LUISS

Master's Degree in Data Science and Management

Course of Machine Learning

A novel approach for detecting signatures: YOLO Neural Network

Prof. Giuseppe Italiano

SUPERVISOR

Prof. Blerina Sinaimeri

CO-SUPERVISOR

Carlotta Cantelmo 746131

CANDIDATE

Academic Year 2022/2023

Contents

Introduction

The subject considered in this thesis is a categorization problem: the objective is to utilize a model that takes documents as input and organizes them based on whether or not they present a signature. The information from this classification allows us to select and organize signed documents. A scanned digital or handwritten signature is generally present on identity documents like identity cards, which comprise most documents under examination. As a result, it's essential to consider the document's shape and orientation, the background color, all of the items on the page, the type of written sentences—handwritten and computerized—and a wide range of other pertinent criteria. Since there are already several ways to accomplish this, the issue of signature identification is considered straightforward and quick to resolve. However, it's crucial to highlight an essential component of these procedures since it will clarify why I've chosen to take on this issue. Nowadays, Machine Learning algorithms are applied to check the signature's authenticity. Banks' mechanism to validate the signature when it is manually entered at the desk to sign any form of the document is a typical illustration of these procedures. In this instance, as with other signature processes, the goal is to confirm the integrity of the signature rather than locate where it is on the page. To guarantee the signature's authenticity, it is considered individually in each step and compared with a corresponding signature in

the dataset. As a result, the signature position identification step uses more straightforward and less accurate models. This phase sometimes misses, and the model is trained directly using datasets that include only signatures without considering the entire document page.
These factors should clarify why I chose to offer this research as a thesis project. The organization of papers is the primary goal of this project, as was already mentioned. We need to use a model whose main objective is to locate the signature on a piece of paper, not to validate it: this is the primary distinction between my research and current models in this area. Finding the location of the signature on the documents is the real thing that matters in this project; the subsequent procedures are nonessential and can always be added in the future. Neural Networks are typically used in projects for the process's last step, validating the signature. The intermediate phases of these undertakings only partially utilized Deep Learning models' accuracy and precision. Therefore, the signature selection stage on the sheet is usually considered a step leading up to the final verification stage, as demonstrated in the following few paragraphs, and needs more consideration: this is the logic behind using Machine Learning models rather than Deep Learning ones. The example that follows will clarify this statement.

The Connected Components method is one of the most often used models for detecting a signature. It operates as follows: the model creates a "mask" that divides the page into sections and, within each mask frame, performs an analysis of these Connected Components. This method bases on the generally accurate notion that when we sign, we tend to link all of the letters together, yet when we type on a computer, all letters are separated. The study of Connected Components assumes that if numerous letters are attached, the likelihood that it is a signature rises significantly. When I first tried this method, it did not produce the desired results because I couldn't observe any marks on the page. As a result, the sheet's sections corresponding to printing, other handwriting, or scanning errors were frequently mistaken for signatures. These circumstances explain why I didn't think this approach was trustworthy and moved on to look for a more accurate solution. After conducting considerable research that consistently produced outcomes similar to the ones just mentioned, I considered changing my perspective and eventually discovered a fresh way to see the project's aim. The goal is to locate a signature on a sheet, which can also be interpreted as finding an item within an image, as I have stated numerous times before. This realization caused me to consider the potential for using some Object Detection methods. Unfortunately, I could not locate any scientific publications in the literature that may have explained to me on a theoretical and technical level whether such a technique can be carried out or done previously. However, I decided to give it a shot. I searched for several Object Detection models to determine which was best. As we will see in the coming chapters, among the existing approaches, I use a Convolutional Neural Network known as YOLO (*You Only Look Once*). In addition to conducting this study, I was able to locate a Notebook on GitHub that used YOLO to carry out the process I needed [10]. All of these factors led me to choose this model. This Notebook was extremely helpful to me in understanding how the selected model operates.

What distinguishes this work is the shift in perspective. I made an effort to come up with a different solution to a problem that is already well-known and frequently addressed using the same set of methods (such as Connected

Components). As previously stated, the identification phase of current initiatives always played a minor role because the primary focus was on their verification rather than the identification of signatures. In contrast to the subsequent verification phase, which typically uses Deep Learning, this phase has never been surpassed using more complicated Deep Learning models, such as a Neural Network. Instead, I preferred to make signature detection the project's main objective and worked to implement a Deep Learning technique to do so. The outcome was very satisfying. Using the Neural Network, Yolo allowed me to shift my perspective, which was the best option for my project.
The usefulness of my research can also be emphasized, in addition to its value explained in the previous paragraphs. For example, a significant corporation that lacks a document cataloging system that meets its requirements could need this project. Due to this, it's crucial to emphasize that the implemented model should also be considered innovative research and a useful cataloging tool that avoids wasting time and money on manual document classification. Furthermore, in the future, we can explain the beneficial applications of this model; for example, in addition to the signatures' location, we can add the detection of identity images or the tax code. All of this is made possible by using Yolo, which enables the inclusion of other object classes in addition to signatures without requiring any other model changes, allowing it to adapt to a company's requests and change along with it as the tasks evolve.

Chapter 1

YOLO

1.1 The Object Detection Method

Humans rely on their eyes to perceive everything; they take in data from the world around them and send it to their brains for analysis and inference-making. Well, it seems very easy, doesn't it? Just by looking, we can learn a great deal about the things we are seeing, including what they are, where they are, and how they are used. But the processing that our brain performs is simply incomparable. The researchers wondered if we would be able to give a machine this intriguing brain function. With this, the machine's duty will be significantly easier since, if it can identify the items in its environment, it can engage with them more effectively—which is the whole point of developing machines—to make them receptive to interaction with humans and more human-like. How can we teach a machine to recognize an object? The field of computer vision known as "Object Detection" was born out of this. It is the study of finding instances of different classes of items (such as a person, book, chair, automobile, bus, etc.) in digitally produced images and videos. Recent years have seen major advancements in automated data processing and interpretation because to the development of Deep Learning. One of the advantages of Neural Networks is the ability to adapt and progressively learn how to compute the solution to a given problem with the aid of learning phases. Training entails adapting the parameters of the various levels throughout the backward propagation phase in order to reduce the difference between the value generated and the expected value. The sub-domains of this Object Detection include things like face detection, activity recognition, image annotation, and many others. Applications for Object Detection can be found in a number of significant fields, including self-driving cars, robots, video surveillance, object tracking, etc.

1.2 YOLO, You Only Look Once

As already said, Object Detection has a wide variety of uses, including autonomous driving, the detection of aerial objects, text detection, surveillance, search and rescue missions, robotics, the recognition of faces and pedestrians, visual search engines, the calculation of things of interest, brand detection, and many more.[5]

The main obstacles to object detection are:

- Processing of low-resolution visual contents.
- Handling varied sized multiple objects in an image.
- Availability of labelled data.
- Handling overlapping objects in visual content.
- The occupancy of an object in an image has an inherent variation, such that objects in an image may occupy majority of the pixels, or 0,70 to 0,80, or very few pixels, or 0,10 or even less. [5]

Fast, precise, and able to distinguish a range of items should all be characteristics of general-purpose object detection. The advent of Neural Networks has accelerated the speed and precision of detection frameworks. However, the majority of detection techniques are still limited to a few types of objects. Compared to datasets for other tasks, like classification and tagging, object detection datasets currently available are small. The most popular detection datasets include tens of thousands to several hundred thousand images along with a few dozens to several hundred tags. Classification datasets have millions of images with tens of thousands or hundreds of thousands of categories. Detection should be scalable to the level of object classification. [8]

The object detection field has advanced tremendously over the past few years as a result of a significant amount of study. The performance of the algorithms has attained levels of accuracy and precision that have never been higher. The fundamental goal in this field is to mimic the functions and skills of human eyes in terms of object recognition after only a single glance. There are now several algorithms that are especially effective at doing this. The detection accuracy is increasing significantly with the introduction of YOLO algorithm (*You Only Look Once*) and its architectural descendants, and it is occasionally better than two stage detectors. YOLO versions are frequently used in a variety of applications due to their quick conclusions and precise identification. For example, *YOLO9000*, one of the various YOLO variations, is a real-time object detector that can recognize over 9000 different item types. The model is trained on more than 9000 classes from ImageNet as well as detection data from COCO using the dataset combination approach and joint training algorithm. [8] In light of this, it is possible to study this innovative model and its object identification technique.

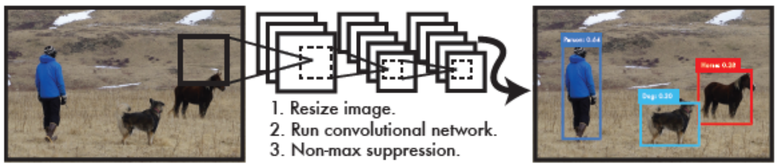


Figure 1.1: The Yolo Detection System [9]

YOLO is a special Neural Network because it has an architecture that is incredibly fast. Because the entire detection pipeline consists of a single network and can be improved end-to-end based solely on detection performance. The model has undergone numerous iterations over the years in an effort to increase accuracy and speed. This algorithm's major objective is to mimic how individuals perceive the world: after taking just one look at something, it can promptly and accurately identify it. [9]

1.2.1 The Algorithm

As was previously said, the YOLO method is a real-time object recognition system for images and videos. It is extremely quick while maintaining a high level of accuracy and precision because of its architecture. YOLO is refreshingly simple. A single neural network predicts multiple bounding boxes and class probabilities for each box. While training on entire pictures, YOLO instantly improves detection performance. It redefines object recognition as a single regression problem by replacing image pixels with bounding box coordinates and class probabilities. It is possible to see a general representation of how it works in Figure 1.1.

It can tell what things are and where they are located by only taking a single look at the image. Unlike classifier-based approaches, YOLO is incredibly quick during testing because it only needs to evaluate one network. As will be covered in more detail in the chapter's following section, the grid design requires spatial variation in bounding box predictions. Due to the fact that it is frequently clear which grid cell an object belongs in, the network can only predict one box for each object. [9]

As a result, J. Redmon et al. [9] provide a novel approach to the object detection issue by integrating the various system components into a single network and developing the first "one-stage" object detection system. By simultaneously reasoning about the full image rather than just a few parts, the network is able to better understand the environment in which the various object classes are situated. Additionally, by doing so, it establishes an implicit connection between object classes that are actually connected.

The authors of YOLO [9] have changed the definition of the Object Detection problem from classification problem to regression problem. A single Convolutional Neural Network is used to forecast the bounding boxes and assigning class probabilities to each of the detected objects shown in an image. As said before, this algorithm just needs to look at the image once to recognize the objects and their locations using boundary boxes. The difficulty comes from precisely identifying many items and determining their exact locations within a single visual input. Two fundamental Convolutional Neural Network (CNN) characteristics, parameter sharing and multiple filters, are capable of successfully addressing this Object Detection issue. When discussing YOLO, a number of other criteria in addition to those already mentioned must be taken into account. Different qualities that explain why this model can be considered as innovative can be found.

- Speed: Because it doesn't deal with complicated pipelines, it is incredibly quick. It has a 45 Frames Per Second (FPS) image processing speed. YOLO is one of the best choices for real-time processing since it can achieve more than twice the mean Average Precision (mAP) compared to other real-time systems. [12]

- High accuracy: The YOLO prediction method provides precise findings with few background mistakes. [6]
- Learning capabilities: The algorithm has great learning capabilities that allow it to pick up on object representations and use them for object detection. [6]
- Good generalization: This is especially accurate for the newest YOLO iterations. With those improvements, YOLO went a little further and offered improved generalization for new domains, making it ideal for applications that require quick and reliable object identification. [12]
- Open source: Making YOLO open source encouraged the community to continuously enhance the model. This is among the factors that have contributed to YOLO's rapid rise. [12]

Understanding what is actually being forecasted is crucial to comprehension of the YOLO algorithm. Predicting an object's class and the bounding box that specifies the object's position is the ultimate goal. There are four descriptors that can be used to describe each bounding box:

- x, y, the center of the box;
- *Width* of the box;
- *Height* of the box;

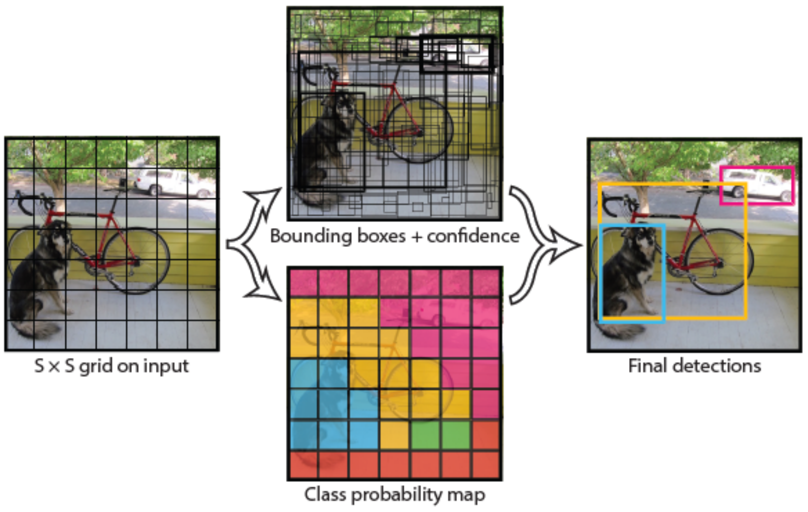


Figure 1.2: The Model [9]

- *Value c*, corresponding to the class of an object.

Additionally, it is forecasted a real number Pc, which represents the probability that an object is present within the bounding box. Instead of looking for potential object-containing regions in the input image, YOLO divides the image into cells, usually on a 19 x 19 grid. Each cell is then accountable for estimating K bounding boxes. Only when the anchor box's centre coordinates fall within a given cell is an object considered to be in that cell. This parameter causes the height and width to always be calculated relative to the entire image size while the centre coordinates are always calculated relative to the cell. [14]

YOLO calculates the likelihood that a specific class is present in the cell during the first run of forward propagation. The class with the highest probability is selected and put in that specific grid cell. For each of the grid cells shown in the image, a similar process takes place. [14] This is shown in

Figure 1.2.

The architecture of this Neural Network will be discussed in further detail in the sections that follow, with an emphasis on both its advantages and disadvantages as well as comparisons to other object recognition models. The framework used is another crucial factor to pay attention to when implementing a Neural Network and working with Deep Learning in general. These are open-source interfaces, frameworks, or technologies that even someone with little background in AI and machine learning may simply integrate. It is possible to upload data and train a Deep Learning model with the support of these frameworks, which will result in precise and understandable predictive analysis.

Here we have some framework examples:

- A recent Deep Learning framework built on Torch is called *PyTorch*. It was created by Facebook's AI research team and made available on GitHub in 2017 for applications involving Natural Language Processing. The reputation of PyTorch is one of simplicity, use, adaptability, effective memory usage, and dynamic computational graphs. Additionally, it feels native, which speeds up processing and simplifies coding.

- A complete, open-source Deep Learning framework called *TensorFlow* was created by Google and released in 2015. It is renowned for its support for many abstraction layers, scalable production and deployment options, and compatibility for various platforms, including Android. It is a symbolic math library for Neural Networks that works best with dataflow programming for a variety of purposes. It offers a flexible, complete ecosystem of community resources, frameworks, and tools that simplify creating and deploying machine learning programs. It is a promising and quickly expanding entering into the Deep Learning arena. Additionally, TensorFlow now uses *Keras*, making it difficult to compare the two.

- Python-based Keras is a powerful high-level Neural Network Application Programming Interface (API). This open-source Neural Network framework can operate on top of CNTK, *TensorFlow*, and *Theano* and is made to allow for quick experimentation with deep neural networks. The three main goals of Keras are modularity, usability, and extensibility. Low-level computations are passed off to the *Backend*, a different library, rather than being handled by it. In the middle of 2017, Keras was adopted and incorporated into TensorFlow. It is accessible to users through the *tf.keras* module. The Keras library can still function independently and individually, though.

Both TensorFlow and PyTorch provide practical abstractions that simplify model creation. They vary because TensorFlow provides a multitude of options, whereas PyTorch takes a more "pythonic" and object-oriented approach. Even though PyTorch is the least well-known of the three main frameworks, it is still used for many Deep Learning projects today. PyTorch is the tool of choice for researchers that need flexibility, debugging tools, and quick training times. TensorFlow is the preferred tool due to its well-documented framework, amount of trained models, and tutorials. Better visualization provided by TensorFlow enables developers to more effectively debug and monitor the training process. But PyTorch only offers a small amount of visualization. Because of the TensorFlow Serving framework, TensorFlow outperforms PyTorch when it comes to delivering trained models into the real world.

PyTorch and Keras are both excellent, but developers who want a plug-and-play framework that enables them to design, train, and evaluate their models rapidly should choose Keras over PyTorch. Additionally, Keras provides simpler model export and more deployment options, but PyTorch is quicker and offers better debugging capabilities than Keras. Both platforms are well-known enough to provide a wealth of educational content. PyTorch offers an exceptional community and active development, while Keras has excellent access to reusable code and tutorials.

While Keras is a high-level neural network library that runs on top of TensorFlow, that is an open-sourced end-to-end platform, library for many Machine Learning tasks. Both offer high-level APIs that make it simple to design and train models, while Keras is more approachable due to the inclusion of Python. When dealing with massive datasets and object detection and needing top-notch functionality and fast performance, researchers turn to TensorFlow. After weighing all of these frameworks, it was decided to employ Yolo on Tensowflow to carry out the signature detection project.

1.2.2 The Architecture

The model has undergone some improvements over time, but the dominant model idea has not changed.

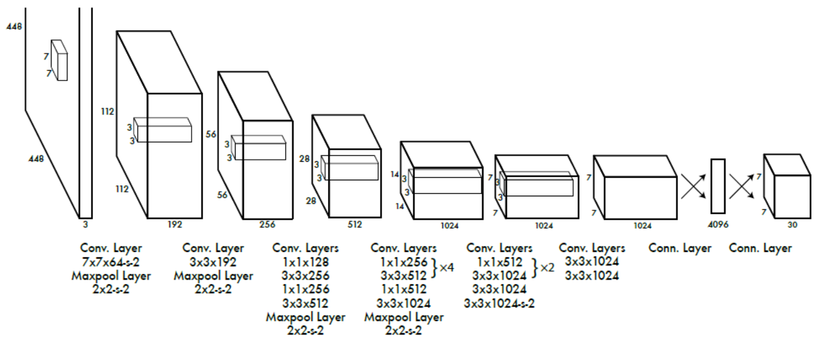


Figure 1.3: The Architecture [9]

Network Design

Convolutional Neural Networks are used to create this model, which Redmon J. et al.[9] evaluate using data from the PASCAL VOC 2007 and 2012 detection dataset. While the fully connected layers of the network anticipate the output probabilities and coordinates, the initial convolutional layers extract features from the image.

The design of YOLO was heavily influenced by that of GoogleNet, although instead of its initial modules, (1 x 1) convolution followed by (3 x 3) convolutional filters were used, with a 7 x 7 filter only present in the first convolutional layer. As seen in Figure 1.3, YOLO has 24 convolutional layers, of which only four are followed by max-pooling layers. These 24 convolutional levels are then followed by 2 fully connected layers. The method makes use of global average pooling and (1 x 1) convolution.

The final level of YOLO employs a linear activation function, while all other levels employ the following activation function: [7]

$$\phi(x) = \begin{cases} x \ se \ x > 0 \\ 0{,}1x \ altrimenti \end{cases}$$

The authors simply use 11 reduction layers followed by 3 x 3 convolutional layers in place of the inception modules utilized by Google Neural Network. The 7 x 7 x 30 tensor of predictions is the network's ultimate output. [9]

Unified Detection

The technique combines many elements of object detection into a single Neural Network, as was already stated in the previous section. To predict each bounding box, this network takes features from the full image. Additionally, it simultaneously predicts all bounding boxes for a picture across all classes. This indicates that the network considers the entire image and all of its objects when making decisions. The Yolo design maintains excellent average precision while enabling end-to-end training and real-time speeds. [9]

The algorithm used in this Object Detection procedure divides the input image/frame into S x S grid cells, and each grid cell predicts B bounding boxes together with their locations and dimensions, the probability that an object will be found in the underlying grid, and conditional class probabilities. The basic idea behind an object being detected by any grid cell is that the object's centre must lie within that grid cell, and the cell in question is responsible for detecting that object if its centre does so. Formally, it forecasts the bounding boxes and confidence levels for each box in a grid. The model's level of confidence in the box's object-containment and how accurately it thinks the box will be predicted are both shown by these confidence scores.

Bounding box-specific parameters include the following: *pc* represents the probability of containing an object in the grid by the underlying bounding box; *bx* and *by* indicate the center of the predicted bounding box; *bh* and *bw* represent the predicted dimension of the bounding box, *p(ci)* means conditional class probability that the object belongs to *ith* class for the given *pc* and *n* is the number of classes/categories. A grid cell predicts (B $\times$ 5 + n) values, where *B* is the number of bounding boxes per grid cell. The output tensor shape would be S $\times$ S$\times$ (B $\times$ 5 + n) as the image is divided into S $\times$ S grid cells.

When the input image is divided into, for instance, 19 x 19 grids and four bounding boxes are predicted for each grid, the final schematic of the output tensor prediction is shown in the following figure, where class probabilities are distributed among all the bounding boxes for a given grid. For each bounding box within a grid, the confidence score *cs* is a value that is assigned to each grid cell and predicts the bounding boxes. It represents both the probability that an item will be located within the bounding box *Pr(Object)* and how correctly the size and position of the bounding box are considered to be related to the object, the *IOU truth pred*. It is calculated by multiplying the probability *pc* by the intersection over union (Figure 1.4) between the predicted and actual bounding box.

Formally, confidence is defined as *Pr(Object) * IOU(truthpred)*. The confidence ratings should be zero if there is no object present in that cell. If not, the confidence score is needed to be the same as the Intersection Over Union (*IOU*) between the predicted box and the actual data. The centre of the box in relation to the boundaries of the grid cell is represented by the *(x, y)* coordinates. Relative to the entire image, the width and height are predicted. The IOU between the projected box and any ground truth box

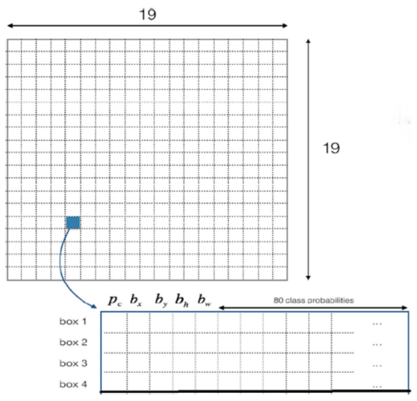


Figure 1.4: Dividing the image into grid cells and predictions corresponding to one grid cell [5]

is represented by the confidence prediction, which is the last step. The C conditional class probabilities, *Pr(ClassijObject)*, are likewise predicted for each grid cell. These probabilities depend on the grid cell in which an object is located. [9]

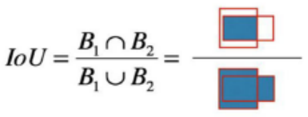


Figure 1.5: IOU metrics representation [4]

These bounding boxes typically vary in size while taking into account various shapes for capturing the various objects. A Bounding Box should be used to identify an object in the image such that the object's center falls inside the Bounding Box. The centers of many objects could, however, coexist in the same box. The boxes that correspond to a single grid cell were referred to by the authors as *Anchor Boxes*, using a distinct terminology. After examining the dataset and its underlying objects, a set of standard bounding boxes known as "Anchor Boxes" were chosen.

By taking into account various width and height combinations, such as square, vertical or horizontal rectangle, etc., to suit the aspect ratio and size of all the items available in the dataset, these selected anchor boxes should reflect the majority of classes/categories. In order to forecast the overlapping bounding boxes for the same object, adjacent grid cells may also predict that object. Because neighbouring grid cells might suppose the item center falls within of them, there would therefore be several predictions.

Figure 1.6a shows the multiple bounding box prediction for an object, whereas Figures 1.6b and 1.6c show the high and low overlapping between the predicted box and the actual ground truth, respectively. It is crucial to find a solution to the problem of the same object being detected by several grid cells or multiple bounding boxes of the same grid cell.

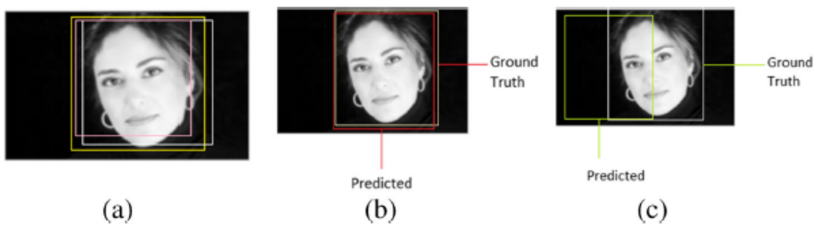


Figure 1.6: Multiple bounding boxes and their overlapping with the ground truth (a) Multiple bounding boxes (b) high overlapping (c) low overlapping [5]

Each bounding box in every grid will now have a class-specific score, box coordinates, and classification output category associated with it. There will be a total of *S2 x B* predicted boxes, and boxes with class scores below a certain minimum threshold will be eliminated. In most Object Detection algorithms, this threshold is typically fixed to 0.5, however it may change based on the dataset and its properties. Low probability of an object being present in that grid or low probability of any class category that optimizes the

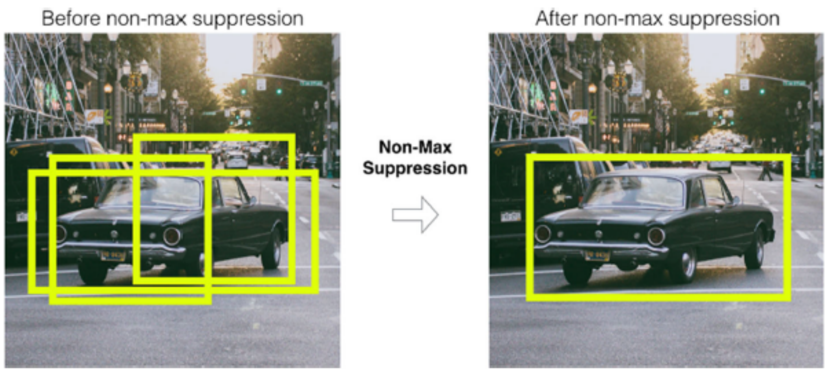


Figure 1.7: The effect of Non-Max Suppression in Object detection using YOLO [5]

class score may be the cause of a low score. A smaller number of bounding boxes remain, but their count is still very high after being eliminated with the help of some threshold.

Non-max suppression, which also has the IoU as a foundation, is the second criterion for eliminating the less important bounding boxes. Fig. X displays the non-max suppression's effect.

Internally, Non Max Suppression makes use of the crucial idea of IoU [4], which may be computed for two boxes as shown in Figure 1.7. The box with the highest-class score is initially chosen. All additional bounding boxes that overlap the selected box will be ignored if their IoU exceeds a certain cutoff.

As previously mentioned, the model predicts each bounding box with five values: x, y, w, h, and the confidence score *Pr(Object) * IOUtruth pred*, where *(x, y)* are the coordinates of the bounding box's centre with respect to the grid cell's edges, and w and h represent the bounding box's width and height, respectively, in relation to the entire image.[7]

Regardless of how many produced bounding boxes are generated, each grid cell forecasts the C classes using its conditional probability *Pr(ClassijObject)*.

The model now calculates a class-specific confidence score for each bounding box by multiplying the conditional probabilities of the classes of the cell by the bounding box's confidence score:

$$
\begin{aligned}
&Pr(ClassijObject) * Pr(Object) * IOUtruth \\
&\qquad = pred = Pr(Classi) * IOUtruthpred
\end{aligned}
$$

The result represents the likelihood of an object belonging to a particular class occurring within the bounding box under consideration as well as the accuracy with which the stated bounding box delimits the object's spatial contours.[7]

Training

The authors [9] claim that YOLO training is divided into two stages: a first pre-training stage in which only the deeper layers are trained, and a second training stage in which the entire network is trained.

Phase I: Pre - Training Only the first 20 convolutional layers, along with an average-pooling level and a fully connected level, are trained when the training process is first started, according to the authors of YOLO. The ImageNet 1000-class competition dataset with input photos of size 224 x 224 pixels is used for this purpose. After about a week of training, the model had an accuracy of about 88 percent on the ImageNet 2012 validation set.

Phase II At this point, J. Redmon et al.[9] removed the last two layers of the network - added specifically for the first part of the training - to add four more convolutional layers, followed finally by two fully connected layers. This time, the model is trained on images with twice the resolution (448 x 448 pixels), as accurate and generic object detection models need precise information to improve. According to the authors' design, the second training phase includes 135 iterations using the training and validation sets for PASCAL VOC 2007 and PASCAL VOC 2012. A batch size of 64, a momentum of 0.9, and a decay of 5 x 10 x 4 were used during training. The model contains a 50% dropout level after the first fully connected layer to prevent coadaptation between fully connected levels, which reduces overfitting. Additionally, data augmentation techniques were employed to reduce this phenomenon: the model transforms the input pictures by adding translations and altering the dimensions by up to 20% from their original features before using them. Additionally, a factor of 0 to 1.5 is used to alter the photos' exposure and saturation. [7]

Both the probabilities of the classes and the coordinates of the bounding boxes for the recognized objects are predicted by the network's final layer. According to the size of the input image, the model normalizes the bounding boxes' width and height so that their values fit inside the real range between zero and one.

In order for the center of the bounding boxes to also fall into the range between zero and one, it is normalized and expressed in reference to the cell of the grid to which it is assigned. With regard to the learning rate, the authors noted that the model often diverges towards unstable gradients when trained from the beginning with high levels of this parameter. For this reason, the learning rate starts from a low value, 10 x 3, and then according to the following pattern: The learning rate gradually grows over the first few epochs until it reaches a value of 10 x 2, then it stays the same until epoch 75, before changing to a rate of 10x3 for the subsequent 30 epochs and a rate of 10 x 4 for the final 30.

The Loss Function The function used to determine the error should be carefully chosen given the model's complexity. Because it is simple to optimize, the authors chose to employ a quadratic error; nevertheless, this decision does not align well with the goal to maximize the mean accuracy of the forecasts. Classification mistakes are compared to location errors using this method. Additionally, many grid cells in each image can be empty. As a result, these cells' confidence score is pushed towards zero, frequently increasing the influence of the gradients produced by them. According to the authors, all of this can make the model more unstable and cause it to diverge.

The authors fix this issue by decreasing the gradient gained from bounding boxes that do not contain objects and increasing the gradient obtained from those that do. They use two parameters, *lcoord* and *lnoobj*, respectively, to do this. These parameters are set to lcoord = 5 and lnoobj = 0.5 by default.

The loss functions have undergone changes over the years. The prediction made by the model for nearby items was inadequate because the original YOLO, version 1, makes the same mistake for both large and small objects. Two objects can only be identified if they appear in the same grid, making it difficult to detect little things. The new loss function is better and more adaptable than the previous one. The proportionality takes the place of the initial difference in the new loss function. YOLOv1's initial loss function used a single loss function for both object categorization and bounding box calculation. Now it is possible to divide the loss function in five parts, each of them is focused on a different aspect of the model: parts 1 and 2 are concerned with the loss of the bounding box coordinates; parts 3 and 4 are concerned with the difference in the confidence that an object is present in the grid; and part 5 is concerned with the difference in class probability.[2]

The loss function of YOLO consists of:

- The classification loss,
- The localization loss (the spatial error between the predicted and the actual bounding box),
- The confidence loss (the overall error of the bounding box, i.e., the error in the calculated confidence score).

We will see each of them in the next pages.

The Classification Loss It determines whether or not the recognized class matches to the object specified in the image by computing the error in the categorization of a particular object. If a grid cell is thought to be in charge of a specific object, then it is also in charge of classifying that thing. The quadratic function used to determine the conditional probabilities of the recognized class with regard to each class is what causes the error in this operation: [7]

$$\sum_{i=0}^{S^2} 1_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p_i}(c))^2 \tag{1.1}$$

The Localization Loss It measures the error in the prediction of bounding boxes, i.e. the height, width and centre coordinates. Only bounding boxes that are in charge of identifying an object are used by YOLO to calculate this mistake. The localization error is computed as follows:

$$\lambda(coord)\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_i^{obj}\left[(x_i-\hat{x_i})^2+(y_i-\hat{y_i})^2\right] \\ +\lambda(coord)\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\left[\left(\sqrt{w}_i-\sqrt{\hat{w_i}}\right)^2+\left(\sqrt{h}_i-\sqrt{\hat{h_i}}\right)^2\right]$$

The *lambda_coord* and *lambda_noobj* are scalars to weight each loss function, It is clear that the error with respect to the size of the bounding box is calculated on the square root of the values. This is done so that faults of different sizes will not be treated equally. For instance, a *2-pixel* inaccuracy can be significant in a bounding box of 30 pixels but insignificant in one of 500 pixels. [7]

The Confidence Loss It measures how objectively the forecast produced with just one bounding box was made. If, given a bounding box, the model detects an object within it, the following formula is used to determine the confidence loss:

$$\sum_{i=0}^{S^2}1_i^{obj}\sum_{i=0}^{B}(C_{ij}-\hat{C_{ij}})^2 \tag{1.2}$$

The majority of bounding boxes are empty. As a result, the model becomes unbalanced and in reality, is trained more frequently to recognize backgrounds than objects. To address this, the factor *l_noobj* already seen before, with a default value of 0.5, is added to the confidence loss for bounding boxes that do not contain objects. [7]

1.2.3 Comparison with Other Object Detection Models

To better understand what may be done to enhance the performance of the studied model, it is useful to examine different Object Detection methods in the pages that follow. This can be useful to determine which features of YOLO need to be improved in order for it to outperform all other models.

- Deformable Parts Model (DPM) DPM detect objects using a Sliding Window method. To extract static characteristics, classify regions, predict bounding boxes for high scoring regions, and perform other tasks, DPM employs a disjoint pipeline. In YOLO, a single Convolutional Neural Network takes the place of all of these various components. The network simultaneously executes contextual reasoning, bounding box prediction, Non-Maximal Suppression, and feature extraction. The algorithm trains the features in-line and makes them more effective for the detection task rather than using static features. The YOLO integrated design produces a model that is faster and more precise than DPM.

- R-CNN This algorithm is considered as one of YOLO's main competitors.
  Instead of using sliding windows to find items in photos, it uses region proposals. Potential bounding boxes are generated by Selective Search, features are extracted by a Convolutional Neural Network, the boxes are scored by an SVM, the bounding boxes are adjusted by a linear

model, and duplicate detections are eliminated by Non-Max Suppression. The outcome is an extremely slow system that takes more than 40 seconds per image during test time because each stage of this intricate pipeline must be carefully tuned independently. YOLO and R-CNN have some parallels. Potential bounding boxes are suggested for each grid cell, and those boxes are then scored using convolutional features. To counteract multiple detections of the same object, YOLO, however, places spatial constraints on the grid cell proposals. Additionally, YOLO suggests 98 bounding boxes per image as compared to around 2000 from Selective Search. YOLO technology then merges these various parts into a single, jointly optimized model.

- Other Fast Detectors Fast and Faster R-CNN They focus on accelerating the R-CNN architecture by sharing computing and substituting neural networks for selective search when proposing regions. Both fall short of real-time performance despite being faster and more accurate than R-CNN. The DPM pipeline's speed is the topic of numerous research projects. They leverage cascades, accelerate HOG computing, and direct processing to GPUs. Only 30Hz DPM, though, operates in real time. YOLO discards the entire detection pipeline instead of trying to optimize individual parts of it and is quick by design. Since they must deal with significantly less variation, single class detectors like those for faces or individuals can be greatly tuned. YOLO is a general-purpose detector that can simultaneously learn to detect a wide range of objects.

- Deep Multibox Instead of using Selective Search, a Convolutional Neural Network is used to anticipate regions of interest. By substituting a single class prediction for the confidence prediction, MultiBox is also capable of doing single object detection. Multi-Box is still only a component of a broader detection pipeline and is unable to detect general objects, necessitating additional picture patch categorization. Both YOLO and MultiBox predict bounding boxes in an image using a Neural Network, while YOLO is a full detection system.

- OverFeat Convolutional Neural Networks are used in this system to accomplish localization and then modify the localizer to carry out detection. Although OverFeat effectively detects sliding windows, the system is still fragmented. Not the performance of detection, but localization, is optimized by Over-Feat. The localizer, like DPM, only considers local data while producing a prediction. Due of OverFeat's inability to reason about global context, extensive post-processing is necessary to create coherent detections.

- MultiGrasp YOLO shares design elements with research on grip detection. The MultiGrasp method for regression to grasps serves as the foundation for the grid approach to bounding box prediction. But unlike object detection, grip detection is a lot easier task. For an image with a single object, MultiGrasp simply needs to anticipate a single graspable zone. Finding a region that is suitable for grabbing is all that is required; it need not estimate the object's size, location, or borders or forecast its class. For many objects belonging to various classes in an image, YOLO forecasts bounding boxes as well as class probabilities. [9]

Since YOLO can also detect objects in real-time, it's crucial to compare it to other real-time object detection algorithms by evaluating how well they perform on PASCAL VOC 2007. Examining the errors made by YOLO and Fast R-CNN, one of the most effective forms of R-CNN, on VOC 2007 is crucial to comprehending the distinctions between YOLO and R-CNN variants. Based on the different error profiles, it is possible to demonstrate that YOLO may be utilized to improve the performance of Fast R-CNN detections by rescoring them and reducing the mistakes from background false positives. On two datasets of artwork, it is possible to demonstrate that YOLO generalizes to new domains better than other detectors by presenting the results from VOC 2012 and contrasting them with state-of-the-art techniques. [9]

Comparison to Other Real-Time Systems Making standard detection pipelines quick is a major focus of object detection research. Only a few researchers have been successful in creating a real-time detection system, despite several attempts. It is particularly helpful to compare the performance of YOLO to the Deformable Parts Model system. There are some differences between YOLO and the GPU implementation of DPM, which operates at 30Hz or 100Hz. Some elements are obvious even though the other efforts fall short of the real-time benchmark when comparing their relative mAP and speed to analyze the accuracy-performance trade-offs accessible to object detection algorithms.

As far as we are aware, Fast YOLO is the quickest object detector currently in use. It is also the fastest object detection method on PASCAL. It is more than twice as accurate at 52,7% mAP as previous real-time detection attempts. YOLO increases real-time performance while increasing mAP to 63,4%. It is also used VGG-16 to train YOLO. This model is substantially slower than YOLO, but it is also more accurate. Although it is useful for comparison to other VGG-16-based detection systems, the remainder of the paper concentrates on faster models because it is slower than real-time. Fastest DPM effectively increases DPM without significant mAP loss, but real-time performance is still 2 times slower. Additionally, it is constrained by DPM's very poor detection accuracy in comparison to neural network methods. Static bounding box proposals are used instead of Selective Search in R-CNN. Even while it is far faster than R-CNN, it still lacks real-time capabilities and suffers greatly in accuracy due to the poor quality of its proposals. [9]

Fast R-CNN accelerates the classification phase of R-CNN, although it still relies on Selective Search, which may generate bounding box proposals after about two seconds per image. It has a high mAP as a result, but at 0,5 fps, it is still not real-time. The latest Faster R-CNN proposes bounding boxes instead of selective search. Their most accurate model performs at 7 fps in the tests, while a smaller, less accurate model performs at 18 fps. Faster R-CNN's VGG-16 variant is 10 mAP faster than YOLO but 6 times slower. Only 2.5 times as slow as YOLO, the Zeiler-Fergus Faster R-CNN is also less accurate. [9]

1.2.4 YOLO's Features

Error Performance Analysis

Redmon J. et al.[9] analyse the YOLO error performance, after completing the training phase on the VOC 2007 dataset and contrasting it to Fast R-CNN, one of the best models, with the best performance using PASCAL setups. The N best forecasts for each object category can be determined

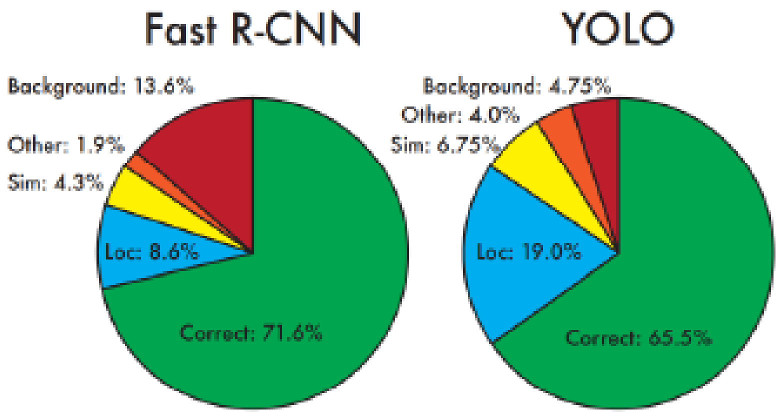


Figure 1.8: Errors analysis Fast R-CNN vs. YOLO [7]

during the testing process.

It is possible to determine whether or not a category is accurate, if an error exists, and what kind of error it is:

- Correct: the appropriate category, IOU higher than 0.5;
- Localization: the appropriate category, IOU between 0.1 and 0.5;
- Similar: a related class, IOU higher than 0.1;
- Other: the inappropriate class, IOU higher than 0.1;
- Background: IOU smaller than 0.1, independently from the category type. [7]

The distribution of various error types on a sample of 20 object classes is depicted in the following Figure 1.8.

It is instantly obvious from a comparison of the test results from YOLO and Fast R-CNN that YOLO has some localization issues. These errors are more prevalent in YOLO compared to all the others, whereas Fast R-CNN produces less errors overall but more background errors. Since 13.6% of Fast

R-CNN predictions are false-positives, no objects are present in the bounding boxes. Looking back at this comparison it is evident that Fast R-CNN tends to produce background forecasts three times more frequently than YOLO. [7]

In order to continue to consider the distinctions between these two approaches, we could compare the most recent release of YOLO, version 5, with the latest and quickest version of Fast R-CNN, known as Faster R-CNN. The performance of both models and their limits can be better understood from the table that is presented below. It is clear from their weaknesses alone that both have an issue with the presence of missing objects, those that the model is unable to recognize. The table demonstrates that performance for YOLO is good in all other aspects, while performance for the second model is not considered as reliable. As a result, YOLO proves to be superior in terms of the speed of inference, the detection of small or distant objects, and the issue of overlapping boxes. Finally, both models are capable of locating items in chaotic conditions, where it is more challenging to locate the target object due to the abundance of other objects.

Features	Yolo v5	Faster RCNN
Inference Speed	YES	NO
Detection of small or far away objects	YES	NO
Little to no overlapping boxes	YES	NO
Missed Objects	NO	NO
Detection of crowded objects	YES	YES

YOLO's Generalizability

The generalization of the data is unquestionably one of YOLO's best features, second only to its speed. The same model, trained on images of the

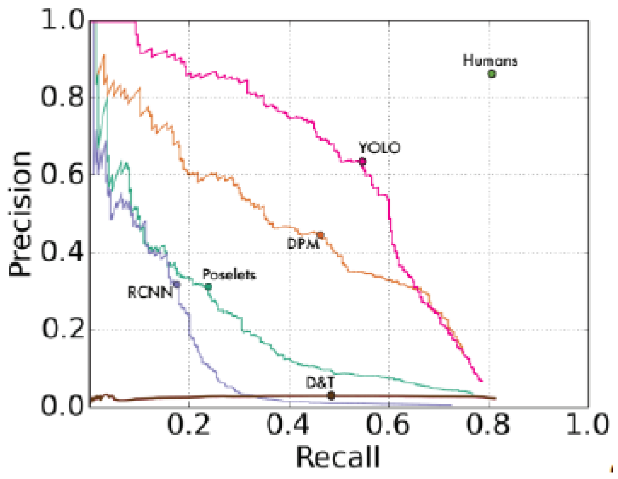


Figure 1.9: Generalization results on the Picasso dataset [7]

natural world, performs well even when used to categorize artistic images, portraits of paintings, and more generally, artificial images. The Picasso dataset and the People-Art Dataset, two datasets created to test person detection algorithms on artistic works, were used by the authors to compare YOLO's performance with that of other object detection systems in order to assess the generalizability of YOLO.

The comparison between YOLO's performance and those of other object detecting algorithms is shown in Figure 1.9. The VOC 2007 dataset was used to train the models for the Picasso dataset training phase, while the VOC 2010 dataset was used to test the models on the People-Art dataset. R-CNN performs exceptionally well on VOC 2007, but this performance significantly declines when the model is applied to creative imagery. [7]

Using a method designed for natural images, R-CNN uses Selective Search to find bounding box proposals. When applied to creative photos, DPM retains its accuracy well. Because DPM has a very good spatial model for identifying the shape and arrangement of objects, it is hypothesized that it functions well with aesthetic imagery. Although DPM does not lose accuracy when applied to fake images like R-CNN does, the authors of YOLO point out that it begins with less accuracy when evaluating natural photographs. [7]

Limitations of YOLO

Since each grid cell can only predict two boxes and only have one class, YOLO places substantial spatial limits on bounding box predictions. The number of neighbouring items that the model can predict is constrained by space. Small items that occur in groups, like flocks of birds, are difficult for the model to handle. The model finds it difficult to generalize to objects with novel or odd aspect ratios or configurations since it learns to predict bounding boxes from data.

Given that the architecture comprises many down sampling layers from the input image, the model also employs rather coarse features for predicting bounding boxes. Last but not least, while being trained on a loss function that roughly represents detection performance, the loss function treats errors in tiny bounding boxes and large bounding boxes equally. The impact of a tiny error in a small box on IOU is substantially greater than that of a small error in a large box. Inaccurate localizations are the primary cause of inaccuracy. [9]

Although YOLO performs well on VOC 2007, it doesn't lose accuracy as much when used on creative photos as the other models under consideration. As DPM, YOLO considers the objects' size, shape, relationship to one another, and locations where they appear more frequently. The authors also discuss how, when considering a small number of pixels, natural and generated images differ greatly, although, at the same time, forms and proportions within the same image are highly similar. Given its design, this contributes significantly to YOLO's ability to generalize object classes and produce effective results when used on simulated photos.[9]

Datasets' availability

The availability of a smaller number of images to use, compared to other types of machine learning algorithms, is another factor that must be considered while analyzing the constraints of this model. Since there are fewer images and datasets accessible for other Machine Learning applications, the majority of object detection algorithms are actually trained on the same collection of data and images. A few hundred photos make up the most well-known and often used dataset in the field of object detection. The datasets utilized for other classification or regression issues, on the other hand, contain hundreds of thousands of data. Since one can train these algorithms with a significantly bigger volume of data than object detection techniques, it is obvious that this results in better results and performance. [9]

1.2.5 Why YOLO?

Understanding if YOLO is the appropriate model to apply in the current scenario being studied is made much easier by all the strenghts and weaknesses of the model that have been so far described. Yolo's single pipeline, which has already been stated, is what makes it so fast. In addition, the generalisability component is very intriguing because it enables a more comprehensive understanding of the image. The employment of the Non-Maximum Suppression approach, which enables the removal of superfluous and unnecessary boxes, is the last factor to take into account. After considering all the features emphasized in this first chapter, it is important to determine whether Yolo should be used instead of a faster or more accurate model. The final choice, in my opinion, was not exclusively based on the mathematical conclusions that have been put forth thus far. Yolo's method of approaching an image, as already said before, is one of the things that really stood out to me about it, and it's also one of the key reasons why I chose to utilize this particular Neural Network for my project. What does this mean? The object detection problem is first considered as a regression problem rather than a classification one, which is an intriguing characteristic. The algorithm's ability to understand how the observed object interacts with the background and other objects in the image by considering the complete scenario as a whole is a second noteworthy innovation. Yolo can recognize objects much more quickly in videos and more precisely in images if we think that it pays attention to the entire context, which is in my opinion a very relevant component.

Because of all these aspects and how accurate and precise the model is as a result, YOLO was picked.

The issue of signature detection is now very widespread, as will be seen in the following chapter, therefore creative solutions are required to make a genuinely fascinating contribution to this field of study. YOLO hasn't yet been used to recognize signatures on identity documents or to classify and organize them, so it looks like an innovative and intriguing decision to take advantage of its capabilities in this area. YOLO's inability to recognize objects it has never seen before is one of its limitations, but in this case, it is obvious that this is not an issue because the only thing that needs to be recognized is the signature. As a result, there is no danger of YOLO encountering objects that are too small to be recognized.

The next chapter will highlight the goal of this project and the practical use of the algorithm after explaining and delving into how Yolo operates, as well as its advantages and disadvantages.

Chapter 2

The Process

2.1 The Project

Nowadays, the problem of detecting and validating signatures is regularly encountered, and there are numerous ways to fix it. This is the reason why this work may initially appear simple and straightforward to complete. However, it is a very intriguing process because there are many factors to take into consideration, considering that many things could have a bad impact on the results. The signatures' issue is usually quite delicate because they are typically written by hand and then scanned. Examples include how the page is laid out, where and how the signature is placed, whether there is any additional handwriting on the page, whether colored backdrops are used, and how saturated and clear the background is. Because of this, it is difficult to come up with a workable solution for this situation.

It is also essential to stress that the type of solution to be utilized must be carefully chosen depending on the results we need and the project's type of aim. Why does this matter? We must choose which method best meets our needs from the several techniques to identify and authenticate a signature.

The process of recognizing a signature came with several challenges. The algorithm is trained to recognize a handwritten sign that is typically found in the bottom left-hand corner, which presents a first barrier. What transpires then if there are additional handwritten signatures on the document or if the signature is not in the usual position? This was the first thing that was examined and, if necessary, corrected. It must be said that Yolo made this situation easier to handle. Yolo is quite good at recognizing only the signature when it comes to other handwritings, and it is also possible to specify a probability cutoff below which handwritings are not taken into account. It must be acknowledged that Yolo still makes mistakes with regard to the second challenge, the location of the signature, although this was solved by adding more pictures to the dataset. This made it possible to increase both the number of pictures used to train the algorithm and the number of pictures used to test it.

The design of the page is still another challenge. The method yields great results when detection is required to occur on vertical sheets with a white background and black writing. However, the model's performance noticeably degrades when pages are used with colored backgrounds and the page orientation is reversed. When the background is darker, Yolo has a very tough time recognizing signatures, especially when the color green is present. Due to this, as we shall see on the next pages, it was necessary to repeat the entire procedure in order to increase the algorithm's performance using a dataset of colored images arranged on horizontal pages. The outcomes of all efforts will be seen in the upcoming chapter.

2.2 The Purpose

The idea of detecting signatures came up while I was looking for a system that could classify papers, particularly identification documents, based on the presence or absence of a signature. First, it is important to clarify some project's aspects in order to reveal its innovative nature. The first point that needs to be emphasized is that this research and the solution it provided must be seen in the context of a broader discussion of the field of signature recognition. The signatures' validation is now the most crucial phase in signature verification efforts. As a result, finding their location on a page or document is just a preliminary task that is addressed using less time-consuming and effort-consuming methods and is consequently given less priority.

On the basis of this supposition, it is clear that the ultimate objective of this initiative is not the same as signature verification. Due to the fact that signature verification is a subsequent, and in this case superfluous, step that may always be added later, and more traditional Machine Learning algorithms won't be used.

A neural network is typically used in projects that attempt to verify a signature after it has been trained with a signature dataset to enable it to assess its legitimacy. The signature detection step is often solved using a Machine Learning algorithm, which determines if the signature is there or not based on the examination of the connected components on the document. Frequently, this step is just skipped, and the Neural Network is trained directly for verification using images of each signature that have already been extracted from the document.

The identification of the signature on a document is equally a crucial step in the process, particularly for companies that require it when classifying digital documents and was given full attention in this thesis topic. Due to this, it was decided to use a Neural Network also during the identification step in order to produce findings that were more accurate. Numerous tests were run, taking into account numerous examples and papers found online, in order to determine which algorithm was the best. A choice was done after carefully examining the outcomes found about a dozen images. First, all other potential Machine Learning algorithms that did not use Deep Learning were excluded and the search was limited to the field of Neural Networks. The most successful were then chosen, taking into account the four primary performance metrics: precision, accuracy, recall, and F1-score.

Finally, the Convolutional Neural Network "YOLO" (*You Only Look Once*) was selected. This is an Object Detection algorithm, as was already seen in the previous chapter. The usage of a Neural Network appeared reasonable because the goal is simply to identify an object (the signature) in a picture or scanned document. When used, YOLO should produce a dataset containing all the document's metadata (including the document's width and length, location, length, and width of the signature, as well as the existence and placement of the logo, along with a final column for specifying whether the document is "signed" or "unsigned").

2.3 The Process

In the next pages will be described all the steps that have been followed, in details.

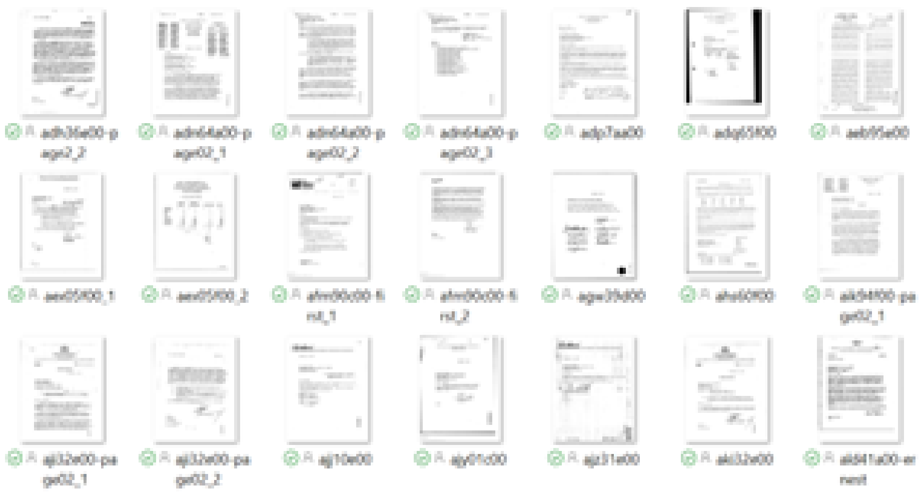


Figure 2.1: Tobacco800 dataset

Data Pre - Processing

I referenced the notebook on Github [10] during the entire process, which demonstrates how to utilize this technique to find signatures and logos on scanned pages.

Data Description Finding a dataset that was appropriate for the project was the first step. The Tobacco800 dataset consists of over 1,200 images of scanned documents in black and white, some of which are signed, some of which are not, some of which are misaligned, and some of which have additional pen writing. At least for the project's initial goal, it was appropriate. The metadata for each image is stored in *XML* files in a separate folder in addition to the one holding the photos. The height and width of the page, the presence or absence of signatures and logos, and, if any were included, their coordinates (x, y, width, and height) were all included in each file. This information was essential for the training process as well as for evaluating the model's effectiveness. In Figure 2.1 is shown a small part of the dataset.

After locating the input data, a data cleaning procedure was required to transform the images and text files into a format that the algorithm could understand without causing errors.

The first step was to add all of this data into a dataset with the following columns:

`filename`, `category`, `xmin`, `xmax`, `ymin`, `ymax`, `labels`, `width`, `height`, `xcenter`, `ycenter`, `width_norm`, `height_norm`, `xcenter_norm` and `ycenter_norm`.

At this point, it was possible to scale data. First, YOLO has an hight and a width that must be the same for all the images: max_height = 640 & max_width = 480. In the following lines of code, is shown the function that pass through each image, read the image and set the maximum height and width, reshaping the entire size of the images.

```python
def scale_image(df):
    df_new = []
    file_name = data['prev_filename']
    X, Y, W, H = map(int, data.x), map(int, data.y), map(int,
    data.width), map(int, data.height)
    for file, x, y, w, h in zip(file_name, X, Y, W, H):
        image_path = BASE_DIR + file
        img = cv2.imread(image_path) #return a numpy array
        page_height, page_width = img.shape[:2]
        max_height = 640
        max_width = 480
```

At this point, it is needed to compute again all the images' coordinates and measures, related to the position of signatures and logos on the image. How to do that? Creating a factor that must be multiplied with all the original coordinates, named the Scaling Factor. As shown in the lines of code below, the scaling factor is equal to (maximum height/height of the page) when the image's height is greater than its width, and it is equal to (maximum width/width of the page) when the image's width is less than the scaling factor.

```
 # computes the scaling factor
      if max_height < page_height or max_width < page_width
:
          scaling_factor = max_height / float(page_height)
          if max_width/float(page_width) < scaling_factor:
              scaling_factor = max_width / float(page_width
)
```

Now we can reshape all the page's measures using this factor: $height * scalingfactor$ and $width * scalingfactor$. The scaling factor is then multiplied by each of the values of x, y, w, and h to recalculate them. Finally, we generate a brand-new dataset containing all fresh values for the scaled images. Additionally, the "category" column, which has the options Logo or Signature, is changed to a numerical value: 0 for Logos and 1 for Signatures. This is the most important column, because it represents the value that Yolo will check when it has to determine if a document is signed or not.

```
  # scale the image with the scaling factor
          img = cv2.resize(img, None, fx=scaling_factor, fy
=scaling_factor, interpolation=cv2.INTER_AREA)
          #INTER_AREA: method that decides which pixel gets
 which value based on its neighboring pixels and the scale
 at which the image is being resized.
      jpg_filename = file[:-4] + '.jpg'
      new_file_path = SAVE_PATH + jpg_filename
      cv2.imwrite(new_file_path, img) # write the scales
image

      # save new page height and width
```

```
        page_height, page_width = page_height*scaling_factor, page_width*scaling_factor
        # compute new x, y, w, h coordinates after scaling
        x, y, w, h= int(x*scaling_factor), int(y*scaling_factor), int(w*scaling_factor), int(h*scaling_factor)
        row = [jpg_filename, x, y, w, h, page_height, page_width]
        df_new.append(row)
    return df_new
scaled_data = scale_image(data)
```

The images can now be converted into the YOLO-required format. How? The following lines of code provide four newly defined functions. Each one reshapes a different measure. X_center and y_center functions will compute the new coordinates for the signatures' and logos' central point. W_norm and h_norm functions will compute the normalized signatures' and logos' width and height.

```
    def x_center(df):
  return int(df.x_scaled + (df.w_scaled/2))
def y_center(df):
  return int(df.y_scaled + (df.h_scaled/2))

def w_norm(df, col):
  return df[col]/df['page_width_scaled']
def h_norm(df, col):
  return df[col]/df['page_height_scaled']
```

At this point, we have almost 1200 images reshaped with the measures needed by the Yolo. Another step is needed to prepare the dataset to the training phase. The data that will be used for training needed to be prepared after the pre-processing phase was finished. To train a Machine Learning model it is necessary to sort the data in two folders, one for the training and one for the test phase, this will ensure that the data used in that training won't be used again in the testing phase, having an effect of the results and the performance parameters. In this case, images are divided randomly by default with percentage of around 60% for the training and around 40% for the testing.

At this point, the situation is as follows:

- Folder with images, which contains “Valid” and “Train”
- Folder with labels, which contains “Valid” and “Train”

The training phase is now ready to begin.

2.3.1 Training

Another important step to ensure the realization of the process is the duplication of Ultralytics' GitHub repository [11], which contains everything needed to use YOLO, is the initial step in order to move forward with the training. Additionally, all *dependencies* and necessary libraries must be installed. As said before, this project will use an algorithm that already exists, we are only going to apply it to a specific case. The directories for the train and test set, that is, the earlier generated folders “Train” and “Val”, the number of classes to be identified, in this case two: signatures and logos, and the class names “Signature” and “Logo” must now all be added to a *.yaml* file. In this way the Neural Network will be able to find the right images to train using this approach. Selecting the model needed is required before training begin. It is possible to select the version of YOLO that best suits the project because, as was demonstrated in the previous chapter, several variations have been created over time. All of the models that were made accessible on Github were tested to see which one generated the best outcomes. The Yolov5s model, which is the fastest and seems to be the most accurate of all the versions available, was chosen. Now it is possible to go on with the training phase. The following line of code shows what the Neural Network needs to start the process. First, the directory for the training python code in the Yolo repository. Secondly, the image height and the number of batches. Third, number of epochs. Then, the directory to the *.yaml* file. Finally, the directory for the chosen version of the model, and the name of the *.yaml* file.

```
!python /content/yolov5/train.py --img 640 --batch 8 --epochs
    100 --data /content/bcc.yaml --cfg /content/yolov5/models
   /yolov5s.yaml --name BCCM
```

The choice of 100 epochs was made since, when a higher number was tried, the findings remained the same and, when the number was increased even further, there was a risk of *overfitting*. Additionally, the 100-epoch results were satisfactory, so there was no need to alter them.

It was required to select the appropriate model version before going on to examine the results in more detail. There are different variations since accuracy and precision have been increased through numerous adjustments over time. Due to this, a test was run throughout both the training and testing phases to determine which model produced the best outcomes.

Every YOLO variant has a unique set of qualities and yields a unique set of outcomes, as seen in the figure above. Yolov5s would seem to be the fastest, while Yolov5x would seem to be the most accurate. It is best to confirm with the available data which version is truly the most suitable to ensure that they are the versions most appropriate for the project's objective. The

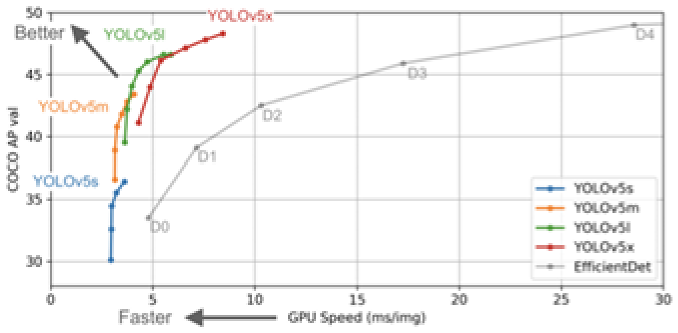


Figure 2.2: Yolo Models performances [13]

four previously specified performance metrics for this model were computed, and they will be covered in more detail on the next pages. You can see how the same process was used for every variation of the model's version 5 in the accompanying figures. Although the performances in the tables are very similar, the model with the best performance was chosen.

The first attempt has been done with the Yolov5x model, because it is slower, but it seems to be also the more accurate.

The results for train and test are the same and are shown in the following table:

Yolov5x	Accuracy	Precision	Recall	F1-score
Signature	0,79	0,61	0,99	0,76
Logo	0,84	0,71	0,90	0,79

After looking at the confusion matrix results, it was possible to compute the four-performance metrics. Accuracy is quite good with 0,79 for signatures and 0,84 for logos. Precision is lower, with a score of 0,61 for signatures and 0,71 for logos. Recall is very high, with a score of 0,99 for signatures and 0,90 for logos. Finally, the signatures' F1 score is 0,76 and logos' score is 0,79.

The first evident aspect is that the performance in detecting logos is higher that signatures. And this is another aspect that must be checked, because we are interested only in detecting signatures, not logos.

At this point we can go on with the second attempt: Yolov5m. This model has been analysed because, as shown in Figure 2.2, it has a good accuracy, even if it is lower than the previous model, but it also keeps a good speed in detecting objects, even if also in this case it is slower than the previous one.

For the train:

Yolov5x	Accuracy	Precision	Recall	F1-score
Signature	0,79	0,60	1	0,76
Logo	0,85	0,71	0,92	0,80

For the test:

Yolov5x	Accuracy	Precision	Recall	F1-score
Signature	0,79	0,60	0,84	0,75
Logo	0,85	0,70	0,92	0,80

Also, in this case it is possible to compare the results between the training and the testing phase. As for the previous model, the results are better for the logos' detection than for signatures' one. Another important aspect is that there is not a big difference between training and testing, there is only a slight decrease of logos' precision, going from 0,71 to 0,70; and another small decrease in F1 signature score that goes from 0,76 to 0,75. Comparing these results to the previous one, it is possible to say that the best model between these two is Yolov5x in detecting signatures.

The final test is done with the Yolov5s model. This version of Yolo is the fastest, but it is also not as good as the others in detecting objects. As shown in the following tables, we have slight differences between training and testing phases.

For the train:

Yolov5x	Accuracy	Precision	Recall	F1-score
Signature	0,74	0,60	0,94	0,70
Logo	0,92	0,69	0,92	0,75

For the test:

Yolov5x	Accuracy	Precision	Recall	F1-score
Signature	0,79	0,62	1	0,76
Logo	0,85	0,70	0,92	0,80

Looking at signatures' detection performance, we can say that there is an increase in all the metrics, and this is a good sign, meaning that the algorithm is trained in an efficient way, and it shows good results in the test. The same happens also for the detection of logos. Comparing these results also with the previous models, it seems to be the one with best performance. For this reason, model 5S ("small") was chosen.

Now that the model has been chosen, it is possible to go on with the training phase, using the line of code previously explained. The results of this phase can be visualize in two different ways, through the Tensorboard or by taking a look at the images of the outcomes. First, the confusion matrix, through which precision, accuracy, F1-score, and recall are determined. Overall, the results are excellent, but they will be examined in further detail in the next chapter.

2.3.2 Performance Measures

The confusion matrix, as shown in Figure 2.3, is especially useful when measuring recall, precision, F1-score, accuracy, and the AUC of a classification model.

- True Positives (TP): The model predicted positive, and the actual label is positive.
- True Negative (TN): The model predicted negative, and the actual label is negative.

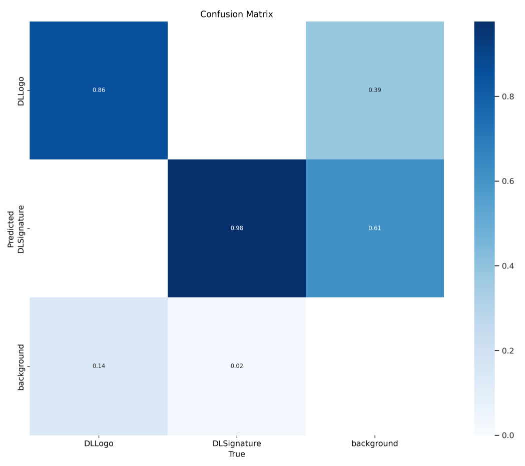


Figure 2.3: An Example of Confusion Matrix

- False Positive (FP): The model predicted positive, and the actual label was negative.
- False Negative (FN): The model predicted negative, and the actual label was positive.

In the multi-class classification problem, there won't be TP, TN, FP, FN values directly as in the binary classification problem. How to find TP, TN, FP and FN for each individual class?

- Accuracy $= TP + TN/TP + TN + FP + FN$
- Precision $= TP/TP + FP$
- Recall $= TP/TP + FN$
- F1-score $= 2TP/2TP + FP + FN$

These are the formulas that have been used to compute the four measures; observing each of them it is possible to analyse the general performance of the model.

Accuracy For tasks requiring balanced classification, accuracy is a useful statistic. When all classes are represented in comparably equal numbers, a classification task is balanced. Examining an illustration of an unbalanced classification task makes it clearer why this is the case. AI accuracy is the ratio of correctly classified data that a trained Machine Learning model produces, or the proportion of correct predictions to all other forecasts combined. It is frequently referred to as ACC. The fact that this metric directly correlates with all values of the confusion matrix is a nice feature of it. These four criteria—true positives, false positives, true negatives, and false negatives—form the foundation of Machine Learning evaluation. Model accuracy is a crucial parameter since it is a very straightforward measure of model performance, as was previously said. It is also a straightforward metric of model error. Accuracy can actually be thought of as $1 - error$.

Accuracy is a highly efficient and useful indicator to assess Machine Learning prediction accuracy in both of its forms. It is one of the metrics that is most frequently employed in research, where it is typical to have clean and balanced datasets to allow for attention on improvements in the algorithmic approach. When datasets with comparable properties are available, accuracy can also be helpful in practical applications. Model accuracy can be easily correlated with a range of financial measures, such as revenue and cost, thanks to its straightforward interpretation. This ease of reporting on the model's worth to all stakeholders increases the likelihood that an ML endeavor will be successful. For tasks requiring balanced classification, accuracy is a useful statistic. When all classes are represented in roughly equal numbers, a classification task is balanced. [1]

F1 - Score A Machine Learning statistic that can be applied in classification models is the F1 score. Using more accurate measures, like as the F1 score, which consider not only the quantity of prediction errors your model generates but also the nature of those errors, can help address issues with class imbalance. The harmonic mean of recall and precision is used to calculate the F1 score. Given that the F1 score is the average of Precision and Recall, Precision and Recall are given equal weight in this score:

- If a model's Precision and Recall are both high, it will receive a high F1 score.
- If Precision and Recall are both poor, the F1 score for the model will be low.
- If one of Precision and Recall is low and the other is high, the model will receive a medium F1 score.

A brief explaination of F1-score:

F1 score	Interpretation
0.9 - 1	Very Good
0.8 - 0.9	Good
0.5 - 0.8	Ok
0 - 0.5	Not Good

Precision A model with more accuracy produces fewer false positives when making predictions, which leads to more accurate outcomes overall. If you want to reduce false positives, precision is a wonderful area to concentrate on.

Precision – Recall Curve By calculating and displaying the accuracy versus the recall for a single classifier at various thresholds, the precision-recall curve is created.

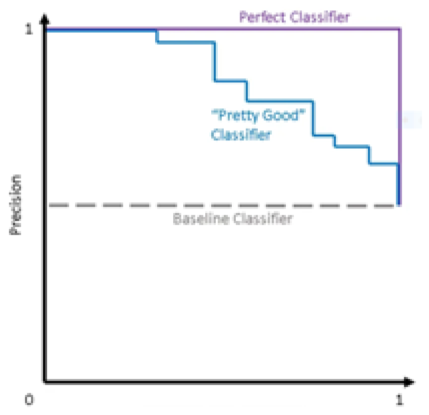


Figure 2.4: The optimal Precision-Recall Curve

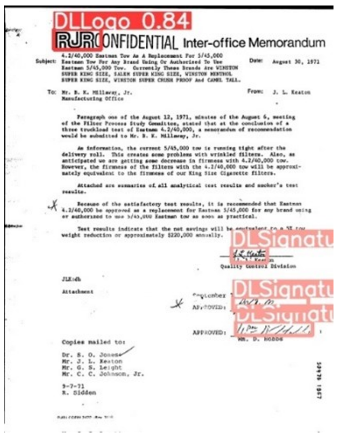
Recall We may look at binary classification to better grasp recall, a metric used for classification. The percentage of data samples that a Machine Learning model properly classifies as belonging to an interest class—the "positive class"—out of all the samples for that class is known as recall, also known as the True Positive Rate (TPR). The predicted labels may accurately or incorrectly classify the actual labels. A confusion matrix can be used to condense this data. Information about true negatives (TN), false negatives (FN), false positives (FP), and true positives (TP) is reported in the confusion matrix. On top of these numbers, Machine Learning recall is computed by dividing the true positives (TP) by all of the positive predictions that should have been made (TP + FN). When attempting to determine "What percentage of positive classifications was identified correctly?" using Machine Learning, recall should be used. When minimizing false negatives is crucial, this metric should be used. This frequently occurs when the opportunity cost of passing up a true positive is large and the cost of acting on a false positive is low. When the use cases are uneven, this frequently occurs.

2.3.3 Inference

Once the results from the training phase are observed and we are satisfied about it, it is possible to go on with the testing phase. This is the step in which the algorithm will pass through new image, that he never saw before. This is a fundamental step, because it is the moment in which we can understand if our model is capable of detecting objects in an image by himself, using all the information he learned during the training phase. Prior to anything else, the trained model must be saved as a *file.pt*. The directory to this file will be inserted, as shown in the following line of code, in the function for testing. Here, we need to insert the Yolo directory for the detection Python code and the directory for the "Val" folder described before, that contains all the image for the testing phase. The function also needs the weights, represented by the *file.pt* saved after the training, that represent the trained model with the best weights; and finally, a command to save a file text that contains all the predicted coordinates for each image.

```
!python Yolov5/detect.py --source Bank_Checks_Database/
    New_format/images/Val --weights Risultati/Checks/Train/
    best.pt --save-txt
```

The outputs are formatted as `class`, `x_center_norm`, `y_center_norm`, `width_norm`, `height_norm` in the text file. The outcomes, which are immediately saved in a different folder, can then be seen. In order to understand how the algorithm has discovered a signature and with what degree of certainty, it will be possible to go back and look at specific images. Alternately, you can go and look at the results of the entire procedure to comprehend the model's performance. Below there is an example of the outcomes in the form of images.


DLLogo 0.84
DLSignatu
DLSignatu
DLSignatu

RJR CONFIDENTIAL Inter-office Memorandum

Subject: 4.2/40,000 Eastman Tow As A Replacement For 5/45,000 Eastman Tow For Any Brand Using Or Authorized To Use Eastman 5/45,000 Tow. Currently These Brands Are WINSTON SUPER KING SIZE, SALEM SUPER KING SIZE, WINSTON MENTHOL SUPER KING SIZE, WINSTON SUPER CRUSH PROOF And CAMEL TALL.

Date: August 30, 1971

To: Mr. B. K. Millaway, Jr. Manufacturing Office

From: J. L. Keaton

Paragraph one of the August 12, 1971, minutes of the August 6, meeting of the Filter Process Study Committee, stated that at the conclusion of a three truckload test of Eastman 4.2/40,000, a memorandum of recommendation would be submitted to Mr. B. K. Millaway, Jr.

As information, the current 5/45,000 tow is running tight after the delivery roll. This creates some problems with wrinkled filters. Also, as anticipated we are getting some decrease in firmness with 4.2/40,000 tow. However, the firmness of the filters with the 4.2/40,000 tow will be approximately equivalent to the firmness of our King Size Cigarette filters.

Attached are summaries of all analytical test results and smoker's test results.

Because of the satisfactory test results, it is recommended that Eastman 4.2/40,000 be approved as a replacement for Eastman 5/45,000 for any brand using or authorized to use 5/45,000 Eastman tow as soon as practical.

Test results indicate that the net savings will be equivalent to a 5% tow weight reduction or approximately $220,000 annually.

[illegible]
J. L. Keaton
Quality Control Division

JLK:dh

Attachment

September
APPROVED: [illegible]

APPROVED: [illegible]
WM. D. HOBBS

Copies mailed to:

Dr. S. O. Jones
Mr. J. L. Keaton
Mr. G. S. Leight
Mr. C. C. Johnson, Jr.

9-7-71
R. Sidden

501781957

Figure 2.5: Yolo result on a single image

This figure shows that Yolo only recognizes signatures and logos. Along with detecting them, it also displays the percentage of accuracy, that means how certain it is that a specific object is indeed a signature or a logo. The creators of Yolo have incorporated a threshold that will be used to determine what value Yolo will base its identifications on. Any object recognized by Yolo with an accuracy of less than 0.25 will not be taken into account because the threshold is by default set at 0.25. This threshold, which is still set at 0,25, can be changed before moving on to the inference step. Yolo also creates images called `batch_labels` in which it juxtaposes multiple images, similar to Figure 2.6, to get a complete view of the output, in addition to the images related to the results that we will see in the next chapter, such as the confusion matrix. This eliminates the need to go and open each file one at a time.

It was possible to enter the findings into a single Pandas data frame once the results were in text file format, from which we could extrapolate

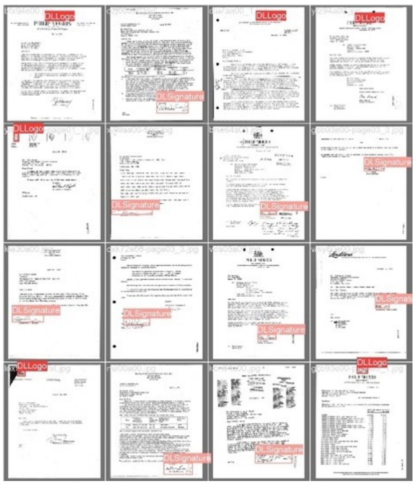


Figure 2.6: Yolo results on images juxtaposed

the number of values that were successfully predicted and those that were inaccurate. When the code is successfully executed, we can see the output is saved in the inference folder. At the end of this process, we will have two different data frames, one with the actual values and the second one with the predicted ones. Using a function to produce an inner joint between the two data frames, it will create a final unique dataset that contains only the values that has been correctly predicted. Thanks to this final Excel file, it will be possible to categorize each document as "Signed" or "Unsigned" after reviewing the results, adding another column at the end called "Result".

2.3.4 Hyperparameters Analysis

YOLO is characterized by the presence of several parameters needed to enhance its performance, as with all Machine Learning models. An ideal selection of these parameters could mean the difference between a successful Deep Learning model and an unsuccessful endeavor. The main hyperparameters of this model will be presented and described in the lines that follow, and all the efforts made during this project to try to increase YOLO's performance in spotting signatures will subsequently be demonstrated.

The hyperparameters that have been analysed are:

- lr0: 0.01 , it is the initial learning rate the indicates how often the neural network refreshes the notions it has learned.
- lrf: 0.01, final OneCycleLR learning rate (lr0 * lrf)
- momentum: 0.937, the SGD momentum/Adam beta1 is the tuning parameter for the gradient descent algorithm; its work is to replace the gradient with an aggregate of gradient.
- weight_decay: 0,0005, the optimizer weight decay $5e - 4$ is a regularization technique, that works by adding a penalty term to the cost function of a network, which has the effect of compressing the weights during the backpropagation process.
- warmup_epochs: 3.0, the warmup epochs (fractions ok) is a phase in the beginning of the neural network training where it starts with a learning rate much smaller, than the "initial" learning rate and then increase it over a few epochs until it reaches that "initial" learning rate.
- warmup_momentum: 0.8, warmup initial momentum
- warmup_bias_lr: 0.1, warmup initial bias lr
- box: 0.05, the box loss gain is a loss that measures how "tight" the predicted bounding boxes are to the ground truth object.
- cls: 0.5, the cls loss gain is a loss that measures the correctness of the classification of each predicted bounding box.

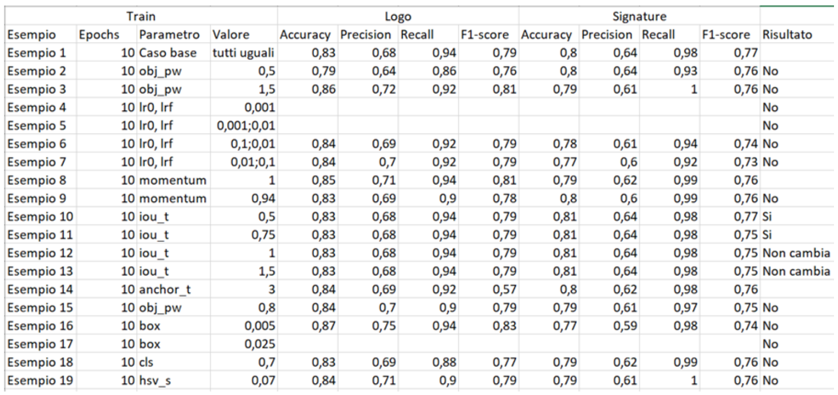
- cls_pw: 1.0, cls BCELoss positive_weight
- obj: 1.0, obj loss gain (scale with pixels)
- obj_pw: 1.0, is the obj BCELoss positive_weight; Increasing it, increases Recall and reduce Precision (more objects are detected).
- iou_t: 0.20, the IoU training threshold is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground truth bounding box. An IoU of 0 means that there is no overlap between the boxes. An IoU of 1 means that they are completely overlapping.
- anchor_t: 4.0, anchor-multiple threshold
- anchors: 3, anchors per output layer (0 to ignore)
- fl_gamma: 0.0, focal loss gamma (efficientDet default gamma=1.5)
- hsv_h: 0.015, image HSV-Hue augmentation (fraction)
- hsv_s: 0.7, image HSV-Saturation augmentation (fraction)
- hsv_v: 0.4, image HSV-Value augmentation (fraction)
- degrees: 0.0, the image rotation (+/- deg) is used to increasing model accuracy by randomly rotating images up to 360 degrees in whole training data.
- translate: 0.1, image translation (+/- fraction)
- scale: 0.5, the image scale (+/- gain) is used to resize an image either to match with grid size or for optimization of results.
- shear: 0.0, image shear (+/- deg)

- perspective: 0.0, image perspective (+/- fraction), range 0-0.001
- flipud: 0.0, the image flip up-down (probability) is used to flip images up and down randomly from the complete dataset for achieving better results.
- fliplr: 0.5, the image flip left-right (probability) is used to flip images left and right randomly from the complete dataset for achieving better results.
- mosaic: 1.0, the image mosaic (probability) is used to increase model accuracy by creating a new image from a combination of multiple images, and then using newly created images for training.
- mixup: 0.0, image mixup (probability)
- copy_paste: 0.0, segment copy-paste (probability)

To determine how each of them affected the model's result, each was examined separately. The following provides a general idea of the more than twenty trials that were conducted. It was chosen to employ a zero case, leaving the original parameter values unaltered, with 10 epochs, in order to reduce the amount of time used when running these tests. To observe the effects of each adjustment on these values, 10 epochs were used in all following testing.

Numerous attempts were made, as can be seen, and the best and most disappointing outcomes will be revealed in the following few lines. The first example is the zero case, as can be seen, where no parameters were altered, and the outcomes are as follows. As for logos:

- Accuracy: 0,83



Train				Logo				Signature				
Esempio	Epochs	Parametro	Valore	Accuracy	Precision	Recall	F1-score	Accuracy	Precision	Recall	F1-score	Risultato
Esempio 1	10	Caso base	tutti uguali	0,83	0,68	0,94	0,79	0,8	0,64	0,98	0,77	
Esempio 2	10	obj_pw	0,5	0,79	0,64	0,86	0,76	0,8	0,64	0,93	0,76	No
Esempio 3	10	obj_pw	1,5	0,86	0,72	0,92	0,81	0,79	0,61	1	0,76	No
Esempio 4	10	lr0, lrf	0,001									No
Esempio 5	10	lr0, lrf	0,001;0,01									No
Esempio 6	10	lr0, lrf	0,1;0,01	0,84	0,69	0,92	0,79	0,78	0,61	0,94	0,74	No
Esempio 7	10	lr0, lrf	0,01;0,1	0,84	0,7	0,92	0,79	0,77	0,6	0,92	0,73	No
Esempio 8	10	momentum	1	0,85	0,71	0,94	0,81	0,79	0,62	0,99	0,76	
Esempio 9	10	momentum	0,94	0,83	0,69	0,9	0,78	0,8	0,6	0,99	0,76	No
Esempio 10	10	iou_t	0,5	0,83	0,68	0,94	0,79	0,81	0,64	0,98	0,77	Si
Esempio 11	10	iou_t	0,75	0,83	0,68	0,94	0,79	0,81	0,64	0,98	0,75	Si
Esempio 12	10	iou_t	1	0,83	0,68	0,94	0,79	0,81	0,64	0,98	0,75	Non cambia
Esempio 13	10	iou_t	1,5	0,83	0,68	0,94	0,79	0,81	0,64	0,98	0,75	Non cambia
Esempio 14	10	anchor_t	3	0,84	0,69	0,92	0,57	0,8	0,62	0,98	0,76	
Esempio 15	10	obj_pw	0,8	0,84	0,7	0,9	0,79	0,79	0,61	0,97	0,75	No
Esempio 16	10	box	0,005	0,87	0,75	0,94	0,83	0,77	0,59	0,98	0,74	No
Esempio 17	10	box	0,025									No
Esempio 18	10	cls	0,7	0,83	0,69	0,88	0,77	0,79	0,62	0,99	0,76	No
Esempio 19	10	hsv_s	0,07	0,84	0,71	0,9	0,79	0,79	0,61	1	0,76	No

Figure 2.7: Results for the hyperparameters analysis

- Precision: 0,68
- Recall: 0,94
- F1: 0,79

And signatures:

- Accuracy: 0,80
- Precision: 0,64
- Recall: 0,98
- F1: 0,77

The values are not significantly different from those obtained with 100 epochs, as demonstrated in the following chapter. It is obvious that all attempts were not conducted at random but rather in accordance with the reasoning and descriptions provided for each parameter in order to increase Precision and Accuracy. It should be noted that none of these tests were found to be ineffective; in fact, the four error measurements were never lower than 0.59. Despite this, it is obvious that the best-performing cases were chosen.

The example with the box parameter of 0.005 is one of the most disappointing situations. Here are the outcomes. As for logos:

- Accuracy: 0,87
- Precision: 0,75
- Recall: 0,94
- F1: 0,83

And signatures:

- Accuracy: 0,77
- Precision: 0,59
- Recall: 0,98
- F1: 0,74

As can be seen, despite being one of the least successful attempts, the outcomes are still excellent. Also keep in mind that the identification of signatures, not logos, is the main focus of this study. I decided to concentrate on the outcomes of the parameters that relate to signatures because of this. As we can see, the results for logos are excellent, but this case study was put on hold because the results for signatures were substantially lower.

The test run with the values *lr0* and *lrf* of 0.1 and 0.01 respectively is another illustration of the "worse" tests. In this instance, the signatures' outcomes were as follows:

- Accuracy: 0,78
- Precision: 0,61
- Recall: 0,94
- F1: 0,74

Although the results were overall positive, they are again rather disappointing; nonetheless, this scenario was also ignored because we know the model is capable of delivering better outcomes. Additionally, more successful endeavors were made. These situations only slightly improved the four performance metrics, with an increase of around 0.1. Two instances saw the same outcome, raising the iou_t to 1 and 1.5.

On the other hand, improvements were seen when the iou_t was increased to 0.5 and 0.75. The outcomes in the first situation are:

- Accuracy: 0,81
- Precision: 0,64
- Recall: 0,98
- F1: 0,77

In the second case are:

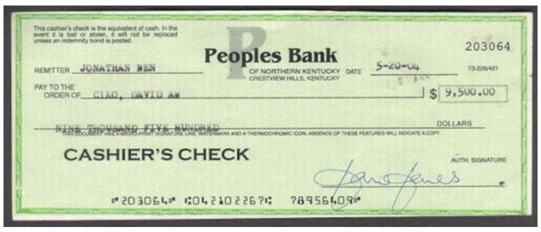
- Accuracy: 0,81
- Precision: 0,64
- Recall: 0,98
- F1: 0,75

When comparing the two, the F1 score is slightly different. For this reason, it was chosen to maintain this parameter's value of 0.5 during the entire operation. This was obviously first and foremost a theoretical investigation to fully comprehend the function of each parameter in the model. After thorough study, it was feasible to go on to comprehending how to apply them to boost performance. Although these are only minor upgrades, they can result in more precise outcomes when employed with datasets that are far larger than the ones being utilized right now, as was previously indicated.

2.3.5 Bank Cheques Dataset

It is vital to focus on a second stage of the process before going on to the following chapter, which will display all of the project's results. As was stated at the beginning of the chapter, when the procedure had been tested on the Tobacco800 dataset, it was important to determine whether Yolo could also recognize signatures on less "conventional" sheets and documents. It was necessary to hunt for another dataset with horizontally colored pages as a result. The project was excellently suited to the "Bank Checks Accounts" dataset that was discovered on Kaggle [3].

The figure below shows an example of the images contained in this dataset.



This cashier's check is the equivalent of cash. In the event it is lost or stolen, it will not be replaced unless an indemnity bond is posted.

203064

Peoples Bank
OF NORTHERN KENTUCKY
CRESTVIEW HILLS, KENTUCKY

REMITTER JONATHAN BEN

DATE 5-20-04

73-226/421

PAY TO THE ORDER OF CIAO, DAVID AW

$ 9,500.00

NINE THOUSAND FIVE HUNDRED DOLLARS

THIS DOCUMENT HAS A MICRO-PRINT SIGNATURE LINE, WATERMARK AND A THERMOCHROMIC ICON. ABSENCE OF THESE FEATURES WILL INDICATE A COPY

CASHIER'S CHECK

AUTH. SIGNATURE

[illegible]

⑈203064⑈ ⑆042102267⑆ 78956409⑈

Figure 2.8: Example of the bank cheques analysed

An early attempt was done to see how much poorer the performance was in terms of accuracy, precision, recall, and F1 score without first training the model on these images. The performance was not as good when utilizing the model trained on the Tobacco800 dataset without going through the training phase. With this new dataset, performance lost about 0.2 accuracy compared to the Tobacco800, making it about 0.6 instead. The performance did decline, but it did not fall to extremely low levels, even though the new configuration had not made it more loose. This is why I believed that the results will get better after the training phase was completed. With this new dataset, a second attempt was made, and all the previously described steps were repeated, but the results, as we will see in the next chapter, were still unsatisfactory, even if they increased a bit. The dataset, which had 167 photos and was therefore insufficient to train a model, was the underlying issue. In contrast to the training with Tobacco800, where a total of about 800 images were used, this first attempt's training phase only used 80 images. Since there were obviously missing data, additional images had to be manually added to the dataset to make up for the problem. The dataset has now grown to about 500 photos, and by replicating them, it can be doubled in the future.

The metadata of this dataset pose another issue. Since Tobacco800 is a dataset created specifically for this kind of task, each image in the dataset comes with an accompanying *XML* file that contains all of the image's metadata. This kind of data is not present in the current Checks dataset, but it is crucial since Yolo uses it to generate the findings and create the confusion matrix and all other performance measurements by comparing predicted and actual data. Due to this, it was required to manually enter all of the coordinates and page measurements into text files, one for each image.

It was feasible to repeat the entire process after these data processing procedures were finished. All of the outcomes will be revealed in the following chapter.

Chapter 3

Results

The outcomes of the project are described in this chapter. These will demonstrate whether employing YOLO is an original and practical method for accelerating and improving signature detection accuracy. As was already discussed in the previous chapter, YOLO was put to the test on two separate datasets to see how well it could recognize signatures on various types of documents. Due to this, the chapter is split into four sections: the first two for the training of both datasets and the last two for testing each one. The first images are from Tobacco800, which was quite helpful because, as was already indicated, it is a dataset made specifically for these kind of projects. With the help of these pictures, the initial attempts were run, and they gave us a practical understanding of how the algorithm works. Additionally, the full study of hyperparameters was done with these images. Given that the data and all the information required for the project are already present and that there are a lot of images—roughly 1200—that are ideal for training a Neural Network, it is evident that the dataset is very useful. Because it was suitable for this job, we were able to thoroughly understand all of YOLO's properties without running the danger of making an error of data uncertainty.

A folder of bank checks make up the second dataset. It could be argued that the issue with these images was exactly the opposite of Tobacco800. In fact, it is a dataset with roughly 160 total pictures that was hard to find on Kaggle [3]. Additionally, not all of the metadata required to evaluate the model's performance is offered. Therefore, it is necessary to expand the amount of images available as a first step. How? By copying the current images and altering their colors, position, and shapes to distinguish them from one another. The dataset now contains roughly 540 photos, however a further 500 images might be added. The metadata must to be added at this point as well; each image needs a text file that contains six values, including the height and width of the page, the coordinates of the point at the top left of the rectangle containing the signature, and the height and width of the rectangle itself. Once this procedure is done, manually, it is simple to repeat every step also applied to the first dataset.

Training - Tobacco800 Dataset The training phase is essential for the model to obtain good performance, as described in the previous chapter. It is the stage where YOLO learns the "game's rules". Both types of images undergo this period. The procedure is slower and takes longer with Tobacco800. Numerous tests have been tried because this is the first dataset used. The training set is made up of about 800 documents on vertical sheets, each with a white backdrop and black writing. The model's output compares actual and predicted values, and based on this comparison, numerical and graphical numbers that reflect the model's overall performance are generated. The Confusion Matrix, Recall, Score F1, and Precision are some measurements that are frequently employed for this purpose, as demonstrated in the previ-

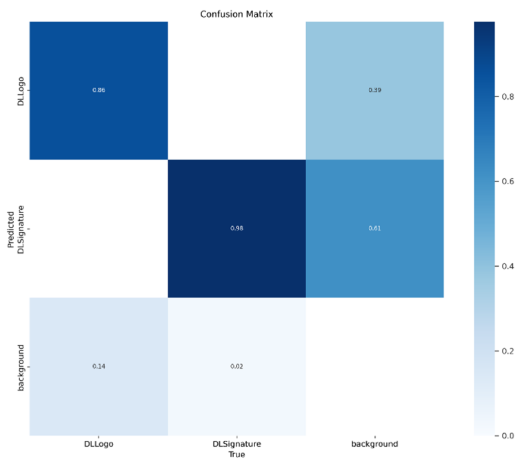


Figure 3.1: Confusion Matrix for Tobacco800 training

ous chapter. We'll now examine each of them separately.

The methods seen in the previous chapter are used to determine Accuracy, Precision, Recall, and Score F1 using the Confusion Matrix. Figure 3.1 illustrates how the model produces a matrix with three entries: DDLogo, DDSingature, and Background. The model is trained for the logo and signature detection; the third item is added automatically because all object identification methods consider also the image's background.

Using the equations from the previous chapter, performance metrics may be generated from this matrix. Consequently, the outcomes of signature detection will be:

- Accuracy: 0,79
- Precision: 0,61
- Recall: 0,98
- F1: 0,76

Using the information from the previous chapter as a guide, it is feasible to say that the degree of accuracy is quite high. The maximum value of accuracy is 1, which denotes the number of accurate predictions the model can make. On the other hand, Precision has a lower value, which, as already indicated, needs to be kept in check in order to reduce the frequency of false positives. It is a quality that can be improved. The third item describes the Recall value. This is one of the most crucial indicators in a Deep Learning model, as we all know, and in this instance, it has a high value. The question that needs to be posed in this situation is what percentage of the values have been accurately identified. By using this indicator, the likelihood of false negatives is reduced. The only thing to keep in mind is that a sufficient amount of each category to be classify, in this case logos and signatures, is required in order to use Recall effectively. The F1 score, which is derived from the Precision and Recall measures, serves as a final indicator of the relationship between these two metrics. The score will decline if either one is low, and the model's overall performance won't be good as a result. With a score of 0.76, we can state that the outcome is acceptable by the standards described in the previous chapter. Looking at the values of Accuracy and Recall, we can say that Recall is quite high while Accuracy can be improved. As a result, the model does a good job overall at identifying signatures.

Even if logos performance analysis is available too, since we have already talked extensively about this before and since the classification of logos is not the core of our project, the description and explanation in this case will not be as detailed as for signatures. Therefore, the model's performance in the specific case of logo identification is as follows:

- Accuracy: 0,82

- Precision: 0,69
- Recall: 0,86
- F1: 0,76

The F1 score is constant, the value of Recall decreases, while the values of Accuracy and Precision rose when compared to the results obtained using the signatures. Therefore, we can say that the model recognizes logos more readily than signatures. This is not surprising because logos are merely small symbols, whereas signatures occasionally fail to be recognized and can be confused with other writing.

The F1-Confidence Curve, depicted in the following graphic, is the model's second output.

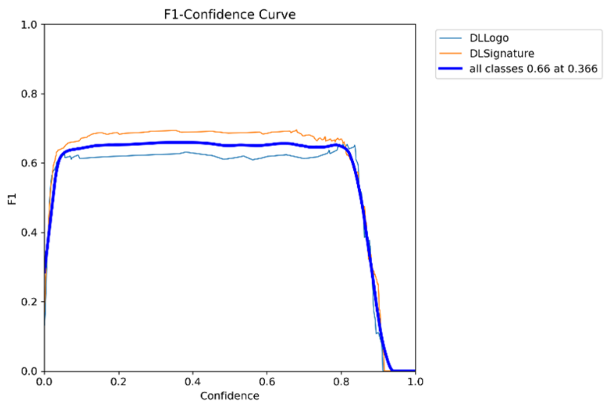


Figure 3.2: F1 - Confidence Curve for Tobacco800 training

There will be a more complete description of the model's performance depending on how the next curves perform. The F1 score represents ideal confidence level that balances the precision and recall values for a given model. We may say that the link between the confidence level and these

indicators is that, normally, as the confidence level rises, precision rises and recall falls. It is feasible to find the point of balance between precision and recall and choose it using the F1 score curve.
The confidence value that maximizes precision and recall based on the F1 curve is 0.366. A higher confidence value is preferred in many situations. Given that the F1 value for this model seems to be about 0.66, using a confidence level of 0.8 could be the best option. The precision and recall numbers at a confidence level of 0.8 further support the idea that this would be a good design point. The precision number is still nearly at its maximum value and the recall value starts to degrade around 0.8.
Now we are able to observe how the Precision-Confidence and Precision-Recall curves behave. Considering the first case, it is shown in the next Figure 3.3:

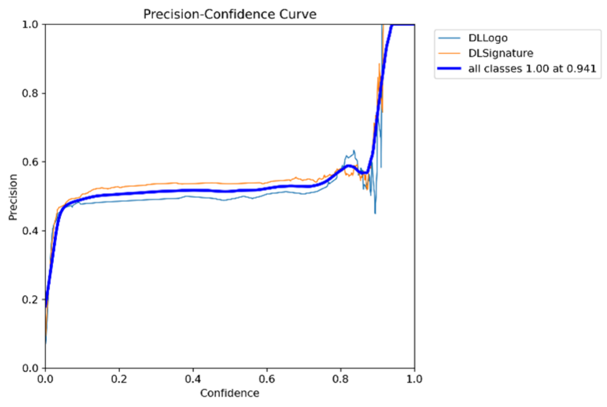


Figure 3.3: Precision - Confidence Curve for Tobacco800 training

The precision will typically increase when the confidence level is raised, while the recall will typically decrease. This phenomenon can be seen in our situation as well: the model's precision rises sharply as confidence rises. The final curve, which is related to confidence, emphasizes the connection between recall and confidence. Once more, Figure 3.4 shows that the curve moves significantly after a given degree of recall. The curve quickly shifts and hits zero at a value of about 0.8. We can see that Confidence has a fairly similar value to Recall when Recall is near 0.8.

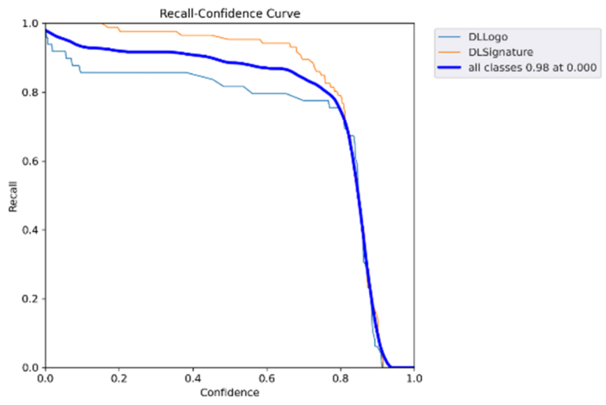


Figure 3.4: Recall - Confidence Curve for Tobacco800 training

The Precision - Recall curve, which connects these two indicators, must also be observed. Plotting the precision on the y-axis and the recall on the x-axis for all threshold values between 0 and 1 yields the precision-recall curve. The precision is very high and the recall is quite low for thresholds that are very high (just below 1.0). By placing ourselves around a Precision value close to 0.6, we can infer from this figure that the Recall will also have a fairly comparable value. A balance between the two metrics is most likely to be found at this position on the graph.

At this point, it is possible to continue with the analysis of the testing phase's findings using the Tobacco800 dataset.

Inference - Tobacco800 Dataset The stage of the process known as inference allows the model's performance to be evaluated using images that it has never seen before and are, therefore, dissimilar to those it has trained on thus far. The trained model is saved in a *.pt* file during the training stage

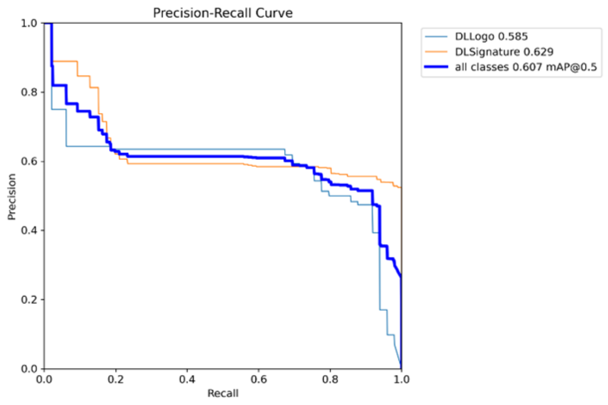


Figure 3.5: Precision - Recall Curve for Tobacco800 training

and used in the testing stage. The line of code that follows demonstrates how:

```
python Yolov5/detect.py --source Tobacco800/New_format/images
    /Val --weights Risultati/Tobacco800/Traitn/best.pt --save-
    txt
```

First things first, the *detect.py* file containing the detection function must be invoked. Second, it is important to specify the location of the images for the model to inspect; in such case, Yolo will discover the images in the $images \geq Valid$ folder. These are the photos used following the initial format conversion phase. Next, choose the template we want to use, and then insert the *best.pt* file, which contains the template trained in the previous stage. These are the fundamental commands that need to be typed in. Additionally, more commands may be inserted to enable the model to generate usable results. Among them are:

- –conf-thres: “default=0.4” is the object confidence threshold
- –iou-thres: “default=0.5” is the threshold for NMS
- –device: is the “cuda device or cpu” command

- –view-img: is the “display results” command
- –save-txt: is the “saves the bbox co-ordinates results to **.txt*” command
- –classes: it permits to “filter by class: –class 0, or –class 0 2 3”

In order to store a *.txt* file containing all the coordinates and measures predicted by the model for each image in a different folder, we chose to only use the *–save-txt* option. We made the decision to save them so that we could go back and evaluate the disparities between the actual data and the data observed and determine whether or not the model is actually accurate. Below is a sample of *.txt* files.

```
1 0.677083 0.292880 0.256250 0.108414
```

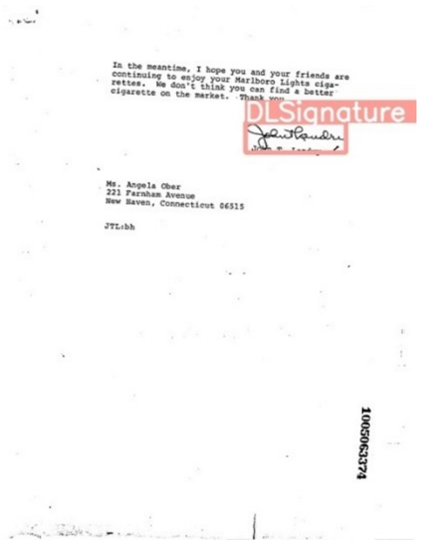
These values correspond to the class, height, and length of the signature rectangle, as well as the x and y coordinates of the upper left point of the rectangle. Below in Figure 3.6 there is a clear illustration of the rectangle discovered.

The next stage adds a second line of code:

```
!python Yolov5/val.py --weights Risultati/Tobacco800/Train/
    best.pt --data bcc.yaml
```

This is quite helpful since it enables to preserve the model's results, which calculate all the performance indicators seen in the previous phase. In this way, it will be feasible to evaluate if the model performs better or worse after training.

The path containing the *val.py* file, with the code needed to validate the trained model, is the first item in the previous line of code. These two


In the meantime, I hope you and your friends are continuing to enjoy your Marlboro Lights cigarettes. We don't think you can find a better cigarette on the market. Thank you

DLSignature

[illegible]

Ms. Angela Ober
221 Farnham Avenue
New Haven, Connecticut 06515

JTL:bh

1005063374

Figure 3.6: Example of signature detection through YOLO

Python files, along with the second *detect.py* file, make up the codes used to assess the trained model. The directory of the *.pt* file created during the training phase must be entered once again to specify which model will be utilized. The *.yaml* file, which contains the paths to the directories “train” and “val” described in the previous chapter, is then inserted. The figures produced by this line of code are as follows:

These are just some of the pictures generated by the model, where the detection of all the images are compared to give an overview of the outcomes. It is clear whether or not the majority of the signatures were accurately recognized at first glance. Following that, it will be feasible to determine whether the model was actually efficient or not by going and measuring it using numerical values.

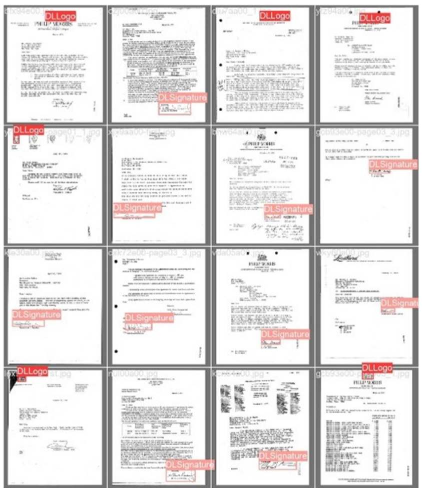


Figure 3.7: Results of signature detection through YOLO

3.1 Performance Metrics

Figure 3.8, already observed, is the confusion matrix from which Accuracy, Precision, Recall, and F1-score are obtained.

If the model can perform well on images other than those on which it is trained, it can be determined by comparing the predicted results with the training results. The outcomes in relation to logos are:

- Accuracy: 0,82
- Precision: 0,69
- Recall: 0,86
- F1: 0,70

These data can be compared to those from the logo training and it can be seen that there are no differences. As a result, the model's ability to recognise logos retains its precision and accuracy. It is now important to verify that

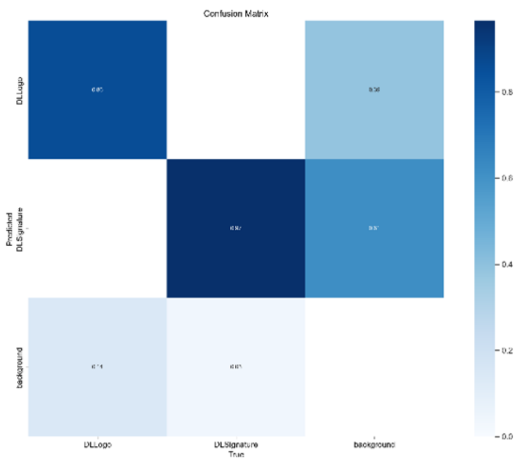


Figure 3.8: Confusion Matrix for Tobacco800 Inference

the model's performance is unaffected when it comes to signatures. The following are the outcomes for signature detection:

- Accuracy: 0,79
- Precision: 0,61
- Recall: 0,97
- F1: 0,75

If these findings are compared to the training outcomes, a small difference is discernible. Accuracy and Precision remain the same while the Recall and F1 score values both fall by 0,01. Thus, even in the context of signatures, performance is still great.

With the confidence curves already seen in the training phase, it is possible to observe these outcomes in much greater depth.

The F1 - Confidence curve's trajectory closely resembles the training one. To avoid a sharp decline in both Confidence and F1 Score, there is a value

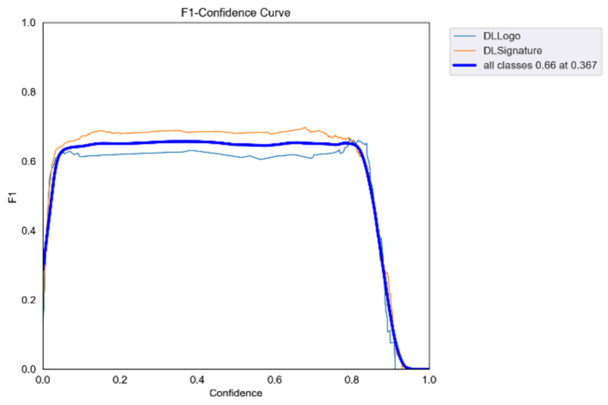


Figure 3.9: F1 - Confidence Curve for Tobacco800 Inference

around which we should maintain our F1-score. Therefore, to keep the confidence level at 0.8, this score must stay close to 0.66. The relationship between confidence and precision is shown in Precision - Confidence Figure 3.10, used to evaluate the degree of confidence once more.

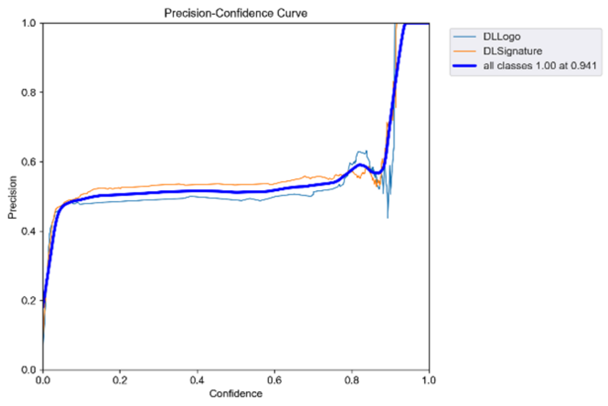


Figure 3.10: Precision - Confidence Curve for Tobacco800 Inference

The pattern always closely resembles the training stage. Once more, the curve shows a significant increasing trend from a 0.8 Confidence level. The link between recall and confidence is represented by the following curve.

Again, it is important to make an effort to maintain a Recall value that

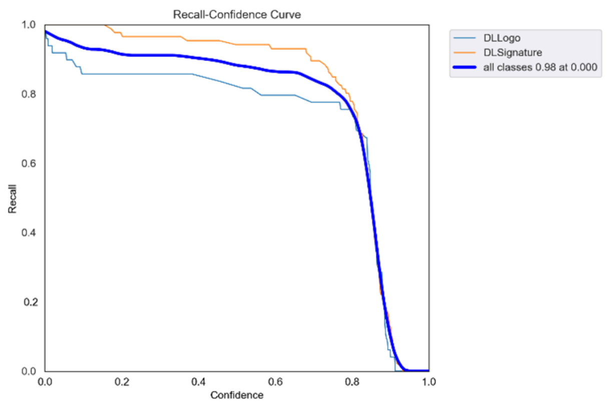


Figure 3.11: Recall - Confidence Curve for Tobacco800 Inference

ensures a Confidence level of roughly 0.8. The Recall number should be in the range of 0.8, as shown in Figure 3.11. The values in each of these confidence curves come out to be very similar to, if not exactly the same as, what was seen at the conclusion of the training phase. This outcome, which is also consistent with the confusion matrix data, confirms that there is little to no performance difference between the two stages. This is a surprising result since it shows that the model can detect new signatures and logos without decreasing accuracy after being trained.
To be fully convinced of the outcomes so far, an additional image, Figure 3.12, can be observed; the relationship between recall and precision. This is a crucial relationship, as already stated for the prior phase. These two variables frequently follow opposing patterns; that is, when one rises, the other falls, and vice versa. Finding an equilibrium point where both exhibit satisfactory values is therefore necessary.

Once more, the curve shows a downward trend and is quite similar to the training step. During the training, we claim that the Precision value stays around 0.6 and the Recall value displays a fairly comparable value, finding equilibrium. The situation following the inference step is extremely similar.

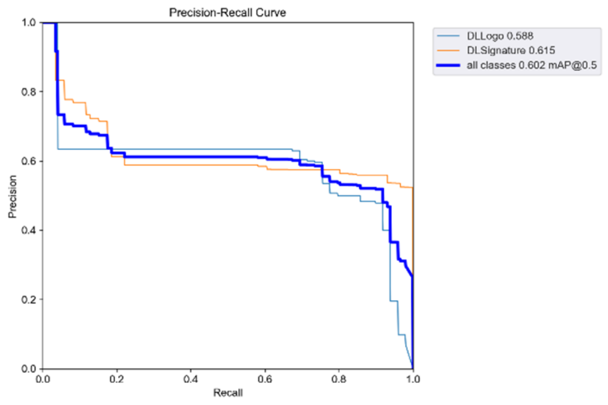


Figure 3.12: Precision - Recall Curve for Tobacco800 Inference

The slope rapidly declines until 0.6 Precision and Confidence is obtained. Because of this, we can assert that even in this final instance, the inference and training outputs are nearly identical. After concluding the work with the Tobacco800 images, it is time to test Yolo's capabilities with another kind of image to determine whether the selected Neural Network is actually the best one for this project's objective.
To continue on this road, a dataset with pictures that were more similar to identity documents is required; these images must to be horizontal with a colored background, much like identity cards, in order to grasp what Yolo's constraints are. On Kaggle [3], a dataset containing precisely these kind of pictures is discovered. It is ideal, as it has 160 images of cheques with logos and signatures.

There are several attempts with these images. In order to see if Yolo can recognize signatures even without being trained on other pictures, an attempt is made directly on the test phase, without training the model with these new images. The outcomes are displayed in Figure 3.13.

Let's just say that the initial attempt is unsuccessful. Without the correct

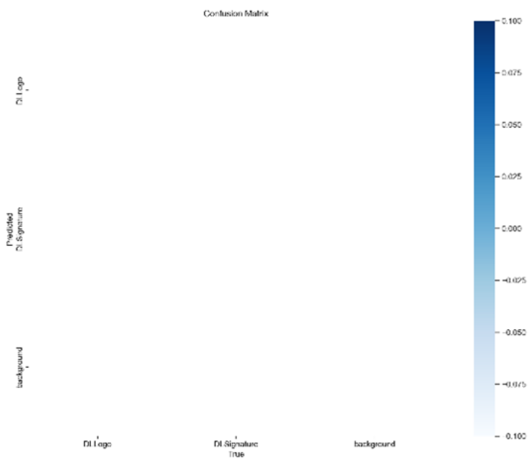


Figure 3.13: First Attempt with Bank Cheques' Dataset

training, Yolo is unable to recognize trademarks or signatures on photos with colored backgrounds. This is mostly because color obscures the signature and prevents the Neural Network from recognizing it, especially when specific colors, such as green, are used. The images below provide an illustration of the issues.

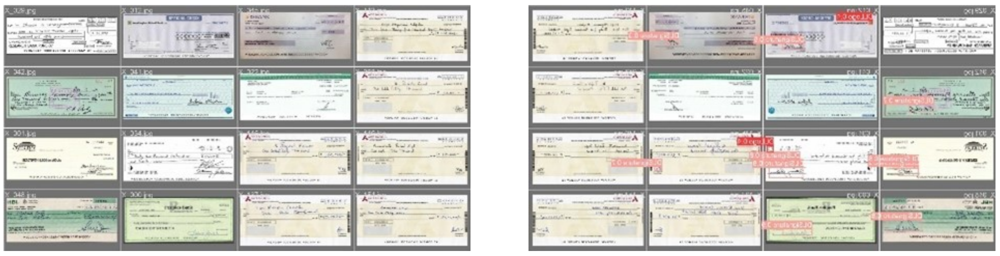


Figure 3.14: First Results with Bank Cheques' Dataset

These pictures offer a broad view of the identifications the model produces. As can be observed, Yolo is unable to find anything in the first set of images, hence there are no results. These are images with a darker outline, as already said, and the black letters on them usually tends to merge with the background color.

The full training and testing procedure outlined in the prior chapter is used with these new photos to address this issue. The dataset's availability of 163 photos presents another challenge. This number is not sufficient to effectively train the model. In truth, Yolo would be trained with only about 80 photos when these few images are split into a train set and a test set. This is insufficient to guarantee that the model is correctly trained. In order to manually increase the amount of photos, this required further processing on the dataset. Each image is copied, then altered to produce new versions of the same image. The number of photographs created is in the neighborhood of 540, and each one comes with a *.txt* file holding all the coordinates required for the first stage of data processing, which was covered in the previous chapter. It must be emphasized that since each measurement is manually taken, its accuracy cannot be completely guaranteed. However, it turns out to be a useful process for our objectives. The outcomes significantly improve with 540 photos and they are described in the following sections.

Training Bank Cheques Dataset The data processing stage came before the training stage. In this case as well, all photos are resized to fit Yolo standards and the coordinates in the *.txt* files are adjusted correspondingly. The training can now go on as planned. The directory containing the modified photos, but the lines of code used are the same as for the prior dataset. Let's now discuss the outcomes.

We can already say from a quick glance of Figure 3.15 that the outcomes will differ from those of the first dataset. This is due to the fact that the background in this instance is colored, making its detection levels significantly higher than in Tobacco800. Performance metrics can still be computed in

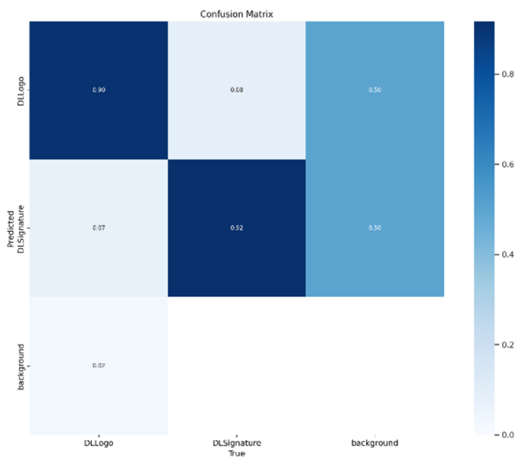


Figure 3.15: Confusion Matrix for Bank Cheques Dataset training

spite of this.

The following results regarding the firms are calculated:

- Accuracy: 0,78
- Precision: 0,61
- Recall: 0,92
- F1: 0,74

With regard to the logos, we can claim that the outcomes are extremely comparable: Precision displays the same result of 0.61; the difference between all other measures is less than 0.01.

Since this is one of the first attempts using different documents than those previously utilized, the results are satisfactory. It is also feasible to see the variations between these outcomes and those attained during the Tobacco800 dataset's training phase. Accuracy for Cheques and Tobacco800 are essentially similar: the first is equal to 0.78 and the second to 0.79. Precision in both instances is the same, 0.61. Cheques' recall slightly declines, while Tobacco800's recall increases to 0.98. The F1 score, which was 0.76 in the first instance, is likewise slightly lower. As a result, it can be claimed that the results are fairly similar and satisfactory for both datasets, showing that, at least during the training stage, with the proper training, the model's performance is not much impacted by the layout and style of the image.

Now that the other curves have been examined, it is possible to determine just how effective the model is.
The confidence curves let us determine the highest value that can be obtained before the confidence level and our results collapse. The curves seems to resemble those of Tobacco800 quite a bit. Once more, the confidence level roses to a peak of roughly 0.8 before abruptly dropping. This 0.8 is equivalent to an F1 score of roughly 0.60. This score tends to fluctuate more, going into a range between 0.53 and 0.64, in contrast to Tobacco800, where it tends to remain more steady as confidence rose.

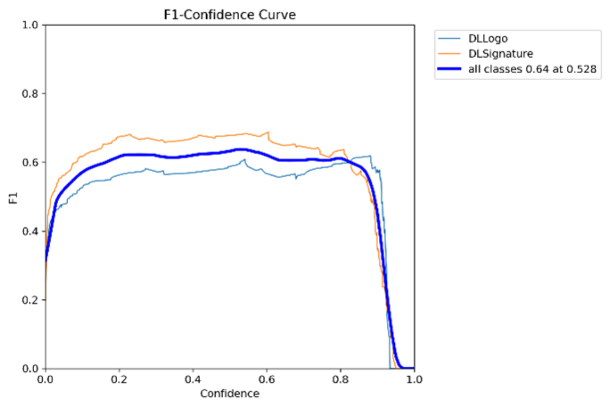


Figure 3.16: F1 - Confidence Curve for Bank Cheques Dataset training

The curve in Figure 3.17 showing confidence in respect to precision is another significant curve. As with Tobacco800, the tendency of the curve is

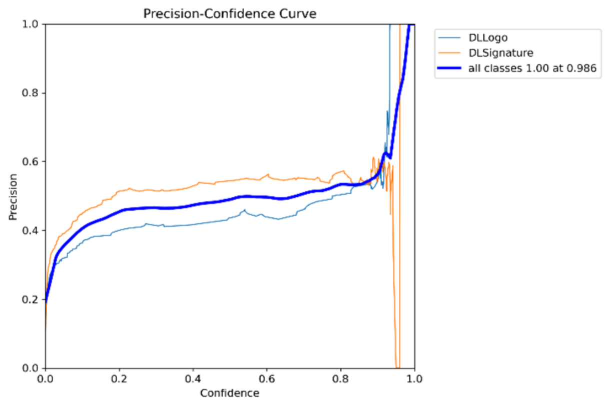


Figure 3.17: Precision - Confidence Curve for Bank Cheques Dataset training

upward also in this case. Upon achieving a Confidence value of roughly 0.9, it can be observed a significant increase and a brief period of decline. The orange curve illustrates this slight decline as a collapse of Precision in the case of signatures, which rises extremely quickly in the following instant.

Despite everything, the curve in Figure 3.17 demonstrates that Precision's performance is lower than Tobacco800's, but it is still a sufficient value for the project's needs.

The link between recall and confidence is depicted in Figure 3.18. Precision and recall are two values that need to be carefully monitored.

As it should, the curve shows a diminishing tendency. At first look, Figure 3.18 appears to have a trend worse than Tobacco800, with values that are occasionally increasing and other times dropping when compared to the curve of the first dataset. Overall, it is clear that the outcome is still positive because the curve has the traditional form of a recall-confidence curve with satisfactory values that are only marginally below Tobacco800's ones.

Finally, in Figure 3.19 we have the relationship between Precision and Recall,

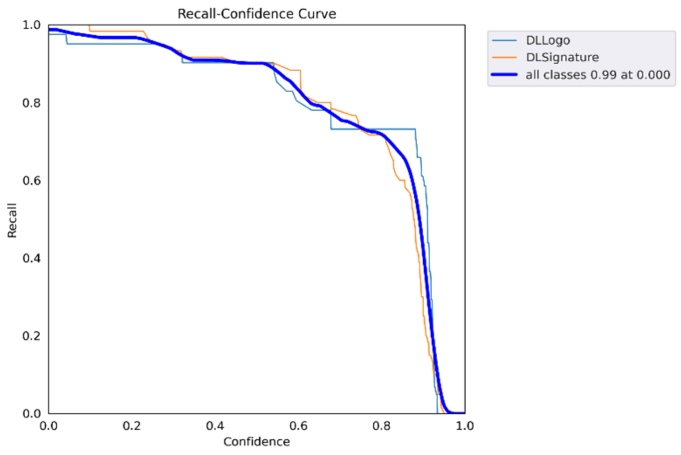


Figure 3.18: Recall - Confidence Curve for Bank Cheques Dataset training

two values that must be balanced in order for the model to perform well.

The shape of the curve slopes downward as it should. We all know that the ideal curve would incline toward the top right corner of the graph. The performance in our situation is obviously inferior, as the curve typically swing between 0.5 and 0.6. The outcomes are also less favorable than Tobacco800's curve, which tended to range between 0.6 and 0.7. In spite of this, the remaining parameters are excellent, and the model's overall performance can still be deemed satisfactory. We can now proceed to track the model's performance during the Inference phase.

Inference – Bank Checks Dataset The following image gives us a first, more broad perspective. We can already detect a significant improvement when comparing it to the images displayed prior to the training period, particularly in the images with a green backdrop where no elements were previously recognized.

Although we can see from Figure 3.20 that there are some irregularities, overall we can claim that things have greatly improved since the last time.

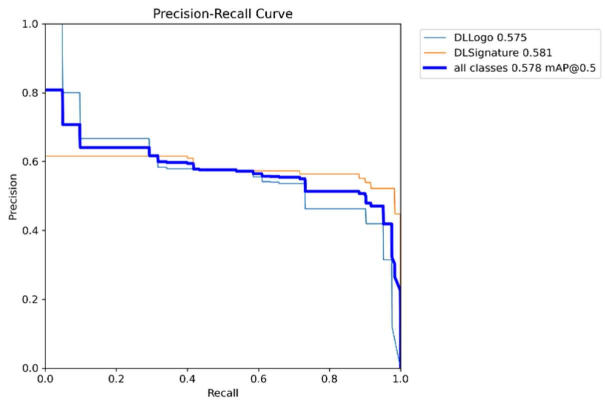


Figure 3.19: Precision - Recall Curve for Bank Cheques Dataset training

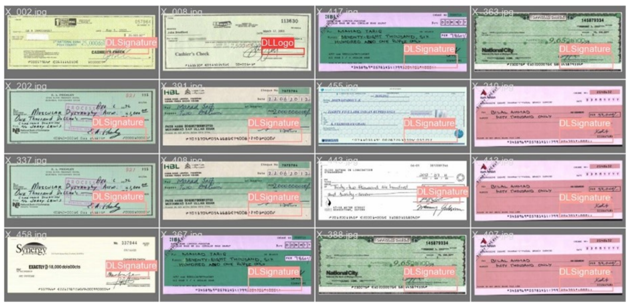


Figure 3.20: Results for Bank Cheques Inference

Let's now discuss the outcomes.

Because these photos have a colored background, which influences how Yolo views the items on the page, it is immediately obvious that the model in this instance has given significantly more weight to the background.

Although the values may initially appear to be considerably different, as we will see, the results are still satisfactory even though the matrix may appear to be very different. We have once more computed the four performance indicators. The results are as follows in terms of the signatures:

- Accuracy: 0,72

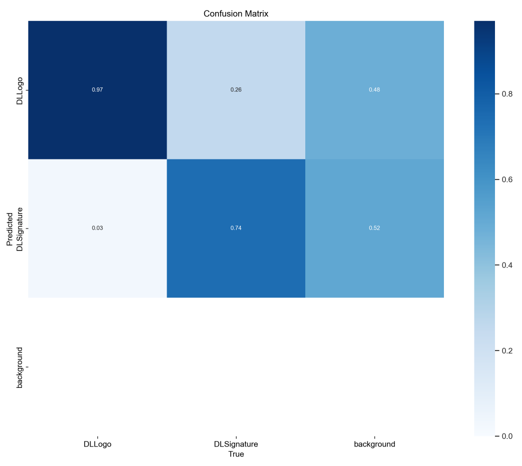


Figure 3.21: Confusion Matrix for Bank Cheques Inference

- Precision: 0,56
- Recall: 0,73
- F1: 0,63

The findings are lower than both the training phase and the first dataset, when they were a bit higher, as was to be expected. This indicates that, aside from conventional white documents, the model has some trouble identifying signatures on images. Let's take a closer look at the curves to better appreciate the challenges this performance presents.

The F1 score is taken into consideration in the first confidence curve we see, Figure 3.22. The curve's trend in this instance does not indicate any issues.

As Confidence rises, the score tends to stabilize around 0.5. The latter significantly falls about 0.9. The curve does not exhibit any anomalous or dissimilar behavior from what has been observed thus far. Undoubtedly, the score value continues to be a little lower than the others previously observed.

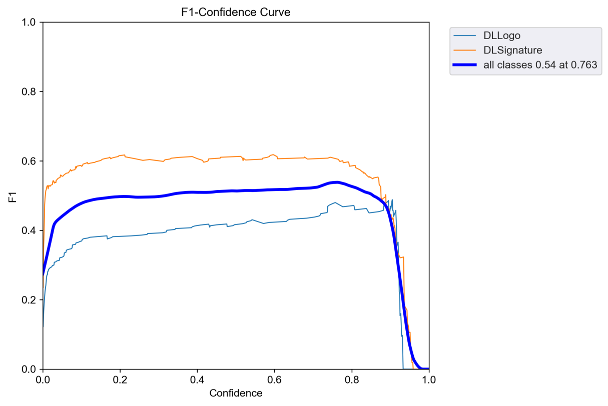


Figure 3.22: F1 - Confidence Curve for Bank Cheques Inference

The following curve demonstrates how the model performs with lower results than the others; in fact, the precision value, when compared to confidence, is the lowest. Although the Precision number is not very high, the curve's tendency is remarkably similar to the other phases so far.

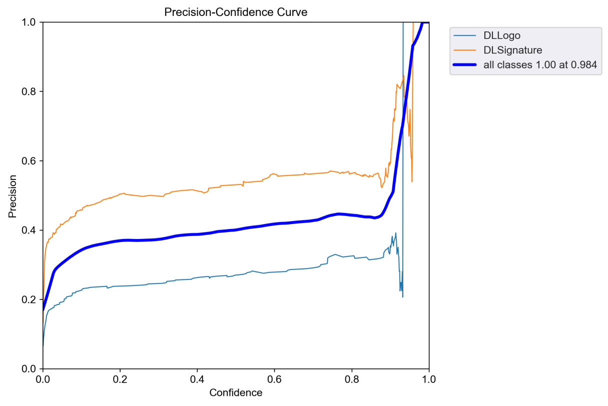


Figure 3.23: Precision - Confidence Curve for Bank Cheques Inference

Contrary to the preceding curves, the following Figure 3.24 displays values inside the usual range. The curve trend is positive and the recall values are satisfactory. Even in this location, it is still possible to see brief intervals where the curve fluctuates, but the curve's overall tendency is still positive.

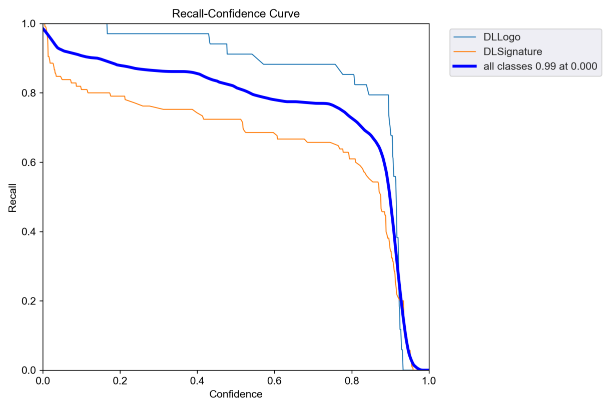


Figure 3.24: Recall - Confidence Curve for Bank Cheques Inference

The real issue with this performance is revealed by the curve in Figure 3.25. It displays extremely poor results, just barely passing the standard required to qualify as a "good" model. The orange curve shows that signatures typically do better than the average of the two classes, whereas logos are the primary source of difficulty. We are aware that recall refers to the percentage of instances from a given class that the model correctly identified as belonging to that class, while precision refers to the proportion of right predictions among all forecasts for a given class. A model with low precision and high recall will produce numerous outcomes, but the majority of the forecasts will be wrong. In contrast, a model with high accuracy and poor recall will produce very few results, but the majority of the predictions will be right. Since there is not a significant difference between those values in our scenario, some predictions will be incorrect by a little margin.

Final Comments on the overall results Overall, it may be concluded that the outcomes are consistent with expectations. Given that the Tobacco800 dataset was created especially for this kind of activity, the first

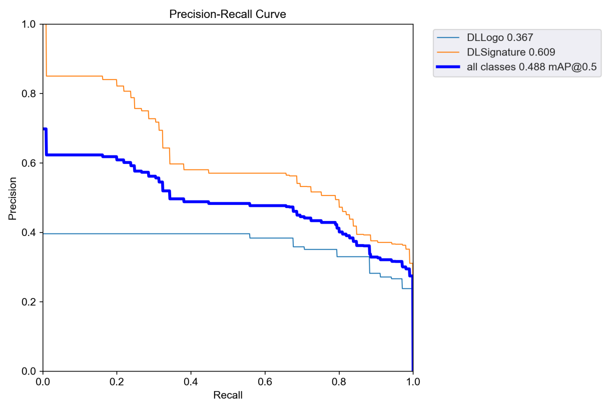


Figure 3.25: Precision - Recall Curve for Bank Cheques Inference

portion of the work, using it, yields better results. The ability to comprehend the model's behavior, outcomes, and performance without being concerned that the data might contain errors was really helpful. As a result of the satisfactory performance with this dataset, we can move on to the following phase.

The project's second phase presents various challenges.
First, the data. The hardest part of the whole project was this. It is extremely dangerous to manually arrange data that will be used as input in a model because there is a very high chance that something will go wrong and affect the model's performance. Although the task is not yet complete, numerous tries were made, many mistakes and revisions were done, and ultimately good results were found. The training phase using the cheques is incredibly fulfilling, with outcomes that are almost identical to those of the project's first phase. They are undoubtedly higher than we anticipated. On the other side, the inference stage goes exactly as predicted. Results are less favorable than in any of the other phases, with a decrease of 0,2 in one of the performance metrics. Nevertheless, given that Yolo has more trouble identifying signatures when they are on colored backgrounds, these are the kinds

of results we might expect. The backdrop component becomes a significant barrier to the project's successful execution. It is clear that more work has to be done, and there are still a number of actions that can be performed to enhance the model's performance and, hopefully, the outcomes of this final stage.

3.2 Future Steps

Clearly, it goes without saying that the outcomes must be sufficient to produce a functional model that can be provided to the client. This means that the project is still on going. The lack of any readily available datasets outside Tobacco800 that can be utilized for this purpose, in my opinion, is the major barrier. In the lack of data, the only option is to create the dataset manually, which has a very high risk. Even though there are just 163 photos in the cheque dataset, finding more proved challenging. The dataset currently contains 540 photos after being manually expanded. Comparing this figure to Tobacco800's 1250 photos, it is still too low. In future, there could also be the possibility to create new images with the help of new AI tools raised in the last months.
The first stage will therefore be to increase the quantity of photographs and attempt to at least double them. Obviously, all of the signature coordinate information in the images will also be manually entered throughout this process. In this way, the dataset will have more than a thousand photos, and the model's performance should increase.
Working with more suitable datasets and a bigger number of images will be required in the next weeks to optimize the model's performance. This phase is essential to comprehend both the nature of the documents to be catalogued

and the kind of photos that will be used to optimize the model.

Conclusion

After this procedure, we can summarize the goal of this project, emphasizing several elements.
The project's main objective is to classify papers of all kinds, notably identification documents, based on whether a signature is present or absent. As a result, it is a classification issue. We tried to view the issue from a different angle among the several methods for identifying document signatures. Rather than concentrating on the models currently used for signature recognition, we use a yet-to-be-employed method. It is a novel approach to recognizing signatures using a Neural Network for Object Detection: a new practical use of a model developed for different use.

Using two different datasets, the procedure breaks into multiple components and thoroughly examines each process stage and the model architecture. We choose Yolo from all available models because of its construction, which makes it quick, precise, and accurate. The findings from the two datasets produced using it also demonstrated its correctness and speed. The results presented in this thesis are merely the first of many steps that will be taken in the future because the journey is still long and unfinished.

The results observed so far highlight some of the algorithm's advantages and disadvantages. As was already indicated, Yolo is speedy. It generates inference results in minutes and contains all previously examined findings. These findings demonstrate how precise and accurate Yolo is in determining the location of the signature or logo. It also detects the background with relatively little inaccuracy. The second dataset-colored backgrounds well illustrated this last aspect, which Yolo could immediately recognize. These colored backgrounds presented a challenge during the process, as was already noted, but an additional data cleaning and pre-processing step swiftly resolved it.

The following actions are essential to enhancing the model's performance even further. First, it was a path built around uniqueness. The primary goal was to devise an alternative solution to a common problem, which has since been resolved in various ways but was usually seen as a passing phase and given little consideration. Identifying signatures was the project's primary objective, allowing us to approach the problem fresh. We can include this procedure in longer processes involving additional steps confirming these signatures after detection, similar to how signature identification can be seen as the project's ultimate goal. Since the Neural Network is a learning process that can constantly be enhanced or prolonged, implying that different classes of things may be added that Yolo could recognize, including, for instance, recognizing the signature and the photo in the documents. Given that the processes outlined above can be completed entirely from scratch to guarantee the project's success, this model has a wide range of potential future advancements.

Finally, it is crucial to clarify that this project is still in progress and that we will keep working on it in the following months to enhance Yolo's performance. Therefore, after this thesis, we can state that although the findings are already satisfactory, they are only the project's starting point.

Chapter 4

Summary

The problem considered in this thesis is a categorization issue: the objective is to utilize a model that takes documents as input and organizes them based on whether or not they bear a signature. The signed documents can then be selected and organized using the information from this classification. Most of the documents in question are identity documents, such as identity cards, on which a signature—typically a digital or handwritten one that has been scanned—is present. As a result, it is essential to consider the document's shape and orientation, the background color, all of the items on the page, the type of written sentences—both handwritten and computerized—and a wide range of other fundamental criteria.

Since there are already several ways to accomplish this, the issue of signature identification is now viewed as being straightforward and quickly resolved. However, it is crucial to bring attention to an essential component of these procedures since it will clarify why We have chosen to take on this issue. Nowadays, signatures are used in Machine Learning algorithms that check the signature's authenticity.

Banks' mechanism to validate the signature when it is manually entered at the desk to sign any form of the document is a typical illustration of these procedures. In this instance, as with other signature processes, the goal is to confirm the integrity of the signature rather than locate where it is on the page. In order to confirm that the signature is authentic, it is considered individually in each of these steps and compared to a matching valid signature that already exists in the dataset. As a result, signature position detection is frequently done using more straightforward and less accurate models. Sometimes, however, this phase is skipped, and the model is trained directly using datasets that include only signatures without considering the entire document page. These factors make it more transparent why we developed this research as a thesis project. The organization of papers is the primary goal of this project, as was already mentioned. To do this, a model must be used whose main objective is to locate the signature on a piece of paper, not to validate it. This is the primary distinction between our research and current models in this area.

Finding the location of the signature on the documents is the only thing that matters in this project; the subsequent procedures are optional and can always be added in the future. Neural Networks are typically used in projects of such nature for the process's last step, validating the signature. For the intermediate phases of these undertakings, Deep Learning models' accuracy and precision have never before been fully utilized. The signature selection stage on the sheet is usually considered a step leading up to the final verification stage, as demonstrated in the following few paragraphs, and needs more consideration. This is the logic behind using Machine Learn-

ing models rather than Deep Learning ones. The example that follows will clarify this statement. The Connected Components method is one of the most often used models for detecting a signature. It operates as follows: the model creates a “mask” that divides the page into sections, and within each mask frame, an analysis of these Connected Components is performed. This method is based on the generally accurate notion that when we sign, we link all of the letters together, yet when we type on a computer, all letters are separated. The study of Connected Components is based on the assumption that if numerous letters are attached, the likelihood that it is a signature rises significantly.
When we first tried this method, it did not produce the desired results because any mark on the page could be observed. As a result, the sheet's sections corresponding to printing, other handwriting, or scanning errors were frequently mistaken for signatures. This is why we thought this approach needed to be more trustworthy and moved on to look for a more accurate solution.

After conducting considerable research that consistently produced outcomes similar to the ones just mentioned, We considered changing our perspective and eventually discovered a fresh way to see the project's aim. The goal is to locate a signature on a sheet, which can also be interpreted as locating an item within an image, as we have stated numerous times before. This realization caused us to consider the potential for using some Object Detection methods. Unfortunately, we could not locate any scientific publications in the literature that may have explained to us on a theoretical and technical level whether such a technique could be carried out and whether it had been done previously. However, we decided to give it a shot.

We searched for several Object Detection models to determine which was best. Many approaches might be used, as seen in the coming pages, but we use a Convolutional Neural Network known as YOLO (*You Only Look Once*). In addition to conducting this study, we were able to locate a Notebook on GitHub that used YOLO to carry out the process I needed [10]. All of these factors led us to choose this model. Furthermore, this Notebook was extremely helpful to us in understanding how the selected model operates.

A new perspective

What distinguishes this work is the shift in perspective. We made an effort to come up with a different solution to a problem that is already well-known and frequently addressed using the same set of methods (such as Connected Components). As previously stated, the identification phase of current initiatives always played a minor role because the primary focus was on their verification rather than the identification of signatures. In contrast to the subsequent verification phase, where Deep Learning is typically used, this phase has never been surpassed by using more complex Deep Learning models, such as a Neural Network. Instead, we preferred to make signature detection the project's main objective and worked to implement a Deep Learning technique to do so. The outcome was very satisfying. Using the Neural Network, Yolo allowed us to shift our perspective, which was the best option for our project.

The usefulness of our research can also be emphasized, in addition to its value explained in the previous paragraphs. A major corporation that lacks a document cataloging system that meets its requirements could need this project. Due to this, it is crucial to emphasize that the implemented model should also be considered innovative research and a useful cataloging tool that avoids wasting time and money on manual document classification. The beneficial applications of this model can be expanded in the future; for instance, the detection of identity images or the tax code might be added in addition to the detection of signatures. All of this is made possible, in large part, by the use of Yolo, which enables the inclusion of further object classes in addition to signatures without requiring any other model modifications. This enables it to be adapted to a company's requests and change along with it as the tasks required evolve.

The Object Detection Problem

Humans rely on their eyes to perceive everything; they take in data from the world around them and send it to their brains for analysis and inference-making. Well, it seems very easy. Just by looking, we can learn a great deal about what we see, including what they are, where they are, and how they are used. However, the processing that our brain performs for it is simply incomparable. The researchers wondered if we could give a machine this intriguing brain function. With this, the machine's duty will be significantly easier since, if it can identify the items in its environment, it can engage with them more effectively—which is the whole point of developing machines—to make them receptive to interaction with humans and more human-like. How can we teach a machine to recognize an object? The field of computer vision known as "Object Detection" was born out of this. It studies finding instances of different class items (such as a person, book, chair, automobile, bus, etc.) in digitally produced images and videos. Recent years have seen significant advancements in automated data processing and interpretation because of the development of Deep Learning. One of the advantages of Neural Networks is the ability to adapt and progressively learn how to compute the solution to a given problem with the aid of learning examples. Training entails modifying the parameters of the various levels throughout the backward propagation phase to reduce the difference between the value generated and the expected value. The sub-domains of this Object Detection include face detection, activity recognition, image annotation, and many others. Applications for Object Detection can be found in several significant fields, including self-driving cars, robots, video surveillance, object tracking, etc.

YOLO, You Only Look Once

Fast, precise, and able to distinguish a range of items should all be characteristics of general-purpose Object Detection. The advent of Neural Networks has accelerated the speed and precision of detection frameworks. The Object Detection field has advanced tremendously over the past few years due to a significant amount of study. The performance of the algorithms has attained levels of accuracy and precision that have never been higher. The fundamental goal in this field is to mimic the functions and skills of human eyes in terms of object recognition after only a glance. There are now several algorithms that are especially effective at doing this.

The detection accuracy is increasing significantly with the introduction of the YOLO algorithm (You Only Look Once) and its architectural descendants, and it is occasionally better than two-stage detectors. YOLO versions are frequently used in various applications due to their quick conclusions and precise identification. YOLO is a particular Neural Network because it has a high-speed architecture. Because the entire detection pipeline consists of a single network and can be improved end-to-end based solely on detection performance. The model has undergone numerous iterations to increase accuracy and speed over the years. This algorithm's primary objective is to mimic how individuals perceive the world: after looking at something, it can promptly and accurately identify it. [9]

The Algorithm

Yolo is extremely quick while maintaining a high level of accuracy and precision because of its architecture. It is refreshingly simple. A single Neural Network predicts multiple bounding boxes and class probabilities for each box. While training on entire pictures, YOLO instantly improves detection performance. It redefines object recognition as a single regression problem by replacing image pixels with bounding box coordinates and class probabilities.

It can tell what things are present and where they are by only taking a single look at an image. Unlike classifier-based approaches, YOLO is incredibly quick during testing because it only needs to evaluate one network. As will be covered in more detail in the following section, the grid design requires spatial variation in bounding box predictions. Due to the fact that it is frequently clear which grid cell an object belongs in, the network can only predict one box for each object. [8]

As a result, J. Redmon et al. [9] provide a novel approach to the Object Detection issue by integrating the various system components into a single network and developing the first "one-stage" Object Detection system. By simultaneously reasoning about the full image rather than just a few parts, the network is able to better understand the environment in which the various object classes are situated. Additionally, by doing so, it establishes an implicit connection between object classes that are actually connected.

The authors of YOLO [9] have changed the definition of the Object Detection problem from classification problem to regression problem. A single Convolutional Neural Network is used to forecast the bounding boxes and assigning class probabilities to each of the detected objects shown in an image. As said before, this algorithm just needs to look at the image once to recognize the objects and their locations using boundary boxes. The difficulty comes from precisely identifying many items and determining their exact locations within a single visual input. Two key Convolutional Neural Network (CNN) characteristics, parameter sharing and multiple filters, are capable of successfully addressing this object detection issue.

The Architecture

The model has improved, but the dominant model idea has stayed the same.

Network Design

Convolutional Neural Networks are used to create this model, which Redmon J. et al.[9] evaluate using data from the PASCAL VOC 2007 and 2012 detection dataset. The design of YOLO was heavily influenced by that of GoogleNet. YOLO has 24 convolutional layers, of which only four are followed by max-pooling layers.

Unified Detection

The technique combines the many elements of Object Detection into a single Neural Network, as stated in the previous section. This network takes features from the entire image to predict each bounding box. Additionally, it simultaneously predicts all bounding boxes for a picture across all classes. This indicates that the network considers the entire image and all of its objects when making decisions. The YOLO design maintains excellent average precision while enabling end-to-end training and real-time speeds. [9]

The bounding boxes typically vary in size while considering various shapes to capture the objects. A bounding box should be used to identify an object in the image such that the object's center falls inside the bounding box. However, he centers of many objects could coexist in the same bounding box. The bounding boxes that correspond to a single grid cell were referred to by the authors as *Anchor Boxes*.

Comparison with Other Object Detection Models

To better understand what may be done to enhance the performance of the studied model, it is helpful to examine different Object Detection methods in the pages that follow. This can help to determine which features of YOLO need to be improved in order for it to outperform all other models.

- Deformable Parts Model (DPM)
  To extract static characteristics, classify regions, predict bounding boxes for high scoring regions, and perform other tasks, DPM employs a disjoint pipeline. In YOLO, a single Convolutional Neural Network takes the place of all of these various components. The network simultaneously executes contextual reasoning, bounding box prediction, non-

maximal suppression, and feature extraction. The YOLO integrated design produces a model that is faster and more precise than DPM.

- R-CNN
  This algorithm is considered as one of YOLO's main competitors. It uses region proposals: potential bounding boxes are generated, features are extracted by a Convolutional Neural Network, the boxes are scored by an SVM, the bounding boxes are adjusted by a linear model, and duplicate detections are eliminated by non-max suppression. The outcome is an extremely slow system because each stage of this intricate pipeline must be carefully tuned independently. YOLO and R-CNN have some parallels. Potential bounding boxes are suggested for each grid cell, and those boxes are then scored using convolutional features. To counteract multiple detections of the same object, YOLO, however, places spatial constraints on the grid cell proposals. YOLO technology merges the various parts into a single, jointly optimized model.

- Other Fast Detectors Fast and Faster R-CNN
  They focus on accelerating the R-CNN architecture. Both fall short of real-time performance despite being faster and more accurate than R-CNN. YOLO discards the entire detection pipeline instead of trying to optimize individual parts of it and is quick by design. YOLO can simultaneously learn to detect a wide range of objects.

- Deep Multibox
  A Convolutional Neural Network is used to anticipate regions of interest. Multi-Box is still only a component of a broader detection pipeline and is unable to detect general objects. Both YOLO and MultiBox predict bounding boxes in an image using a Neural Network, while YOLO

is a full detection system.

- OverFeat
  Although OverFeat effectively detects sliding windows, the system is still fragmented. Not the performance of detection, but localization, is optimized by Over-Feat. It only considers local data while producing a prediction.

- MultiGrasp
  YOLO shares design elements with research on grip detection. For an image with a single object, MultiGrasp simply needs to anticipate a single graspable zone. For many objects belonging to various classes in an image, YOLO forecasts bounding boxes as well as class probabilities. [9]

Error Performance Analysis

Redmon J. et al.[9] analyse the YOLO error performance, and compare it to Fast R-CNN. It is possible to determine whether or not a category is accurate, if an error exists, and what kind of error it is:

- Correct: the appropriate category, IOU higher than 0.5;
- Localization: the appropriate category, IOU between 0.1 and 0.5;
- Similar: a related class, IOU higher than 0.1;
- Other: the inappropriate class, IOU higher than 0.1;
- Background: IOU lower than 0.1, independently from the category type. [7]

From a comparison of the test results from YOLO and Fast R-CNN that YOLO has some localization issues. These errors are more prevalent in YOLO compared to all the others, whereas Fast R-CNN produces less mistakes overall but more background errors. [7] They both have an issue with the presence of missing objects, those that the models are unable to recognize. The Yolo performance is good in the majority of aspects, while performances for the second model are not considered as reliable. As a result, YOLO proves to be superior in terms of the speed of inference, the detection of small or distant objects, and the issue of overlapping boxes. Finally, both models are capable of locating items in chaotic conditions, where it is more challenging to locate the target object due to the abundance of other items.

Limitations of YOLO

YOLO places substantial spatial limits on bounding box predictions. The number of neighbouring items that the model can predict is constrained by space. Small items that occur in groups are difficult to handle. The model needs help generalizing objects with novel or odd aspect ratios or configurations since it learns to predict bounding boxes from data.
Given that the architecture comprises many down sampling layers from the input image, the model also employs rather coarse features for predicting bounding boxes. Last but not least, while being trained on a loss function that roughly represents detection performance, the loss function treats errors in tiny bounding boxes and large bounding boxes equally. The impact of a tiny error in a small box on IOU is substantially more significant than that of a small error in a large box. Inaccurate localizations are the primary cause of inaccuracy. [9] As DPM, YOLO considers the objects' size, shape, relationship to one another, and locations where they appear more frequently.

The authors also discuss how natural and generated images differ significantly when considering a small number of pixels. At the same time, forms and proportions within the same image are highly similar. Given its design, this contributes significantly to YOLO's ability to generalize object classes and produce effective results when used on simulated photos.[9]

Data Availability

The availability of a smaller number of images to use, compared to other types of Machine Learning algorithms, is another factor that must be considered while analyzing the constraints of this model. Since there are fewer pictures and datasets accessible for other Machine Learning applications, the majority of Object Detection algorithms are actually trained on the same collection of data and images. A few hundred photos make up the most well-known and often used dataset in the field of Object Detection. On the other hand, the datasets utilized for other classification or regression issues contain hundreds of thousands of data. Since one can train these algorithms with a significantly volume of data than Object Detection techniques, it is evident that this results in better outcomes and performance. [9]

Why YOLO?

Understanding if YOLO is the appropriate model to apply in the current scenario being studied is made much easier by all the features and restrictions of the model described so far. Yolo's single pipeline, which has already been stated, is what makes it so fast. In addition, the generalisability component is very intriguing because it enables a complete understanding of the image. The employment of the Non-Maximum Suppression approach, which enables the removal of superfluous boxes, is the last factor to consider. After all the factors emphasized in this first section, it is essential to determine whether Yolo should be used instead of a faster or more accurate model. The final choice was not exclusively based on the mathematical conclusions that have been put forth thus far. Yolo's method of approaching an image, as already said before, is one of the things that really stood out to us about it, and it's also one of the key reasons why we chose to utilize this particular Neural Network for the project. What does this mean? The Object Detection problem is first considered as a regression rather than a classification problem, which is an intriguing characteristic. The algorithm's ability to understand how the observed object interacts with the background and other objects in the image by considering the complete scenario as a whole is a second noteworthy innovation. Yolo can recognize objects much more quickly in videos and, more precisely, in images if it pays attention to the entire context, which is, in our opinion, a very relevant component. Because of all these aspects and how accurate and precise the model is as a result, YOLO was picked.

The issue of signature detection is prevalent nowadays. Therefore creative solutions are required to make a genuinely fascinating contribution to this field of study. YOLO has yet to be used to recognize signatures on identity documents or to classify and organize them, so it looks like an innovative and intriguing decision to take advantage of its capabilities in this area. YOLO's inability to recognize objects it has never seen before is one of its limitations, but in this case, it is obvious that this is not an issue because the only thing that needs to be recognized is the signature. As a result, there is no danger of YOLO encountering objects that are too small to be recognized.

The Project

Nowadays, the problem of detecting and validating signatures is regularly encountered, and there are numerous ways to fix it. This is the reason why this work may initially appear simple and straightforward to complete. However, it is a very intriguing process because there are many factors to consider;many things could hurt the results. For example, signatures are usually quite delicate because they are typically handwritten and scanned. Examples include how the page is laid out, where and how the signature is placed, whether there is any additional handwriting on the page, whether colored backdrops are used, and how saturated and transparent the background is. Because of this, it is not easy to come up with a workable solution for this situation.
It is also essential to stress that the solution to be utilized must be carefully chosen depending on the results we need and the project's type of aim. Why does this matter? We must choose which method best meets our needs from the several techniques to identify and authenticate a signature.

The process of recognizing a signature came with several challenges. First, the algorithm is trained to recognize a handwritten sign typically found in the bottom left-hand corner, presenting a first barrier. What transpires if there are additional handwritten signatures on the document or if the signature is not in the usual position? This was the first thing that was examined and, if necessary, corrected. Yolo made this situation easier to handle than we had anticipated. Yolo is quite good at recognizing only the signature when it comes to other handwritings, and it is also possible to specify a probability cutoff below which handwritings are not taken into account. It must be acknowledged that Yolo still needs to improve on the second challenge, the location of the signature, although this was solved by adding more photos to the dataset. This made it possible to increase both the number of pictures used to train the algorithm and the number of pictures used to test it.

The design of the page is still another challenge. The method yields excellent results when detection is required on vertical sheets with a white background and black writing. However, the model's performance noticeably degrades when pages are used with colored backgrounds and the page orientation is reversed. When the background is darker, Yolo has an adamant time recognizing signatures, especially when the color green is present. Due to this, as we shall see on the folowing pages, it was necessary to repeat the entire procedure to increase the algorithm's performance using a dataset of colored images arranged on horizontal pages. The outcomes of all efforts will be seen in the next pages.

The Purpose

The thought of using this kind of algorithm as a thesis project first emerged during the research for a system that could classify papers, particularly identification documents, based on the presence or absence of a signature. First, it is essential to clarify some project's aspects in order to reveal its innovative nature. The first point that needs to be emphasized is that this research and its solution must be seen in the context of a broader discussion of the field of signature recognition. The signatures' validation is now the most crucial phase in signature verification efforts. As a result, finding their location on a page or document is just a preliminary task that is addressed using less time-consuming and effort-consuming methods and consequently given less priority.

Based on this supposition, it is clear that the ultimate objective of this initiative is not the same as signature verification because signature verification is subsequent, and in this case superfluous, step that may always be added later; more traditional Machine Learning algorithms will not be used.

A Neural Network is typically used in projects that attempt to verify a signature after being trained with a signature dataset to enable it to assess its legitimacy. The signature detection step is often resolved using a Machine Learning algorithm, which determines if the signature is there or not based on the examination of the connected components on the document. Frequently, this step is just skipped, and the Neural Network is trained directly for verification using images of each signature that have already been extracted from the document.
Identifying the signature on a document is equally a crucial step in the process, particularly for companies that require it when classifying digital documents and we were given full attention in this thesis topic. Due to this, it was decided to use a Neural Network also during the identification step in order to produce more accurate findings.
Numerous tests were run, taking into account numerous examples and papers found online, in order to determine which algorithm was the best. Finally, a choice was made after carefully examining the outcomes found in about a dozen images. First, all other potential Machine Learning algorithms that did not use Deep Learning were excluded and the search was limited to the field of Neural Networks. The most successful ones were then chosen, taking into account the four primary performance metrics: precision, accuracy, recall, and F1-score.

Finally, the Convolutional Neural Network "YOLO" (You Only Look Once) was selected. This is an Object Detection algorithm, as was already seen in the previous section. Using a Neural Network appeared reasonable because the goal is to identify an object -the signature- in a picture or scanned document. When used, YOLO should produce a dataset containing all the document's metadata (including the document's width and length, location, length, and width of the signature, as well as the existence and placement of the logo, along with a final column for specifying whether the document is "signed" or "unsigned").

The Process

The following pages will describe all the steps that have been followed in detail.

Data Pre - Processing

We referenced the notebook on Github [10] during the process, demonstrating how to utilize this technique to find signatures and logos on scanned pages.

Data Description

Finding a dataset that was appropriate for the project was the first step. The Tobacco800 dataset consists of over 1,200 images of scanned documents in black and white, some of which are signed, some of which are not, some of which are misaligned, and some of which have other pen writing. At least, for the project's initial goal, it was appropriate. The metadata for each image is stored in *XML* files in a separate folder in addition to the one holding the photos. The height and width of the page, the presence or absence of sig-

natures and logos, and, if any were included, their coordinates (x, y, width, and height) were all included in each file. This information was essential for training and evaluating the model's effectiveness.
After locating the input data, a data cleaning procedure was required to transform the images and text files into a format that the algorithm could understand without causing errors.
After choosing all the needed features, it was possible to go on with the next step, the training phase. First, the correct version of the model was choosen, the Yolov5s or "small", that presented the best performances: it is the fastest and one of the more accurate version. After this choiceand the proper organization of folders that divide images in the train and test set, it was possible to start the training phase.

At the end of this process, the model will compute the performance metrics. First is the confusion matrix which determines precision, accuracy, F1-score, and recall. The results are excellent, but they will be ex-examined in further detail in the following pages. Then, all the curves that describe the confidence level are also available: the Precision-Confidence curve, the Recall-Confidence curve, the F1-Confidence curve.

Once the results from the training phase are observed and we are satisfied with it, it is possible to go on with the testing phase. This is the step in which the algorithm will pass through a new image, that it has never seen before. This is a fundamental step, because it is the moment in which we can understand if our model can detect objects in an image by himself, using all the information he learned during the training phase.
Focusing on a later stage of the process is vital before going on to the fol-

lowing sections, which will display the project's results. As was stated at the beginning of the chapter, when the procedure had been tested on the Tobacco800 dataset, it was essential to determine whether Yolo could also recognize signatures on less "conventional" sheets and documents. It was necessary to hunt for another dataset with horizontally colored pages as a result. The project was excellently suited to the "Bank Checks Accounts" dataset that was discovered on Kaggle [3]. Repeating the entire process after these data processing procedures were finished was feasible. All of the outcomes will be revealed in the following sections.

Results

The outcomes can be seen in this part after all the computational procedures and the project, in general, have been thoroughly detailed. This will demonstrate whether YOLO is an original and practical method for accelerating and improving signature detection accuracy. As discussed in the previous chapter, YOLO was tested on two separate datasets to see how well it could recognize signatures on various types of documents.

The first one is Tobacco800, which was quite helpful because, as was already indicated, it is a dataset explicitly made for this kind of activity. With the help of these kind of documents, the initial tests were run, and they also gave us a practical understanding of how the algorithm functions as a whole. Additionally, this kind of paper was done with the complete study of hyperparameters. Given that it already contains all the data and information required for the project and that there are roughly 1200 images in it that are ideal for training a Neural Network, it is evident that the dataset is simple to utilize. Because the dataset was so simple to be used, we could thoroughly understand all of YOLO's properties without running the danger of making an error owing to data uncertainty.

Bank cheques make up the second dataset. The issue with these images was precisely the opposite of Tobacco800. It was a dataset with roughly 160 total pictures that was hard to find on the Kaggle site. [3] Not all of the metadata required to evaluate the model's performance was offered. Therefore, expanding the number of photographs available was necessary as a first step. How? Copying the current images and altering their colors, position, and shapes to distinguish them. The dataset now contains roughly 500 photos. However a further 500 images might be added. The metadata had to be added at this point as well; each image had a text file with the same name that contained six values, including the height and width of the page, the coordinates of the point at the top left of the rectangle containing the signature, and the height and width of the rectangle itself. Once this procedure was finished, repeating every step with the first dataset was simple.

Overall, the outcomes are consistent with expectations. Given that the Tobacco800 dataset was created especially for this activity, the first portion of the work, which used it, yields better results. The ability to comprehend the model's behavior, outcomes, and performance without being concerned that the data might contain errors was really helpful. As a result of the satisfactory performance with this dataset, we can move on to the following phase. The project's second phase presented various challenges. First, the data. The hardest part of the whole project was this. It is hazardous to manually arrange data that will be used as input in a model because there is a high chance that something will go wrong and affect the model's performance. Although the task was incomplete, numerous tries were made, mistakes and revisions were made, and ultimately good results were found. The training phase using the checks was incredibly fulfilling, with outcomes almost identical to those of the project's first phase. They were undoubtedly higher than we expected. On the other side, the inference stage went precisely as predicted. Results are less favorable than in any of the other phases, with a decrease of 0,2 in one of the performance metrics. Nevertheless, given that Yolo has more trouble identifying signatures when they are on colored backgrounds, these are the kinds of results we might have expected. The backdrop component becomes a significant barrier to the project's successful execution. More work has to be done, and several actions can still be performed to enhance the model's performance and, hopefully, the outcomes of this final stage.

Future Steps

The project is still ongoing. The primary barrier is the lack of any readily available datasets outside Tobacco800 that can be utilized for this purpose. In the need for more data, the only option is to create the dataset manually, which has a very high risk. Even though there are just 163 photos in the cheque dataset, finding more proved challenging. The dataset currently contains 540 pictures after being manually expanded. Comparing this figure to Tobacco800's 1250 photos, it still needs to be higher. The first stage is to increase the number of photographs and attempt to double them at least. This could also be done using the new AI tools to generate new cheque pictures automatically. The signature coordinates information in the images will be manually entered during this process. The main objective is to reach more than a thousand photos in the dataset, and the model's performance should increase. Working with more suitable datasets and more images will be required in the following weeks to optimize the model's performance. This phase is essential to comprehend the nature of the documents that need to be cataloged and the images that will need to be used to optimize the model.

Conclusion

After this procedure, we can summarize the goal of this project, emphasizing several elements.
The project's main objective is to classify papers of all kinds, notably identification documents, based on whether a signature is present or absent. As a result, it is a classification issue. We tried to view the issue from a different angle among the several methods for identifying document signatures. Rather than concentrating on the models currently used for signature recognition, we use a yet-to-be-employed method. It is a novel approach to recognizing signatures using a Neural Network for Object Detection: a new practical use of a model developed for different use.

Using two different datasets, the procedure breaks into multiple components and thoroughly examines each process stage and the model architecture. We choose Yolo from all available models because of its construction, which makes it quick, precise, and accurate. The findings from the two datasets produced using it also demonstrated its correctness and speed. The results presented in this thesis are merely the first of many steps that will be taken in the future because the journey is still long and unfinished.

The results observed so far highlight some of the algorithm's advantages and disadvantages. As was already indicated, Yolo is speedy. It generates inference results in minutes and contains all previously examined findings. These findings demonstrate how precise and accurate Yolo is in determining the location of the signature or logo. It also detects the background with relatively little inaccuracy. The second dataset-colored backgrounds well illustrated this last aspect, which Yolo could immediately recognize. These colored backgrounds presented a challenge during the process, as was already noted, but an additional data cleaning and pre-processing step swiftly resolved it.
The following actions are essential to enhancing the model's performance even further. First, it was a path built around uniqueness. The primary goal was to devise an alternative solution to a common problem, which has since been resolved in various ways but was usually seen as a passing phase and given little consideration. Identifying signatures was the project's primary objective, allowing us to approach the problem fresh. We can include this procedure in longer processes involving additional steps confirming these signatures after detection, similar to how signature identification can be seen as the project's ultimate goal. Since the Neural Network is a learning process that can constantly be enhanced or prolonged, implying that different classes of things may be added that Yolo could recognize, including, for instance, recognizing the signature and the photo in the documents. Given that the processes outlined above can be completed entirely from scratch to guarantee the project's success, this model has a wide range of potential future advancements.
Finally, it is crucial to clarify that this project is still in progress and that we will keep working on it in the following months to enhance Yolo's perfor-

mance. Therefore, after this thesis, we can state that although the findings are already satisfactory, they are only the project's starting point.

Bibliography

[1] *Accuracy Explaination.* `https://www.iguazio.com/glossary/model-accuracy-in-ml/`.

[2] Tanvir Ahmad et al. "Object Detection through Modified YOLO Neural Network". In: *Scientific Programming* (June 2020). DOI: `10.1155/2020/8403262`.

[3] *BCSD: Bank Checks Segmentation Database.* `https://www.kaggle.com/datasets/saifkhichi96/bank-checks-signatures-segmentation-dataset`.

[4] *Definition of Intersection Over Union.* `https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`.

[5] Tausif Diwan, G. Anirudh, and Jitendra V. Tembhurne. "Object Detection Using YOLO: Challenges, Architectural Successors, Datasets and Applications". In: *Multimedia Tools Appl.* 82.6 (Aug. 2022), pp. 9243–9275. ISSN: 1380-7501. DOI: `10.1007/s11042-022-13644-y`. URL: `https://doi.org/10.1007/s11042-022-13644-y`.

[6] *Introduction to YOLO Algorithm for Object Detection.* `https://www.kaggle.com/datasets/saifkhichi96/bank-checks-signatures-segmentation-dataset`.

[7] Erich Kohmann. "Tecniche di deep learning per l'object detection". PhD thesis. URL: http://amslaurea.unibo.it/19637/.

[8] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: (2016). arXiv: 1612.08242 [cs.CV].

[9] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[10] *Signature Detection through YOLO GitHub Notebook.* https://github.com/amaljoseph/Signature-Verification_System_using_YOLOv5-and-CycleGAN.

[11] *Ultralytics GitHub repository.* https://github.com/ultralytics/yolov5/tree/master.

[12] *YOLO Object Detection Explained.* https://www.datacamp.com/blog/yolo-object-detection-explained.

[13] *Yolo versions comparison.* https:https://machinelearningknowledge.ai/a-brief-history-of-yolo-object-detection-models/?utm_content=cmp-true.

[14] *You Only Look Once.* https://towardsdatascience.com/yolo-you-only-look-once-3dbdbb608ec4.