5-2021

# Object Detection and Recognition Using YOLO: Detect and Recognize URL(s) in an Image Scene

John Ajala
*St. Cloud State University*

# Object Detection and Recognition Using YOLO: Detect and Recognize URL(s) in an Image Scene

by

John Temiloluwa Ajala

A Thesis

Submitted to the Graduate Faculty of

St. Cloud State University

in Partial Fulfilment of the Requirements

for the Degree of

Master of Science

in Computer Science

June, 2021

Thesis Committee:
Julstrom Bryant, Chairperson
Sarnath Ramnath
Anda Andrew

**Abstract**

The world in the 21st century is ever evolving towards automation. This upsurge seemingly has no decline in the foreseeable future. Image recognition is at the forefront of this charge which seeks to revolutionize the way of living of the average man. If robotics can be likened to the creation of a body for computers to live in, then image processing is the development of the part of its brain which deal with identification and recognition of images.

To accomplish this task, we developed an object detection algorithm using YOLO, and acronym for “You Only Look Once”. Our algorithm was trained on fifty thousand images and evaluated on ten thousand images and employed a 21 x 21 grid. We also programmed a text generator which randomly creates texts and URLs in an image. A record of useful information about the location of the URLs in the image is also recorded and later passed to the YOLO algorithm for training.

At the end of this project, we observed significant difference in the accuracy of URL detection when using an OCR software or our YOLO algorithm. However, our algorithm would be best used to specify the region of interest before converting to texts which greatly improves accuracy when combined with OCR software.

**Keywords:** Object Detection; Image Recognition; OCR; AI; World Wide Web; YOLO.

**Acknowledgements**

I would like to express my deepest appreciation to God for life. I would also like to extend my deepest gratitude to all the Professors at St. Cloud State University, who helped make this academic journey worthwhile. Learning Software Development concepts, Database concepts, Data Structure and Algorithm concepts, Machine Learning concepts, and other important concepts relevant to “real-time” applications of Software Development was an awesome experience. I am extremely grateful for everything I got to learn in this University.

I would like to thank my friends, Alienyi Daniel, Ojo Ore, Egwu Benedict, Nwachi Eurel, et al. for their constant support and encouragement. Finally, and most importantly, I want to thank my family for believing in me and constantly encouraging me throughout this journey.

# Table of Contents

Page

Chapter Page

## List of Figures

Figure Page

Figure Page

# Chapter I: Introduction

## Overview

In this chapter, we introduce the background and focus area of our thesis. We highlighted our motivations, scope, and thesis organization. We will explore YOLO's potential in recognizing objects and creating bounding box around them.

## Motivation

Image recognition/image processing is in the forefront of Artificial Intelligence today. It is however far from perfection. Seemingly simple scenarios, such as object detection, face recognition, removing motion blur, etc. and more complex scenarios such as compression artefacts, scratch detection, sensor noise, and spilling detection are applications of image recognition/image processing.

Digitized images are often represented as a two-dimensional (2D) array of pixels values. Each pixel value which makes up the color scheme of the image is often influenced by an array of factors such as light intensity. Visual scene is projected unto a surface, where receptors (natural or artificial) produce values that depend on the intensity of incident light.

These exciting concepts are however hard to implement. Forming an image leads to loss of details of information while collapsing a three-dimensional (3D) image into a two-dimensional image. Many other factors are responsible for why image recognition/ image processing is hard. Some of such factors are noise in the image (pixels values that are off from its surrounding pixels), mapping from scene to image etc.

In recent years, during the ImageNet Large Scale Visual Recognition Competition (ILSVRC, 2015), computers were going better than humans in the image classification task [35].

In 2016, a faster object detector, YOLO, was proposed to implement object detection in real-time situation. Our motivation is to apply YOLO to object detection task of URL links within an image scene. We will also be comparing the speed and accuracy of this with an OCR software.

**Research aim**

The research aim of this thesis is design and implement an AI algorithm to detect URL links within image scenes using YOLO. Subsequently, we will use an Optical Character Recognition (OCR) software to convert these images to text and use regular expression to search for links within the text. Both approaches will be compared for accuracy. We also proposed a few improvements and future work later.

**Problem description and Research Questions**

The first research question of this study is to verify whether YOLO is a good model for general object detection or not. Furthermore, given the nature of URLs and the fact that this class of classification does not yet exist, whether YOLO is a potential candidate for URL detection will be considered.

The second research question is to evaluate the accuracy of our YOLO detection model and the deep learning optical character recognition model, pytesseract.

**What is an image in Computer Science?**

An image (from Latin: imago), is an artefact that depicts visual perception, such as a photograph or other two-dimensional picture, that resembles a subject - usually a physical object - and thus provides a depiction of it [1]. An image is a visual representation of something [2]. According to The Merriam-Webster Dictionary, an image is.

*"a: a visual representation of something: such as*

*(1): a likeness of an object produced on a photographic material*

*(2): a picture produced on an electronic display (such as a television or computer screen)*

*b: the optical counterpart of an object produced by an optical device (such as a lens or mirror) or an electronic device" [3]*

While the orthodox idea of an image is that it usually only two dimensional and thus must appear on a screen, the technical visage of the word suggests that an image can also be three-dimensional. Thus, images may be two-dimensional, such as a photograph or screen display, or three-dimensional, such as statues or holograms. They may be captured by optical devices – such as cameras, mirrors, lenses, telescopes, microscopes, etc. and natural objects and phenomena, such as the human eye or water.

More interestingly perhaps, is that the term image lends itself to more than physical applications. The term also lends itself to mental and metaphysical uses. As such, an image can be interpreted to mean, a mental image - an impression created in the imagination of a person upon the suggestion of a particular concept, and a divine image- an image believed to be transmitted into the human psyche by a supernatural entity. An image may also be classified as being either fixed or volatile. A volatile image is one that exists only for a short period of time. This may reflect an object by a mirror, a projection of a camera obscura, or a scene displayed on a cathode ray tube. A fixed image, also called a hard copy, is one that has been recorded on a material object, such as paper or textile by photography or any other digital process [4].

The term ‘image’ in Computer Science is employed in a more technical scope. It is both restricted as well as specific in its application. Image processing for instance is a set of computational techniques for analyzing, enhancing, compressing, and reconstructing images. Its main components are importing, in which an image is captured through scanning or digital photography; analysis and manipulation of the image [5], In information technology, the term has several usages.

Firstly, it may be employed in its base form, in the way, an image is a picture that has been made or copied and stored in electronic form. An image can also be described in terms of vector graphics or raster graphics. An image stored in raster form is sometimes called a bitmap. Bitmap images are made from small parts called pixels. Vector images are made using coordinates and geometry. Images can be compressed to reduce file size [6].

Common image file formats online include:

1. **JPEG** (pronounced JAY-peg) is a graphic image file produced in compliance to the standard set by the Joint Photographic Experts Group, an ISO/IEC group of experts that develops and maintains standards for a suite of compression algorithms for computer image files. JPEGs usually have a .jpg file extension.
2. **GIF** (pronounced JIF by many, including its designer; pronounced GIF with a hard G by many others) stands for Graphics Interchange Format. The GIF employs the use of 2D raster data type and is encoded in binary. GIF files ordinarily have the .gif extension.GIF89a is an animated GIF image, formatted according to GIF Version 89a. One of the merits of this format is that it possesses the ability to create an animated image that can be played after transmitting to a viewer page that moves - for example, a twirling

icon or a banner with a hand that waves or letters that magically get larger. A GIF89a can also be specified for interlaced GIF presentation.

3. **PNG** (pronounced ping) is an acronym for Portable Network Graphics. It is a file format used for image compression and was designed to provide a number of enhancements over the GIF format. Like a GIF, a PNG file is compressed in lossless fashion, this means that all image information is restored when the file is decompressed during viewing. Files typically have a .png extension.

4. **SVG** is Scalable Vector Graphics, the description of an image as an application of XML. Any program such as a browser that recognizes XML can display the image using the information provided in the SVG format. Scalability means that the file can be viewed on a computer display of any size and resolution, whether the small screen of a smartphone or a large widescreen display in a PC. Files usually have .svg extension.

5. **TIFF** (Tag Image File Format) is a common format for exchanging raster graphics (bitmap) images between application programs, including those used for scanner images. A TIFF file can be identified as a file with a .tiff or ".tiff" file name suffix.

Furthermore, the term is also often used to mean a disk image, a disk image is a copy of the entire contents of a storage device, such as a hard drive or DVD. The disk image represents the content exactly as it is on the original storage device, including both data and structure information.

Lastly, another use of the term image is for a section of random-access memory (RAM) that has been copied to another memory or storage location [2].

**What is Object Detection?**

Object detection is a computer vision implementation that makes a system (an algorithm) about to estimate the location of objects in a digitized scene such as an image or video. Usually, a bounded box is wrapped around the detected object which helps humans locate the object quicker than unprocessed images. For this discourse, an object is the representation of a physical object (URL) in an image. In image processing, it is an identifiable portion of an image that can be interpreted as a single unit [7]. This creates a sharp contrast to the layman's idea that an image or an object are interchangeable.

Usually, an image may contain one or more objects, the discernibility of which is of up-most importance. For instance, in a single image the objects contained can range from a single unit to as many objects of as numbers, bordering on infinity.

Although "detection" could mean locating a hidden concealed object, detection may also mean the ability of an intelligence to signify the existence and identification of an object. The object in question does not have to be hidden. This later form is the form in which we based this thesis.

Interpreting the object localization for object detection can be done in various ways, including creating a bounding box around the object or marking every pixel in the image which contains the object (called segmentation) [8]. Thus, given an image or video stream, an object detection model should be able to identify which of a known set of objects might be present and provide information about their positions within the image [9].

Object Detection is widely employed in various computer vision tasks such as image annotation, activity recognition, face detection, face recognition, video object co-segmentation. It is

also used in tracking objects. For example, tracking a ball during a football match, tracking movement of a cricket bat, or tracking a person in a video are some of its various uses [10].

Fundamentally, two approaches to image detection exist, they are machine learning-based approaches and deep learning-based approaches [11]. In more traditional ML-based approaches, computer vision techniques are employed to analyze various features of an image, such as the color histogram or edges, to identify groups of pixels that may belong to an object. These features are then inputted into a regression model that predicts the location of the object along with its label [11].

Some Machine learning approaches are Viola–Jones object detection framework based on Haar features, Scale-invariant feature transform (SIFT) and Histogram of oriented gradients (HOG) features [10]. On the other hand, deep learning-based approaches employ convolutional neural networks (CNNs) to perform end-to-end, unsupervised object detection, in which features do not need to be defined and extracted separately [11]. Some Deep learning approaches are: Region Proposals (R-CNN,Fast R-CNN,Faster R-CNN, cascade R-CNN.), Single Shot MultiBox Detector (SSD), You Only Look Once (YOLO), Single-Shot Refinement Neural Network for Object Detection (RefineDet),Retina-Net and Deformable convolutional networks.

**What is Image Recognition?**

Object detection is often confused with image recognition (Figure 1). A picture of a dog receives the label “dog”. A picture of two dogs, still receives the label “dog”. Object detection, on the other hand, draws a box around each dog and labels the box “dog”. The model predicts where each object is and what label should be applied. In that way, object detection provides more information about an image than recognition [11].

Recognition in this context is the ability of an intelligent system to identify an object based on certain similarities that it shares with another object that the intelligence has previously encountered. Recognition may be based on inference or relation, that is, a situation whereby an intelligence is able to recognize an object because it recognizes similarities in form and properties. Recognition may also occur because the Artificial intelligence has encountered the exact specimen at a previous instance.

In human beings' recognition is a cognitive process that happens seamlessly and almost instantly without any hitch. The human brain is capable of learning and adapting information with minimal effort such that even humans that are still in the developmental stages of their existence can recognize objects and patterns easily. Humans can recognize a multitude of objects in images with little effort, even though the image of the objects may vary somewhat in different viewpoints, in many different sizes and scales or even when they are translated or rotated. Objects can even be recognized when they are partially obstructed from view [12]. Artificial intelligence, however, does not innately possess this cognitive ability. For Artificial Intelligence to acquire this level of skill they must acquire training. This training is usually acquired by 'teaching' the A.I. using coding, datasets, and databases. This task is still a challenge for computer vision systems given these A.I. systems need to be trained for each class of object it is meant to recognize.

The object recognition problem can be defined as a labelling problem based on models of known objects. Formally, given an image containing one or more objects of interest (and background) and a set of labels corresponding to a set of models known to the system, the system should be able to accurately assign correct labels to regions, or a set of regions, in the image. The

object recognition problem is closely tied to the segmentation problem: without at least a partial recognition of objects, segmentation cannot be done, and without segmentation, object recognition is not possible [13].

Object recognition is an extremely difficult computational problem. The core problem is that each object in the world can cast an infinite number of different 2-D images onto the retina as the object's position, pose, lighting, and background vary relative to the viewer. Yet the brain solves this problem effortlessly. Progress in understanding the brain's solution to object recognition requires the construction of artificial recognition systems that ultimately aim to emulate our own visual abilities, often with biological inspiration [14].

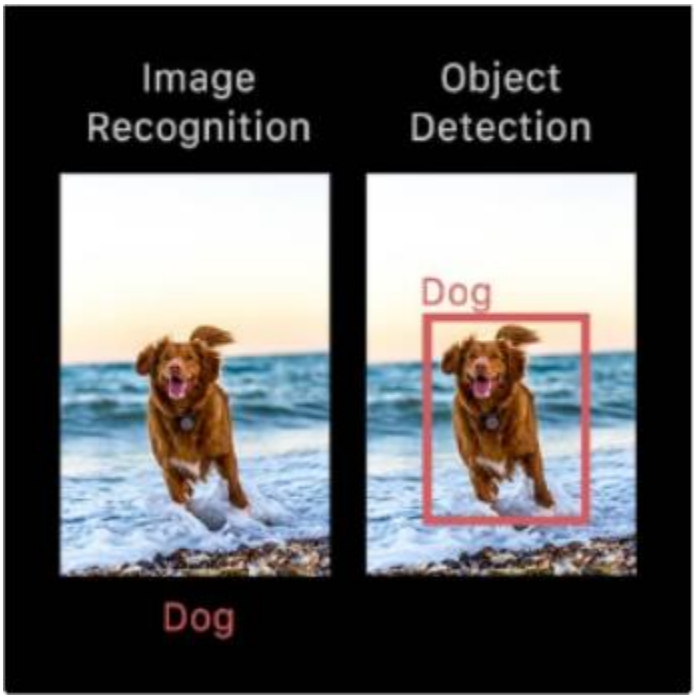


**Figure 1.** Image Recognition and Object Detection difference [36].

# Chapter II: Background

## Overview

In this Chapter, we will review the basic concepts of general neural networks, explain a subclass of neural network, Convolutional Neural Network (CNN), Batch Normalization, Max Pooling, Activation Function, and Rectified Linear Unit (RELU). We will also describe a specific CNN based object detection model, YOLO, which is the most accurate real-time object detector [35].

The neural networks described in this thesis are the feed-forward neural networks, in which information is transmitted in a feedforward manner.

## Image Processing

Image processing is a technique that is used to detect objects/patters and perform some operations on an image usually with the aim to retrieve value information from the image or enhance it. The processing of digital images can be divided into several classes: image enhancement, image restoration, image analysis, and image compression [36].

**Image Enhancement:** This is a technique used to adjust digital images for better suitable displays or analysis. An image can be enhanced using median filtering, linear contrast adjustment, histogram equalization, etc.

**Figure 2.** Enhancing grayscale images with histogram equalization [37].

**Image Restoration:** Restoration is usually performed on burry or noisy images. Images can be blurred due to many factors like relative motion the camera and the object. Noises are often caused by environmental conditions such as rain, snow, etc. or even by thermal signals [36].

**Image Analysis:** Images that contain useful information can be used to outline objects and describe them. Some examples of image analysis are edge extraction, image segmentation, and texture and motion analysis [36].

**Image Compression:** When an image is compressed, the image byte is minimized without degrading the quality of the image. The advantage of compression is more images can be stored on a disk or memory after compression [36].

**Artificial intelligence and learning**

The term artificial intelligence was first used in 1956, at a computer science conference in Dartmouth. It described an attempt to digitally model how the human mind worked and based on this knowledge create more advanced computers. Even though some truly groundbreaking progress has been made, today, artificial intelligence is ubiquitous, but computers are still far from modelling human intelligence to perfection. The goal of the "Strong" AI is to become artificial persons: machines that have all the mental powers we have, including phenomenal consciousness. "Weak" AI, on the other hand, seeks to build information-processing machines that appear to have the full mental repertoire of human persons (Searle 1997). Weak, or narrow AI on the other hand is created to be quite adept at carrying out the purpose that it was defined for, but it will not pass for human in any other field outside of its defined capacities. An example of this is Deep Blue, the first computer to defeat a human in chess. It was able to defeat the famous chess master Garry Kasparov in 1997.

To facilitate this growth and development in the field, machines must learn how to behave and this process if facilitated through a system known as machine learning.

**Machine Learning**

Machine learning is an area within Artificial Intelligence (AI), that focuses on decision making and predictions. The primary aim is to allow the computer to be further developed without any human intervention, it will be trained based on observations and data [36]. Machine learning algorithms are often categorized as supervised or unsupervised, where supervised algorithms apply what has been learned in the past to new data using labeled examples. This type of problem can be either predicting a continuous quantity (Regression) or discrete class labels (Classification). Unsupervised learning includes grouping a set of uncategorized data by finding structures or patterns (Clustering) or finding relationships between variables in big data (Association).

Machine learning is a subset of the larger field of Artificial Intelligence that "focuses on teaching computers how to learn without the need to be programmed for specific tasks [15]," "In fact, the key idea behind ML (sic) is that it is possible to create algorithms that learn from and make predictions on data [15]." To educate a machine. Three important components must be present, namely, a dataset, features, and an algorithm. These components will be discussed serially in subsequent paragraphs.

A dataset is a pool of samples compiled with the intention of educating a machine on how to go about a task. These pools can sometimes be numbers, images, texts, or any other kind of data. It can sometimes be quite strenuous and expensive to compile a data set depending on how

specialized the task is expected to be. For instance, a data set of exclusive Rolex watches would be easier to compile than a data set of all types of cars in existence.

Features are important pieces of data that act as the key to the solution of the task. They show the machine what to look for in an image. The way features are defined vary based on what the machine is expected to look out for. For instance, to predict the price of an apartment, it would prove to be too difficult to prove by way of linear regression the cost of the property based on the combination of its length and width However, it may be easier to predict a price based on the correlation between the price and location of a building. Therefore, there is no gainsaying the fact that the accuracy of the features provided affects the eventual performance of the machine. Which is what occurs when the machine is trained with labeled data which contain the "right solutions", and a validation set. During the learning process, the program is expected to get the "right" solution. And then, the validation set is used to tune hyperparameters to avoid overfitting. However, in unsupervised learning, features are learned with unlabeled input data. In his case the machine is not given any features to familiarize itself with, it learns to notice the patterns by itself.

An algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of problems or to perform a computation [16]. Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, and other tasks [17]. The instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output".

**Machine Learning Evaluation**

Evaluating a machine learning model requires that our dataset be split appropriately. It is typically to have 70% training dataset and 30% for testing. However, it is also advisable to have about 20% validation dataset to make it possible to evaluate the model while still building and tuning the model. This helps to avoid overfitting and discourages using the training dataset for testing. After shuffling the data, a split of 60% training dataset, 20% validation dataset, and 20% testing dataset is advisable.

Speed, precision, recall, and accuracy are important metrics of a machine learning algorithm to be evaluated. The accuracy of an object detection model is calculated using the Mean Average Precision (mAP) [36].

Accuracy is the ratio of correct predictions to the total number of input samples.

$$accuracy = \frac{\text{correct predictions}}{\text{all predictions}}$$

Precision is the ratio of relevant examples (true positives) among all the examples which are predicted to belong in a certain class [38].

$$precision = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Recall is the ratio of examples which were predicted to belong to a class with respect to all the examples that truly belong in that class [38].

$$recall = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$$

True Positive (TP) [36]: The prediction that was made is true, and the actual output is positive.

False Positive (FP) [36]: The prediction that was made is false, and the actual output is positive.

False Negatives (FN) [36]: The prediction that was made is false, and the actual output is negative.

**Deep Learning**

Apart from the machine learning earlier discussed there are new evolved specialized forms of learning which have grown apart and form a subset of machine learning known as Deep Learning. Deep learning is a more effective branch of machine learning based off the perceived structure of the human brain, Deep learning algorithms use complex multi-layered neural networks, where the level of abstraction increases gradually by non-linear transformations of input data.

In a neural network, information moves from one layer to another over a set of over-connecting channels. These channels are called weighted channels due to the amount of value attached to them.

All neurons possess a unique number known as bias. This bias combined with the weighted sum of inputs reaching the neuron is then applied to the activation function. The result of the function determines if the neuron gets activated. Every activated neuron passes on information to the following layers. This continues up to the second last layer. The output layer in an artificial neural network is the last layer that produces outputs for the program [18]. The field of deep learning is very useful in fields such as object recognition as well as voice recognition.

In 2013, R-CNN was the first algorithm to apply deep learning to Object detection. It outperformed the previous algorithms by up to 30% on the Visual Object Classes Challenge

(VOC2012). This affirmed its place as the frontrunner of the object detection industry. It beats the previous ones by more than 30% on the VOC2012 and was therefore a huge improvement in the fields of Object detection.

Object Detection involves two difficulties: locating objects and suppling the appropriate classifications.

**Neural Network**

Neural network is a computer system modeled on the human brain and the nervous system. Wikipedia defines it as a network of circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes. Thus, a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems.

**Neuron**

Neuron, in biology, is the basic working unit of the brain, an electrically excitable cell that communicates with other cells via specialized connections called synapses. A neuron is the atomic element of a neural network. A perceptron is the mathematical model of a biological neuron where electric signals are represented as numerical values.

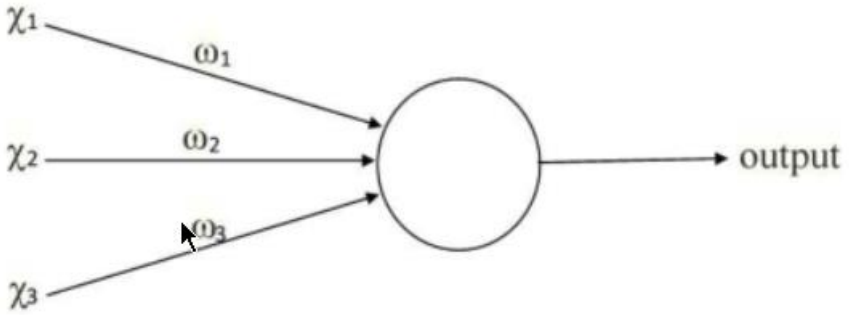


**Figure 3.** Structure of Perceptron [35].

$$output = \begin{cases} 0 & if \quad \sum \omega_j \chi_j > threshold \\ 1 & if \quad \sum \omega_j \chi_j \leq threshold \end{cases}$$

where $\omega_j$ is the weight for input $\chi_j$.

**Figure 4.** Output of a Perceptron [35].

A perceptron has many inputs and one output. It calculates the weighted sum of the input values to represent the total strength of the input signals and applying a step function as the activation function on the sum to output the result. The output generated is fed into other perceptron and the final output is bounded between 0 and 1. A multi-layered perceptron is called neural networks.


**Figure 5.** Activation functions. (1) the left curve is a sigmoid function curve. (2) the right curve is a tanh function curve [35].

**Structure of Neural Networks**

Neural networks typically consist of multiple layers and each layer has many neurons as shown in Figure 6. The first layer is known as the input layer where each circle represents an input neuron. The rightmost layer is the output layer and contains output neuron(s), and all the

middle layers are hidden layers. General neural networks have fully connected neurons and neurons between adjacent layers are connected for passing information. For example, if two adjacent layers have $m$ and $n$ neurons respectively, the total number of connections will be $m \times n$ [35].

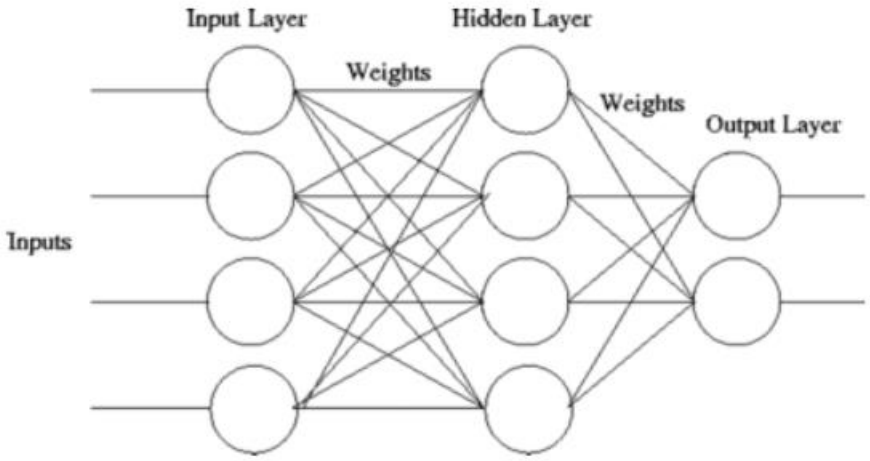


**Figure 6.** A sample fully connected neural network with only one hidden layer [35].

**Cost Function and Training**

Training involves fine-tuning a bad performing neural network (due to randomly initialized weights) into a network with high accuracy. Training is equivalent to minimizing loss function. Neural networks learn from the errors between the predictions and ground truths by updating its parameters (weights) [35]. A cost function is a measure of error between the predicted value of a model and what the actual value is. The error range continues to shrink during the training process until the value stays constant which indicates the training process is complete. Figure 7 shows a visual representation of a training process. The bowl-shaped error surface indicates the error sliding from the edge to the bottom (where the minimum error is) [35].

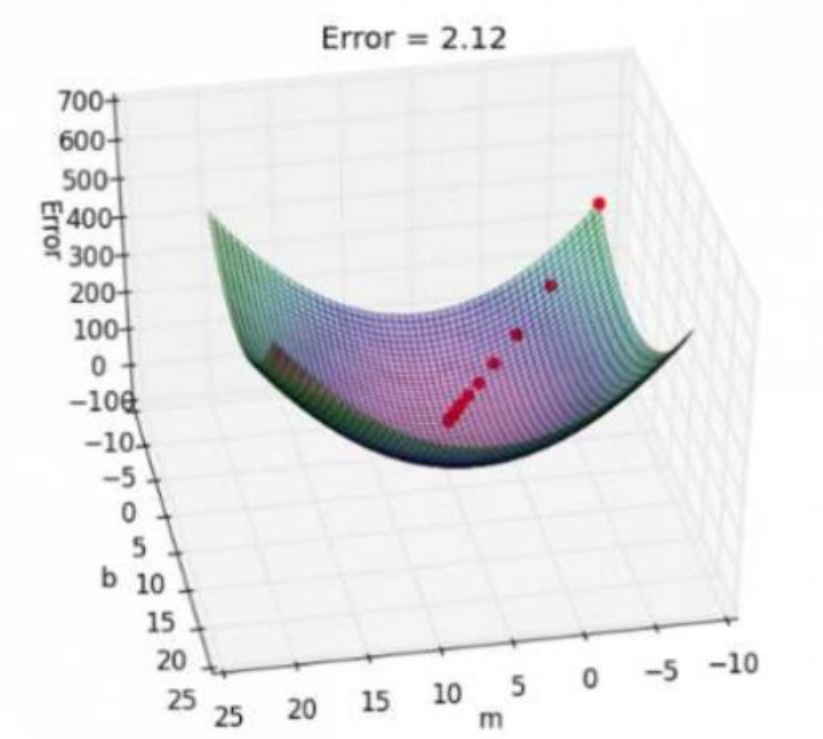


**Figure 7.** Visualization of a Training Process [35].

**Convolutional Neural Network (CNN)**

Convolutional Neural Network (CNN) first surfaced around 1998 [35] but was proved to be an efficient tool for image classification in 2012 at the Large Scale Visual Recognition Challenge. CNN is a deep learning algorithm which can take an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods, filters are hand-engineered, with enough training, CNN can learn these filters/characteristics [39].

**Pooling**

After each convoluted operation, pooling operation follows to further simplify the information. The pooling process simply compresses previous feature maps into condensed feature maps. The two most common pooling methods are average-pooling and max-pooling. Average-pooling means to output the average value among the pooling filed, and max-pooling is to select the maximum value. Figure 8 shows a max pooling operation with a $2 \times 2$ pooling filter. The size of the feature map is reduced from $24 \times 24$ to $12 \times 12$. We will be using max-pooling for our experiment [35].

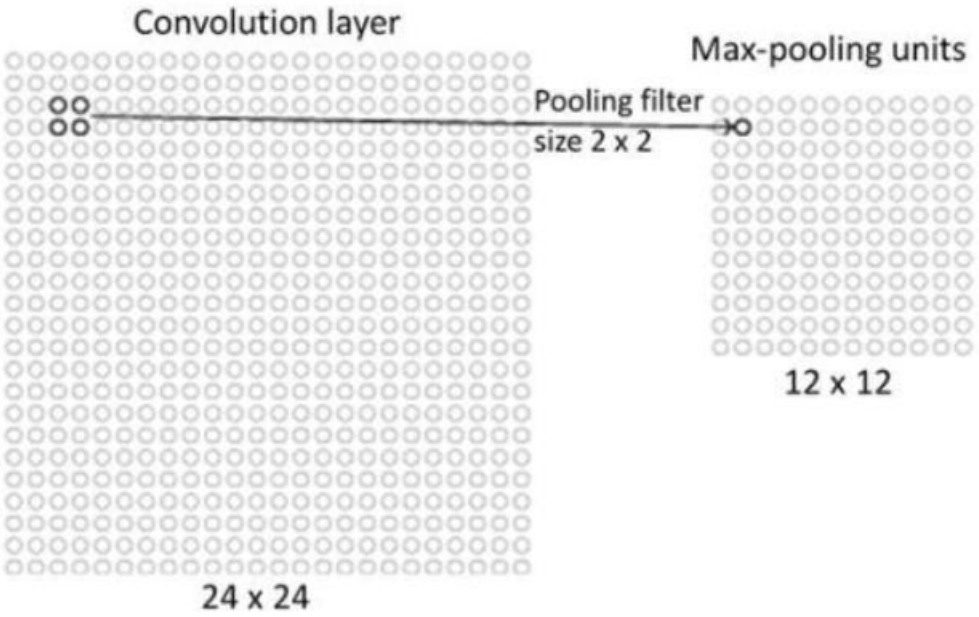


**Figure 8.** Max-pooling. Pooling from 24 × 24 to 12 × 12 [35].

**General Object Detection Model**

Figure 9 is an object detection structure which has a region proposal component followed by a CNN classifier. Researchers use region proposal methods to produce a bunch of candidate regions, each of which may contain one kind of object. Each region is then passed through the

CNN classification algorithm. The model is to convert a multiple object detection problem into a single object classification problem. However, these region proposal methods are much slower than classification part, which becomes the bottleneck for the whole system. The drawback of this structure is that we cannot tradeoff accuracy for detection speed in time critical applications [35].

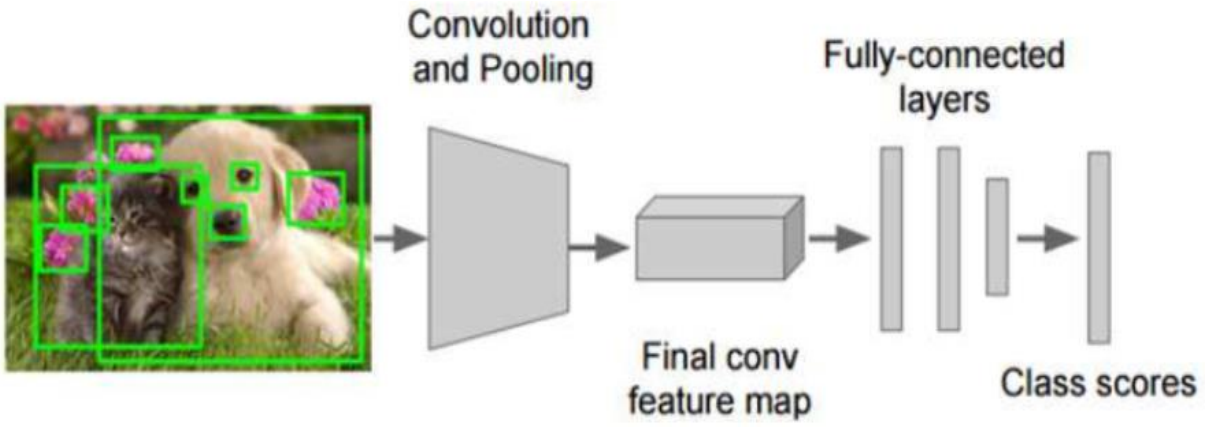


**Figure 9.** Structure of a CNN Based Object Detection Model [35].

**Unified Detection Model – YOLO**

The YOLO model was first brought into existence by Joseph Redmon in his paper "You only look once, Unified, Real-time object detection". The mechanism for the algorithm employs the use of a single neural network that takes a photograph as an Input and attempts to predict bounding boxes and class labels for each bounding box directly. Although this offered less predictive accuracy, which was mostly due to more localization errors, it boasted speeds of up to 45 frames per second and up to 155 frames person on speed optimized versions of the model [19].

*"Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second …" [21]*

To begin with the model operates by splitting the inputted image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within it. Each grid cell predicts a bounding box involving the x, y coordinate and the width and height and a metric of valuation of quality known as a confidence score. A class prediction is also based on each cell.

To supply more emphasis an instance will be provided. For example, an image may be divided into a $7 \times 7$ grid and each cell in the grid may predict 2 bounding boxes, resulting in 94 proposed bounding box predictions. The class probabilities map and the bounding boxes with confidences are then combined into a final set of bounding boxes and class labels.

The YOLO was not without shortcomings, the algorithm had a number of limitations because of the number of grids that it could run on as well as some other issues which will be addressed subsequently. Firstly, the model uses a $7 \times 7$ grid and since each grid can only identify an object, the model restricts the maximum number of objects detectable to 49. Secondly, the model suffers from what is known as a close detection model, since each grid is only capable of detecting one object, if a grid cell contains more than one object it will be unable to detect it. Thirdly, a problem might arise because the location of an object might be more than a grid, thus, there exists a possibility that the model might detect the object more than once [20]. Due to the aforementioned problems encountered when running YOLO, it was fairly obvious that localization error and other problems of the system needed to be addressed. As a result of that, YOLOv2 was created as an improvement to deal with the issues and questions posed by its predecessor. Therefore, localization errors as well as errors of real were significantly addressed in the new

version. The model was updated by Joseph Redmon and Ali Farhadi to further revamp model performance in their 2016 paper named “YOLO9000: Better, Faster, Stronger [19]”.

**Structure of YOLO**

YOLO is implemented as a convolution neural network and has been evaluated on the PASCAL VOC detection dataset. It consists of a total of 24 convolutional layers followed by 2 fully connected layers. The layers are separated by their functionality in the following manner:

1. First 20 convolutional layers followed by an average pooling layer and a fully connected layer is pre-trained on the ImageNet 1000-class classification dataset.
2. The pretraining for classification is performed on dataset with resolution $224 \times 224$.
3. The layers comprise of $1 \times 1$ reduction layers and $3 \times 3$ convolutional layers.
4. Last 4 convolutional layers followed by 2 fully connected layers are added to train the network for object detection.
5. Object detection requires more granular detail hence the resolution of the dataset is bumped to $448 \times 448$.
6. The final layer predicts the class probabilities and bounding boxes.

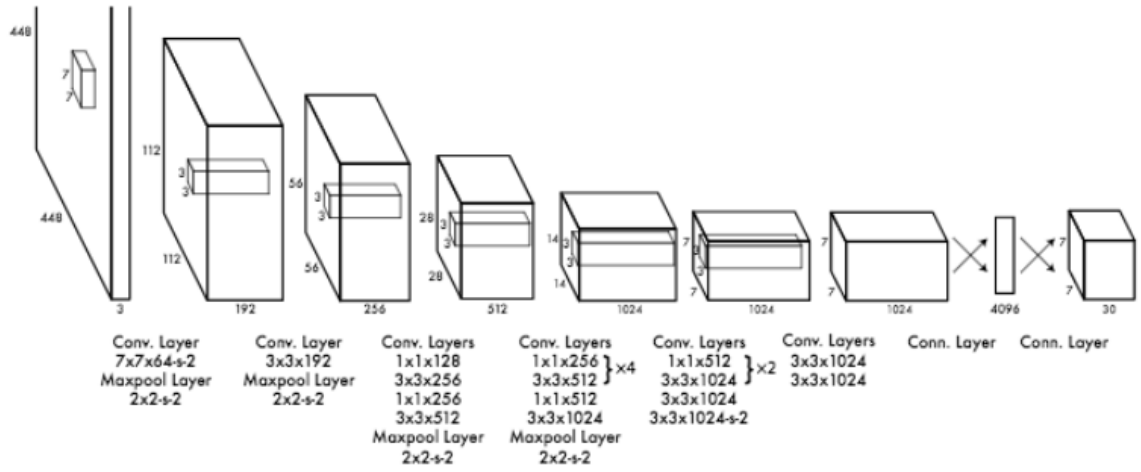


**Figure 10.** YOLO Structure [35].

The final layer uses a linear activation whereas the other convolutional layers use leaky ReLU activation.

The input is 448 × 448 image, and the output is the class prediction of the object enclosed in the bounding box.

**Network Structure**


**Figure 11.** YOLO Network Architecture [42].

The whole system can be divided into two major components: Feature Extractor and Detector; both are multi-scale. When a new image comes in, it goes through the feature extractor

first so that we can obtain feature embeddings at three (or more) different scales. Then, these features are feed into three (or more) branches of the detector to get bounding boxes and class information [42].

**Intersection over Union (IoU)**

YOLOs default metric for measuring overlap between two bounding boxes or masks is Intersection over union ($IoU$). Any algorithm that provides predicted bounding boxes as output can be evaluated using $IoU$ [20]. If the prediction is completely correct, $IoU = 1$. The lower the $IoU$, the worse the prediction result.

To apply Intersection over union, certain parameters must be met to evaluate an (arbitrary) object detector, they are: The ground-truth bounding boxes these are the hand labeled bounding boxes from the testing set that specify where in the image our object is and the predicted bounding boxes from our model. If these two sets of bounding boxes are present it is possible to apply Intersection over union. Thus, computing Intersection over Union can therefore be calculated as; The area of overlap divided by the area of union [22].

Plainly put, the intersection over union is a ratio of the similarity of the ground truth bounding box and the predicted bounding box, thus predicting a rough estimate on how much an Artificial intelligence can rely on the predictions made by the algorithm. This is an improvement over binary models which label predictions as either correct or incorrect. Also due to the variance in parameters between the model and the object it is quite unrealistic to have a 100% match between the $(x, y)$ coordinates a predicted bounding box and the $(x, y)$ coordinates of the ground truth box [22]. Thus, this equation ensures that boxes with a larger area of overlap get higher scores than those with lesser areas thus cementing Intersection over union as excellent metric for

evaluating custom object detectors. Generally, any score greater than 0.8 is a good score [23]. This application of intersection over union is used during the testing phase at the completion of the training [24].

Another interesting application of this model occurs where there is more than one bounding box for the same object. the metric helps to eliminate bounding boxes with lesser scores. How this is done is that if there are two bounding boxes with very high confidence scores, then the chances are that the boxes are detecting the same object therefore the box with the higher confidence rating is selected. However, where confidence rating of the two boxes is low, it is likely that they are predicting separate objects of the same class, such as two different cars in the same picture. This application is used after the model has completed training and is being deployed [24].

As earlier mentioned, Intersection over union is a good metric for evaluating the quality of a task. Before further elucidation, it is pertinent to define some terminologies first, these terminologies are true positive, true negatives, false negatives, and false positives. Simply put, a true positive is any correctly drawn annotation with a value greater than 0.5, a true negative is when an F1 refuses to draw any annotation because there simply is not one to be drawn. There is no value here because no annotation is drawn, thus there is no way to calculate true negatives. A false negative occurs where there are missing annotations while a false positive occurs where there are incorrectly drawn annotations that have an $IoU$ score of less than 0.5. An equation known as accuracy is usually used to measure the performance of a task because it is an incredibly straight forward measurement as well as for its simplicity. it is simply a ratio of correctly drawn annotations to the total expected annotations (ground truth). Thus, it is calculated as being

equal to the sum of True positive and True negative divided by the sum of True positive, False positive, True negative and False negative.

**Direct Location Prediction**

In the early versions of the YOLO model, there were no constraints on the location prediction. The predicted bounding box was not tethered therefore it could occur far from the original grid location. This anomaly resulted in a very unstable model [25]. The bulk of the instability resulted from predicting the $(x, y)$ locations for the box. In region proposal networks the network predicts values $t_x$ and $t_y$ and the $(x, y)$ center coordinates are calculated as:

$$x = (t_x * w_a) - x_a$$

$$y = (t_y * h_a) - y_a$$

For example, a prediction of $t_x = 1$ would shift the box to the right by the width of the anchor box, a prediction of $t_x = -1$ would shift it to the left by the same amount [25]. This formulation did not exist within any boundaries and as such, any anchor box could end up at any point in the image regardless of what location predicted the box. with random initialization it took the model an obscene amount of time to stabilize to predict sensible offsets [25]. The subsequent versions of YOLO diverged from this approach and devised a means to properly tackle the situation. YOLOv2 bounds the location using logistic activation.

Sigma $(\sigma)$, which ensures that the value remains between 1 and 0 [26]. Given the anchor box of size $(p_w, p_h)$ at the grid cell with its top left corner at $(c_x, c_y)$, the model predicts the offset and the scale, $(t_x, t_y, t_w, t_h)$ and the corresponding predicted bounding box has center $(b_x, b_y)$ and size $(b_w, b_h)$. The confidence score is the sigmoid $(\sigma)$ of another output $t_o$. Since

the location prediction is constrained, the parameterization is easier to learn, making the network more stable. The employment of dimension clusters along with directly predicting the bounding box's center location improves YOLO by almost 5% over the version with anchor boxes n, m.

**Bounding Box Prediction**

A bounding box is a relatively closed space within which points, objects, or a group of objects may be contained. For YOLO, the cell in which the center of an object resides, is the cell responsible for detecting that object. Each cell will predict B bounding boxes and a confidence score for each box. The confidence score will be from '0.0' to '1.0', with '0.0' being the lowest confidence level and '1.0' being the highest; if no object exists in that cell, the confidence scores should be '0.0', and if the model is completely certain of its prediction, the score should be '1.0'. These confidence levels capture the model's certainty that there exists an object in that cell and that the bounding box is accurate. Each of these bounding boxes is made up of 5 numbers: the x position, the y position, the width, the height, and the confidence. The coordinates '(x, y)' represent the location of the center of the predicted bounding box, and the width and height are fractions relative to the entire image size. The confidence represents the Intersection over Union (IoU) between the predicted bounding box and the actual bounding box, referred to as the ground truth box [27]. In general, there are five types of bounding boxes, i.e., a surrounding sphere (SS), an axis-aligned bounding box (AABB), an oriented bounding box (OBB), a fixed-direction hull (FDH), and a convex hull (CH) [29]. The AABB refers to a box whose axis is parallel to the coordinate axis. It is the rectangle formed by selecting the maximum and minimum horizontal and

vertical coordinates in each vertex of the two-dimensional shape and is one of the most commonly used bounding box types [29]. Bounding box functions to mark the object area that has been detected [28].

In digital image processing, the bounding box is the coordinates of a rectangle wishing which an object may be contained when it is placed over a page, a canvas, a screen, or any other similar bi-dimensional background [30]. In the field of object detection, a bounding box is usually used to describe an object location. The bounding box is a rectangular box that can be determined by the $x$ and $y$ axis coordinates in the upper-left corner and the $x$ and $y$ axis coordinates in the lower-right corner of the rectangle [31]. The first version of YOLO directly predicted all four values which describes a bounding box. the x and y coordinates of each bounding box are defined relative to the top left corner of each grid cell and normalized by the cell dimensions such that the coordinate values are bounded between 0 and 1. However in YOLOv2 there was a shift in paradigm and the algorithm employed dimensional clusters in place of anchor boxes, 4 coordinates are predicted for each bounding box, $t_x, t_y, t_w, t_h$, If the cell is offset from the top left corner of the image by $(c_x, c_y)$ and the bounding box prior has width and height $p_w, p_h$, then the predictions correspond to [32]:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

**Figure 12.** Bounding Box prediction formula [32].

During training, the sum of squared error loss is used Assuming that the ground truth for some coordinate prediction is $\hat{t}_*$ our gradient is the ground truth value (computed from the ground truth box) minus our prediction: $\hat{t}_* - t_*$. This ground truth value can be easily computed by inverting the equations above. YOLOv3 predicts an objectness score for each bounding box using logistic regression. This should be 1 if the bounding box prior overlaps a ground truth object by more than any other bounding box prior is not the best but does overlap a ground truth object by more than some threshold the prediction is ignored. We use the threshold of .5. usually, a system only assigns one bounding box prior for each ground truth object. If a bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predictions, only objectness [33].

The concept of a bounding box prior was introduced in YOLOv2. Previously, the model was expected to provide unique bounding box descriptors for each new image, a collection of bounding boxes is defined with varying aspect ratios which embed some prior information about the shape of objects we are expecting to detect. Redmon offers an approach towards discovering the best aspect ratios by doing k-means clustering (with a custom distance metric) on all of the bounding boxes in the training dataset [33].

Thus, instead of predicting the bounding box dimension directly, the task is reformulated to simply predict the offset from the bounding box prior in order to fine-tune the predicted bounding box dimensions. The result of which is that it makes the prediction task easier to learn.

**Objectness (and assigning labeled objects to a bounding box)**

"objectness" score $p_{obj}$ is trained to approximate the intersection over union between the predicted box and the ground truth label. When the loss during training is calculated, Objects are

matched to whichever bounding box prediction on the same grid cell produced the highest $IoU$ score. For unmatched boxes, the only descriptor which will be included in the function is $p_{obj}$.

Upon the introduction of additional bounding box priors in YOLOv2, it was possible to assign objects to whichever anchor box on the same grid cell has the highest IoU score with the labeled object.

YOLO (version 3) redefined the "objectness" target score $p_{obj}$ to be 1 for the bounding boxes with highest $IoU$ score for each given target, and the score 0 for all remaining boxes. However bounding boxes which have a high $IoU$ score above a defined threshold but not the highest score when calculating the loss will not be included. This simply means that it does not produce the most appropriate prediction because it is not the best possible prediction [33].

**Class Labels**

YOLO (version 3) uses sigmoid activations for multi-label classification, noting that SoftMax (from the previous versions) is not necessary for good performance. This choice will depend on your dataset and whether or not your labels overlap (e.g., "golden retriever" and "dog").

**Output Layer**

YOLO (version 3) has 3 output layers. These output layers predict box coordinates at 3 different scales. The output prediction is of the form $width \times height \times filters$.

YOLO (version 3) replaced the skip connection splitting for a more standard feature pyramid network output structure. With this method, there is an alternate between outputting a prediction and up sampling the feature maps (with skip connections). This allows for predictions

that can take advantage of finer-grained information from earlier in the network, which helps for detecting small objects in the image [33].

**Pytesseract**

Tesseract is an open-source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. Tesseract does not have a built-in graphics user interface (GUI) [43]. Pytesseract is the tesseract framework for python language.

Legacy Tesseract 3.x was dependent on the multi-stage process where we can differentiate steps into word finding, line finding, and character classification. Word finding was done by organizing text lines into blobs, and the lines and regions are analyzed for fixed pitch or proportional text. Text lines are broken into words differently according to the kind of character spacing. Recognition then proceeds as a two-pass process. In the first pass, an attempt is made to recognize each word in turn. Each word that is satisfactory is passed to an adaptive classifier as training data. The adaptive classifier then gets a chance to, more accurately, recognize text lower down the page [43].

Modernization of the Tesseract tool was an effort on code cleaning and adding a new Long Short-Term Memory (LSTM) model. The input image is processed in boxes (rectangle) line by line feeding into the LSTM model and giving output.

After adding a new training tool and training the model with a lot of data and fonts, Tesseract achieves better performance. Still, not good enough to work on handwritten text and weird fonts. It is possible to fine-tune or retrain top layers for experimentation [43].

**Preprocessing for Tesseract**

Tesseract does various image processing operations internally (using the Leptonica library) before doing the actual OCR. It generally does a very good job of this, but there will inevitably be cases where it is not good enough, which can result in a significant reduction in accuracy. Pre-processing includes but are not limited to inverting images, rescaling, binarization, noise removal, dilation, and erosion, rotation / deskewing, border removal, etc. [44].

To avoid all the ways your tesseract output can drop, you need to make sure the image is appropriately pre-processed. Given that our data is custom generated, we typically had a clear and sharp images devoid of noise. As a result, we did not require pre-processing except for rescaling of the image to our model's specification.

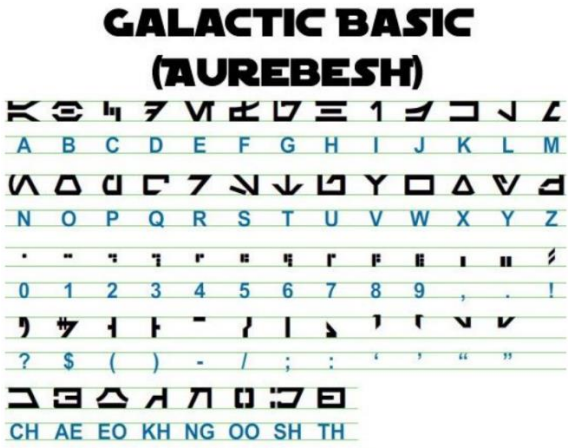


**Figure 13.** Image before pre-processing [43].

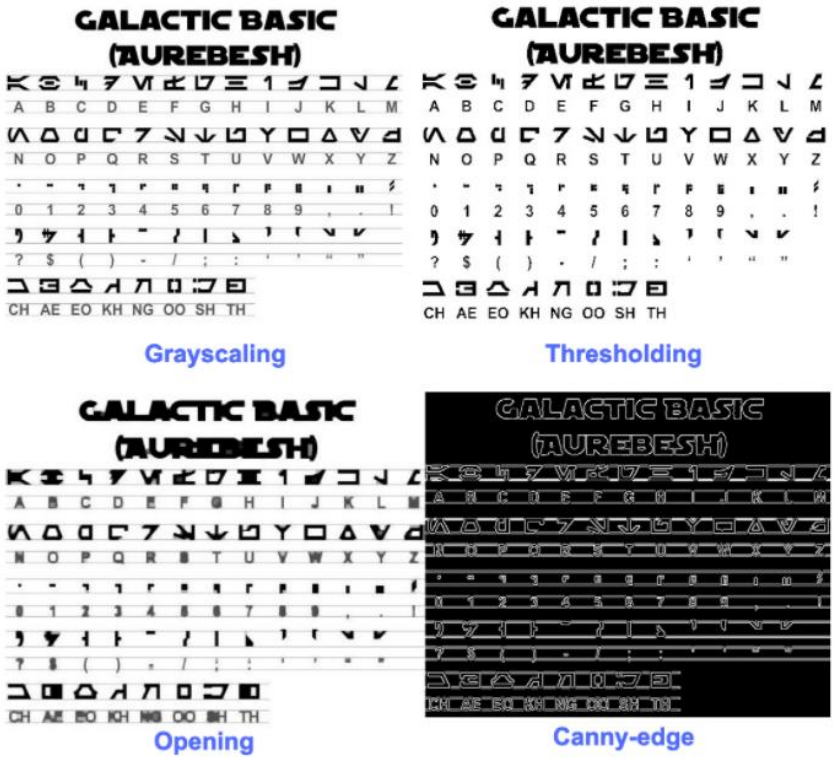


**Figure 14.** Image after pre-processing [43].

**Regular Expressions**

A regular expression (regex or regexp for short) is a special text string for describing a search pattern. You can think of regular expressions as wildcards on steroids. You are probably familiar with wildcard notations such as *.txt to find all text files in a file manager. The regex equivalent is «.∗\.$txt$» [45]. Regular expressions offer a limited but powerful metalanguage to describe all kinds of formats, protocols, and other small textual languages. Regular expressions arose in the context of formal language theory, and a primary use has been as part of scanners in compilers. However, nowadays their applications extend far beyond those areas. For example, regular expressions have been used in editors for structured text modification [46]. They are used

in network protocol analysis and for specifying events in a distributed system. Regular expressions are used for virus detection using signature scanning, in mining the web, and as alternatives types for XML data [46]. Moreover, there are many uses of regular expressions outside of computer science, for example, in sociology (for characterizing events that led to placing of a child in foster care) or biology (for finding DNA sequences) [46]. In addition to specific applications, many generic types of information, such as phone numbers or dates, are often presented and exchanged in specific formats that are described using regular expressions [46].

Regular expressions, though powerful, are often problematic due to its complexity (obscure, and hard to understand and reuse), errors (subtle and hard to detect faults), and version proliferation (making it hard to select the right one for a specific task). Some complex regular expression could exceed 100 characters, while some can go as high as over 4000 characters. This makes it difficult to scale regular expressions.

# Chapter III: Experiment

## Overview

In real-life application, speed, accuracy, and resources tradeoffs must be made. The internet is a great resource for different classes of trained data ready for use. Unfortunately, there is not one class ready made for recognition of URLs in an image. Most tests and trainings are done with datasets, and their results are measured in mean Average Precision (mAP) at Intersection over Union (IoU) threshold. IoU measures the overlap between two regions.

In this Chapter, we will highlight the custom generation of our training data from our word bank of 56000 (Figure 15) words.

## Data Preparation

A dataset is a collection of data. Image datasets are used to train and benchmark object detection algorithms [40]. For the algorithm to be trained, we need to create images with URLs in them and store the coordinates of the URLs. This would be used by the training algorithm to know what is right (a URL) and use that to learn what is wrong (not a URL).

Python was our choice programing language and because YOLO is just a technique for detection using convolutional neural network, we made use of pythons' keras library as well.

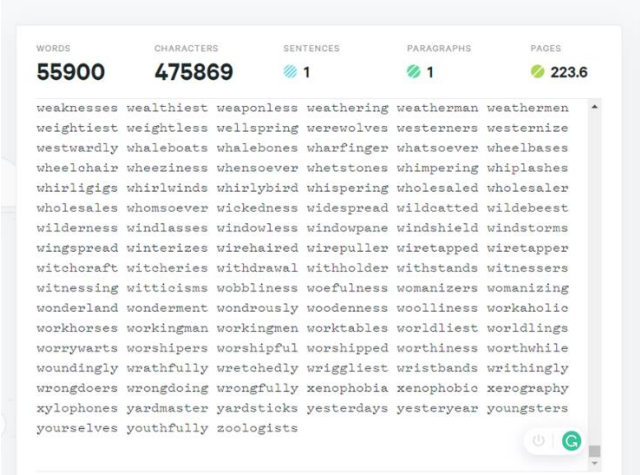


**Figure 15.** Word bank used to generate our training image data.

Figure 16 shows a sample image generated from our word bank. This process is known as tagging, and ours is automated. A total of sixty thousand images were generated for this experiment, each having varying number of URL(s) in them and some having no URL at all, giving the algorithm a large enough sample size. A record of the x and y coordinate of any URL generated is stored and is later passed into our detection algorithm.

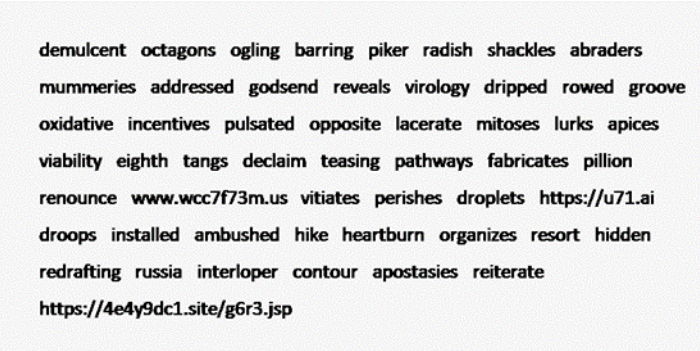


**Figure 16.** Sample generated image using our custom-built tagging tool.

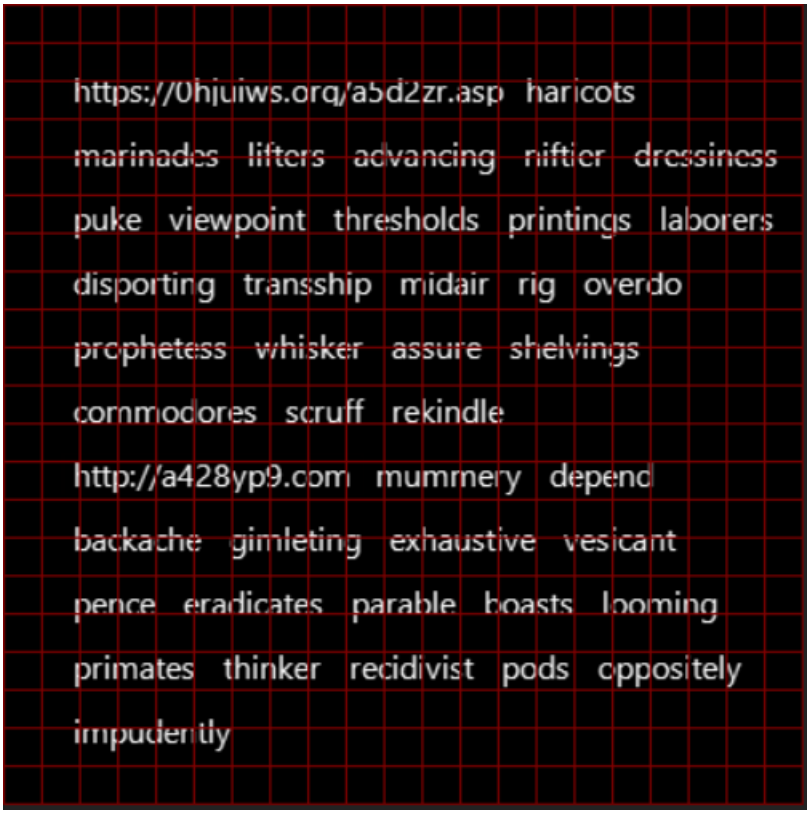


**Figure 17.** Sample grid generated by YOLO for our detection algorithm.

**Training**

Training, according to Longman Dictionary [34], is the process of teaching or being taught the skills for a particular job or activity. In this context, it refers to the process of teaching an algorithm towards a specific task for which it will be used. These algorithms also learn from experience without being explicitly programmed. In our experiment, we labelled ten thousand images, each with varying number of URLs and some with no URL at all. We trained our algorithm using fifty thousand images. These images have varying font size. Our image is sectioned into $21 \times 21$ grid cells (Figure 17), each is responsible for predicting K bounding boxes. The grid cell was selected because we would be working with only text. An object is considered to lie

in a specific cell only if the center co-ordinates of the anchor box lie in that cell. Due to this property, the center co-ordinates are always calculated relative to the cell, whereas the height and width are calculated relative to the whole image size. Using the Equation below, YOLO determines the probability that a cell contains a certain class. The class with the maximum probability is chosen and assigned to that grid cell. This is repeated for all grid cells in the image. The probability that there is an object of certain class 'c' is:

$$score_{c,i} = p_c \times c_i$$

Setting 0.3 filter size and 0.7 $IoU$ threshold, we applied non-max suppression to select the bounding box with the highest $IoU$ threshold. Batch normalization is then applied to add noise to the inputs of every later, discouraging overfitting preventing our model from producing deterministic values for a given training example. We later applied max pooling to down-sample out input (the batch normalized output) followed by our activation function. We settled with Rectified Linear Units (ReLU) – see Figure 18.

At the end of the training process, provided the trained algorithm with a new set of data (with ten thousand images) to test our result. This new dataset is modelled like the training data.

Model: "model"

| Layer (type) | Output Shape | Param # | Connected to |
|---|---|---|---|
| input_1 (InputLayer) | [(None, 400, 400, 1) | 0 | |
| conv2d (Conv2D) | (None, 396, 396, 30) | 780 | input_1[0][0] |
| batch_normalization (BatchNorma | (None, 396, 396, 30) | 120 | conv2d[0][0] |
| max_pooling2d (MaxPooling2D) | (None, 198, 198, 30) | 0 | batch_normalization[0][0] |
| activation (Activation) | (None, 198, 198, 30) | 0 | max_pooling2d[0][0] |
| conv2d_1 (Conv2D) | (None, 194, 194, 30) | 22530 | activation[0][0] |
| batch_normalization_1 (BatchNor | (None, 194, 194, 30) | 120 | conv2d_1[0][0] |
| activation_1 (Activation) | (None, 194, 194, 30) | 0 | batch_normalization_1[0][0] |
| conv2d_2 (Conv2D) | (None, 194, 194, 30) | 22530 | activation_1[0][0] |
| batch_normalization_2 (BatchNor | (None, 194, 194, 30) | 120 | conv2d_2[0][0] |
| conv2d_3 (Conv2D) | (None, 194, 194, 30) | 22500 | activation[0][0] |
| add (Add) | (None, 194, 194, 30) | 0 | batch_normalization_2[0][0]<br>conv2d_3[0][0] |
| max_pooling2d_1 (MaxPooling2D) | (None, 97, 97, 30) | 0 | add[0][0] |
| activation_2 (Activation) | (None, 97, 97, 30) | 0 | max_pooling2d_1[0][0] |
| conv2d_4 (Conv2D) | (None, 93, 93, 30) | 22530 | activation_2[0][0] |
| batch_normalization_3 (BatchNor | (None, 93, 93, 30) | 120 | conv2d_4[0][0] |
| activation_3 (Activation) | (None, 93, 93, 30) | 0 | batch_normalization_3[0][0] |
| conv2d_5 (Conv2D) | (None, 93, 93, 30) | 22530 | activation_3[0][0] |
| batch_normalization_4 (BatchNor | (None, 93, 93, 30) | 120 | conv2d_5[0][0] |
| conv2d_6 (Conv2D) | (None, 93, 93, 30) | 22500 | activation_2[0][0] |
| add_1 (Add) | (None, 93, 93, 30) | 0 | batch_normalization_4[0][0]<br>conv2d_6[0][0] |
| max_pooling2d_2 (MaxPooling2D) | (None, 46, 46, 30) | 0 | add_1[0][0] |
| activation_4 (Activation) | (None, 46, 46, 30) | 0 | max_pooling2d_2[0][0] |
| conv2d_7 (Conv2D) | (None, 42, 42, 30) | 22530 | activation_4[0][0] |
| batch_normalization_5 (BatchNor | (None, 42, 42, 30) | 120 | conv2d_7[0][0] |
| activation_5 (Activation) | (None, 42, 42, 30) | 0 | batch_normalization_5[0][0] |
| conv2d_8 (Conv2D) | (None, 42, 42, 30) | 22530 | activation_5[0][0] |
| batch_normalization_6 (BatchNor | (None, 42, 42, 30) | 120 | conv2d_8[0][0] |
| conv2d_9 (Conv2D) | (None, 42, 42, 30) | 22500 | activation_4[0][0] |
| add_2 (Add) | (None, 42, 42, 30) | 0 | batch_normalization_6[0][0]<br>conv2d_9[0][0] |
| average_pooling2d (AveragePooli | (None, 21, 21, 30) | 0 | add_2[0][0] |
| activation_6 (Activation) | (None, 21, 21, 30) | 0 | average_pooling2d[0][0] |
| conv2d_10 (Conv2D) | (None, 21, 21, 5) | 155 | activation_6[0][0] |
| batch_normalization_7 (BatchNor | (None, 21, 21, 5) | 20 | conv2d_10[0][0] |
| activation_7 (Activation) | (None, 21, 21, 5) | 0 | batch_normalization_7[0][0] |

Total params: 204,475
Trainable params: 204,045
Non-trainable params: 430

**Figure 18.** Our detection model summary.

## Evaluation Metric

Intersection over Union (IoU) is an evaluation metric (between 0 and 1) that measures the overlap between the boundaries of the ground truth of an annotation and the predicted boundary. IoU evaluates whether a prediction is "good enough" [36]. The closer the prediction is to 1, the closer to perfect it is. Figure 19 illustrates the graphical view of the equation below.

$$IoU = \frac{Area\ of\ Intersection/Overlap}{Area\ of\ Union}$$

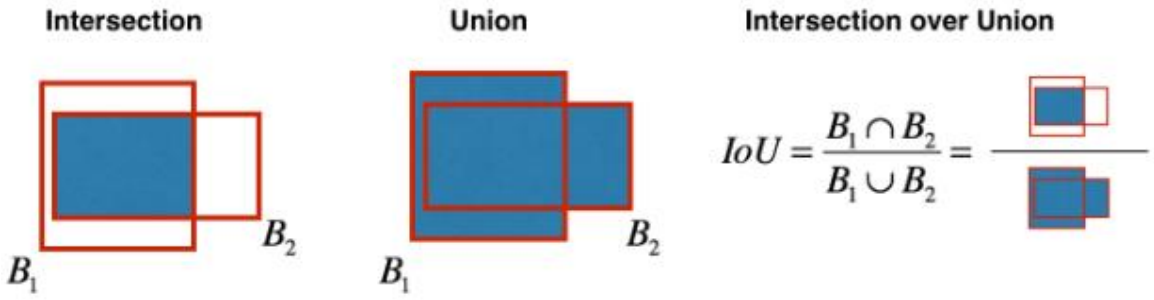


**Figure 19.** Graphical View of the IoU equation [41].

In general, when $IoU \geq 0.5$, we consider the prediction correct. Knowing IoU, we calculate precision and recall. We also calculated the confidence score. Confidence score reflects how likely the box contains an object and how accurate the bounding box is.


**Figure 20.** The higher the IoU, the better the performance [24].

# Chapter IV: Results

## Overview

The training of our object detection algorithm was over 200 epochs. It took approximately 8 hours, efficiently making use of a core i7 NVIDIA GeForce GTX 1650 with Max-Q Design computing power. During each epoch, the confidence metric improved over time, making detection closer to perfect.


**Figure 21.** (a) Object detection algorithm after 10 epochs (b) Object detection algorithm for an image without any link (c) Improved Object detection algorithm after 200 epochs.

## Detection Accuracy

Using the online Kaggle training compute power, we were able to achieve better accuracy. Running the training on 500 epochs brought the result close to perfect.

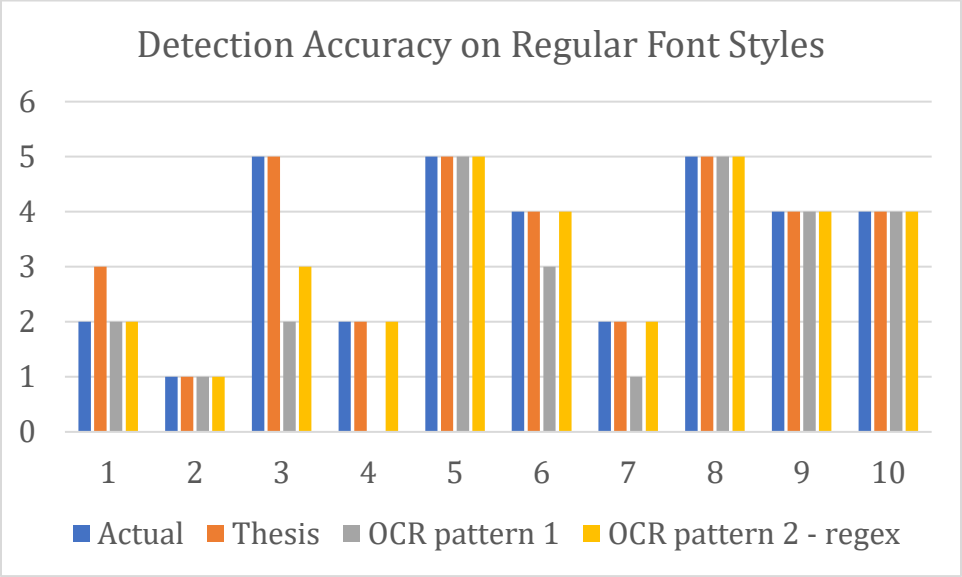


**Figure 22.** Accuracy between two Optical Character Recognition techniques and our model on ten im-ages using regular font style.

The OCR patter 1 technique used is a python library called “urlextract” while for the other pattern, we used regular expression. Before selecting the regular expression to use, we tried different ones from stackoverflow and chose the most accurate (see Figure 23).

```
WEB_URL_REGEXS = r"""(?i)\b((?:https?:(?:/{1,3}|[a-z0-9%])|[a-z0-9.\-]+[.](?:com|net|org|edu|gov|mil|aero|asia|biz|cat|coop|info|int|jobs|mobi|museum|name|post|pro|tel|travel|xxx|ac|
ad|ae|af|ag|ai|al|am|an|ao|aq|ar|as|at|au|aw|ax|az|ba|bb|bd|be|bf|bg|bh|bi|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cc|cd|cf|cg|ch|ci|ck|cl|cm|cn|co|cr|cs|cu|cv|cx|cy|cz|dd|de|dj|dk|dm|do|
dz|ec|ee|eg|eh|er|es|et|eu|fi|fj|fk|fm|fo|fr|ga|gb|gd|ge|gf|gg|gh|gi|gl|gm|gn|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|hr|ht|hu|id|ie|il|im|in|io|iq|ir|is|it|je|jm|jo|jp|ke|kg|kh|ki|km|kn|kp|
kr|kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|lv|ly|ma|mc|md|me|mg|mh|mk|ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|mv|mw|mx|my|mz|na|nc|ne|nf|ng|ni|nl|no|np|nr|nu|nz|om|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|ps|
pt|pw|py|qa|re|ro|rs|ru|rw|sa|sb|sc|sd|se|sg|sh|si|sj|Ja|sk|sl|sm|sn|so|sr|ss|st|su|sv|sx|sy|sz|tc|td|tf|tg|th|tj|tk|tl|tm|tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|us|uy|uz|va|vc|ve|vg|vi|vn|
vu|wf|ws|ye|yt|yu|za|zm|zw)/)(?:[^\s()<>{}\[\]]+|\([^\s()]*?\([^\s()]+\)[^\s()]*?\))|\([^\s]+?\))+(?:\([^\s()]*?\([^\s()]+\)[^\s()]*?\)|\([^\s]+?\)|[^\s`!()\[\]{};:'".,<>?«»“”‘’])|(?:(?
<!@)[a-z0-9]+(?:[.\-][a-z0-9]+)*[.](?:com|net|org|edu|gov|mil|aero|asia|biz|cat|coop|info|int|jobs|mobi|museum|name|post|pro|tel|travel|xxx|ac|ad|ae|af|ag|ai|al|am|an|ao|aq|ar|as|at|
au|aw|ax|az|ba|bb|bd|be|bf|bg|bh|bi|bj|bm|bn|bo|br|bs|bt|bv|bw|by|bz|ca|cc|cd|cf|cg|ch|ci|ck|cl|cm|cn|co|cr|cs|cu|cv|cx|cy|cz|dd|de|dj|dk|dm|do|dz|ec|ee|eg|eh|er|es|et|eu|fi|fj|fk|fm|
fo|fr|ga|gb|gd|ge|gf|gg|gh|gi|gl|gm|gn|gp|gq|gr|gs|gt|gu|gw|gy|hk|hm|hn|hr|ht|hu|id|ie|il|im|in|io|iq|ir|is|it|je|jm|jo|jp|ke|kg|kh|ki|km|kn|kp|kr|kw|ky|kz|la|lb|lc|li|lk|lr|ls|lt|lu|
lv|ly|ma|mc|md|me|mg|mh|mk|ml|mm|mn|mo|mp|mq|mr|ms|mt|mu|mv|mw|mx|my|mz|na|nc|ne|nf|ng|ni|nl|no|np|nr|nu|nz|om|pa|pe|pf|pg|ph|pk|pl|pm|pn|pr|ps|pt|pw|py|qa|re|ro|rs|ru|rw|sa|sb|sc|sd|
se|sg|sh|si|sj|Ja|sk|sl|sm|sn|so|sr|ss|st|su|sv|sx|sy|sz|tc|td|tf|tg|th|tj|tk|tl|tm|tn|to|tp|tr|tt|tv|tw|tz|ua|ug|uk|us|uy|uz|va|vc|ve|vg|vi|vn|vu|wf|ws|ye|yt|yu|za|zm|zw)\b/?(?!@)))
"""
```

**Figure 23.** Regular Expression used to recognize a URL in a string of text.

The reason for our result in image 1 (see Figure 24) detecting an extra image is because of overfitting. However, if converted into text, would be quickly eliminated because of the nature of the area of interest (see Figure 25). With improved tunning and more training, we should ex-perience better accuracy.


**Figure 24.** Our model experiencing overfitting.

The observed reason for the errors when the urlextract library and the regular expression technique were used is, the Optical Character Recognition (OCR) accuracy. The OCR we used is the pytesseract image to string library. It is a deep learning model that detects, recognizes and outputs texts based on learned patterns. This technique could get worse in an unconstrained environment where the level of noise and distortions is not controlled.

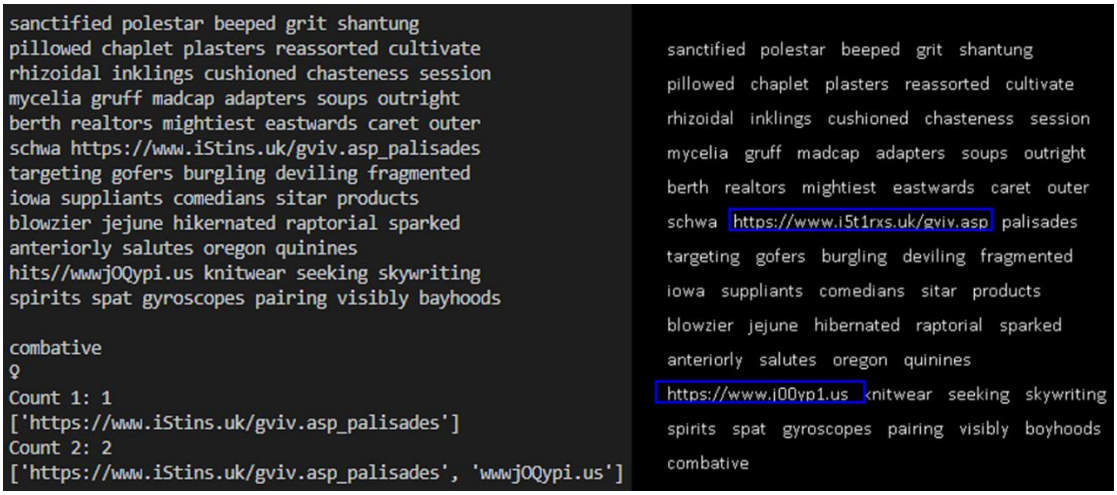


**Figure 25.** OCR image conversion to text versus our object detection model.

**Font Style and Size**

Font style and size significantly affects the visibility of a text. This is also an important factor to consider when creating a computer vision model to recognize texts. There are different categories of font styles. We will consider serifs and sans-serifs briefly. A serif is a decorative stroke that finishes off the end of a letters stem (sometimes also called the “feet” of the letters). A sans serif is a font that does not.

OCR software “won the battle” on italicized texts. This is simply because our model was trained on strictly serif and sans-serif fonts. Verdana (sans-serif), Arial (sans-serif), Segoe UI (sans-serif), Calibri (sans-serif), Cambria (serif) font style were used to train our model. The font size of our training images ranged between 14 pixels to 18 pixels. Our algorithm got better accuracy when the test images have similar texts font style.

In Figure 27, our model was evaluated on about 100 images with texts of font family of Brush Script MT and Bauhaus and font size between 24 and 28.


**Figure 26.** Example italicized text.

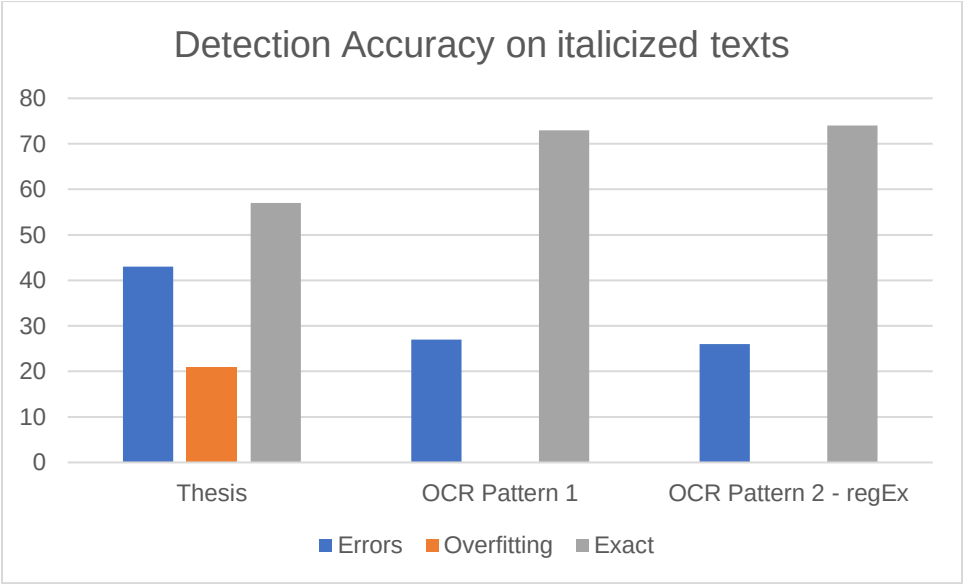


**Figure 27.** Detection Accuracy between Optical Character Recognition techniques and our model on ten images using italicized font style.

Training our model on these variants of font styles (and font size) should fix the less accuracy gotten from italicized texts in images.

**Input Size**

Using 400 x 400 input size image for training, we tested our algorithm on larger image sizes 1200 x 1200 and 600 x 600. Given that pytesseract OCR has been trained on a wide range of data, our model failed to perform optimally.

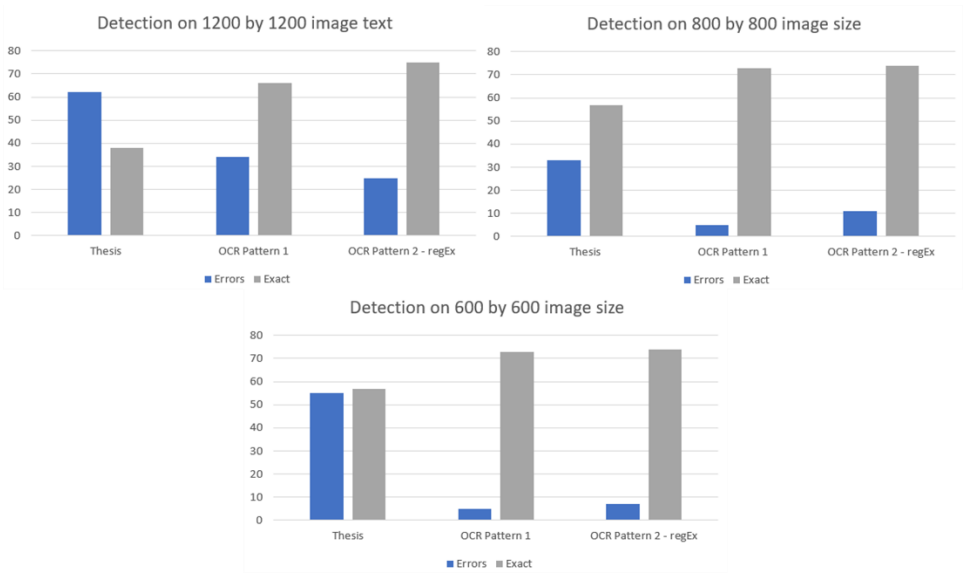


**Figure 28.** Detection on different input image sizes.

For better performance, comprehensive training is required with larger data set consisting of variety of input sizes and italicized texts with different font family, all of which pytesseract has.

# Chapter V: Conclusion

## Overview

The title of this project is “Object Detection and Recognition Using YOLO: Detect and Recognize URL(s) in an Image Scene”. This translates into the usage of object detection and recognition technology to successfully identify URL in images.

To achieve this, a custom YOLO algorithm is trained on fifty thousand images and evaluated on ten thousand images. The algorithm employs a 21 x 21 grid. When set up this way, it allows the bounding boxes generated to capture the objects properly with minimal chances of overfitting.

To generate training data, a text generator was programmed to generate images which randomly contain both texts and URLs and records useful information about the location of the URLs in a text file in JSON format. About fifty thousand images are used to train the model in about 500 epochs, allowing for a very dependable performance model.

## Application

The application of the algorithm is quite restricted, with its main use being the detection of URLs in an image scene. On its own, the application of the model is not wide and varied. However, in combination with other modifications, such as the augmentation this algorithm can prove to be very useful and beneficial. The project has a few applications in various fields. These applications will be described in subsequent paragraphs.

The model can be particularly useful in the field of tourism. Tourist can take a picture of a URL and have the information of the website displayed without having to type on their key-

board. Business owners, shop owners, and their customers can advertise and publicize information by leaving URLs to the services they offer on their advertisement spaces and users can pick up the URL(s) by the click of a button. The model can also be used to capture reference URLs while listening to a presentation at a conference. The uniqueness in of this model is that valuable time need not be wasted by users and eliminating the possibility of not being able to completely type out a URL in time in the case of a presentation. Given the ability of smart phone cameras in recent times, taking a picture of a URL while in transit should result in a "not too blurry" image which can be passed in as an input to our model and the URL(s) of interest retrieved.

The software can also be utilized at auctions that provide URL(s) to read more about an item. With the utilization of this model, information may be passed by appending URL(s) containing the full description of the items linked to them.

Another situation where the use of this model is in museum and zoos. Less space needs to be taken up when describing an item. A brief description of an item and a link to the full description would make museums and zoos less congested with placards. If implemented in the form of an application, could provide real-time data on items.

As opposed to QR codes, this model does not require any special software to encode the URL. QR codes cannot be handwritten but if our model is extended to handwriting recognition, it would be able to detect URLs.

**Limitations of Work**

One of the major limitations of this work is handwriting detection. The data which the algorithm is trained with is restricted entirely to digitally written text. As such, URLs written in freehand have a significantly lower chance of detection by the model.

Furthermore, the algorithm can only recognize horizontal texts. As a result of this, it is limited in functionality such that vertical as well as diagonal texts will be impossible to detect by the algorithm.

Due to the type of data the algorithm has been trained with; it is also impossible for the algorithm to detect links that span multiple lines. This is because the bounding boxes trained are not big enough to encompass texts that span multiple lines.

**Future work**

Implementing an OCR alongside our model would make converting URLs in a picture to clickable text. This would greatly reduce the need for advanced software like QR codes generators and QR code detectors. A person would need to simply have a URL displayed and anybody may freely access it when equipped with our modelled algorithm.

The second improvement would be to attempt to increase the range of the URLs that can be detected by the model. So far, our model was trained on slightly varying font size (see Figure 29). In general, font type should not create a significant difference in the accuracy of our model. It would be great to detect vertically aligned URLs (although that is not a common case in real life), and to detect URLs at different angles. Another case would be detecting URLs written backwards.

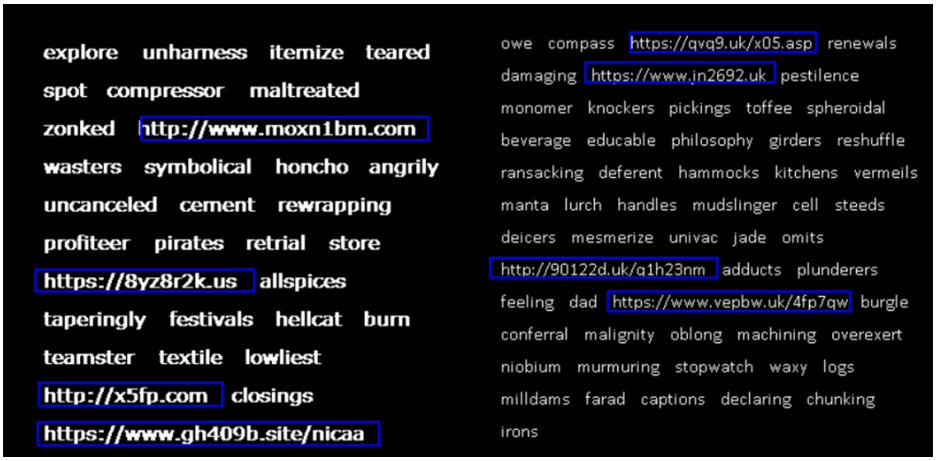


**Figure 29.** Our models performance on slightly varying font size.

Finally, the future iterations of this model could be trained on handwritings and more noisy environments and fonts to improve its ability to recognize a wider range of texts effectively.

## References


[1] P. Chakravorty, "What Is a Signal? [Lecture Notes]," in IEEE Signal Processing Magazine, vol. 35, no. 5, pp. 175-177, Sept. 2018, doi: 10.1109/MSP.2018.2832195.

[2] M. Rouse, "image" [Online]. Available at: https://whatis.techtarget.com/definition/image. [Accessed: 01/10/20]

[3] Merriam-Webster "image" [Online]. Available at: https://www.merriam-webster.com/dictionary/image, [Assessed: 01/10/20]

[4] Wikipedia, "Image" [Online]. Available at: https://en.wikipedia.org/wiki/Image, [Assessed: 01/10/20]

[5] The Editors of Encyclopedia Britannica, "Image-processing" [Online], Available at: https://www.britannica.com/technology/image-processing. [Assessed: 01/10/20]

[6] Wikipedia, "Encoding images" [Online], Available at: https://www.bbc.co.uk/bitesize/guides/zqyrq6f/revision/3. [Assessed: 02/10/20]

[7] Object (image processing) [Online], Available at: https://en.wikipedia.org/wiki/Object_(image_processing) , [Assessed: 02/10/20]

[8] P. Ganesh, Object Detection: Simplified [Online], Available at: https://towardsdatascience.com/object-detection-simplified-e07aa3830954, [Assessed: 02/10/20]

[9] Tensorflow, Available at: https://www.tensorflow.org/lite/models/object_detection/overview, [Assessed: 03/10/20]

[10] Wikipedia, "Object detection" Available at: 'https://en.wikipedia.org/wiki/Object_detection, [Assessed: 02/10/20]

[11] Fritz, "Object Detection Guide", Available at: https://www.fritz.ai/object-detection/, [Assessed: 03/10/20]

[12] Wikipedia, "Outline of object recognition", Available at: https://en.wikipedia.org/wiki/Outline_of_object_recognition , [Assessed: 02/10/20]

[13] Object Recognition, Available at: cse.usf.edu/~r1k/MachineVisionBook/MachineVision.files/MachineVision_Chapter15.pdf, [Assessed: 03/10/20]

[14] N. Pinto, D. D. Cox, and J.J. DiCarlo, "Why is Real-World Visual Object Recognition Hard?" (2008) *PLoS* Comput Biol 4(1): e27.

[15] A. Gulli and P. Sujit, "Deep Learning with Keras" (2017), Available at: https://1lib.us/book/3411804/7ea47a?id=3411804. [Assessed: 03/10/20]

[16] "The Definitive Glossary of Higher Mathematical Jargon - Algorithm". Available at: https://mathvault.ca/math-glossary/ [Assessed: 03/10/20]

[17] "Definition of ALGORITHM". Merriam-Webster Online Dictionary. Available at: https://www.merriam-webster.com/dictionary/algorithm [Assessed: 04/10/20]

[18] Y. Gavrilova, "Artificial Intelligence vs. Machine Learning vs. Deep Learning: Essentials" Available at https://serokell.io/blog/ai-ml-dl-difference [Assessed: 04/10/20]

[19] J. Brownlee, "A Gentle Introduction to Object Recognition with Deep Learning" (2018) Available at: https://machinelearningmastery.com/object-recognition-with-deep-learning/ [Assessed: 01/10/20]

[20] A. Kamal, “YOLO, YOLOv2 and YOLOv3: All You want to know”(2019) Available at: https://medium.com/@amrokamal_47691/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899 [Assessed: 04/10/20]
[21] You Only Look Once: Unified, Real-Time Object Detection, 2015. Available at: https://arxiv.org/abs/1506.02640, [Assessed: 04/10/20]
[22] A. Rosebrock., “Intersection over Union (IoU) for object detection” (2016) Available at: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, [Assessed: 04/10/20]
[23] I. Tan, “Measuring Labelling Quality with IOU and F1 Score”, Available at: https://medium.com/supahands-techblog/measuring-labelling-quality-with-iou-and-f1-score-1717e29e492f , [Assessed: 01/10/20]
[24] StackOverflow, “Intersection Over Union (IoU) ground truth in YOLO”, Available at: https://stackoverflow.com/questions/61758075/intersection-over-union-iou-ground-truth-in-yolo, [Assessed: 02/10/20]
[25] J. Redmon & A. Farhadi, (University of Washington), “YOLO9000: Better, Faster, Stronger” Available at: https://pjreddie.com/media/files/papers/YOLO9000.pdf, [Assessed: 02/10/20]
[26] “YOLO v2 – Object Detection” Available at: https://www.geeksforgeeks.org/yolo-v2-object-detection/ [Assessed: 04/10/20]
[27] A. Aggarwal, “YOLO Explained” *Available at:* https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31 [Assessed: 04/10/20]
[28] K. Mahesh Babu, M.V. Raghunadh, *Vehicle number plate detection and recognition using bounding box method*, May 2016, pp 106–110
[29] L. Cai, F. Jiang, W. Zhou, and K. Li, Design and Application of An Attractiveness Index for Urban Hotspots Based on GPS Trajectory Data, (Fellow, IEEE), pg 4
[30] Wikipedia, “Minimum bounding box” Available at: https://en.wikipedia.org/wiki/Minimum_bounding_box, [Assessed: 04/10/20]
[31] Dive into Deep Learning, “13.3. Object Detection and Bounding Boxes” Available at: https://d2l.ai/chapter_computer-vision/bounding-box.html , [Assessed: 04/10/20]
[32] J. Redmon & A. Farhadi, YOLOv3: An Incremental Improvement, University of Washington Available at: https://arxiv.org/abs/1804.02767 [Assessed: 05/10/20]
[33] YOLO: You Only Look Once, Available at: jeremyjordan.me/object-detection-one-stage/#yolo, [Assessed: 05/10/20]
[34] Longman Dictionary, Definition of training, Available at: https://www.ldoceonline.com/dictionary/training [Assessed: 05/10/20]
[35] Guangrui Liu “Real-Time Object Detection for Autonomous Driving Based on Deep Learning” (2017), Available at: https://tamucc-ir.tdl.org/handle/1969.6/5637 [Assessed: 03/18/21]
[36] A. Abdulkader & C. Vlahija “Real-time vehicle and pedestrian detection, a data-driven recommendation focusing on safety as a perception to autonomous vehicles”, Available at: http://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1479957&dswid=-3676 [Assessed: 03/18/21]

[37] Enhancement methods in image processing Available at: https://www.mathworks.com/discovery/image-enhancement.html. [Assessed: 03/18/21]
[38] Evaluating a machine learning model. Available at: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/. [Assessed: 03/18/21]
[39] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Available at https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Assessed: 03/18/21]
[40] J. Jokela "Person Counter Using Real-Time Object Detection and a Small Neural Network" Turku University of Applied Sciences. Available at: https://www.theseus.fi/bitstream/handle/10024/153489/Jokela_Jussi.pdf?sequence=1&isAllowed=y. [Assessed: 03/18/21]
[41] Manishgupta "YOLO – You Only Look Once". Available at: https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4. [Assessed: 03/18/21]
[42] E. Y. Li "Dive Really Deep into YOLO v3: A Beginner's Guide". Available at: https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e. [Assessed: 03/18/21]
[43] F. Zelic & A. Sable "A comprehensive guide to OCR with Tesseract, OpenCV and Python.". Available at: https://nanonets.com/blog/ocr-with-tesseract/. [Assessed: 03/18/21]
[44] Tessdoc. Available at: https://github.com/tesseract-ocr/tessdoc/blob/master/ImproveQuality.md. [Assessed: 06/04/21]
[45] J. Goyvaerts "Regular Expressions: The Complete Tutorial". Available at: https://www.regular-expressions.info/print.html. [Assessed: 06/04/21]
[46] M. Erwig & R. Gopinath, "Explanations for Regular Expressions". Available at: https://web.engr.oregonstate.edu/~erwig/papers/ExplRegExp_FASE12.pdf. [Assessed: 06/04/21]