is initialized with random numbers. When this convolutional layer receives pixel values of input data, the filter will convolve over each patch of the input matrix. The output of the convolutional layer is usually passed through the ReLu activation function to bring non-linearity to the model. It takes the feature map and replaces all the negative values with zero [20].

The pooling layer is added in succession to the convolutional layer to reduce the dimensions. We take a window of say 2x2 and select either the maximum pixel value or the average of all pixels in the window and continue sliding the window. So, we take the feature map, perform a pooling operation, and generate a new feature map reduced in size. Pooling is a very important step in the ConvNet as reduces the computation and makes the model tolerant towards distortions and variations [20].

The convolutional layer was responsible for the feature extraction. But, What about the final prediction? A fully connected dense neural network would use a flattened feature matrix and predict according to the use case [20].

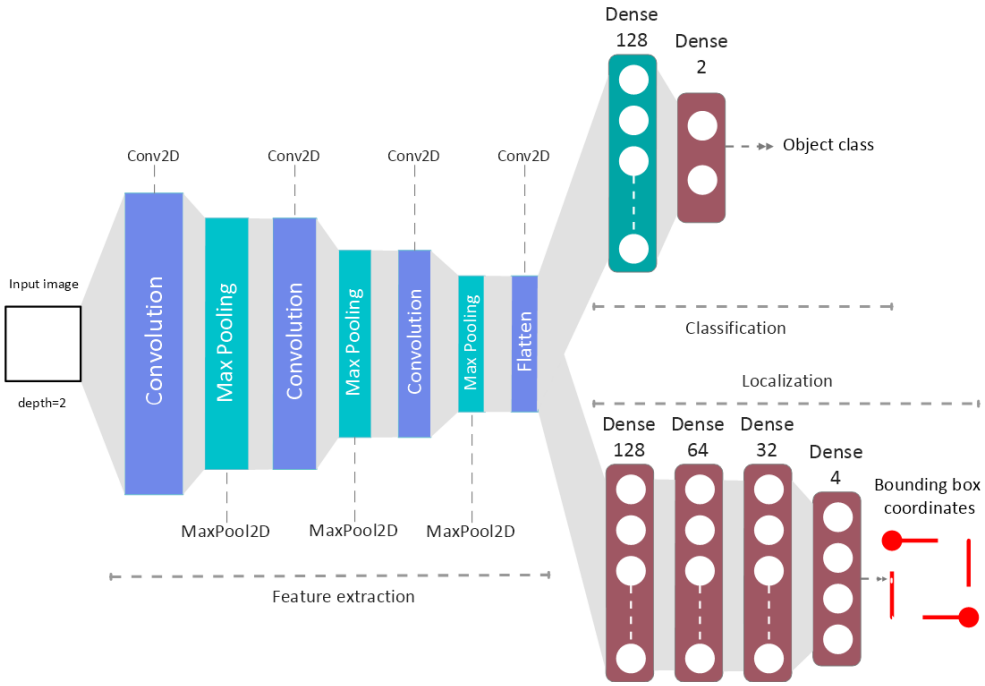

Figure 1.18: Architecture of CNN model (classification/localization

1.6 Conclusion

As shown above in this chapter we presented a brief approach about machine learning to set up the plate for our interest in most common ML subfield. In this case is deep learning. Then, we started deal with the fundamental's studies of deep learning field including neural network concepts (neural network algorithms: forward-propagation, back-propagation), perceptron components. Also, we addressed one of the most widely used deep neural networks CNN, we did envelop every CNN component and layer from convolutional layer to the last layer known as FC. All these notions let us proceed towards the next chapters, where we going to implement all these theorical notions to construction prediction model.

Chapter 2: Object detection

Abstract

Object detection has been attracting a lot of interest, and recently a lot of various computational models have been designed. In this chapter, we explained various object detection models including RCNN, FAST RCNN, FASTER RCNN and YOLO.

2.1 Introduction

When carefully observing the brain, multiple processing levels can be identified. It is understood that at each level the brain can learn features or representations at escalating heights of abstraction. For instance, the typical design of the visual cortex suggests that (roughly speaking) the brain initially extracts edges followed by patches, then surfaces, then objects, and so on. This is one of the fundamental ways in which the brain performs vision [21].

Convolutional Neural Networks (CNNs) emerged from the study of the brain's visual cortex, and they have been used in image recognition since the 1980s. In the last few years, CNNs have managed to achieve superhuman performance on some complex visual tasks.

In this chapter we will discuss one of the most important visual tasks which is object detection, the application of this latter is used in many domains such as in the field of military, security, medical and augmented reality. We will first, understand what is exactly object detection (Classifying multiple objects in an image and placing bounding boxes around them), then we will discuss some of the best CNN architectures in this regard.

2.2 Object Detection

Object detection is an automated computer vision technique for locating instances of objects in digital photographs or videos. Specifically, object detection draws bounding boxes around one or more effective targets located in a still image or video data. An effective target is the object of interest in the image or video data that is being investigated. The effective target (or targets) should be known at the beginning of the task [22].

Object detection can be categorized into two research topics:

- Detection of specific instance
- General object detection

The first type aims towards the detection of a particular object while the second type gives the exact locations and classifying all the elements in the vision, The main process of object detection is shown in Figure2.1, this process localizes and classifies the target. In supervised

learning, the effective targets should be known at the beginning of the task; the effective target in this case is the object of interest in the image or video data that is being investigated.

Before we start setting the main techniques used in recent years for object detection, we need to differentiate between object detection, semantic segmentation and instance segmentation, as these terms can be confusing for anyone new to the field.

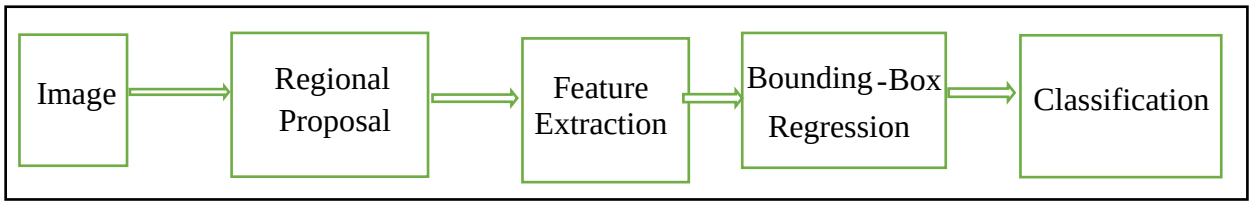

Figure 2.1:main process of object detection

2.3 Object detection, semantic segmentation, instance segmentation differences

2.3.1 Object detection

In object detection, each image pixel is classified whether it belongs to a particular class (e.g. face) or not It refers to the method of identifying and correctly labelling all the objects present in the image frame. In practice, this consists of two steps:

- **Object Localization:** Here, a bounding box or enclosing region is predicted in the tightest possible manner in order to locate the exact position of the object in the image.
- **Image Classification:** The localized object is then fed to a classifier which labels the object.

The figure below shows the output for object detection applied on a real image of sheep, where on the left it shows one box determining the sheep and classifying it, and on the right each sheep is localized and classified.

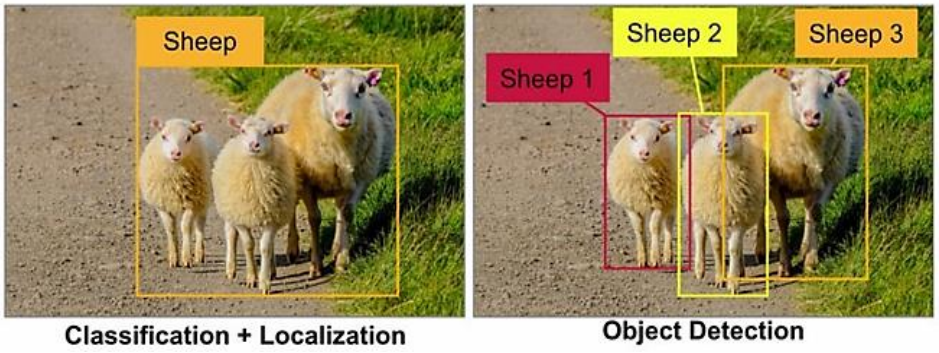

Figure 2.2:Detect one single object(right), detect multiple objects (left).

2.3.2 Semantic segmentation

If you are familiar with the term segmentation, you already know it is just simply linking pixels to determine objects and does not differentiate any instance inside of it. Same for semantic segmentation, it refers to the process of linking each pixel in the given image to a particular class label. For example, in the figure 2.3 the pixels are labelled as road, sheep and grass.

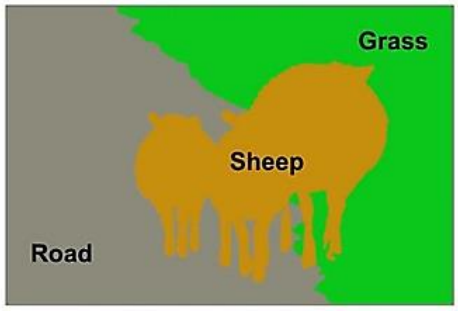

Figure 2.3:Semantic detection

2.3.3 Instance segmentation

Instance Segmentation is one step ahead of semantic segmentation. Instead of assigning same pixel values to all objects of same class, it associates a class label to each pixel similar to the semantic segmentation, except that it treats multiple objects of the same class as individual objects separate entities.

the end result will have an image where all the objects are separated by pixel boundaries. For example, in the figure2.4 below we can differentiate each sheep zone.

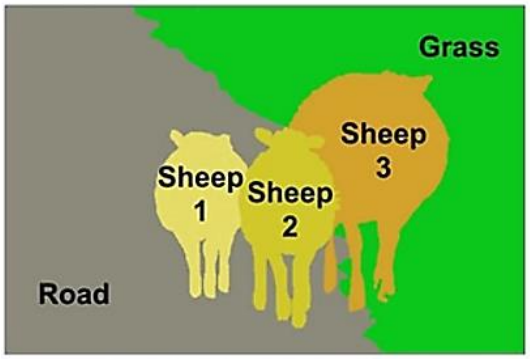

Figure 2.4:Instance segmentation.

In our project we will focus on object detection, which have been for a long time an interesting field for researchers, but significant results were only produced in the recent years owing to the rise of Convnets as feature extractors and Transfer Learning as method of passing on previous knowledge. There are multiple deep detection models, it won't possible for us to cover up all of them, we will therefore detail some recent techniques that exist for this matter, and we will focus on the CNN as this is the technique, we will adopt to achieve our object detection objective.

2.4 Object Detection using Convolutional Neural Networks

In recent years, the research community invented the (CNN), Deep Neural-Networks, CNN's (Convolutional Neural Network) different models had attained amazingly accurate results for object detection.

The CNN architecture of object detection is based on two different methods, the methods can be divided into two main classes:

- Two-stage methods
- One-stage methods

Two-stage methods firstly generate some candidate object proposals and then classify those proposals into the specific categories while one-stage methods simultaneously extract and classify all the object proposals. Generally speaking, two stage methods have a relatively slower detection speed and higher detection accuracy, while one-stage methods have a much faster detection speed

and comparable detection accuracy. Later we will introduce you the metrics used to evaluate the accuracy of object detection detectors.

In the following part of this section, two-stage methods and one-stage methods are introduced, respectively [23].

2.4.1 Two-Stage Detector

Two-stage frameworks divide the detection process into the region proposal and the classification stage. These models first propose several object candidates, known as regions of interest (RoI), using reference boxes (anchors). In the second step, the proposals are classified and their localization is refined. Will take R-CNN, or Region-based Convolutional Neural Network.

2.4.1.1 Two-Stage Detector

Two-stage frameworks divide the detection process into the region proposal and the classification stage. These models first propose several object candidates, known as regions of interest (RoI), using reference boxes (anchors). In the second step, the proposals are classified and their localization is refined. Will take R-CNN [22], or Region-based Convolutional Neural Network, as an example of two-stage methods.

2.4.1.1.1 R-CNN for object detection

R-CNN consists of 3 simple steps:

1. An algorithm called Selective Search used to generate proposals.
2. Run a convolutional Neural Net (CNN) on top of each of these region proposals.
3. Take the output of each CNN and feed it into an SVM to classify the region and to a linear regressor to tighten the bounding box of the object, if such an object exists.

These 3 steps are illustrated in the image below:

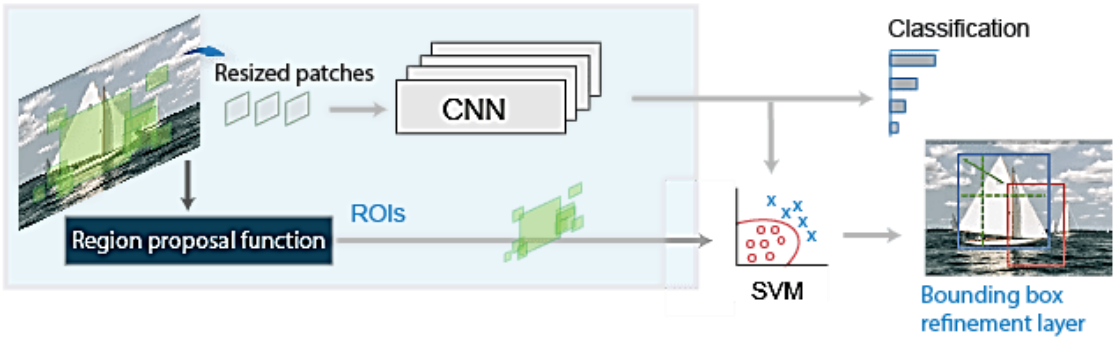

Figure 2.5:R-CNN object detector.

To sum it up; in this technique we first propose regions, second extract features, and finally classify those regions based on their features. This method turns object detection into an image classification problem. Knowing that R-CNN is a very intuitive method, but at the same time it is very slow.

Later, Faster-RCNN proposed a scheme in which features are shared between both stages, achieving a significant efficiency improvement. The main insight of Faster R-CNN was to replace the slow selective search algorithm with a fast neural net. Specifically, it introduced the region proposal network (RPN). Faster R-CNN uses a convolutional backbone network, such as VGG or ResNet, which outputs global feature maps. These convolutional maps are shared between the Region Proposal Network (RPN) and the detection network, which reduces the cost of generating proposals externally.

2.4.2 One-stage Detector

On the other hand, one-stage detectors contain a single feed-forward fully convolutional network that directly provides the bounding boxes and the object classification. The Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO) were among the first to propose a single unified architecture, without requiring a per-proposal computation. Single-Shot Multibox Detector (SSD) was the first one-stage detector to achieve an accuracy reasonably close to the two-stage detectors while still retaining the ability to work in real-time. We will take YOLO [24].

2.4.2.1 You Only Look Once (YOLO)

YOLO detectors are another anchor-based alternative that divides the image into regions and predict bounding boxes and probabilities for each region. It is one of the faster object detection algorithms which make it a very good choice when we need real-time detection, without loss of too much accuracy. This technique uses Convolutional Neural Networks for object detection.

The YOLO model was first described by Joseph Redmon, et al. in the 2015 paper titled You Only Look Once: Unified, Real-Time Object Detection [24].

The first three YOLO versions have been released in 2016, 2017 and 2018 respectively. However, in 2020, within a period of only few months, three major versions of YOLO have been released named YOLO v4, YOLO v5 and PP-YOLO.

YOLO involves a single neural network trained end-to-end that takes a picture as input and predicts bounding boxes and class labels for each bounding box directly. Let's look up close at the general architecture of an YoloV4 detector to understand.

The figure2.6 below shows an example of an output image of YoloV4.

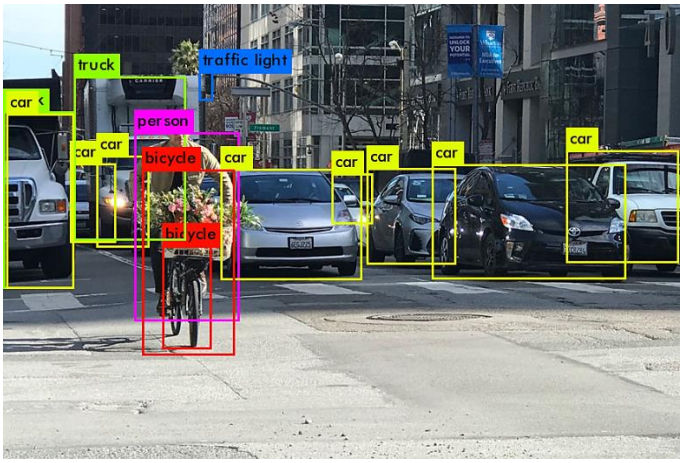

Figure 2.6:Objects detected by YoloV4.

2.4.2.1.1 YoloV4 architecture

First, Let's take a look at the main components of a modern one-stage object detector. The figure 2.7 is taken from YOLO v4 [25] paper It represents the general architecture of YoloV4 which is a one-stage detector.

As you can see in the figure 2.7, YOLOv4 consists of:

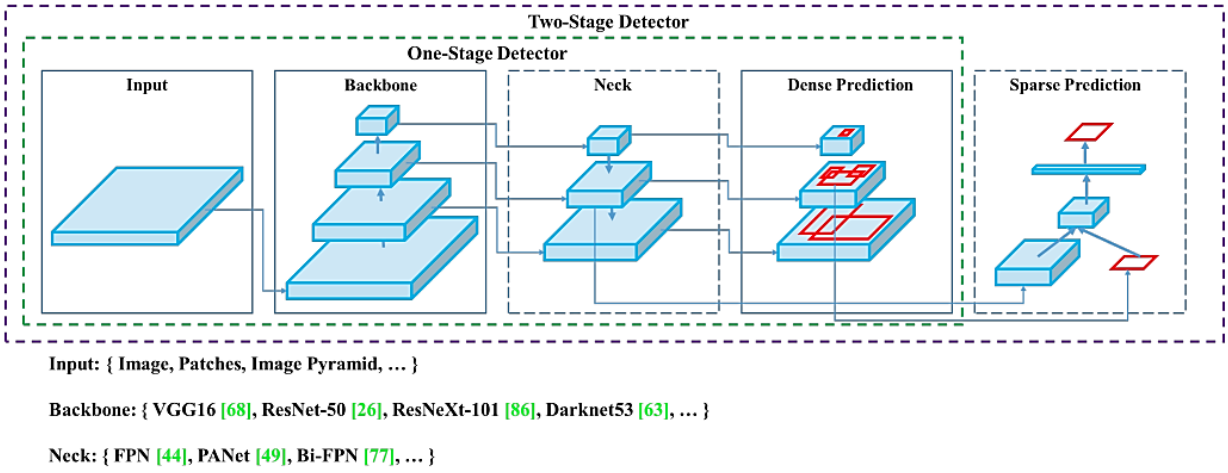

Figure 2.7:YoloV4 architecture.

1. Backbone;
2. Neck;
3. Head.

A. The Backbone:

Refers to the feature-extraction architecture. They are pre-trained on image classification datasets, like ImageNet, and then fine-tuned on the detection dataset. In YOLOv4, backbones can be VGG, ResNet, SpineNet, EfficientNet, ResNeXt, or Darknet5. Tiny YOLO has only 9 convolutional layers, so it's less accurate but faster and better suited for mobile and embedded projects. Darknet53 (The backbone used in YOLOV3) has 53 convolutional layers, so it's more accurate but slower. Later we will introduce you the metrics use to evaluate the accuracy of object detection detectors.

B. The Neck:

These are extra layers that go in between the backbone and head. They are used to extract different feature maps of different stages of the backbone, in other terms its purpose is to add extra information in the layers. (Feature Pyramid Networks, Path Aggregation Network ...).

C. **Head**.

This is a network in charge of actually doing the detection part (classification and regression) of the bounding boxes.

2.4.2.1.2 Interpreting the YOLO Output

Input to the network is a batch of images, and the features learned by the convolutional layers are passed onto a classifier/regressor which makes the detection prediction. And the output is a list of bounding boxes along with recognized classes.

To understand the whole process let's assume that we have an input image. First, we divide it into a grid of dimensions equal to that of the final feature map. For example, if the input image is of 416 x 416 cells and stride is 32, then the feature map will be of 13 x 13 cells. Then we divide the input image into 13 x 13 cells. The input to the network is a batch of images, and the features learned by the convolutional layers are passed onto a classifier/regressor which makes the detection prediction.

Instead of predicting the absolute size of boxes with regard to the entire image, YOLO introduces what is known as an Anchor Box, a list of predefined boxes that best match the desired objects, which are calculated using k-mean clustering. Yolo V3 uses three anchor boxes, each of the 13 x 13 cells thus encodes information about three boxes.

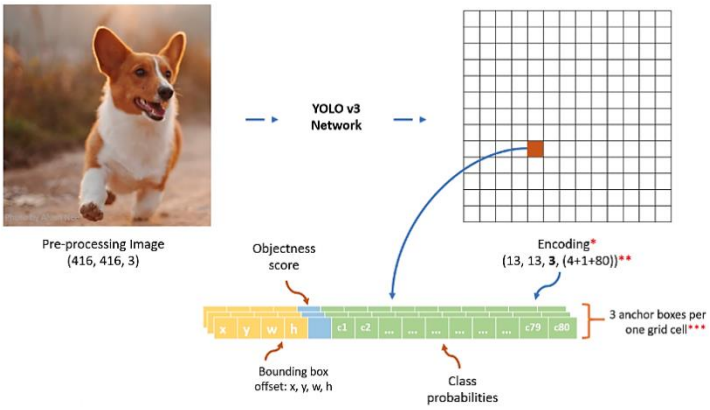

Figure 2.8: YOLOv3 Network prediction process

The output is a list of bounding boxes along with the recognized classes. Each bounding box is represented by 6 numbers $(p_c, b_x, b_y, b_h, b_W, c)$. where p_c is the probability that it belongs to a particular class, b_x and b_y are the centre of the bounding box, b_h and b_W are the height and width of the bounding box respectively, and c is the available total number of classes. If we expand c into an 80-dimensional vector (The number of classes), each bounding box is then represented by 85 numbers.

The algorithm may find multiple detections of the same object. Non-max suppression is a technique by which the algorithm detects the object only once. To remove the duplicates, the bounding boxes that have the class probability above a threshold value are selected and used to locate the object within the image.

So, we saw how YOLO learns and how it is able to detect all the images in a single go. Now, we will introduce you to the metrics used to evaluate the accuracy of our proposed object detection detector in the practical part.

2.5 Evaluation of Object Detection Model

The MSE often works fairly well as a cost function to train the model, but it is not a great metric to evaluate how well the model can predict bounding boxes. The most common metric for object detection is IoU which stands for Intersection over Union.

2.5.1 Intersection over union (IOU)

We have explained above some of the methods for object detection but still, we didn't clarify how to determine the accuracy of these methods. Intersection over Union is a test to ascertain how close is the prediction to the actual truth. The IOU is calculated using this equation:

$$IoU = \frac{A \cup B}{A \cap B} \qquad 2.1$$

The figure below represents two overlapped boxes (red and blue empty boxes), The overlapping region is the region marked with a black pen over the combination of both.

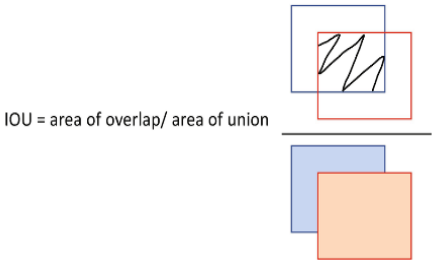

Figure 2.9:Intersection over Union is used to measure the performance of detection

It is obvious that IoU loss, only works when the predicted bounding boxes overlap with the ground truth box. IOU loss would not provide any moving gradient for non-overlapping cases.

So, if we get a higher value of Intersection over Union, it means the overlap is better. Hence, the prediction is more accurate and better as depicted below.

We can say that the Intersection over Union allows us to measure and compare the performance of various solutions. It also makes it easier for us to distinguish between useful bounding boxes and not-so-important ones [26].

2.5.2 Mean Average Precision – mAP

Mean Average Precision (mAP) is one of the most comm only used evaluation metrics for object detection models, mAP is a metric based on Average Precision (AP) which is used to calculate the precision and recall of a model. This is done using the following formulas

$$Precision = \frac{TP}{TP + FP} \quad 2.2$$

$$recall = \frac{TP}{TP + FN} \quad 2.3$$

Precision in equation 2.2 is defined as the ability for an object detection model to identify objects, this is represented in percentage. Recall in equation 2.3 is defined as the ability for an object detection model to identify all objects in sample data, this is also represented in percentage. To determine TP (True Positive), FP (False Positive) and FN (False Negative) and also define when

a correct or incorrect detection is made, Intersection Over Union (IoU) is needed. A threshold is also predefined to determine if the object was actually detected, this threshold is often 50%.

If the object detection system detects an object within the IoU threshold it is seen as a True Positive, else it is seen as a False Positive. False Negatives occurs when an object detection system fails to detect an object that is present in an image. AP is used to calculate the mAP of an object detection model and is the average AP for all different classes of objects that the model is designed to detect, given by equation 2.4 where N represents the total number of classes and APi is the average precision for each class [27].

$$mAP = \frac{1}{N}\sum_{i=1}^{N} APi \qquad 2.4$$

2.6 Conclusion

In recent years, with the development of Deep Learning in object detection, multiple deep detection models are proposed. Throughout this chapter we have discussed one of the most important visual tasks which is the object detection, where we have focused on the CNN architecture of object detection and its two main classes: the two-stage and one-stage methods.

It should be noted that object detection has not been used much in various areas where it could be of great help. In the next chapter we will propose and implement our object detector on real medical images, we will also apply Yolo5 and compare them.

Chapter 3: Model implementation and experimental results

Abstract

Early diagnosis of brain tumors plays an important role in a patient's treatment and makes it easy easier to save his or her life. A major challenge in brain tumor treatment planning and quantitative evaluation is the determination of the tumor extent. In this chapter we have implemented two different neural network-based object detection models that include our constructed CNN model and pre-trained YOLOV5 model. Then, we have evaluated and compared the experimental results of these models.

3.1 Introduction

In this chapter, we propose an efficient and skillful method which helps in the detection of the brain tumor without any human assistance based on both Convolutional Neural Network and YOLOV5. First, we will describe the brain image dataset we have used to train our constructed convolutional neural network. Then detail the various CNN architectures that we have reconstructed to enhance the results each time. Lastly, we have applied YOLOv5 on our dataset to see how is efficient our method compared to YOLOv5; which is a family of object detection architectures and models pretrained on the COCO dataset (COCO is a well-known large-scale object detection, segmentation, and captioning dataset.) and at the same time learning how to apply a pretrained model on a real dataset and do retrain it on costume data.

3.2 Data

The dataset that we used for training our model was published in Kaggle website by Navoneel Chakarabarty [28], it contains 253 brain tumor images of two classes images with brain tumor and images with no brain tumor as shown in Table 3.1. Also, each image in the dataset had a different size.

Table 3.1: Number of images per classes

Classes	Number of images	Total
Brain tumor	98	253
No brain tumor	155	

These images dataset is generated through the scans, and they are examined by the radiologist. A manual examination can be error-prone due to the level of complexities involved in brain tumors and their properties. Figure 3.1 shows sample from the dataset class brain tumor.

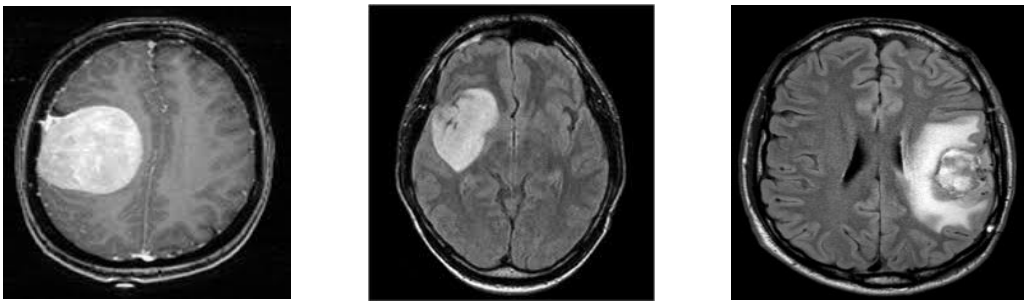

Figure 3.1: Sample from the dataset class brain tumor

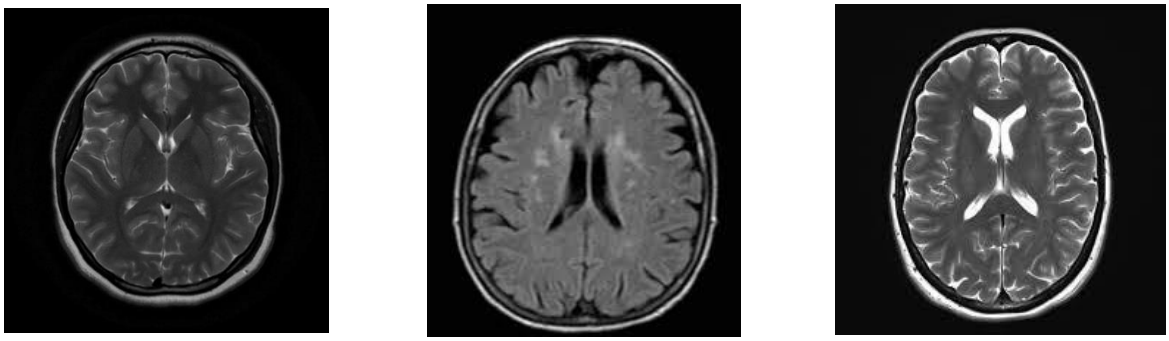

Figure 3.2: Sample from the dataset class no brain tumor

The dataset does not have bounding boxes around the objects. So, we had to label it manually using LabelImg, a graphical image annotation tool, this creates XML files in PASCAL VOC format. We should mention that this task was one of the hardest and most time consuming part; we have spent time to choose the right tool since there are many open-source image labeling tools, and then more extra time to get the labeled

We marked the location of the brain tumor by drawing a bounding box around the tumor to be detected (Figure 3.3).

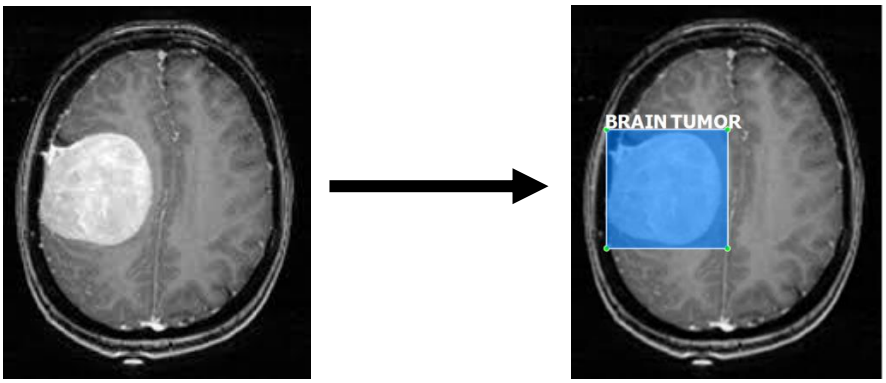

Figure 3.3: Mark the location of the brain tumr in form of bounding box using labelImg

These annotated images are saved in xml format Figure 3.4 show an xml file structure generated by LabelImg tool The XML file contains information about the image including the name of the image and the name of the dataset. The size and depth of the image are recorded under the ‘size” tab. The information under the ‘object’ tab is the content we marked before. Then ‘name’ tab records the class of the object. And ‘bndbox’ tab records the specific location of the object.

```
<annotation>
        <folder>MYDATASETS</folder>
        <filename>aug_Y1_0_2900.jpg</filename>
        <path>C:\Users\ANG KID LAGUZA\Desktop\MYDATASETS\aug_Y1_0_2900.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>300</width>
                <height>300</height>
                <depth>1</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>BRAIN TUMOR</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>153</xmin>
                        <ymin>112</ymin>
                        <xmax>271</xmax>
                        <ymax>209</ymax>
                </bndbox>
        </object>
</annotation>
```

Figure 3.4: Xml file for the annotated image exported by labelImg tool

3.3 Preprocessing

For the preprocessing we uploaded our dataset and annotations XML files to Roboflow. Roboflow is a Computer Vision development framework for better data collection to preprocessing, and model training techniques. Roboflow has public datasets readily available to users and has access for users to upload their own custom data also. Roboflow accepts various annotation formats. In data pre-processing, there are steps involved such as image orientations, resizing, contrasting, and data augmentations [43].

As we have said before our dataset has different sizes, so we won't be able to feed them directly to the network. In addition, large image sizes have implications towards not only neural network training time but also memory issues. We resized the images to $300x300$ pixels smaller which in turn make the training step faster.

We also adjusted the contrast of the images in order to improve the normalization and line detection in varying lighting conditions, and make our model better at edges detection.

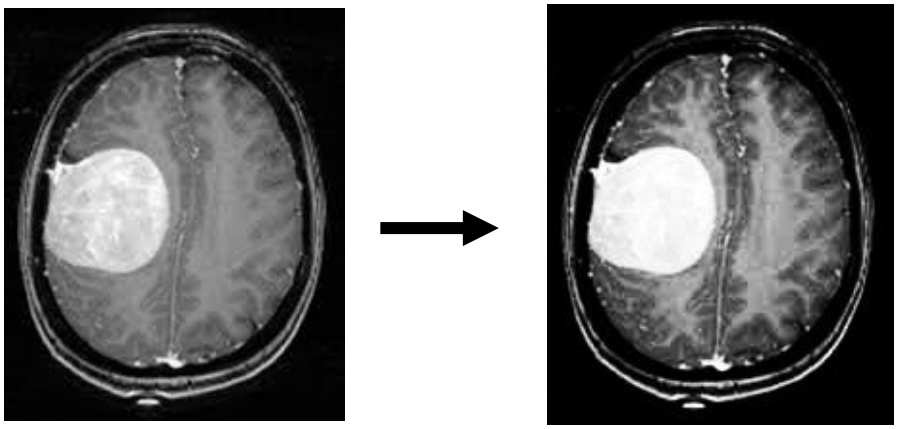

Figure 3.5: Image before and after contrast adjustment applied

3.4 Data augmentation

We know previously mentioned that overfitting can happen when neural networks weights memorize training data rather than generalize the input to discover patterns in the data. This is usually the case in small datasets. So, it was obvious to us that 253 images was too small an amount of data if we really wanted to have good training results. With this limited dataset brain tumor

images, our neural networks will be for sure at a high risk of overfitting. To avoid this, we needed to apply data augmentation.

We implemented a several forms of data augmentation:

- Rotation: Between - 45º and 45º
- Horizontal-flip: -180º or 180º
- Vertical-flip: -180º or 180º
- Outputs per training example: 3
- Brightness: between 0% and + 54
- Noise: up to 10% of pixels
- Bounding box exposure: Between -8% and +8%

After we applied data augmentation process, we have obtained 1256 images of brain tumor; which is still not enough but acceptable. Figure 3.6 show sample from dataset after augmentation.

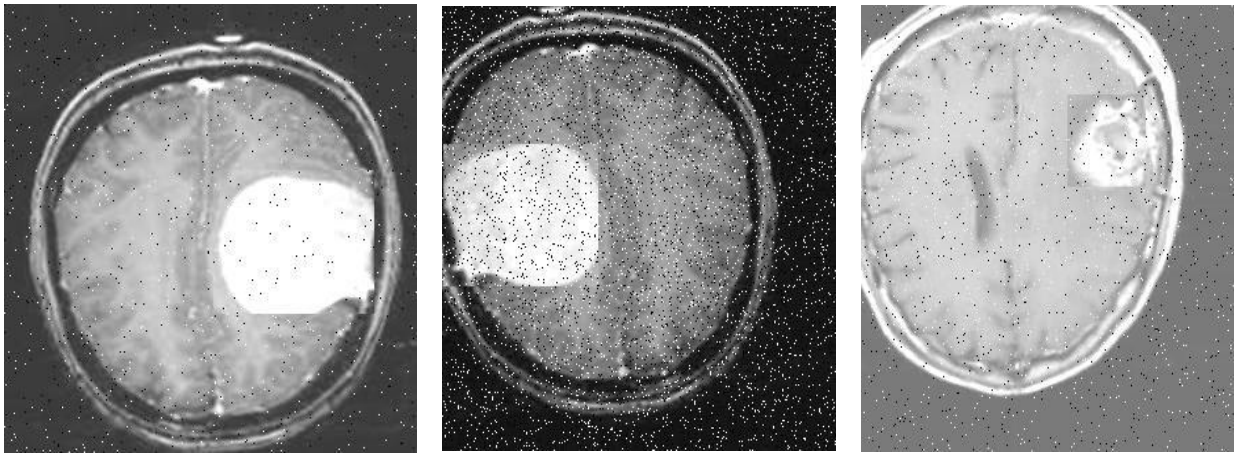

Figure 3.6: sample from dataset after augmentation

Roboflow gives you the ability to split your dataset to whatever we want, so after we did the data augmentation, we split our data 80% for training, 10% for validation and 10% for testing. Table 3.2 show on number and percentage our split dataset.

Table 3.2: Number of training image, validation and testing

	Datasets			
	Training	Validation	Testing	Total
Percentage	80%	10%	10%	100%
Number	1004	126	126	1256

After that, we exported our augmented dataset into two versions with different formats, first version had CSV format so we can use it for train our constructed model and the second version is TXT annotationis and YAML config that we will use later to train YOLOv5 on our custom dataset.Table 3.3 shows the content of the CSV file.

Table 3.3:Content of CSV file

Filename	***Width***	***height***	***class***	***xmin***	***ymin***	***xmax***	***ymax***
aug_Y154_0_1224.jpg	300	300	BRAIN_TUMOR	1	160	47	213
aug_Y19_0_1106.jpg	300	300	BRAIN_TUMOR	57	81	128	137
38 no.jpg	300	300	NO_BRAIN_TUMOR	0	0	0	0
41 no.jpg	300	300	NO_BRAIN_TUMOR	0	0	0	0
aug_Y37_0_5320.jpg	300	300	BRAIN_TUMOR	157	95	235	169
aug_Y77_0_390.jpg	300	300	BRAIN_TUMOR	83	143	202	221

Each XML file contain annotation parameters of one single image, so we converted all the XML files into one CSV file that gathers all the annotation parameters of the whole dataset. Now, our file is organized and ready for use.

3.5 Construction of our proposed convolution neural network detector

For our implementation, we reconstructed the convolution neural network that we have introduced in chapter 1 using Keras as it is an effective high-level neural network, Application Programming Interface (API) written in Python to create our CNN model. We have used Google Collaboratory platform to execute our code due to the performance of GPU that saved for us so much time. In the next section, we are going to describe the architecture of our CNN detector in detail

In the experimental part, we have tested three combinations of layers as our first combination of layer used to construct the CNN model was really terrible at prediction. So, we had to try other combinations in order improve the results.

In the first combination of layers, we have dealt with the problem of detection as a regression task so we did not include an output for classification. The output of the CNN had just four neurons to give us the bounding box annotations parameters $(xmin, ymin, xmax, ymax)$. To construct the CNN, we used Keras sequential model, which is a linear stack of layers:

- Convolutional layer with 64 filters of size 5 x 5 and stride of 1
- Max-pooling layer with pool size 2 x 2
- Convolutional layer with 128 filters of size 5 x 5 and stride of 1
- Max-pooling layer with pool size 2 x 2
- Convolutional layer with 256 filters of size 5 x 5 and stride of 1
- Max-pooling layer with pool size 2 x 2
- Fully Connected layer with 100 neurons, in form of three dense layers: input dense layer with 64 neurons, hidden dense layer 32 neurons, output dense layer 4 neurons, all FC layer neurons have ReLu as activation function.

In the second combination of layers, we added another convolutional layer with 32 filters of size 5 x 5 and stride of 1 as input layer of the model and another dense layer with 128 neurons as input for the fully connected layer. Also, we changed the activation function for the output dense layer to sigmoid function.

The last combination of layers, was the same as the second combination but we added another fully connected layer for classification of 2 output neurons for classified image if they contain brain tumor or not with SoftMax as activation function for the last layer. In order to apply this combination we needed to use Keras functional API because in this case our model has multi-output. In other words a double fully connected layer, we need these two fully connected layers to be trained at the same time and get their inputs from the same source. In Figure 3.7 we can see two main outputs one for the bounding box named “box_output” with 4 neurons, and the other one for

the classification of the brain tumor; if it exists or not in the images named “class_output” with 2 neurons. We added a couple of dropout layers to the FC to help in the regularization and over fitting Figure 3.7 illustrates the architecture of the last combination of layers.

3.6 Training

3.6.1 CNN model Training

We already split our data into training, validation and testing data. We used validation data for updating weights while it gave a glimpse into how the neural network was improving over time. After the training phase was completed, the test data was then used to see how well the neural networks predicted tumors from new images.

A variety of hyperparameters are available to alter. We list the hyperparameters that produced the highest accuracies.

- Learning rate: 0.001
- Adam optimizer: beta1=0.9, beta2=0.999
- Batch size and epochs we tried different value during the testing the models
- First and second combination layers: we used Mean Square Error as Loss function
- Third combination layers: we used Mean Square Error loss function for bounding-box output and Categorical Cross Entropy for classification output.

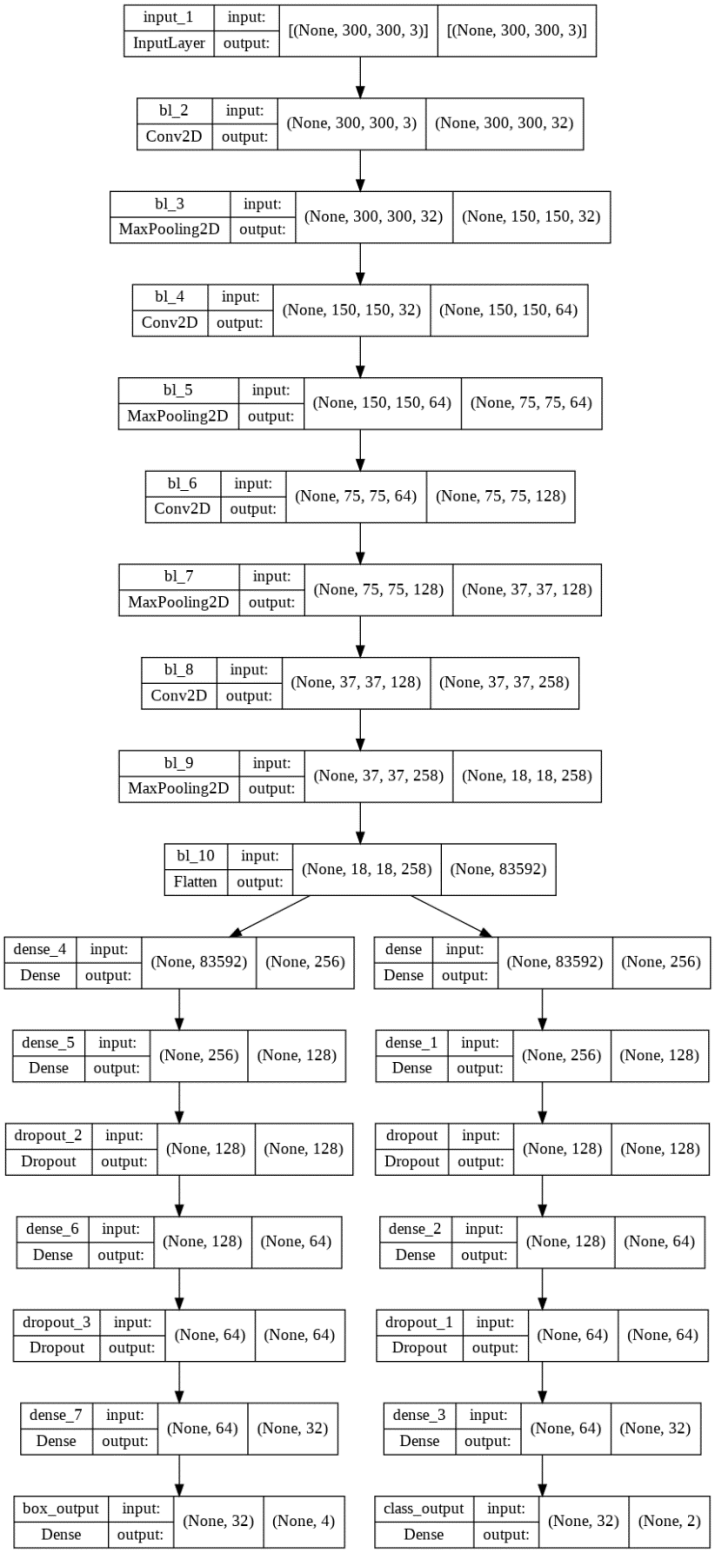

Figure 3.7: show the architecture of the last combination of layers.

Table 3.4: The total number of params in each combination

Combination layers	Total params
First	22,983,398
Second	11,135,398
Third	22,505,160

Table 3.5: show batch size and epochs value that we tried during each test

Combination layers	Test	Batch size	Epochs
First	1	50	10
	2	32	30
Second	1	32	10
	2	32	100
Third	1	16	100

To avoid neglecting any detail you may be interested in knowing, Figure 3.8 shows an organigram of the whole process of training our convolutional neural network detector.

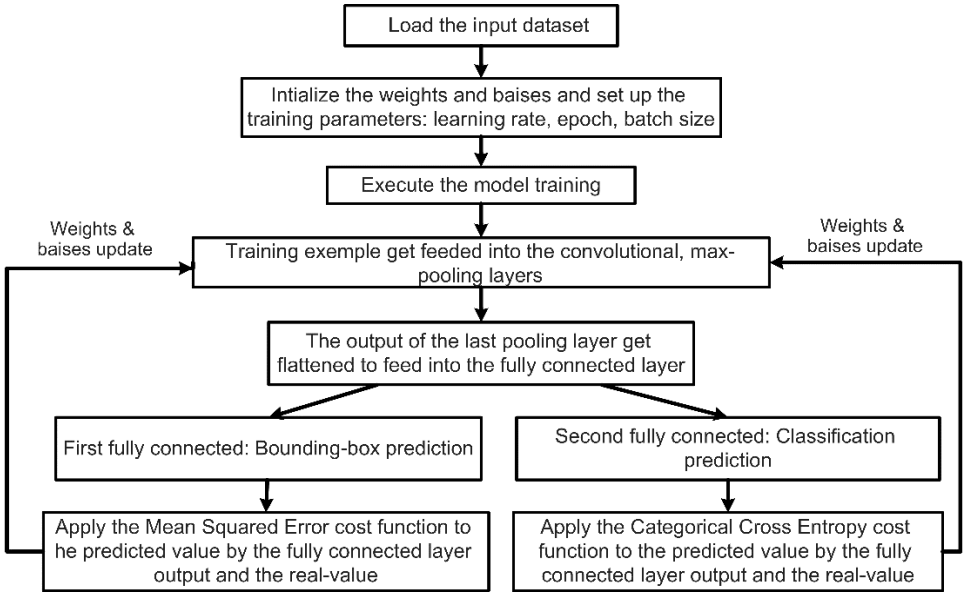

Figure 3.8: Training Convolutional neural network model organigram

Keras library gave us the ability to see the feature maps of the convolutional neural network, and the ability to see what your model tries to focus on learning. Figure 3.9 shows our model feature maps in the first layer.

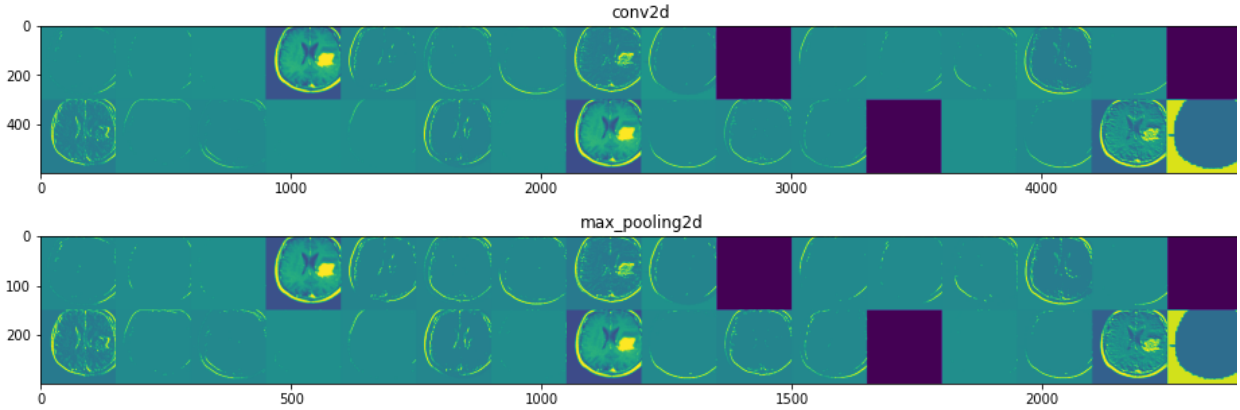

Figure 3.9:Show the feature maps of the first convolution and max-pooling layer

As we can see each feature maps from first convolutional layer focuses on different regions in the image, there is a feature map that concentrates on the brain edge, others focus in brain tumor and some of them focus on the whole brain texture and so on. Also, we have the ability to see a scale displayed in the figure so we can observe that max-pooling layer decreased the size of the data. Figure 3.10 Show the feature maps of the last convolutional and max-pooling layer

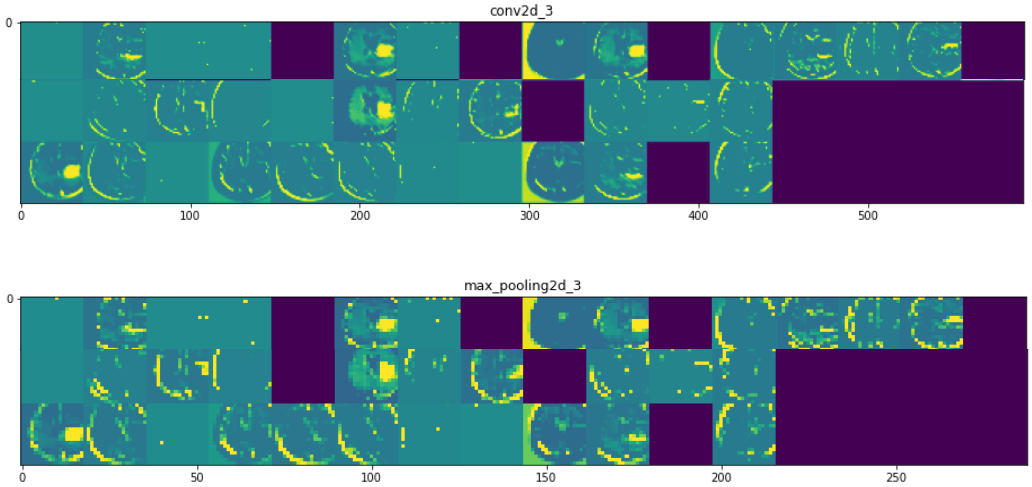

Figure 3.10:Show the feature maps of the last convolutional and max-pooling layer

Figure 3.10, we can see the last layer detected way more details in images if we compare it to first layer. Also, we can see the difference in size in the max pooling.

After constructing our detector, next we will see how to train YOLOv5

3.6.2 YOLOv5 training

we know that YOLOv5 is already trained on COCO dataset, but we need to train it our brain tumor dataset to predict a new object. Table 3.6 shows batch size and epoch values of YOLOv5 training on our custom dataset also the model summary.

Table 3.6: Shows batch size, epoch values and layers, parameters number

	Model summary		Batch size	Epoch
	Layers	Parameters		
YOLOv5	232	7246518	16	100

We plotted Graph of YOLOv5 training loss, recall and precision metrics and mAP, using TensorBoard, TensorBoard is a Tensorflow library for providing the measurements and visualizations needed during model training.

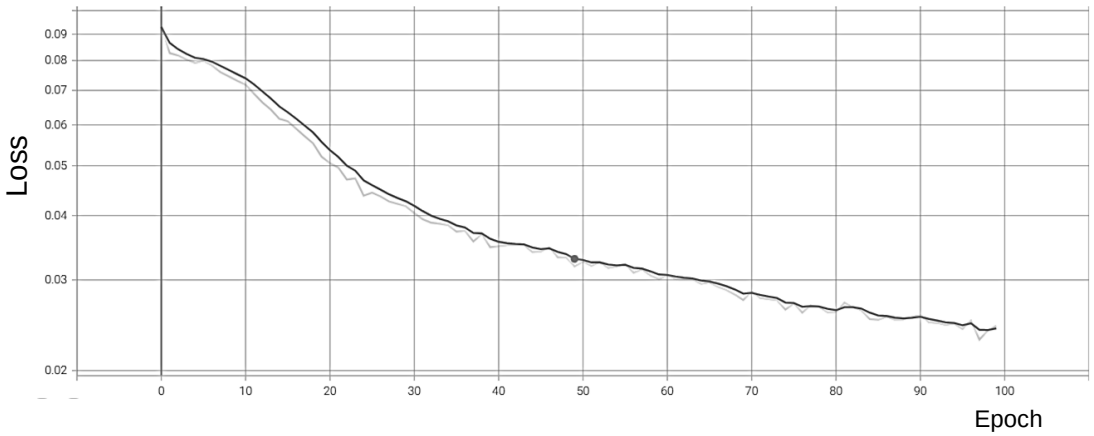

Figure 3.11:Graph of YOLOv5 training loss

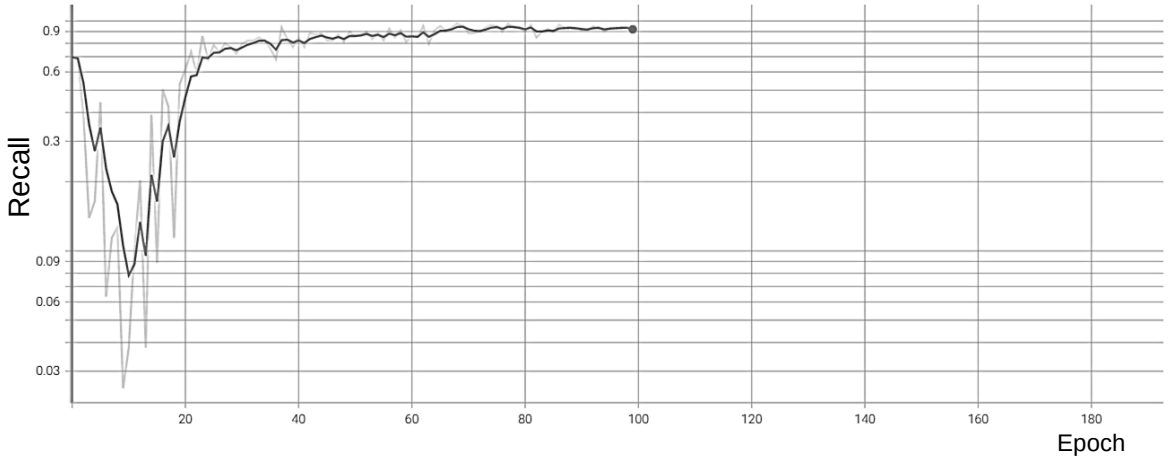

Figure 3.12:Graph of YOLOv5 metrics recall during training

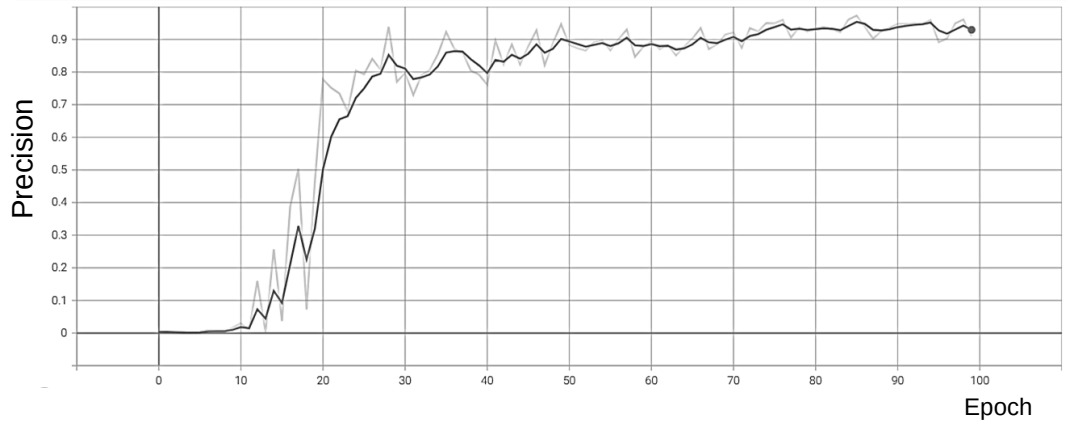

Figure 3.13: Graph of YOLOv5 metrics precision during training

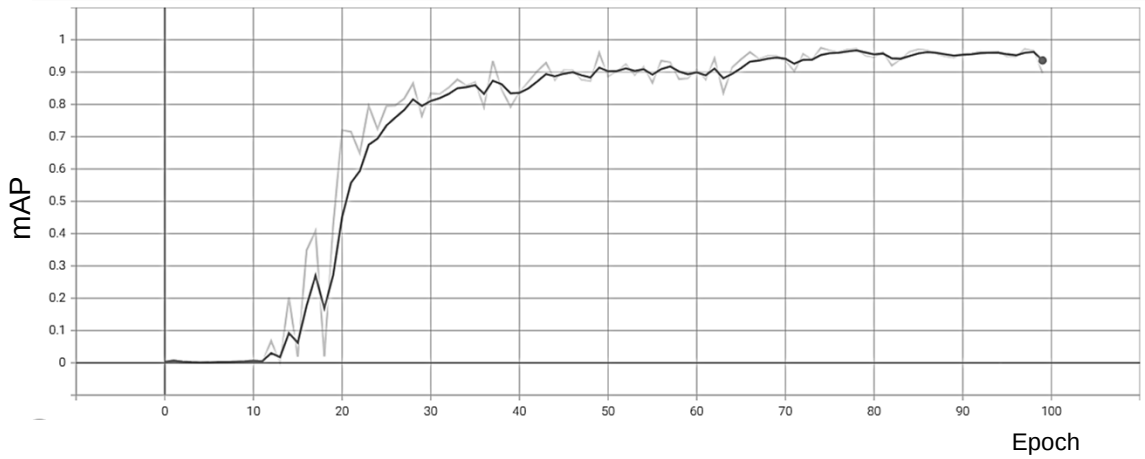

Figure 3.14:Graph of YOLOv5 mAP values during training

3.7 Results

3.7.1 CNN model Results

In this section we going to see our different results for each combination. You can refer directly to Table 3.5 to be able to follow our results analysis.

3.7.1.1 First combination (Test one) results:

As can be seen in the figure 3.15 below, the first combination outputs unacceptable results since there is no overlapping between the red box which represents the model prediction and the green box which represents the real box which means all IoU values are equal to 0 as shown in the figure 3.15 , above the red boxes.

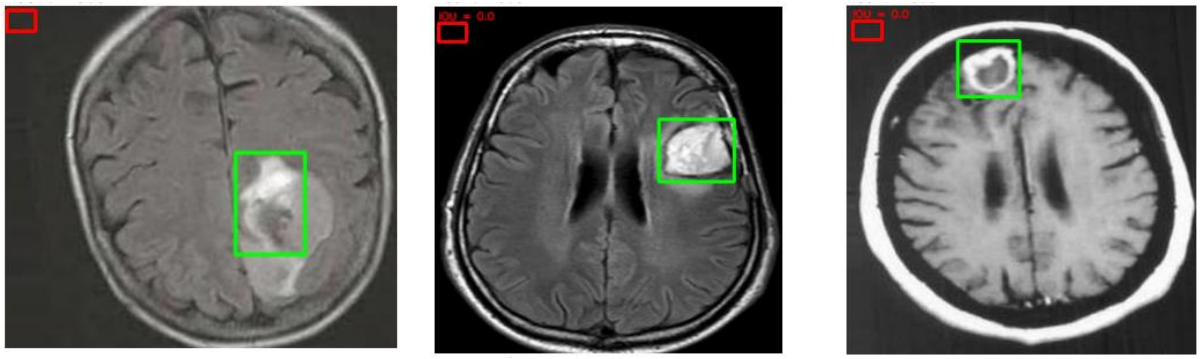

Figure 3.15: First combination layer prediction results

3.7.1.2 First combination (Test two) results

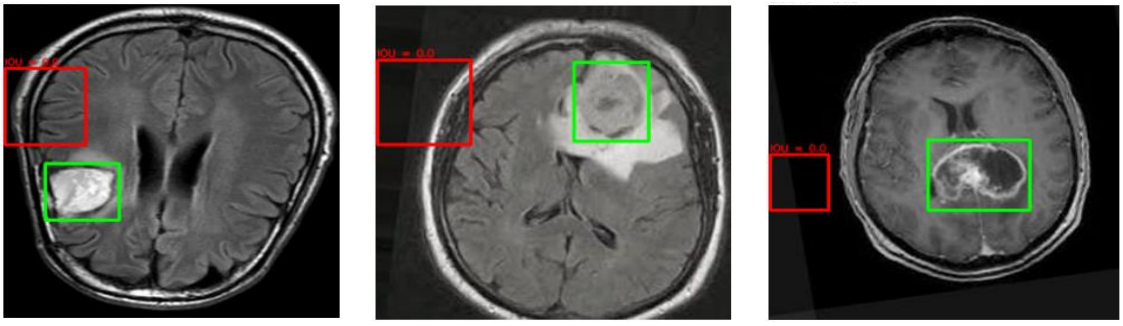

Figure 3.16:Sample from first combination test two results with bad prediction

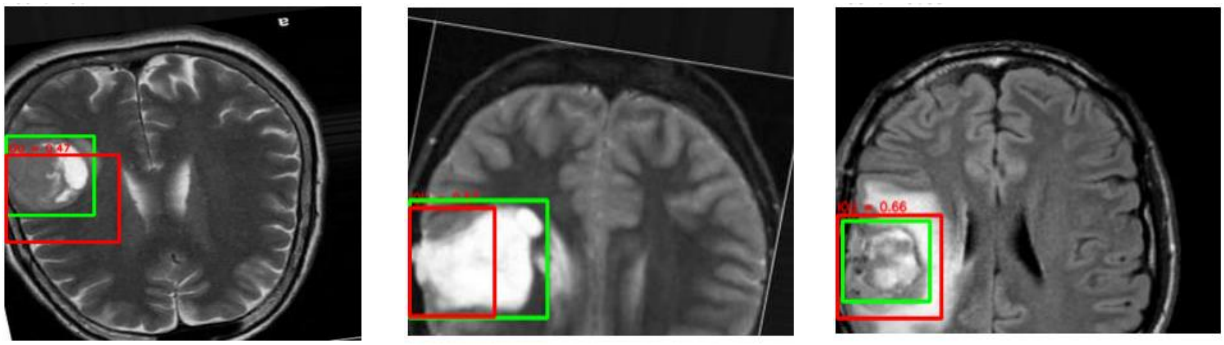

Figure 3.17:Sample from first combination test two results with a good prediction

By analyzing these images, we can notice that our model gave a good prediction in the images where the brain tumor localization was near to left side, that means when $xmin$ (parameter of brain tumor localization) almost equals zero our model gave us really good results as we can see in figure 3.17. On the other hand, when brain tumor localization was far from left side our model couldn't predict the location of the green box, figure 3.16. To overcome this problem, we have proposed another combination and did further test on it. The IoU values can be seen above the red boxes

3.7.1.3 Second combination (Test one) results

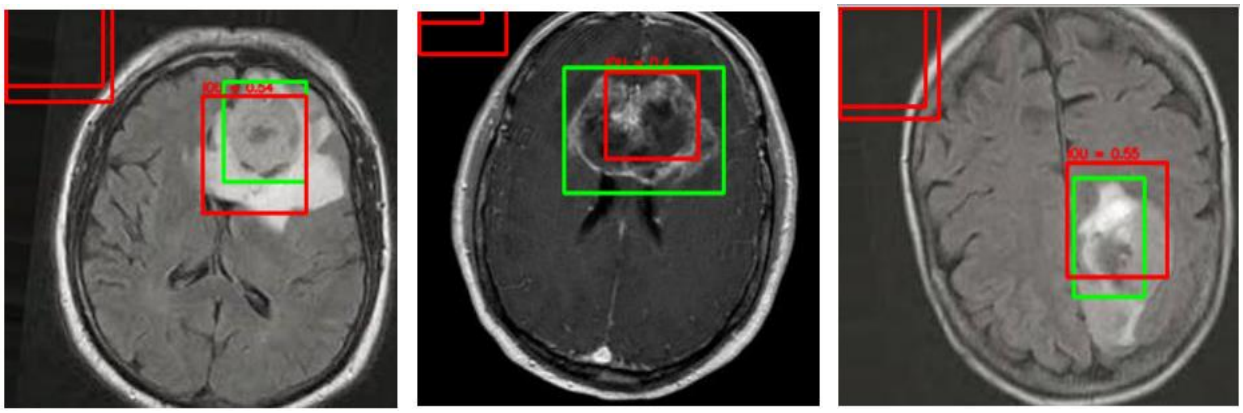

Figure 3.18: Sample of second combination test one results

As can be observed, there is a big difference between the second and first combination results. We can see that the obtained results are better, The IoU obtained are higher. The multiple red bounding boxes are due to the execution of our code multiple times without restarting the

runtime in Google Collaboratory. Google Collaboratory saved any previous executions so when we re-executed the code it was going to plot new results over the last results.

3.7.1.4 Second combination (Test two) results

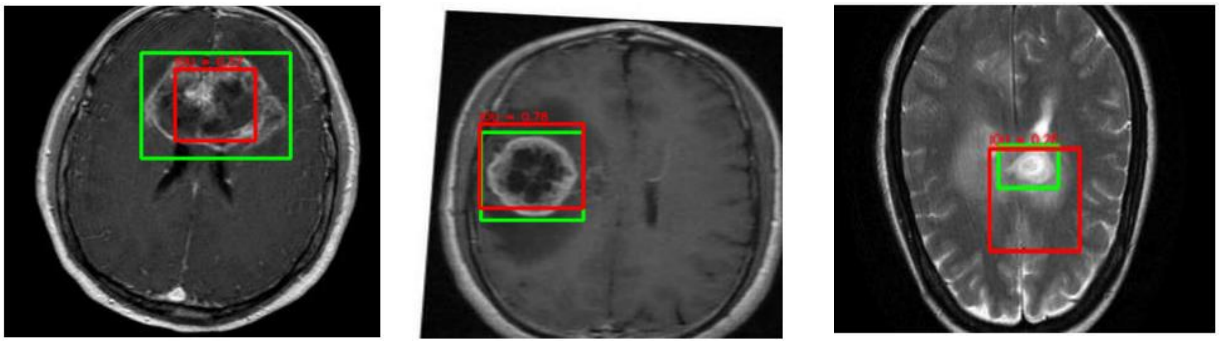

Figure 3.19: Sample from results of the second combination test two

This is the best result so far in comparison to the previous results, In this case the model was be able to detect even small brain tumor and gave higher IoU number.

3.7.1.5 Third combination results

The last combination was designed to predict the bounding-box annotations and classified the input:

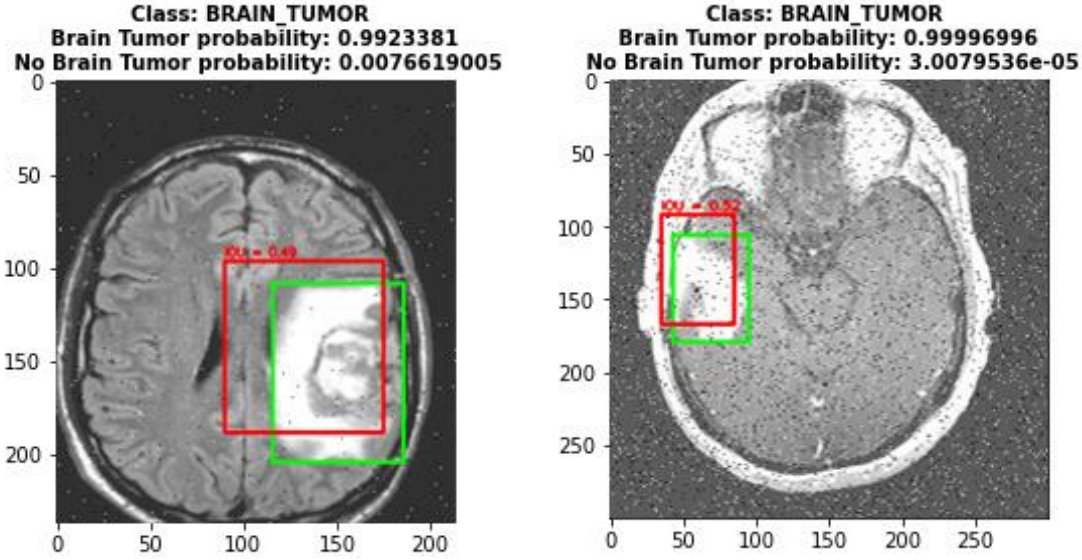

Figure 3.20:Simple of results from third combination with class brain tumor

Figure 3.20 shows the results of our third model combination on images that contain brain tumor. The model did really good at classification, as we can see on the top of each image there are two outputs. First, brain tumor probability we have gotten 0.99 means that these two images were classified as brain tumor images. Second, no brain tumor probability equals to 0.00 and 3.00 meaning that no brain tumor exists. Also, the model did really good at bounding-box annotations prediction.

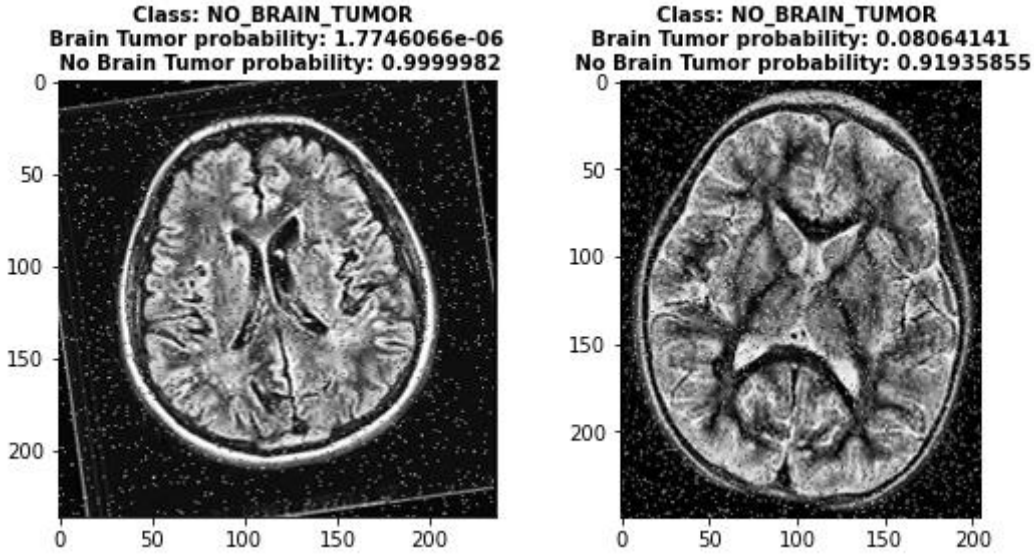

Figure 3.21:Sample from results of the last combination with class no brain tumor

The Figure 3.21 shows sample from results of the last combination with class no brain tumor, and this model did really good also at no brain tumor class.

3.7.2 Evaluation of the CNN model on test data

We evaluated the different combinations with testing dataset. So, we can see how it going to perform with unseen data. Table 3.7 show MSE calculated in each test:

Table 3.7: MSE calculated in each test for the first and the second combination

Combination layers	**Test**	**Evaluation [Metrics=Mse]**
First	1	0.16479593515396118
	2	0.10943473875522614
Second	1	0.09307263180613518
	2	0.03434731811285019

For The third combinations we did one test, the Table 3.8 of evaluation is bet different:

Table 3.8:Third combination evaluation

	Evaluation			
	Total loss	Classification loss	Bounding-box [Metrics=Mse]	Classification [Metrics=Accuracy}
Third	7.6591e-04	2.4991e-05	1.4092e-04	1.0

3.7.3 YOLOv5 Results

After we trained the YOLOv5 on our brain tumor dataset, we tested the performance on the unseen data (test dataset). Figure 3.22 shows the results of the YOLOv5 on test data:

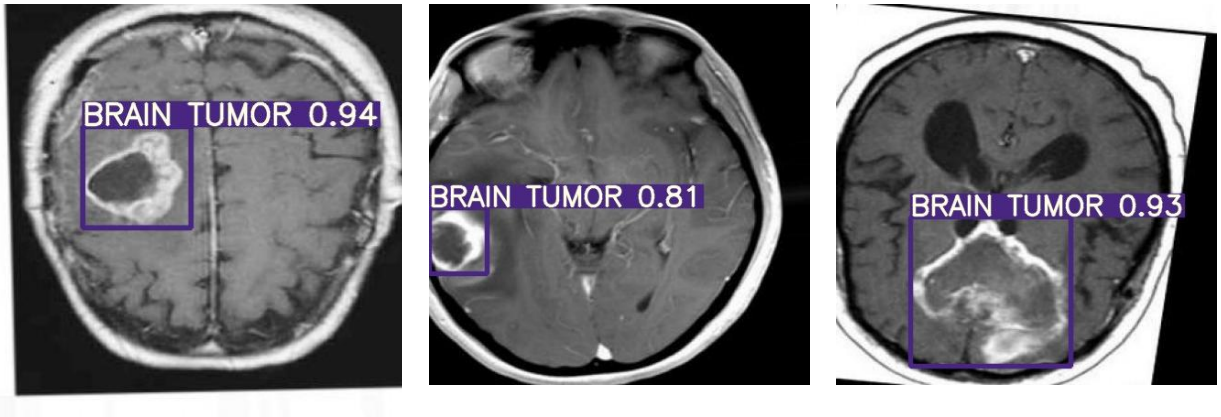

Figure 3.22:Sample from the results of the YOLOv5 on test

We can see in figure 3.22 that YoloV5 gave a high IoU values. In each image we have printed the Intersection Over Union value in purple.

After completing the test task, we observed that even strong models such as YOLOv5 can fail sometimes. Figure 3.23 shows a couple of results where YOLOv5 wasn't be able to detect the brain tumor in them.

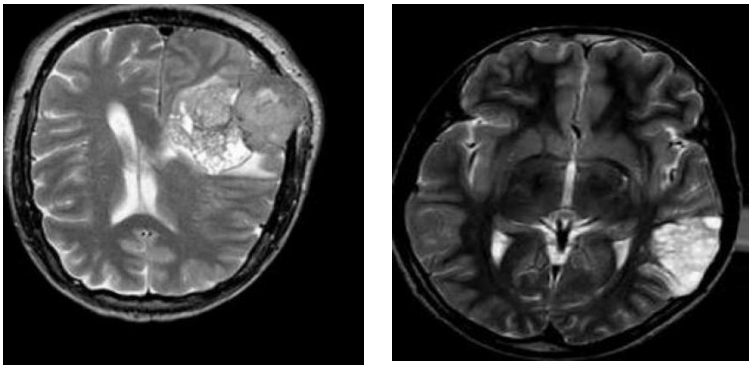

Figure 3.23: Result that YOLOv5 wasn't able to detect the brain tumor

3.7.4 Comparison of the result

Before we do the comparison, we need to clarify that our goal from this comparison isn't really to compete YoloV5 detector, but to understand this pretrained model's efficiency in object detection. By comparing the intersection values of the YOLOv5 model and our detector (third combination), we have seen that YOLOv5 was way better at detecting big and small tumors. It was able to localize the tumor with high accuracy, as demonstrated in the IoU values. Although, our model didn't give comparable results due to many factors, but it was acceptable for us as it is our first try to construct an object detection detector. Our third combination succeeded in detecting the tumor and classifying the images.

Figure 3.24 shows a comparison between YOLOv5 and our CNN model in terms of IoU values.

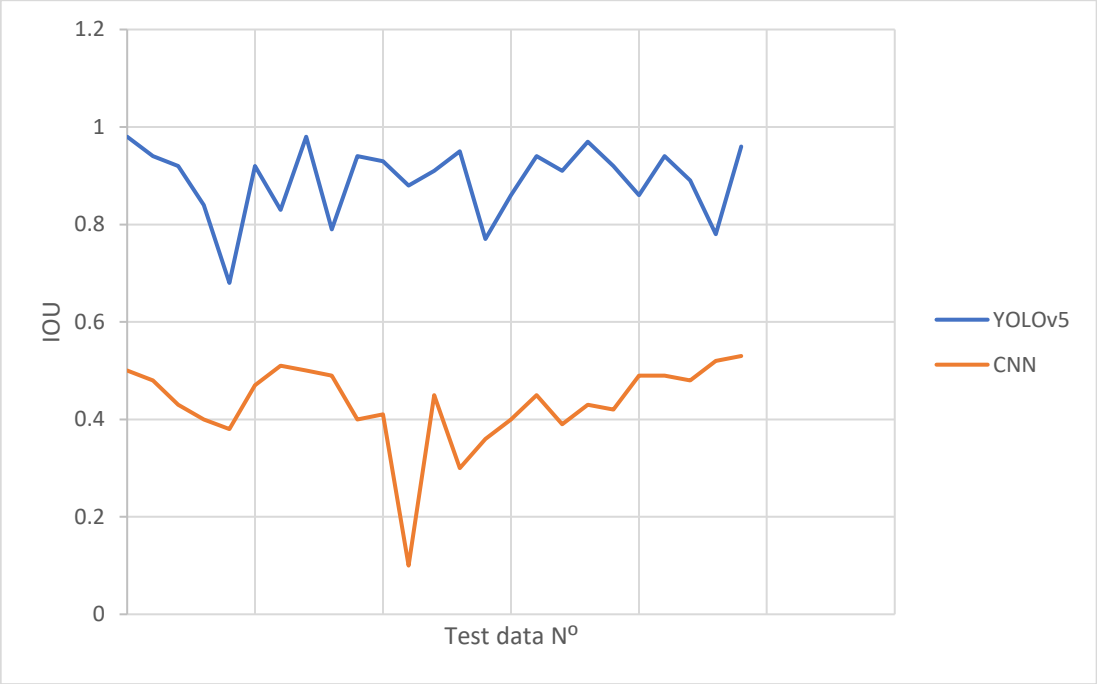

Figure 3.24:Graph show YOLOv5 and CNN Model IoU values on the test data

3.8 Conclusion

Considering the needs to detect brain tumor, along this chapter we have explained all the steps that we have followed to develop our CNN model for tumor detection. First, we have applied data preprocessing and data augmentation, then we detailed the different CNN models we have used. We have proposed three combinations, the first one failed in detecting the tumor, so it was necessary to enhance the results, the third was be able to detect the tumor, it gave us acceptable IoU values. Also, we trained YoloV5 detector on the same dataset to see how approximate our model performs to the YoloV5 detector.

GENERAL CONCLUSION

We have come a long way since chapter 1, where we managed to build our own object detector with convolutional neural networks.

To recap, in chapter 1, we introduced the domain of machine learning, deep learning neural networks, and convolutional neural networks. We have focused on CNNs as they are widely used in object detection application.

In chapter 2, we have understood what is exactly object detection (Classifying multiple objects in an image and placing bounding boxes around them). We have clarified the main differences between object detection, semantic segmentation and instance segmentation. We also gave an example of the two main classes method of object detection: R-CNN and YOLO for the two-stage and one-stage methods respectively. Finally, we explained the metrics used to evaluate the accuracy of our proposed object detection detector in the practical part.

In chapter 3, we have implemented an object detection model able to localize and classify brain tumors on real medical images. We have explained all the steps we followed throughout this experimental part starting from data preprocessing and data augmentation and ending with the implementation of our detector. We have also trained YoloV5 detector on the same dataset to see how approximately our model performs compared to it.

This work has introduced us to a new field of Deep Learning which is a very large and promising specialty to us as students who want to pursuit our studies as researchers.

REFERENCES

[1] Xinyi Zhou, Wei Gong, WenLong Fu, Fengtong Du, “Application of Deep Learning in Object Detection”, 2007, IEEE article

[2] S. Ray. “A Quick Review of Machine Learning Algorithms”. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon). 2019, pp. 35–39. doi: 10.1109/COMITCon.2019.8862451.

[3] James Loy, “Neural Network Projects with Python: The ultimate guide to using Python to explore the true power of neural networks through six projects” 1st Edición, Edición Kindle

[4] Ye, Ning Zhang, Yingya Wang, Ruchuan Malekian, Reza. “Vehicle trajectory prediction based on hidden Markov model”, Research article, 2016-07

[5] Arnav Thakur ,Reza Malekian. “Fog Computing for Detecting Vehicular Congestion, an Internet of Vehicles Based Approach: A Review”. In: IEEE Intelligent Transportation Systems Magazine 11.2 (2019), pp. 8–16. doi: 10. 1109/MITS.2019.2903551.

[6] Bojan Furundzic, Fabian Mathisson, “Dataset Evaluation Method for Vehicle Detection Using TensorFlow Object Detection API”, Malmo University Faculty of Technology and Society Computer Engineering Bachelor Thesis.

[9] Michael A.Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

[12] Mohit Goyal, Rajan Goyal, Brejesh Lall, Studies in Computational Intelligence (2020)

[14] Murphy, Kevin P. (2012). Machine Learning: A Probabilistic Perspective. Cambridge: MIT Press. p. 247. ISBN 978-0-262-01802-9.

[15] Buduma, Nikhil; Locascio, Nicholas (2017). Fundamentals of Deep Learning : Designing Next-Generation Machine Intelligence Algorithms. O'Reilly. p. 21. ISBN 978-1-4919-2558-4.

[16] R.C Joshi Rahul Chauhan Kamal Kumar Ghanshala. “Convolutional Neural Network (CNN) for Image Detection and Recognition”. In: 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). 2018.

[18] Azulay, Aharon; Weiss, Yair (2019). "Why do deep convolutional networks generalize so poorly to small image transformations?". Journal of Machine Learning Research. 20 (184): 1–25. ISSN 1533-7928.

[21] Katleho L Masita, Ali N Hasan, ThokozaniShongwe, Deep Learning in Object Detection:

[22] Paper, D. (2021). Deep Learning with TensorFlow Datasets. In: State-of-the-Art Deep Learning Models in TensorFlow. Apress, Berkeley, CA. https://doi.org/10.1007/978-1-4842-7341-8_4

[23] Aneela Aslam, Aun Irtaza, Nudrat Nida, Object Detection and Localization in Natural Scenes Through Single-Step and Two-Step Models (IEEE article) Published: 2020

[25] Bochkovkly, A., Wang, C.Y., & Liao.H.Y.M. (2020), Yolov4 : Optimal speed and accuracy of object detection arXiv preprint arXiv:2004.10934

[26] Himanshu Singh," Practical Machine Learning and Image Processing: For Facial Recognition, Object Detection, and Pattern Recognition Using Python " Apress; 1st ed. édition (26 février 2019)

[27] Xiaoyue Jiang • Abdenour Hadid • Yanwei Pang Eric Granger • Xiaoyi Feng Editors "deep learning in object detection and recognition "Springer Verlag, Singapore; 1st ed. 2019 édition (27 novembre 2019)

[37] Di Feng et al. "Deep Multi-Modal Object Detection and Semantic Segmentation for Autonomous Driving: Datasets, Methods, and Challenges". In: IEEE Transactions on Intelligent Transportation Systems 22.3 (2021), pp. 1341–1360. doi: 10.1109/TITS.2020.2972974

WEBOGRAPHY

[7] Lisa Tagliaferri, “An Introduction to Machine Learning”, https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning#deep-learning, September 28, 2017

[8] Nilesh Barla, A Gentle Introduction to Deep Learning—the ELI5 Way, https://www.v7labs.com/blog/deep-learning-guide, June 20, 2022

[10] Pragati Baheti, “Neural Network Architectures”, https://www.v7labs.com/blog/neural-network-architectures, June 20, 2022

[11] Pragati Baheti, “The Essential Guide to Neural Network Architectures”, https://www.v7labs.com/blog/neural-network-architectures-guide, June 20, 2022

[13] Shruti Jadon, "Introduction to Different Activation Functions for Deep Learning", https://medium.com/@shrutijadon/survey-on-activation-functions-for-deep-learning-9689331ba092. Mar 16, 2018

[17] From Wikipedia, the free encyclopedia, “en.wikipedia.org/wiki/Convolutional_neural_network#cite_note-:5-61”

[19] Nitish Kumar, “Understanding The Concept Of Convolutional Neural Networks (CNNs)”, https://www.marktechpost.com/2022/01/24/a-detailed-understanding-of-convolutional-neural-networks/, January 24, 2022.

[20] Pragati Baheti, “A Comprehensive Guide to Convolutional Neural Networks”, https://www.v7labs.com/blog/convolutional-neural-networks-guide, June 20, 2022.

[24] Nirmala Murali, ‘Image Classification vs Semantic Segmentation vs Instance Segmentation’,https://nirmalamurali.medium.com/image-classification-vs-semantic-segmentation-vs-instance-segmentation-625c33a08d50, Apr 29, 2021

[28] NAVONEEL CHAKRABARTY, https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection, 2019

Abstract

This master's thesis tackles the problem of Object Detection which is one of the most famous and extensively researched topics in the field of Computer Vision. There are two main methods for object detection using convolutional neural networks: two stage and one stage methods. We explain two examples of object detection architectures **R-CNN**, and **YOLOv5** respectively. We propose in this thesis our CNN architecture to detect tumors in a real MRI dataset and compare our results to **YOLOv5** model.
Keywords: Deep Learning, Deep Neural Network, CNN, Computer Vision, Object detection, R-CNN, YOLOv5

Résume

Ce mémoire de maîtrise aborde le problème de la détection d'objets qui est l'un des sujets les plus célèbres et les plus étudiés dans le domaine de la vision par ordinateur. Il existe deux méthodes principales pour la **détection d'objets** à l'aide de **réseaux de neurones convolutifs** : les méthodes à deux étapes et à une étape. Nous expliquons deux exemples d'architectures de détection d'objets R-**CNN** et **YOLOv5** respectivement. Nous proposons dans cette thèse notre architecture **CNN** pour détecter les tumeurs dans un jeu de données IRM réel et comparer nos résultats au modèle **YOLOv5**.
Mots-clés : Apprentissage profond, Neurones Profonds, Réseau de Neurones profond, CNN, Détection Objets, R-CNN, YOLOv5, Vision par ordinateur.

ملخص

تتناول أطروحة الماجستير هذه مشكلة اكتشاف الأشياء التي تعد واحدة من أشهر الموضوعات التي تم بحثها على نطاق واسع في مجال رؤية الكمبيوتر. هناك طريقتان رئيسيتان لاكتشاف الأشياء باستخدام الشبكات العصبية التلافيفية: طريقتان وطريقة مرحلة واحدة. نفسر مثالين على بنيات اكتشاف الكائنات R-CNN و YOLOv5 على التوالي. نقترح في هذه الأطروحة بنية CNN الخاصة بنا لاكتشاف الأورام في مجموعة بيانات التصوير بالرنين المغناطيسي الحقيقية ومقارنة نتائجنا بنموذج YOLOv5.

كلمات مفتاحية: التعلم العميق ، الشبكة العصبية العميقة ، سي إن إن ، رؤية الكمبيوتر ، كشف الأشياء ، R-CNN، YOLOv5