URL Detection based on YOLO Network in Various Conditions*

Leila Boussaad[1,2,*,†], Aldjia Boucetta[1,2,†], Aymene Zeroual[3,†] and Wail Ala-eddine Zeroual[3,†]

[1]Management Sciences dept., Batna1 University, Batna, 05000, Algeria
[2]LAMIE Laboratory, Computer science dept., Batna2 University, Batna, 05000, Algeria
[3]Computer Science dept., Batna2 University, Batna, 05000, Algeria

Abstract

Object detection is a key technique in computer vision, as it is considered a necessary step in any recognition process. It is the procedure for determining the instance of the class to which the object belongs and estimating its location by displaying its bounding box. It was widely accepted that advances in object detection have generally gone through two periods: "the traditional object detection period", where detection was performed through classical machine learning techniques, and "the deep learning based detection period", where classical machine learning techniques have been completely replaced by methods based on deep neural networks. In this paper, we will focus on object detection based on deep learning. The main objective is to carry out a comparative study of three models of the YOLO family, already proven to be effective for object detection that are YOLOv3, YOLOv4, and YOLOv5 in the context of the detection of URLs in photos taken by a mobile phone. The experimental results, expressed in terms of average precision, showed the generalization ability of the three models, YOLOv3, YOLOv4, and YOLOv5. In addition, the stability of the YOLOv4 model against several difficulties added to the images.

Keywords

Object Detection, Deep Learning, Convolutional Neural Networks (CNN), YOLO, URL Detection

1. Introduction

Object detection plays a central role in any recognition system, encompassing the task of identifying an object's class and estimating its spatial coordinates by delineating a bounding frame around the object. Recent advancements in deep learning-based object detection have delivered remarkable outcomes. However, the real-world implementation of object detection faces a host of challenges when confronted with actual images, including factors like noise, occlusion, lighting fluctuations, rotations, and others. These elements have a pronounced impact on the precision of object detection and demand thorough scrutiny during the detection process.

Conversely, the web has consistently served as a medium that allows the transfer of data in a simple and fast way. It counts as a necessary tool in modern life, offering a multitude of prospects for both individuals and large corporations.

World Wide Web, often referred to as the Web or WWW, encompasses all publicly accessible websites and pages that users can access on their local devices via the Internet. These pages and documents are interconnected through hypertext links, which users can click to access information. This information can take various forms, including text, images, audio, and video. To visit a website, a specific page on a site, or more precisely, an "online resource" (such as content or an online service), users can enter its address, known as a Uniform Resource Locator (URL), into the browser's address bar. The URL is indispensable for pinpointing a particular page within the vast sea of billions of web pages. Each web resource possesses a unique URL, which serves as the web address displayed in your browser.

To be more efficient and remove the step of entering the URL, especially with increasing processing capabilities such as the availability of smart phones and visual input devices such as cameras built into smartphones, this process can be divided into three steps: image acquisition, URL localization, and URL recognition. In this paper, we will mainly focus on the second step, which plays a crucial role in the localization of URLs in a captured image containing text. Object detection methods are well suited to accomplish this process. The detection of a URL can be useful in several fields, particularly in the field of tourism. The tourist can take a picture of a URL and view website information without having to type on their keyboard. Businesses, store owners, and their customers can advertise and post information by leaving URLs to the services they offer on their ad slots, and users can retrieve the URL(s) by clicking a button. The model can also be used to capture URL references

6th International Hybrid Conference On Informatics And Applied Mathematics, December 6-7, 2023 Guelma, Algeria
*Corresponding author.
†These authors contributed equally.
boussaad.mous@gmail.com (L. Boussaad);
boucetta_batna@yahoo.fr (A. Boucetta);
aymenezeroual@gmail.com (A. Zeroual);
wailalaeddinezeroual@gmail.com (W. A. Zeroual)

while listening to a presentation at a conference. Given the capability of smart phone cameras lately, taking a picture of a URL in transit should result in a "good quality" image that can be passed as input to a model, and the URL(s) of interest will be recovered.

The primary goal of this study is to implement and assess the effectiveness of three established models in the domain of object detection—specifically, YOLOv3, YOLOv4, and YOLOv5—for the identification of URLs within images characterized by numerous challenges. This includes the application of traditional image processing techniques like rotation and noise addition, which exist in real-world scenarios.

The subsequent sections of the manuscript are designed as follows: Section II provides a concise overview of the key concepts underpinning this paper. Section III outlines the evaluation process we employed. Section IV is dedicated to the presentation of experimental results and ensuing discussions. Finally, in Section V, the paper draws its conclusions.

2. Backgrounds

Prior to delving into the process and the diverse techniques employed in object detection, it is essential to establish a precise comprehension of object detection itself. Frequently, this term is used interchangeably with techniques like image classification, object recognition, segmentation, and more. Nonetheless, it is imperative to acknowledge that many of the techniques mentioned are distinct tasks typically encompassed within the broader realm of object detection. Treating them as synonyms is inaccurate, as each corresponds to a task of equal importance. Thus, we can distinguish these computer vision tasks:

Image classification is about predicting the class of an element in an image, while **object localization** is about locating the presence of objects in an image and indicating their location using a bounding box (see figure 1(a)), and **object detection** is about locating the presence of objects with a bounding box and the types or classes of objects located in an image. Figure 1 (b) clearly shows the result of an object detection process in a road scene.

Another extension of this division of computer vision tasks is **semantic image segmentation** where instances of recognized objects are indicated by highlighting specific pixels of the object. This technique gives a precise location (at the pixel level) of an object and the pixels found. The pixels produced can also be called a mask (see figure 1 (c)). Combining semantic segmentation with object detection leads to **instance segmentation**, which first detects object instances and then segments each into detected boxes (in this case called regions of interest). In other words, each object in the image gets its own unique mask even if there are other objects with the same class (see figure 1 (d)).

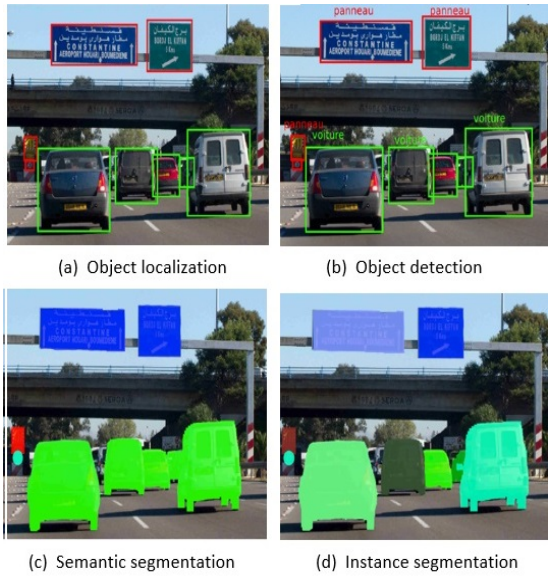

Figure 1: The different computer vision tasks.

In this paper, we will focus on object detection, where it is widely accepted that progress in this field has generally crossed two periods: "the traditional object detection period (before 2014)" and "the period of deep learning-based detection (after 2014)."[1].

During the traditional object detection period (before 2014), object detection predominantly relied on classical machine learning techniques. Among the notable methods that emerged during this period, three significant approaches are worthy of mention. These include the Viola-Jones detector, originally developed in 2001 by Paul Viola and Michael Jones [2] for real-time human face detection, and it has demonstrated its utility in diverse applications. The Histogram of Oriented Gradients (HOG), introduced in 2005 by N. Dalal and B. Triggs [3], represents an enhancement over SIFT descriptors, shape contexts, and contour orientation histograms. HOG provides a robust, scale-invariant solution. Additionally, there is the Deformable Partial Model (DPM) [4], initially proposed by P. Felzenszwalb in 2008 as an extension of the HOG detector. DPM introduced a novel strategy involving learning the components and their overall structure.

Despite the success of traditional approaches, the effort required to create effective and efficient detection models remains significant. Therefore, they have been completely replaced by methods based on deep neural networks, resulting in greater accuracy and generalization. In the era of deep learning, object detection is

grouped into two classes: "two-step detection" and "one-step detection". Typically, an object detector solves two successive tasks: finding an arbitrary number of objects (perhaps even zero) and classifying each object and estimating its size using a bounding box. Methods that combine both tasks in one step are called single-step detectors.

One-stage detectors skip the region proposal step and perform detection directly on a dense sampling of locations. They generally consider all positions in the image as potential objects and try to classify each region of interest as a background or target object.

Within the realm of single-step object detection algorithms, a noteworthy mention goes to the YOLO (You Only Look Once) family of algorithms [5], which serves as the central focus of this investigation. This approach employs a single, fully trained neural network that receives an image as input and directly generates predictions for bounding boxes and their associated class labels.

In the subsequent sections, we provide a concise overview of the three models under consideration.

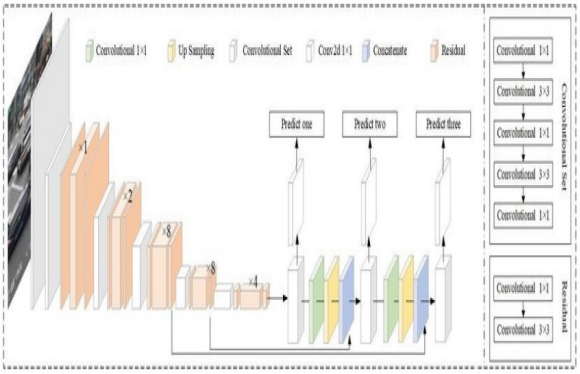

Figure 2: YOLOv3 Architecture.

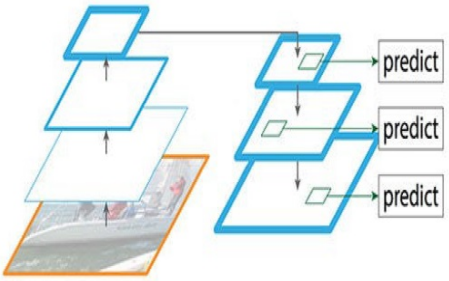

Figure 3: Feature Pyramid Network (FPN).

2.1. YOLOv3 [6]

YOLOv3 is primarily comprised of two key components: a feature extractor and a detector. The initial step involves passing the image through the feature extractor known as Darknet-53. Darknet-53 is responsible for processing the image and generating feature maps at various scales. These feature maps at each scale are subsequently directed into distinct branches of the detector. The detector's primary function is to process these multiple feature maps at diverse scales, culminating in the creation of output grids that contain objectivity scores and bounding boxes. The complete architecture of YOLOv3 is illustrated in Figure 2.

Darknet-53 integrates both residual blocks and Feature Pyramid Networks (FPNs), as illustrated in Figure 3. Serving as a feature extractor, Darknet-53 accepts single-scale images of arbitrary dimensions as input and yields appropriately scaled multi-level feature maps. This feature design enables exceptional performance across a broad range of input resolutions.

2.2. YOLOv4 [7]

Developed in 2020 by Alexei Bochkovsky, the YOLOv4 architecture features CSPDarknet53 as its backbone, which builds upon the foundation of DarkNet-53. It incorporates a CSPNet strategy, as referenced in [8], to partition the base layer's feature map into two segments and subsequently reunite them through a multi-step hierarchy. This division and reunification approach facilitates more degraded flow within the network. Following the backbone, YOLOv4 adopts PANet, as cited in [9], as a parameter aggregation method from various levels of the backbone for distinct detector levels, deviating from the FPN approach employed in YOLOv3.

Furthermore, an SPP block, as referenced in [10], is introduced for the notable expansion of the receptive field. This block effectively isolates crucial contextual features while maintaining minimal impact on network operational speed. Finally, YOLOv3 is employed as the network's head, tasked with extracting pertinent features. The figure 4 provides a clear structure of the YOLOv4 architecture.

2.3. YOLOv5

Developed by Ultralytics in 2020, this development marked a substantial enhancement in facilitating real-time object detection. The shift from Darknet to PyTorch as a framework played a pivotal role in this improvement. Darknet, known for its complexity in configuration and limited production readiness, was surpassed by PyTorch, leading to significant reductions in both training and prediction times.

The model's architecture bears similarities to YOLOv4, incorporating CSPDarknet53 as the backbone, SPP and

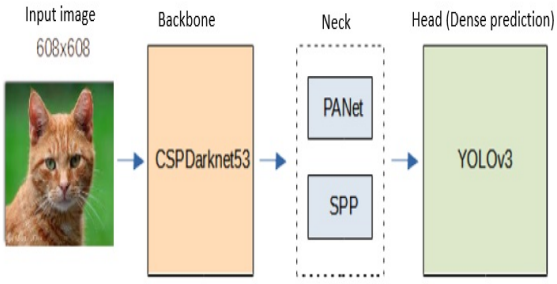

Figure 4: YOLOv4 Architecture.

PANet for the neck, and employing YOLOv3 as the head, as illustrated in Figure 5.

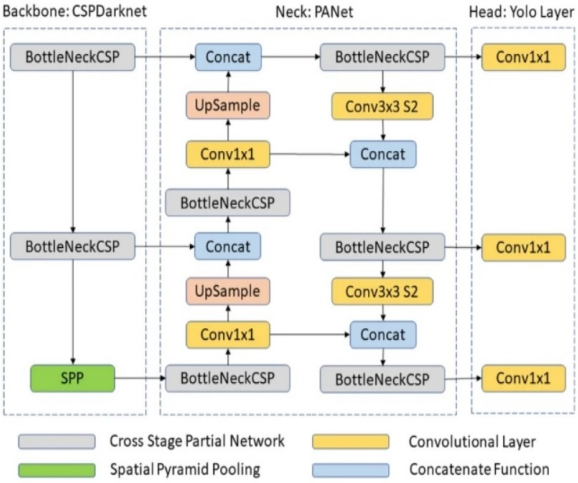

Figure 5: YOLOv5 Architecture.

3. Methodology Steps Description

This section offers an in-depth clarification of the evaluation methodology employed in this study. The research involves a comparative analysis of three deep models within the YOLO family: YOLOv3, YOLOv4, and YOLOv5. These models are single-stage detectors recognized for their fast, high-accuracy detection capabilities and are ideally suited for deployment on low-end systems, such as embedded platforms.

We commence this section by delineating the primary stage, which entails the creation and preparation of the training dataset. Subsequently, we delve into the training process for the three selected models. The section culminates with a series of tests conducted on images presenting various challenges, aimed at assessing the models' generalization aptitude across distinct usage scenarios featuring diverse content.

3.1. Creating and preparing the Training and Testing Datasets

The chosen object detection algorithms are rooted in deep learning, and their intricate architecture necessitates training on specific datasets to attain the desired objectives. The dataset plays a pivotal role in influencing the models' performance; thus, it is imperative to have a robust dataset in order to achieve optimal performance.

In the context of this research, our focus lies on the application of various models for the detection of URLs. A challenge surfaced during the preliminary phase of our study, as no standardized database was readily available for conducting a comprehensive evaluation and comparison of these models. In response, we undertook the task of creating our own dataset, comprising a total of 160 images, all of which feature URLs.

The URL starts with three consecutive letters 'w' and a dot, followed by a label. The label is a series of English letters from a to z (not case-sensitive) and can also contain digits from 0 to 9. Hyphens can be added, but not at the beginning or at the end, and adding more than one consecutively is not allowed. The label length is between 3 and 63 characters maximum. In the end, after a point, an extension is added. The most used extensions are ".com", ".net" and ".org". This part can be called the domain name.The URL can start with a protocol such as http://, but modern web clients like browsers automatically add the protocol before the URL if it doesn't contain one. The URL can also contain, after the extension name, more data, such as the filename /index.html or subdirectories like /dir1/dir2 (see figure 6).

Figure 6: URL structure.

For our image dataset, we created random tag names according to the previously listed conventions with the defined extension added at the end, which is:.com,.net,.org,.fr,.dz,.ca,.uk. These extensions are widely spread, especially in our region. We added a few URLs with additional data at the end, but for the majority of images, we focused heavily on the domain name (label and extension).

Object detection techniques exclusively process pixel-level data, which implies that they perceive distinct variations between the letter 'A' in one font style and the same letter 'A' in a different font style. Moreover, there can be substantial disparities between a handwritten letter and

its printed equivalent, despite both conveying the same semantic meaning through different visual representations (see figure 7). So, as a starting point for creating the dataset, the printed URLs are written using the popular Arial font. and for color, black is chosen.

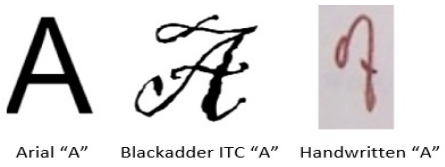

Figure 7: The letter ’A’ written by different fonts.

The majority of global printing is performed on A4 paper, and thus, our dataset will exclusively feature sample URLs that have been printed on A4 paper. These URLs will be juxtaposed with randomly generated text written in various languages, encompassing Latin, Arabic, Chinese, Russian, and Indian scripts.

After the generation of numerous simulated images, the next step consists of a manual labeling process. During this phase, each image’s linked URL box is defined and manually set. Subsequently, these annotations are stored using the YOLO image annotation format, wherein each image corresponds to an individual text annotation file, denoted by the same name as the image itself. Each line within the annotation file serves to define a ground-truth object present within the image, represented like this:

"$< objectclass >< x >< y >< width >< height >$"

This dataset format is compatible exclusively with Darknet-based versions of YOLO, namely YOLOv3 and YOLOv4, and is not compatible with YOLOv5. To accommodate YOLOv5, we adopted the Roboflow web platform, which serves as a comprehensive solution for hosting, annotating, and converting datasets across diverse formats.

3.2. Model training

The models are trained on 80 % (127 images) of all the data. The training is carried out in a Google Colab environment. The training parameters are:

- In the case of YOLOv3, the input images are set to dimensions of 416×416, and the training covers 30 epochs, accumulating a total training duration of 7 hours.
- In the case of YOLOv4, the input images are configured at dimensions of 608×608, and the training persisted for 30 epochs, amounting to a total training duration of 7 hours.
- For YOLOv5, the medium model is chosen, balancing precision and speed more effectively. The input image dimensions are set to 640×640 with 30 epochs. Remarkably, unlike its predecessors, this model required just 15 minutes to complete the training process, showcasing exceptional speed.

The training of YOLOv3 and YOLOv4 is executed within the Darknet framework, whereas YOLOv5 is trained using PyTorch. In all cases, official pre-trained weights are chosen to apply transfer learning. The final weights of the models are uploaded to the local machine to be used in the evaluation phase.

4. Model evaluation, results and discussion

The evaluation is conducted through a two-part process. In the initial stage, we assess the models’ capacity for generalization in URL detection by considering their overall performance, which involves the utilization of the entire test dataset. In the subsequent stage, we subject the three models to testing under various conditions commonly encountered in photos taken with mobile phones, thereby evaluating their stability.

In the field of object detection, the evaluation of model performance relies on several crucial metrics that provide a comprehensive assessment of a model’s object detection capabilities. In the context of this work, we have utilized the defined metrics below:

The Intersection over Union (IoU), also known as the Jaccard Index, quantifies the similarity between predicted bounding boxes and actual bounding boxes. Formally, IoU equals the intersection between the real and predicted bounding boxes divided by their union. The figure 8 clearly illustrates this concept of IoU. IoU ranges from 0 to 1; the closer the actual and predicted bounding boxes, the closer the IoU measure is to 1 (see Figure 9).

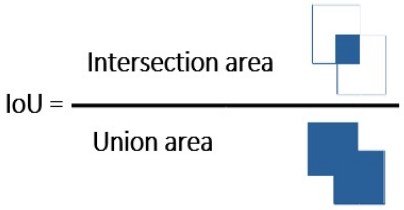

Figure 8: Intersection over Union (IoU) metric.

Precision and recall. Precision assesses the proportion of correct predictions among all positive predictions, while recall measures the proportion of true positives identified among all actual objects.

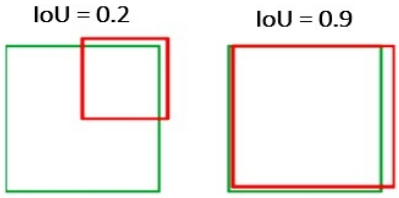

Figure 9: Examples of the Intersection over Union (IoU) metric.

Finally, the precision-recall curve illustrates the trade-off between precision and recall for different confidence thresholds, providing an overall view of the model's performance across a range of confidence thresholds.

4.1. Generalization ability evaluation

The evaluation involves a subset of 20% of the complete dataset, comprising 33 images. We have selected an IoU (Intersection over Union) threshold of 0.5 for this assessment. Results are depicted in figure **??**.

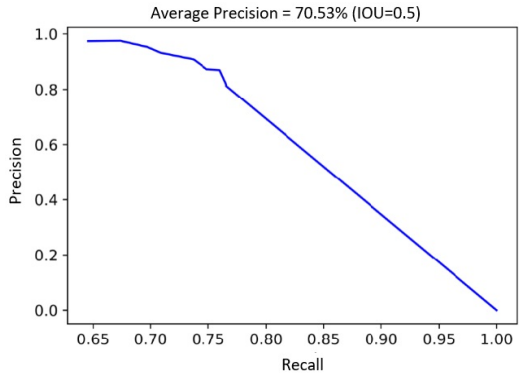

Figure 10: Precision-Recall curve (YOLOV3).

In the evaluation of our object detection models, distinct performance characteristics emerge. YOLOv3, for instance, achieved an average accuracy of **70.53%**. Notably, precision begins to decline once recall surpasses 75%.

Conversely, YOLOv4 demonstrates a notably higher average accuracy of **90.91%**, with a subsequent drop in precision observed after reaching a recall rate of 98%. As for YOLOv5, it achieves an average accuracy of **88.63%**, with precision exhibiting a decrease once recall exceeds 91%.

Despite the limited dataset size, the above observations underscore the model's ability to generalize effectively.

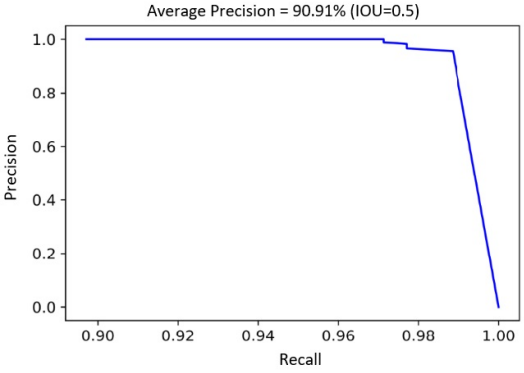

Figure 11: Precision-Recall curve (YOLOV4).

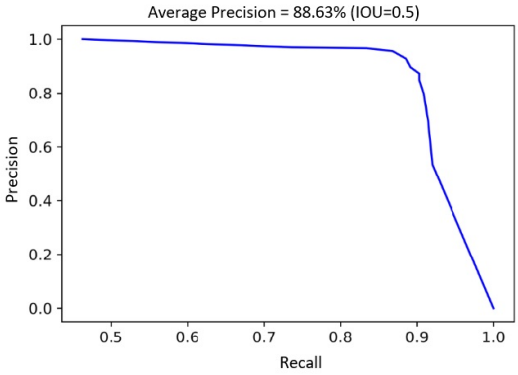

Figure 12: Precision-Recall curve (YOLOV5).

4.2. Evaluation in different conditions

In this section, we assess the model's performance using images that present challenges not encountered in the training dataset. These challenges include:

- Distinct typestyles, such as Algerian, Bradley Hand ITC, and Jokerman.
- Different background colors and character colors.
- Rotation of images of 90° and 180°.
- URLs prefixed with the https:// protocol tag.
- Handwritten URL characters.
- Images with Gaussian Noise.

The table below showcases the models' performance as measured by the average precision (AP) for each difficulty category.

From results presented in Table 1, we can draw the following conclusions:

4.2.1. Testing with distinct typestyles:

This challenge had a relatively minor impact on the models' performance, as all three models demonstrated

Table 1
Average Precision (AP) of YOLOv3, YOLOv4, and YOLOv5 for different difficulties

Difficulty	Models		
	YOLOv3	*YOLOv4*	*YOLOv5*
Colors	31.17%	54.55%	27.27%
Font	72.73%	84.29%	88.07%
HTTP Protocol	12.12%	28.9%	13.64%
Handwritten characters	0.0%	0.0%	0.0%
Rotation (180°)	27.27%	18.18%	18.18%
Rotation (90°)	0.0%	0.0%	0.0%
Gaussian noise	36.36%	37.36%	83.98%

closely aligned average accuracies. YOLOv3 yielded the lowest average accuracy of 72.7%, while YOLOv5 achieved the highest average accuracy of 88.07%, and YOLOv4 84.2%. Furthermore, figures 13, 14, and 15 provide a comprehensive representation of URLs successfully identified within text samples rendered in three different fonts.

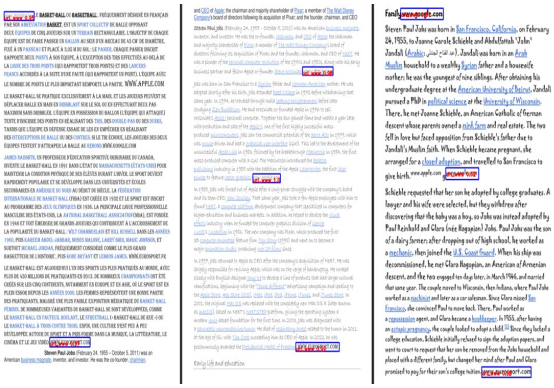

Figure 13: Different character polices (YOLOv3).

4.2.2. Testing with different color polices and backgrounds:

This challenge posed a significant impact on the models' performance. YOLOv3, for instance, exhibited the inability to detect URLs against a colored background, yet it demonstrated success in detecting URLs with colored characters, achieving an average accuracy of 31.71%. In contrast, YOLOv4 emerged as the top-performing model in this context, attaining an average accuracy of 54.44%. YOLOv4 managed to successfully detect all URLs with colored characters and a majority of URLs against colored backgrounds. Conversely, YOLOv5 displayed the weakest performance with an average accuracy of 27.27%, struggling to detect URLs with colored objects, as well as

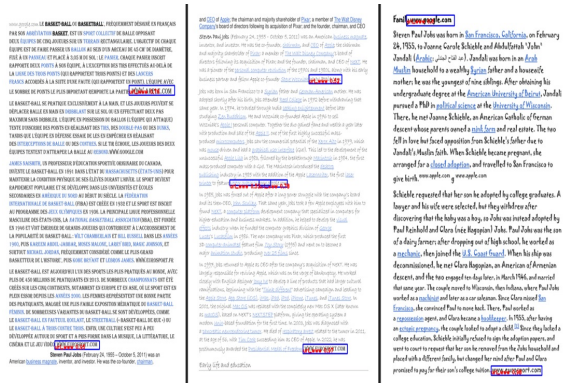

Figure 14: Different character polices (YOLOV4).

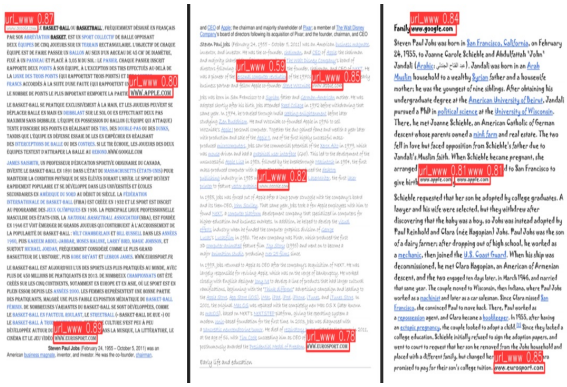

Figure 15: Different character polices (YOLOV5).

those against colored backgrounds. Additionally, figures 16, 17, and 18 provide a visual representation of the identification of URLs within text samples that feature colored characters and are placed against colored backgrounds.

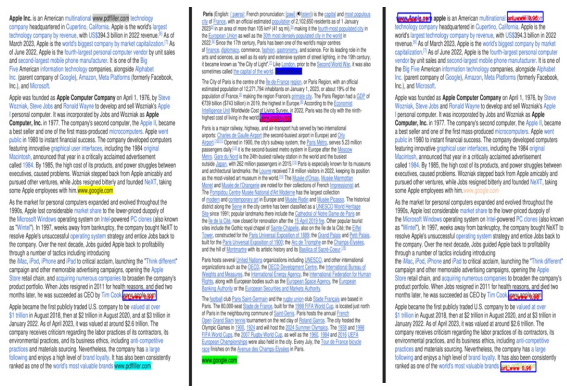

Figure 16: Different color polices and backgrounds (YOLOV3).

Figure 17: Different color polices and backgrounds (YOLOV4).

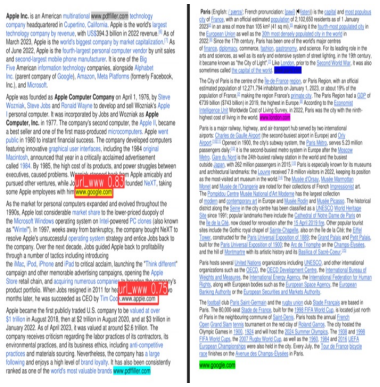

Figure 18: Different color polices and backgrounds (YOLOV5).

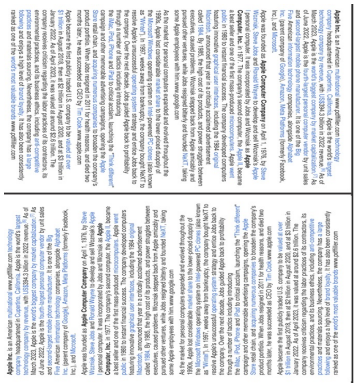

Figure 19: Image rotations -90°, 180° (YOLOV3).

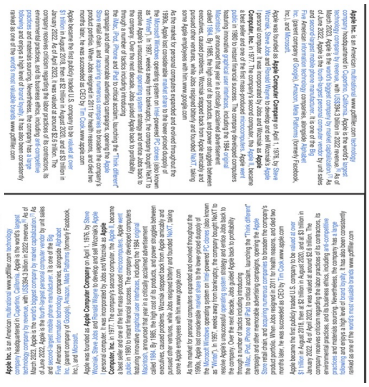

Figure 20: Image rotations -90°, 180° (YOLOV4).

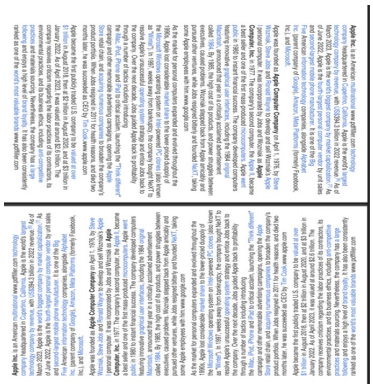

Figure 21: Image rotations -90°, 180° (YOLOV5).

4.2.3. Testing with different image rotations (90° et 180°):

Rotating the images by 180° had no notable influence on the models' performance. However, when rotated by 90°, the models' performance was significantly decreased, with all three models achieving an average accuracy of 0%. as evident in figures 19, 20, and 21.

4.2.4. URL prefixed with the http:// protocol tag:

In the case of URLs prefixed with the http:// tag, the models encountered difficulties in their detection, with some models failing to identify the complete URL, recognizing only the domain name. YOLOv3 exhibited an average accuracy of 12.12%, YOLOv4 outperformed the others with the highest accuracy at 28.9%, and YOLOv5 achieved an accuracy of 13.64%. Examples of detections for this particular challenge are illustrated in Figures 22, 23 and 24.

4.2.5. Handwritten URL characters:

In this case, all models were unable to identify the URL, resulting in an average accuracy of 0% across the board. Figure 25 provides an image depicting a sheet with var-

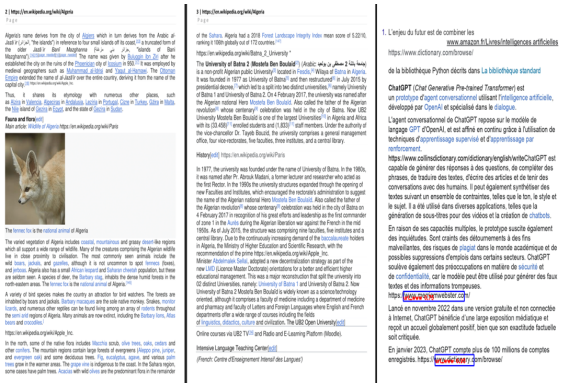

Figure 22: URL prefixed with the http:// protocol tag (YOLOV3).

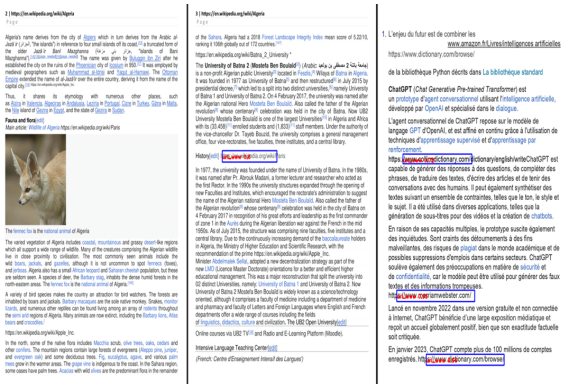

Figure 23: URL prefixed with the http:// protocol tag (YOLOv4).

Figure 24: URL prefixed with the http:// protocol tag (YOLOv5).

ious handwritten URLs that remained unrecognized by all three models.

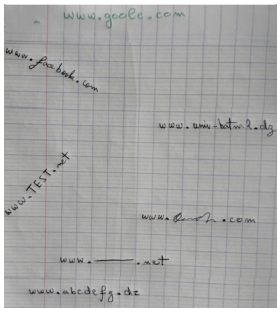

Figure 25: Handwritten URL characters (YOLOv3, YOLOv4, and YOLOv5).

4.2.6. Gaussian noise addition:

The introduction of noise had a profound impact on the models' accuracy, leading to the detection of false objects. YOLOv3's accuracy dropped to 36.36%, and YOLOv4 also experienced a decrease, with an accuracy of 37.36%. In contrast, YOLOv5 achieved the highest accuracy of 83.98% under these conditions (see figures 26, 27, and 28).

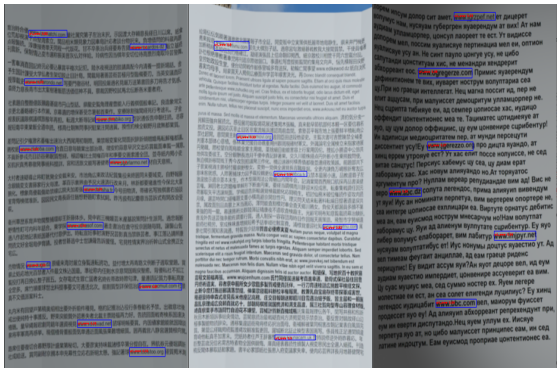

Figure 26: Images with Gaussian noise (YOLOv3).

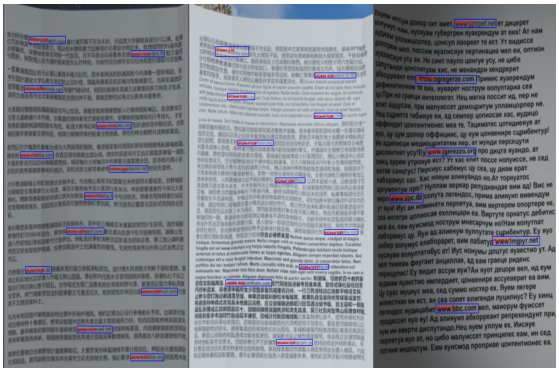

Figure 27: Images with Gaussian noise (YOLOv4).

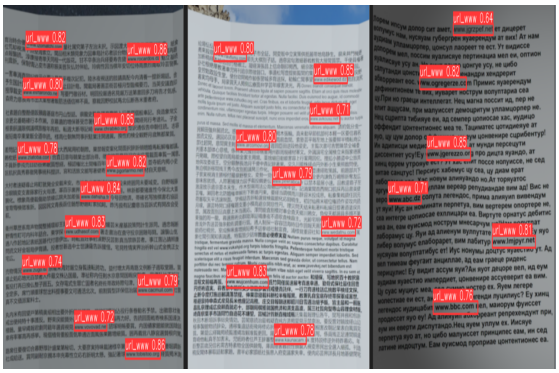

Figure 28: Images with Gaussian noise (YOLOv5).

5. Conclusion

In this study, we investigated a very interesting topic in the field of computer vision, specifically focusing on object detection, a pivotal stage in recognition processes. Our primary goal was to assess the generalization capability and robustness of three distinct models—YOLOv3, YOLOv4, and YOLOv5—in the context of URL detection within mobile phone-captured images.

The experimental results, expressed in terms of average precision, allowed us to deduce the following conclusions:

The three models gave very satisfactory generalization results, and the best is YOLOv4.

Concerning stability for several difficulties, the 3 models did not completely recognize URLs rotated by a 90° rotation angle, where the average precision achieved is 0:0%. Also, for handwritten URLs, all three models provided an average accuracy of 0.0%.

To improve these results, we propose to:

- increase the size of the dataset.
- Augment the image set with images containing different difficulties for the training dataset.
- Test other versions of the YOLO family, even other models of the two-stage detector family.

References

[1] Z. Zou, K. Chen, Z. Shi, Y. Guo, J. Ye, Object detection in 20 years: A survey, Proceedings of the IEEE 111 (2023) 257–276. doi:10.1109/JPROC.2023.3238524.

[2] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, volume 1, 2001, pp. I–I. doi:10.1109/CVPR.2001.990517.

[3] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), volume 1, 2005, pp. 886–893 vol. 1. doi:10.1109/CVPR.2005.177.

[4] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Transactions on Pattern Analysis and Machine Intelligence 32 (2010) 1627–1645. doi:10.1109/TPAMI.2009.167.

[5] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788. doi:10.1109/CVPR.2016.91.

[6] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767 (2018).

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

[8] C.-Y. Wang, H.-Y. Mark Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, I.-H. Yeh, Cspnet: A new backbone that can enhance learning capability of cnn, in: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2020, pp. 1571–1580. doi:10.1109/CVPRW50498.2020.00203.

[9] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 8759–8768. doi:10.1109/CVPR.2018.00913.

[10] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, IEEE Transactions on Pattern Analysis and Machine Intelligence 37 (2015) 1904–1916. doi:10.1109/TPAMI.2015.2389824.