Undergraduate Final Project

A Computational Approach to Screen Scratch Detection and Analysis for Improved Maintenance of LCD Terminals

Hiago Lopes Cavalcante

advised by
Prof. Dr. Tiago Figueiredo Vieira

Universidade Federal de Alagoas
Institute of Computing
Maceió, Alagoas
September 27th, 2024

UNIVERSIDADE FEDERAL DE ALAGOAS
Institute of Computing

A COMPUTATIONAL APPROACH TO SCREEN SCRATCH DETECTION AND ANALYSIS FOR IMPROVED MAINTENANCE OF LCD TERMINALS

Undergraduate Final Project submited to the Institute of Computing at the Universidade Federal de Alagoas as a partial requirement for obtaining the degree of Computer Engineer.

Hiago Lopes Cavalcante

Advisor: Prof. Dr. Tiago Figueiredo Vieira

Examining Board:

Erick de Andrade Barboza	Prof. Dr., UFAL
Ícaro Bezerra Queiroz de Araújo	Prof. Dr., UFAL

Maceió, Alagoas
September 27th, 2024

Catalogação na Fonte
Universidade Federal de Alagoas
Biblioteca Central
Divisão de Tratamento Técnico

Bibliotecário: Marcelino de Carvalho Freitas Neto – CRB-4 - 1767

C376c Cavalcante, Hiago Lopes.

A computational approach to screen scratch detection and analysis for improved maintenance of LCD terminals / Hiago Lopes Cavalcante. – 2024.

29 f. : il.

Orientador: Tiago Figueiredo Vieira.

Monografia (Trabalho de conclusão de curso em Engenharia de Computação) - Universidade Federal de Alagoas, Instituto de Computação. Maceió, 2024.

Texto em inglês.

Bibliografia: f. 28-29.

1. YOLOv8 (Detecção de objetos). 2. Cartão de crédito - Terminais (Computador). 3. Inspeção visual automática. 4. Visão computacional. I. Título.

CDU: 004.35

UNIVERSIDADE FEDERAL DE ALAGOAS/UFAL
Instituto de Computação – IC
Campus A. C. Simões - Av. Lourival de Melo Mota, BL 12
Tabuleiro do Martins, Maceió/AL - CEP: 57.072-970
Telefone: (082) 3214-1401

Trabalho de Conclusão de Curso - TCC

Formulário de Avaliação

Nome do Aluno HIAGO LOPES CAVALCANTE

Nº de Matrícula 18112108

Título do TCC (Tema)

Screen Scratch Detection and Analysis for Improved Maintenance of LCD Terminals

Banca Examinadora

2087442 - TIAGO FIGUEIREDO VIEIRA
Nome do Orientador

Documento assinado digitalmente
gov.br TIAGO FIGUEIREDO VIEIRA
Data: 27/09/2024 15:38:49-0300
Verifique em https://validar.iti.gov.br

2343385 - ERICK DE ANDRADE BARBOZA
Nome do Professor

Documento assinado digitalmente
gov.br ERICK DE ANDRADE BARBOZA
Data: 27/09/2024 17:08:43-0300
Verifique em https://validar.iti.gov.br

2995499 - ICARO BEZERRA QUEIROZ DE ARAUJO
Nome do Professor

Documento assinado digitalmente
gov.br ICARO BEZERRA QUEIROZ DE ARAUJO
Data: 29/09/2024 17:55:51-0300
Verifique em https://validar.iti.gov.br

Data da Defesa

27 / 09 / 2024

Nota Obtida

9,0 (Nove)

Observações

Coordenador do Curso
De Acordo

Documento assinado digitalmente
gov.br JOBSON DE ARAUJO NASCIMENTO
Data: 30/09/2024 22:48:59-0300
Verifique em https://validar.iti.gov.br

Assinatura

Acknowledgements

Firstly, I wish to express my heartfelt gratitude to my parents for the countless opportunities they have provided me throughout my undergraduate studies. I am deeply indebted to my family for their unwavering love and support. In particular, I would like to thank my mother, Sysleide, and my father, Hélvio, for their lifelong dedication to my personal growth and education. I would also like to extend my sincere thanks to my godparents, Luciene and José, whose encouragement and influence have played a significant role in shaping my academic journey.

I want to express gratitude for the friendships I cultivated throughout this period and prior, they were fundamental throughout the process: Luana, Bruna, João Pedro, John, Lucas Massa, Jhonnye, Hugo, Cabral, Mateus, Ruan, Marcus, Naricla, Raquel, Chrys, and Yana.

I am grateful to the entire teaching staff and other employees at the Computing Institute. I would like to express my sincere gratitude to my advisor, Dr. Tiago Figueiredo Vieira, for providing unwavering support and insightful feedback, inspiring me to strive for excellence. I would also like to thank Professor Erick Barboza for introducing me to academic research and encouraging me to continue pursuing research and developing my skills.

September 27th, 2024, Maceió - AL

Resumo

Fabricantes de terminais de cartão de crédito e débito emprestam seus produtos aos varejistas, atribuindo a responsabilidade pela reparação ou substituição dos dispositivos pós-verificação de defeitos funcionais, ou estéticos. Um operador logístico, que processa cerca de dez mil produtos diariamente, deve avaliar a condição de cada dispositivo por meio de testes e inspeções visuais manuais. Dependendo da avaliação, um terminal pode ser devolvido ao cliente ou enviado ao centro técnico do fabricante para reparo. Este processo de inspeção é altamente subjetivo e propenso a erros humanos, pelo fato de que diferentes operadores podem avaliar o mesmo dispositivo de maneira inconsistente. Para resolver essas inconsistências, desenvolvemos um sistema de inspeção visual para avaliar as condições das telas LCD desses terminais de pagamento. O hardware inclui uma estrutura impressa em 3D para otimizar a aquisição de imagens, um esquema de iluminação ativa e uma câmera para a obtenção das imagens dos terminais. A aplicação desktop integra um modelo de detecção de objetos YOLOv8, treinado com os nossos dados, para destacar defeitos na tela dos terminais, que possui por uma interface intuitiva com várias funcionalidades. Esse sistema melhorou a eficiência e a precisão das inspeções, proporcionando aos operadores uma ferramenta confiável que garante consistência. O modelo final alcançou uma precisão média (mAP) de 77,4% e uma velocidade de processamento de 88 milissegundos em CPU. Além disso, a interface da aplicação em tempo real oferece usabilidade de baixa latência, aprimorando mais o processo de inspeção. Após um rigoroso treinamento e testes do modelo, realizados em colaboração com o fabricante dos terminais, o sistema demonstrou sua confiabilidade para implantação em nível de produção.

***Palavras-chave*: Detecção de Objetos, Terminais de Cartão de Crédito, Inspeção Visual, YOLOv8, Visão Computacional.**

Abstract

Manufacturers of credit and debit card terminals lend their products to retailers, retaining responsibility for repairing or replacing devices upon verification of malfunctions or cosmetic defects. The logistic operator, which processes up to ten thousand products daily, must assess each device's condition through tests and manual visual inspections. Depending on the evaluation, a terminal may be returned to the client or sent to the manufacturer's tech center for repair. This inspection process is highly subjective and prone to human error, as different inspectors may assess the same device inconsistently. To address these inconsistencies, we developed a visual inspection system specifically for evaluating the LCD conditions of payment terminals. The hardware includes a 3D-printed structure to optimize image acquisition, an active illumination scheme, and a camera to capture images of the terminals. The web application features a YOLOv8 object detection model to highlight defects on the terminal's screen, complemented by a user-friendly interface with various functionalities. Our system significantly improved the efficiency and accuracy of inspections, providing operators with a reliable tool that ensures consistency. The final model achieved a mean Average Precision (mAP) of 77.4% and a processing speed of 88 milliseconds on CPU. Additionally, the real-time application interface offers low-latency usability, further enhancing the inspection process. After thorough model training and testing conducted in collaboration with the terminal manufacturer, the system has demonstrated its reliability for deployment at the production level.

***Keywords*: *Object Detection*, *Credit Card Terminals*, *Visual Inspection*, *YOLOv8*, *Computer Vision*.**

List of Figures

List of Symbols

α Learning rate.

$|B|$ Batch size

σ Activation Function

L Loss function.

W Image width.

H Image height.

List of Abbreviations

YOLO You Only Look Once

mAP Mean Average Precision

AI Artificial Intelligence

FPS Frames Per Second

IoU Intersection over Union

GIoU Generalized Intersection over Union

CNN Convolutional Neural Network

TFT Thin Film Transistor

LCD Liquid Crystal Display

XOR Exclusive OR

DPMs Deformable Part Models

R-CNNs Region-based Convolutional Neural Networks

LED Light-emitting Diode

GPU Graphics Processing Unit

PCB Printed Circuit Board

Summary

Chapter 1

Introduction

Engaging with the constant transformations and challenges of our contemporary world, society is influenced by new technologies that bring innovation to many fields and is affected by their impact every second. Hwang (2018) says that artificial intelligence is a key point in these scenarios since it is incorporated in multiple areas and has had many breakthroughs through research that built inventions, which has assisted many working fields, including object detection.

Object detection is an important task widely adopted in many scenarios, focused on detecting instances of objects in images, from industry prototypes to assist their workers and responsibilities to personal use through a mobile application, for example. Throughout the years, the fast evolution of deep learning techniques has improved and created many approaches to multiple challenges in object detection [Zou et al. (2023)], which has been responsible for the advances in various contemporary and traditional applications, such as autonomous driving, facial recognition, and anomaly detection.

Following the preceding, it is explicit the uses and applications of object detection using neural networks and traditional techniques for a long time. Such approaches are well perceived and will promote many new methodologies, including defect detection in many devices using computational power to identify these objects, which introduces a new outlook for the field.

1.1 Motivation

In light of object detection breakthroughs, device maintenance has been one area that has utilized these improvements, especially in repair centers where various tasks are executed manually. With the promotion of deep learning pipelines, these tasks can be accelerated and are suitable for automation, assisting operators with their reports and creating a new environment for repairing devices.

To ensure the quality of devices entering the market, cosmetic defect detection is a critical process in the industry. It is responsible for detecting any type of visual defects

in products prior and after release. These defects can be categorized as *mura*, a Japanese word meaning blemish, adopted in English to name imperfections of a display that are seemingly visible. There are multiple types of *mura* defects, such as line *mura*, dot *mura*, area *mura*, among others.

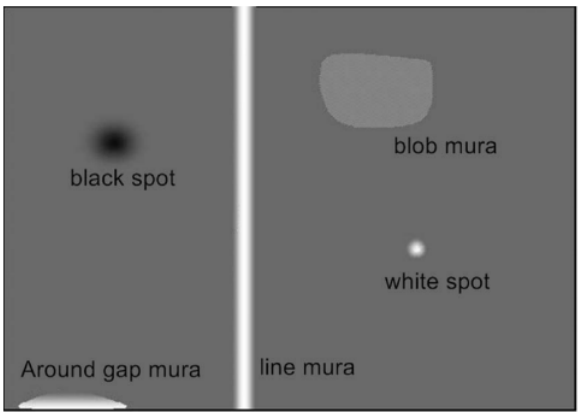


Figure 1.1: Examples of *mura* in TFT LCDs. Source: Li et al. (2012)

As these industry devices become more complex with different materials and textures, defect detection is becoming an even more intricate challenge, especially considering the specific purpose of each device in each company. Within these concepts, cosmetic defect inspection in devices appears as decisive in enhancing the efficacy of repair centers in industry as its benefits bring much value to many contexts, creating a faster and more precise solution, as much as a well-defined methodology for solving said obstacles.

Despite that, the defect detection process still decays in terms of precision, since operators might have a divergence of manual inspections on the same device. This ambiguity and the ongoing exchange of detections on the same device may also influence the pace of these procedures. These challenges inspire a necessity for a different approach to object detection, using neural networks, considering how versatile and robust it works[Zhao et al. (2019)].

1.2 Objectives

1.2.1 General Objectives

This project involves developing an application approaching the detection of scratches on powered off LCD screens of credit card machines in a desktop application, in order to provide notable data to the operator in a commercial environment.

1.2.2 Specific Objectives

1. Obtain terminals for data collection.

2. Build the booth fit for the terminals.
3. Seek and select state-of-the-art, open-source computer vision models focused on detection through a literature review.
4. Organize and train scratch detection models on screens.
5. Evaluate the trained models and integrate the one with the best exhibiting optimal performance into the application.

1.3 Work Organization

This work was organized into chapters that report the steps followed during the development of the solution, going into theoretical details and application of the technologies involved. In Chapter 1, it is described the motivation behind this work, followed by its objectives and background contexts for the development of this project. In the next chapter, it is provided a framework for understanding the subjects approached in this work, serving as the basis for our analysis and interpretation of gathered information, defined as the theoretical foundation chapter (Chapter 2). In this chapter, it is shown the concepts of neural networks, convolutions, and evaluation metrics used in this work. In succession, a detailed sequence of steps is presented, known as methodology, outlining the project's progression. This includes explanations of the tools employed, the methods for data collection and processing, the techniques for obtaining the results, the details of the model's architecture, and deployment stages. These steps were essential to ensure better reproducibility in future research and can be found in Chapter 3. Subsequently, the results and discussions chapter is introduced, presenting the findings throughout the entire project, with a description of each step and its corresponding outcome deriving from the methodology displayed in the chapter 3. Also, this chapter exhibits the issues raised during the project's duration, descriptions of the datasets obtained and their results after training, and details of the desktop application where the model was integrated. At last, there are the conclusions about this work, exhibiting the approaches and results in a summarized manner.

Chapter 2

Theoretical Foundation

In this chapter, we introduce the fundamental concepts applied in this work, including neural networks, their history, structure, and functionality of neurons. In the same prospect, we address the challenges during neural network training, in the means of detecting these problems and assessing how to resolve them. Later, the focus shifts to convolution and its application on neural networks, with convolutional neural networks, also known as CNNs, their complexity and frequent use in computer vision tasks, such as pattern recognition and anomaly detection. Furthermore, evaluation metrics will be introduced, measuring and validating the performance of the trained models, including precision, recall, and mAP. Within the scope of these topics, best practices for implementing and training neural networks are explored in this chapter as well.

2.1 Neural Networks and Deep Learning

Neural networks and deep learning have transformed multiple technological domains, achieving milestones that have impacted the society we inhabit. Deep learning can be described as a remarkable standard in machine learning, since it has outperformed some traditional techniques in domains such as cybersecurity, robotics, natural language processing, and still has a broad range of classic applications using its techniques [Alzubaidi et al. (2021)]. Deep learning models have their roots in biological systems and classic linear models. Based on artificial neural networks, these models seek to emulate the brain's structure and functionality through the training of deep multilayer networks, thereby allowing AI systems to execute intricate tasks [Wang et al. (2017)].

The first concepts of neural networks can be traced back many decades ago, inspired by the structure and functionality of the human brain, especially the biological neurons. In a biological neuron, dendrites receive input signals from other neurons, process and transmit through the axon to other neurons.

In the 1940s and 1950s, the concept of the perceptron by Frank Rosenblatt and the establishment of an initial computational model of a neuron was essential to the advance-

ments of the subject in question. A perceptron can be described as a simple artificial neuron, designed for binary classification. It operates by taking input values, applying weights to them and their sum, passing the result through an activation function, and generating an output value, similar to the biological neuron. The development of the perceptron has laid the foundation for more complex neural network architectures and has influenced the field of deep learning. By demonstrating its potential for learning, adapting, and improving, the perceptron paved the way for the creation of many deep learning models [Rosenblatt (1958)].

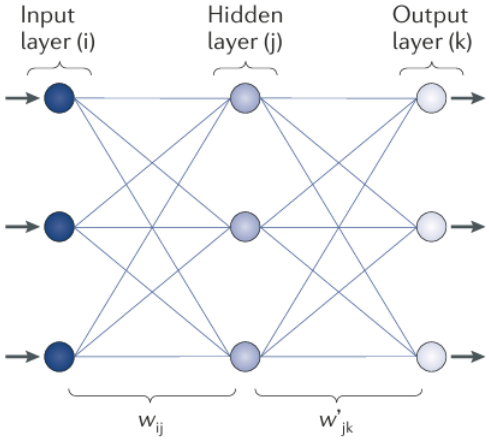


Figure 2.1: Example of a neural network, including the input layer, hidden layers, and output layer. Source: Yuste (2015)

However, a significant limitation of the perceptron is its inability to solve nonlinear problems, such as the XOR problem (see Figure 2.2), which involves a binary classification task where the output is true only if the inputs are different. Using a single-layer perceptron, the data points are not linearly separable, which means this model cannot complete this task on its own. The lack of capability of the perceptron to handle nonlinear problems highlighted the urge for more complex architectures, leading to the development of multilayer neural networks, or multilayer perceptrons (MLPs), and other architectures [Yanling et al. (2002)].

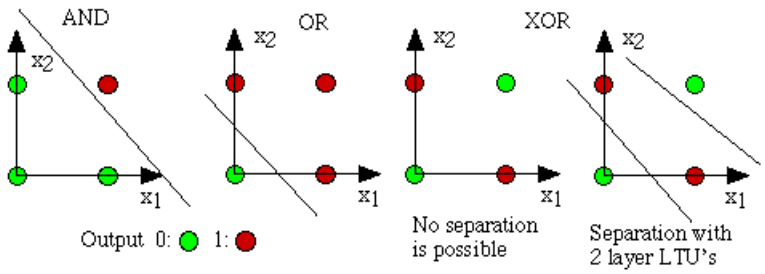


Figure 2.2: Description of the XOR problem. The separation is not possible using a linear approach. Source: Image from internet.

As a last enhancement to initial neurons, we have a neural network. A neural network

is a computational model compiled in nodes, known as neurons, and layers connecting these nodes, as seen in Figure 2.1. The workflow in this structure involves setting data in the input layer, which will pass through the hidden layers with weights and biases applied, propagating throughout the entire network and performing linear and nonlinear operations, then finally generating a result from the output layer. This pipeline describes the forward propagation and will be essential to improve the learning of the network.

Subsequent evolutions, especially the introduction of the backpropagation algorithm in the 1980s, significantly enhanced the training of these networks. The use of backpropagation in neural networks provided an efficient way of calculating the gradient of the loss function by inspecting the network weights, making error optimization easier. Specifically, the loss is propagated backward through the network to calculate gradients, applying the chain rule and computing the derivative of the loss in the means of each weight and bias. Then, using these gradients, the network can now update the weights to optimize the loss. Optimization algorithms like Stochastic Gradient Descent (SGD), as seen in Figure 2.3, are frequently used to promote these calculations [Bottou (2012)]. In summary, backpropagation aids and maintains robustness in training, ensuring that the network is learning and addressing incorrect outputs based on its previous weights.

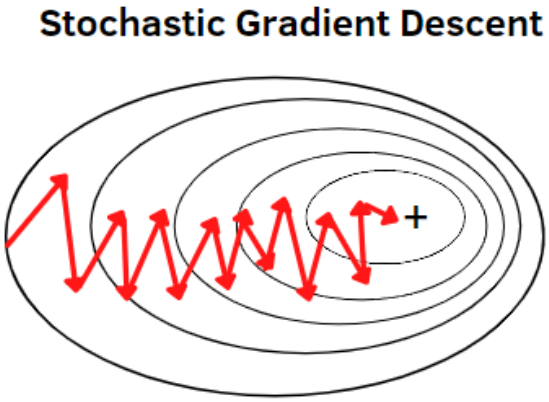


Figure 2.3: Graphical representation of Stochastic Gradient Descent (SGD). Source: Wagh (2022).

Nonetheless, it was not until the early 21st century, with the advent of powerful computational resources and large datasets, that deep learning began to thrive. These technological evolutions enabled the development of deep neural networks capable of processing and learning from large amounts of data, leading to achievements in many fields such as computer vision, natural language processing, and autonomous systems.

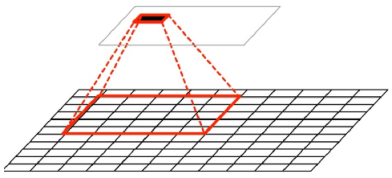
2.2 Convolutional Neural Networks

Complex problems may require complex solutions, especially when dealing with large-scale data like images, videos, or audio. In a society where data is generated and transmitted

at an exponential rate, it is necessary to develop efficient approaches to these types of data. When using high-quality images in neural networks, for example, its size usually will define the dimension of the input, and considering a black and white image of 1000×1000 pixels, it's a 10^6 size input layer. When scaling upwards, these inputs might make the learning process more complex using the classic approaches. This is when a new concept is introduced: convolution.

$$s[i,j] = (I * K)[i,j] = \sum_m \sum_n I[i+m, j+n] K[m,n] \tag{2.1}$$

A convolution is a mathematical operation that lays the foundation of convolutional neural networks. It consists in passing a kernel matrix K, usually of size $n \times n$, applying a weighted sum of each term of the kernel and each pixel of the image matrix I, as seen in the expression 2.1. When each pixel is multiplied by its corresponding term in the kernel, the sum of these calculations will replace the value of the $n \times n$ pixels over which the kernel has passed [Goodfellow et al. (2017)], as can be seen in Figure 2.4.



(a) Convolution

Figure 2.4: Example of convolution operating within an image. Source: Gu et al. (2015).

After a convolution is applied to an image, the resulting matrix representing the input image will exhibit a decrease in its dimensions. As a result, this dimensionality reduction will be one of the primary reasons for the creation and widespread usage of convolutional neural networks (CNNs), assisting models in converging results. Also, small and meaningful features such as edges can be detected by making the kernel smaller than the input. In this way, we reduce the memory and quantity of operations requirements of the model and improve its statistical efficiency. Within this context, convolutional neural networks (CNNs) are a special type of neural network for processing data that has a known, grid-like topology [Goodfellow et al. (2017)].

Generally used for problems involving images, a CNN has a specific structure dedicated to dimensionality reduction, detailed feature extraction, and spatial hierarchy preservation, which consists of three main stages: convolution, activation, and pooling. After the convolution, each modified input will pass through a nonlinear activation function, similar to a classic neural network, and then modify the output of this layer on the pooling layer. A pooling layer will be tasked with changing the value of a region of the output with a

statistic computation of the nearby values. For example, the mean pooling operator will modify the value of certain regions of the output to its mean value. If the output from the convolution is a 16×16 grid, a mean pooling might operate through windows of 2×2 values divided equally into four regions from the grid, resulting in a new grid with size 2×2, which will be passed to the next layer and so on. An example of pooling can be seen in Figure 2.5.

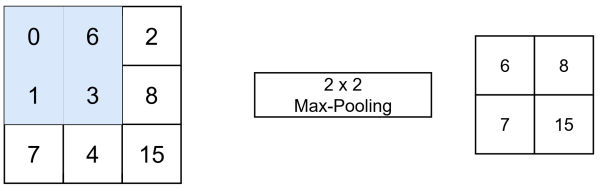


Figure 2.5: Example of pooling operation. Source: Author.

As a result of the evolution of convolutional neural networks, many architectures were developed throughout the years, with a primary focus on improved performance and minimizing inference time. Considering these advances, a new architecture arises with better performance for object detection using CNNs: **YOLO**. "You Only Look Once: Unified, Real-Time Object Detection", by Redmon et al. (2016), introduced a new approach to unified object detection, achieving good performance and better FPS compared to other state-of-the-art models at the time.

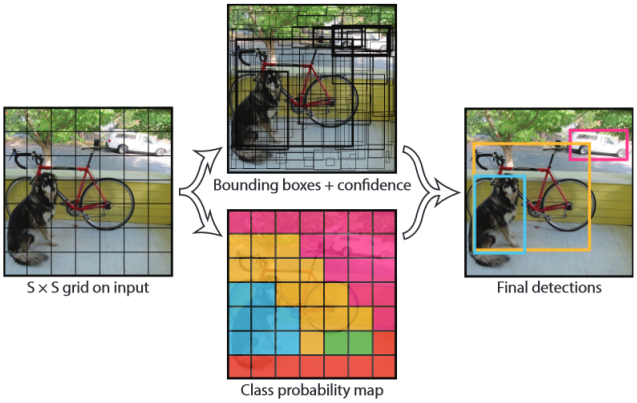


Figure 2.6: Example of the YOLO pipeline, setting bounding boxes and class probabilities to configure the final detections. Source: Redmon et al. (2016).

YOLO redefines object detection as a singular regression task, mapping image pixels to bounding box coordinates and class probabilities, which achieves high speed. Unlike sliding window and region proposal-based methods, such as DPMs and R-CNNs, YOLO conducts a global analysis of the image when making predictions and processes the entire image during both training and testing phases, inherently encoding contextual information during training. In summary, YOLO divides the input image into an S $\times$ S grid. The grid

cell in which the center of an object is located is responsible for detecting that object, as is shown in figure 2.6.

In the means of improvements, YOLO has evolved gradually since its first launch (v1). With the new versions launching, many features were introduced, improving efficiency and utility, including handling higher-resolution images, implementation of anchor boxes, changes in the backbone, new activation functions for faster convergence, new functionalities such as image segmentation and pose detection [Ultralytics (2023)]. As a major feature, it introduces multiple-scale detection, making it easier to detect objects of different sizes. Also, implemented different IoU loss calculations, such as GIoU, a new computation of Intersection over Union (IoU) that considers the size and shape of bounding boxes, which will improve the precision of detections [Li et al. (2022)].

In the Figure 2.7, it is explicited the difference in performance between the YOLO models for each version (v5, v6, v7, and v8) and size (n, s, m, l, and x). The nano model (n) is usually the fastest and lightest in parameters among the models.

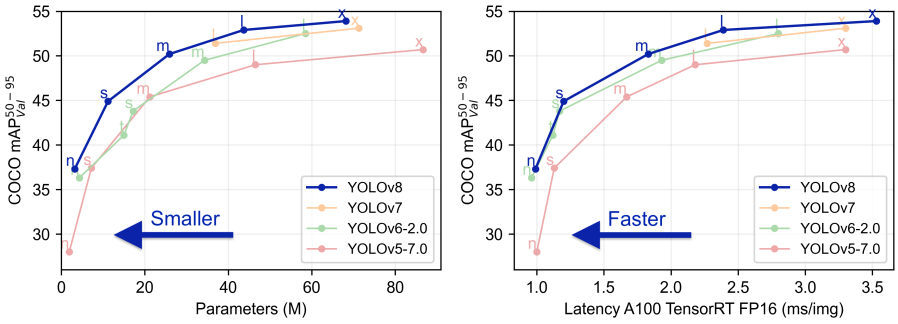


Figure 2.7: Performance comparison of YOLO versions over the years. Source: Ultralytics (2023).

2.3 Evaluation Metrics

This section introduces the metrics used in this project, aiming to demonstrate the model's performance after the training, validation, and testing phases. Among the metrics, we define precision, recall, and mAP, which are widely used in object detection pipelines. In the same prospect, we evaluate the impact of these metrics in the results, intending to assert value on evaluations and maintain the robustness of future outcomes and evaluations.

2.3.1 IoU

The Intersection Over Union metric (IoU), as seen in the expression 2.2, measures the accuracy of an object detector, where P is the predicted bounding box and GT is the ground-truth bounding box. It represents a ratio between the overlap region between the bounding boxes and the union of these areas.

$$IoU = \frac{P \cap GT}{P \cup GT} \tag{2.2}$$

2.3.2 Precision

As a metric, precision can be described as the proportion of true positive (TP) items among all positive detections made by the model, including true positives (TP) and false positives (FP). This means this metric measures the accuracy of the positive predictions along all detections. For a better description, the metric may be calculated as follows:

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

2.3.3 Recall

Recall can be described as the proportion of true positive (TP) items among all actual positive cases, including true positives (TP) and false negatives(FN). This means this metric measures how well the model correctly detects negative instances of detection. For a better description, the metric may be calculated as follows:

$$Recall = \frac{TP}{TP + FN}$$

2.3.4 Mean Average Precision (mAP)

mAP, which stands for Mean Average Precision, calculates the average precision for each class that the model can identify. In this work, mAP50 was used in major cases as an evaluative metric, which considers an intersection of 50% between the predicted and actual bounding boxes (Intersection over Union, IoU). This metric amplifies the initial concept by computing the metric across multiple object classes. For a better description, the metric may be calculated as the area under the curve of metrics cited above, precision and recall, considering a threshold on confidence. An example of the computation of this metric may be observed in figure 2.8:

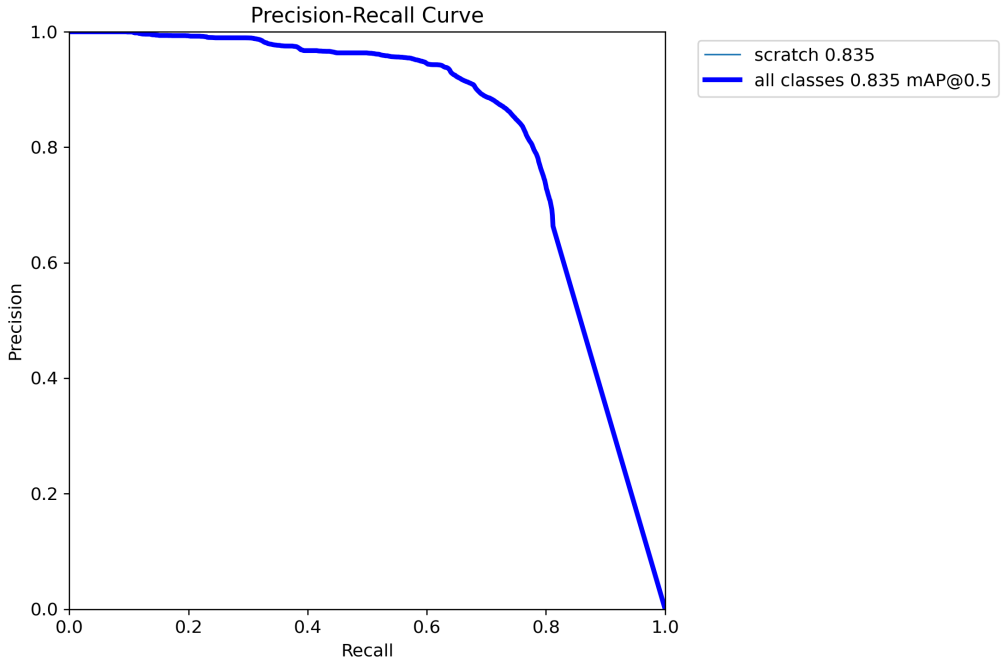


Figure 2.8: Precision-recall curve from a training using YOLOv8. Source: Author, 2023.

Chapter 3

Methodology

In this chapter, we introduce the methodology applied in this work, including a description of the hardware utilized in this project, steps for data collection, describing how the images were obtained for training and evaluation, and restrictions for dataset creation and processing. In the same prospect, a stage of data preprocessing was necessary for the construction of the dataset, as much as a description of the architecture defined for this work and training, for future analysis and integration into the solution.

3.1 Data Collection

In this section, we discuss the process of acquiring images for training, a description of the hardware used in data collection and training, and the pipeline describing all the steps for dataset creation.

3.1.1 Hardware

For the hardware for data collection, a booth was designed and built with the help of a 3D printer, aiming at the fitting of a credit card machine terminal, specifically the model Move 5000. It has a placement tailored for this model and is designed to have an interior as dark as possible, enhancing the visualization on the display. To assist the booth with better lighting, two green LED lights were placed vertically inside the booth, one for each side of the terminal.

For the data collection, a low-cost webcam is necessary and should have an easy way of integrating with the application and fitting over the booth. In the means of light management, a PCB was designed and linked with a microcontroller for lighting control. Using a PCB for this purpose assists the user by assuring a controlled environment, where the application has access to lighting control directly instead of the user.

In the means of training, a laptop and a desktop trained the models, each on their specific purpose and scale. The laptop has the GPU NVIDIA RTX 3060 and the desktop

Figure 3.1: Final design of booth designed for this project. Source: Author.

uses an NVIDIA RTX 3090. Since the desktop is built as a machine learning rig, designed for training with large datasets, consequently, it is the primary computer chosen for training in this work.

3.1.2 Pipeline for Data Acquisition

For data acquisition, the steps consisted of setting the webcam enclosed on top of the booth, turning on the lights inside, and capturing the images as pictures and videos using a computer. This procedure demands a level of caution, as any noise can interfere with data collection and corrupt the database properties.

After the images are stored on a PC, we need to annotate all video frames and images for training. This procedure requires a tool to facilitate this process, which highlights the functionalities and robustness of CVAT. Computer Vision Annotation Tool CVAT.ai Corporation (2023), also known as CVAT, is a tool designed for annotation of various sources of image data and different deep learning formats, such as PASCAL, COCO, YOLO, and onwards.

This tool makes selecting and framing bounding boxes easier through an intuitive and fast interface accessed through their website or Docker container [CVAT.ai Corporation (2023)]. In the context of annotations for the YOLO architecture, it is essential to keep consistency in the file format across the dataset. A YOLO label file is defined as a file with the input name and populated with 5-size tuples containing five numbers: class label, x center, y center, width, and height coordinates of the bounding box. These values follow a precise format where all values are normalized based on the maximum image width and height value. The calculations for these values go as follows:

$$X_{center} = \frac{X_{min} + X_{max}}{2 * W_{image}} \qquad Y_{center} = \frac{Y_{min} + Y_{max}}{2 * H_{image}}$$

$$W_{box} = \frac{X_{max} - X_{min}}{W_{image}} \qquad H_{box} = \frac{Y_{max} - Y_{min}}{H_{image}}$$

After annotating all the data, the dataset is split into two parts: training and validation. For the training section, we configure a threshold of 60 to 70% of the entire dataset to its usage, including images and labels in YOLO format, already provided by CVAT after annotation. In the same prospect, the other 40/30% of the dataset is allocated to the validation section. A threshold is outlined in this work to ensure that multiple sizes and structures of datasets were built throughout the project duration, varying the ratio of training/validation dataset sizes.

3.2 Data Preprocessing

For the data preprocessing, some procedures were executed, including data augmentation. The data preprocessing procedures were executed in the programming language Python for easier replicability and experiment testing. In the data augmentation phase, the Python library *Albumentations* operated in the dataset, alternating between changes in brightness, rotation, translation, and tuples mixing these operations.

The second approach involved creating artificial scratches on the screens of the terminals. Using a sharp tool, some scratches were made on these screens to expand the dataset by adding images to those that already existed. This approach follows a series of extra details involving some new textures of scratches [Wunsch et al. (2023)], since those from real-life usage are quite smooth and smaller compared to hand-made scratches.

3.3 Model architecture

YOLOv8 (version 8) is a state-of-the-art architecture that represents various improvements from the initial model from 2015. It presents many key innovations aiming at better speed and accuracy on detection tasks. Recognized across industries, this architecture is acknowledged for its proficiency in achieving real-time performance while preserving consistency of detection results, making it a robust solution for applications aiming for accurate object detection within image data.

This architecture implements an anchor-free model, which generates predictions from the center of an object instead of a known anchor box. It was generally used in previous versions of YOLO and, in this new version, there is a reduction of box predictions, speeding up the Non-Maximum Suppression (NMS), a usually complicated and longer process. Overall, there are some changes in the neck and bottleneck of the architecture,

including changes in convolution kernel sizes and output concatenation. One major feature launched with this version is the addition of mosaic augmentation in training, aiming at new learning of objects in different locations. This method involved stitching four images from the input data together, creating new surroundings for the placement of the detections.

3.4 Training

For the training of this model, hyperparameters are set for better comprehension of performance and evaluation for future training sessions. The parameters used are listed as follows in table 3.1. They were used in many contexts, such as the dataset used, training-validation split, computer used for training, and different lighting situations. These parameters would be chosen based on preliminary tests that indicated the optimal balance between training speed and model accuracy. The epochs and batch size were defined depending on the training device and computational power available. The other hyperparameters were set to their default values.

Parameter	**Value(s)**
Epochs	50, 200
Patience	50, 200
Batch size ($\|B\|$)	4, 8, 16
Image size	512
Optimizer	Adam
Learning Rate (α)	0.01
Activation Function (σ)	Mish Function [Ultralytics (2023)]

Table 3.1: Hyperparameters used in the model

Training files were implemented for training and evaluation of results. The programming language used was Python for easier replicability and experiment testing. Scripts for dataset creation, data augmentation, and neural network running were implemented in this language and ran on the hardware available for the task, including a machine learning rig, cited in the last chapter.

3.5 Deployment

For the deployment, the development of a desktop application integrated the trained model, using techniques for better usability and fast inference, once this application runs in real-time. To ensure clarity, this work will emphasize the applicability and efficiency of the model itself, omitting the desktop application development and assembly of hardware, while providing more detailed insights into the model's functionality.

Chapter 4

Results and Discussions

With the booth assembly, a controlled environment would be established for capturing images and, consequently, for building the database for training. Upon evaluating the booth's structure, capturing images of the terminal screens with the most stable lighting was achievable. With green lighting and a black interior, much of the noise around the terminal was reduced, allowing the lighting to highlight the screen and the scratches. Therefore, it was possible to build this image database for training with YOLOv8. As the project progressed, some terminals and screens were obtained, which could be placed and adjusted without limitations between the terminals for multiple tests and database construction for the project.

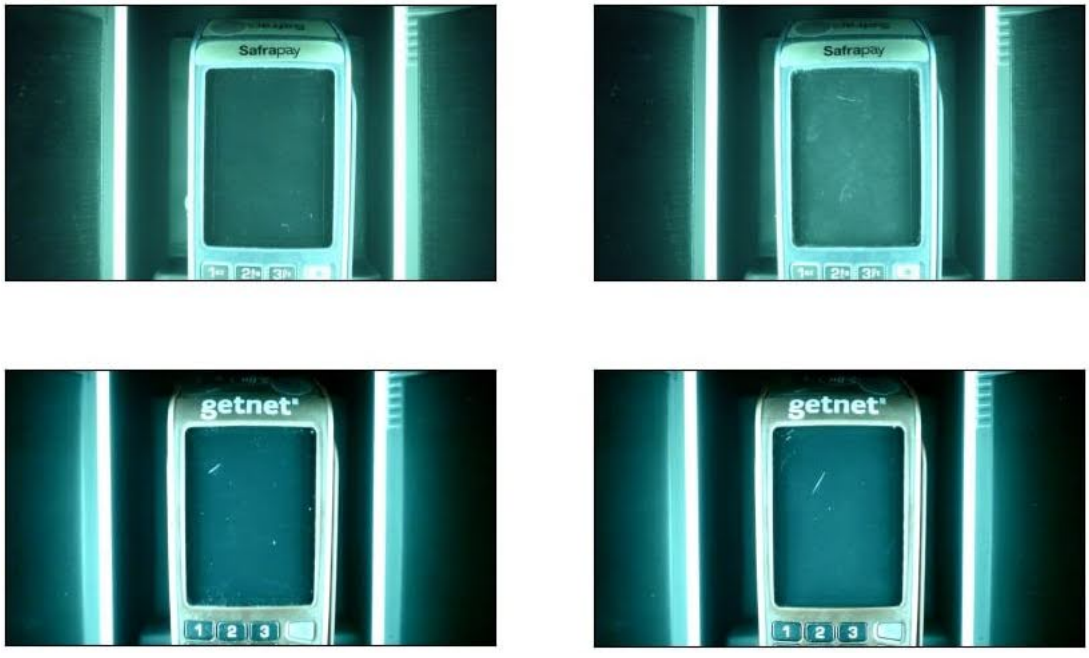


Figure 4.1: Examples of images from the database used in the project. Source: Author.

It is important to note that in the industry, there is the concept of the mura effect, which in Japanese means irregularity, encompassing various types of imperfections pri-

marily in mass-produced products. In the context of this project, scratches, stains, or cracks on the screens could be considered mura defects. Therefore, there is an abstraction of this concept for defects that appear on these devices in other forms, resulting of misuse, for example. From the large-scale use of these terminals, scratches on these screens may arise as factory defects or from user usage, with the friction of fingernails or pointed objects on the screens.

In the development of this database, one of the approaches used to expand the database and the types of scratches was to manually apply scratches to some terminal screens, obtaining a new dataset with artificial scratches. In this way, there would be a synthetic data augmentation, creating a different bias in the type of scratch found on the screens [Wunsch et al. (2023)], since most of the scratches on the screens are smaller and have a distinct shape. However, as the project progressed, greater instability was detected in the detection of scratches when a merge of databases including manual and artificial scratches was used, as the model detected manual scratches well and did not detect the original scratches as effectively.

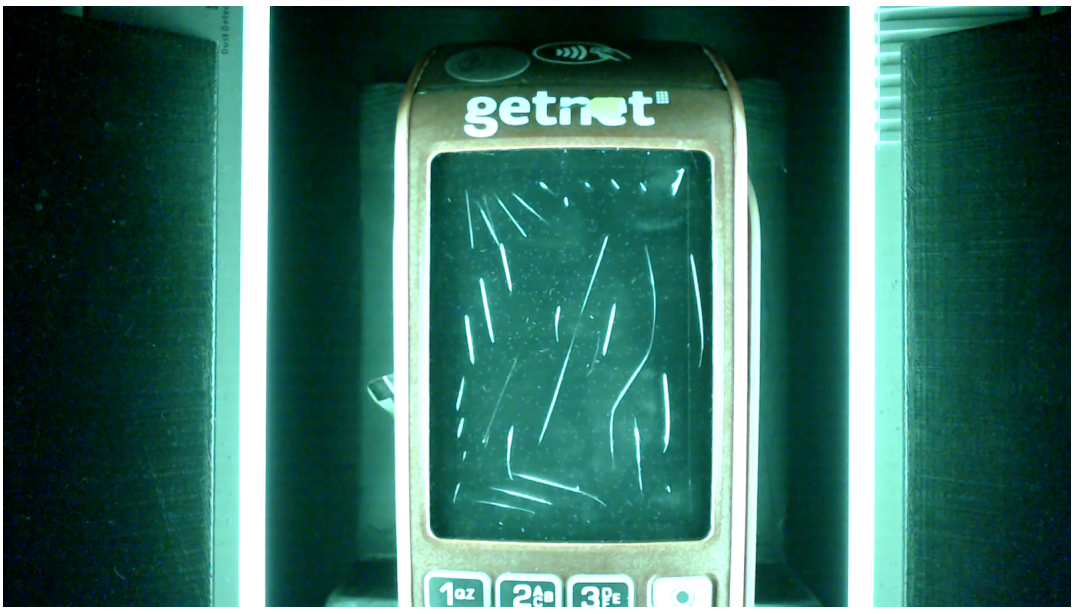


Figure 4.2: Example of the screen from the database with artificial scratches. Source: Author.

Another recurring issue in detecting scratches on the screens is their similarity to much smaller noisy objects that can be confused with scratches, such as dust. By isolating external light and maintaining controlled lighting inside, any small noise becomes very noticeable in the detection, since the majority of the image tends to be black, intensifying noise and these small objects. Additionally, a new complication in the construction of this database was the ambiguity of the concept of a scratch in detection, since, given the context presented to the webcam, there is a limitation of exposure, focus, and overall lighting that limits the visualization of what could be considered a scratch.

All collected screens had a label pointing to a scratch if there was one, which would be an initial annotation for the database, as shown in the figure below. According to the project definition, operators manually inspect and mark a possible scratch, which can be omitted by other operators who will also inspect the same screen. These situations create some ambiguity for the model. Furthermore, the concept of a scratch is also viewpoint-dependent, as scratches may be detectable from one angle and not from others. In the same prospect, this concept affects the database construction, as there are annotations to be done, and something that one operator might consider a scratch might be another type of irregularity on the screen.

Figure 4.3: Example of the screen with a scratch and its manually made annotation. Source: Author.

In the initial phases of the project, other databases were used in training to amplify the base and provide a starting point to test detection with similar scratches on real screens. One of the databases created included internet images of screens similar to those that would be used, such as cellphone and tablet screens.

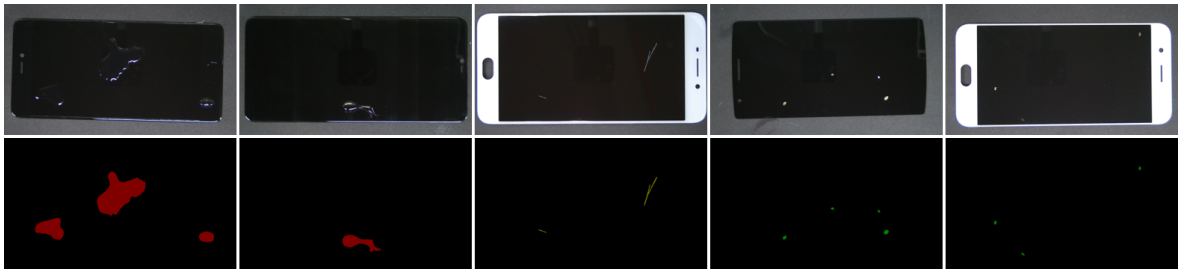


Figure 4.4: Example of an image from the database Internet **BI-400** of cellphone screens with scratches. Source: Zhang et al. (2022).

To ensure an artificial intelligence model that meets the requirements of this project,

it was necessary to conduct a review of the literature for methods and state-of-the-art approaches that are suitable for detecting scratches on screens. Therefore, in the exploration of appropriate methodologies for this problem, the chosen model was a state-of-the-art model in unified object detection: YOLO. Defining a favorable approach, it was imperative to ensure that this model would also be maintained for a considerable period, which was indeed the case: during the course of the project, YOLO had already evolved to its eighth version, thus being selected for the project.

In selecting YOLOv8, it was necessary to determine which version of v8 would be used, and for testing purposes, the nano (n) version was chosen due to its reduced size and shorter inference time. This choice aligns with the primary objective of this project, which is to assist industry operators in conducting inspections with greater efficiency and precision, this emphasizes the necessity for a real-time application. For the sake of elucidation, this model will be referred to as YOLOv8n or v8n.

With YOLOv8n, it was possible to conduct training using the data from the database assembled for the project, selecting hyperparameters, and training the model that would be utilized.

Below, the table displays the metrics achieved after training with various database scenarios and model configurations. All the training procedures were performed with YOLOv8. The metric results can be seen in Figures 4.5, 4.6 and 4.7. The terms in the table are as follows:

Dataset name	**Description**
BI-400	Cell phone screens with scratches, as cited in Figure 4.4. This dataset is consistent and well-structured, with stable lighting and few variations in the screens between images. Source: Zhang et al. (2022).
Internet-ALL	Images of scratched screens of cell phones, tablets, and other various devices from the internet with screens similar to those of the terminals.
Parts-AR	Screens defined for the project with artificial scratches.
Parts-SA	Screens defined for the project without artificial scratches.

Table 4.1: Description of each distinct dataset.

Model	Precision	Recall	mAP50
Parts-AR - VF[1]	0.89	0.717	0.822
Parts-SA - UNI[2]	0.57	0.51	0.547
BI-400 + Parts-SA - UNI[2]	0.855	0.52	0.615
BI-400 + Parts-AR - VF[1]	0.82	0.83	0.87
BI-400 + Internet-ALL + Parts-AR - UNI[2]	0.822	0.53	0.609
Parts-SA + Parts-AR - UNI[2]	0.624	0.596	0.662
Parts-SA - VF[1]	**0.839**	**0.654**	**0.774**

Table 4.2: Model performance on distinct datasets.
Inference time of the best model was 88ms on average.
[1]VF: The images are derived from video frames recorded for a maximum of 10 seconds from each screen, resulting in a larger dataset.
[2]UNI: one image per screen, resulting in a dataset with fewer than 100 images.

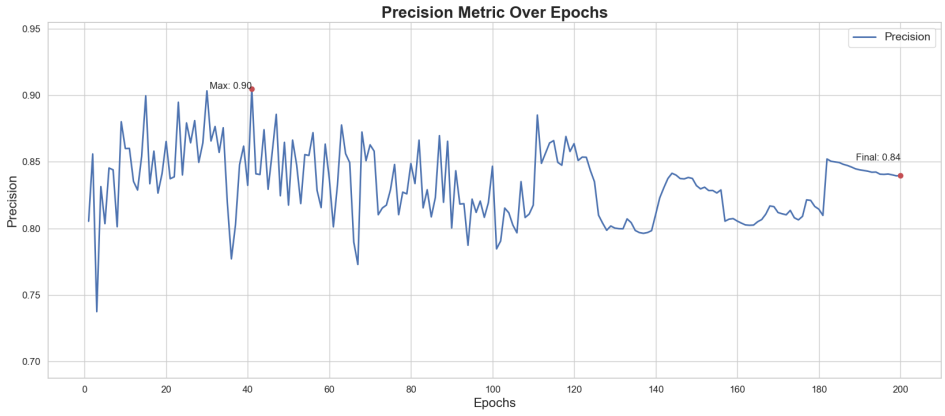


Figure 4.5: Precision curve of the best training. Source: Author.

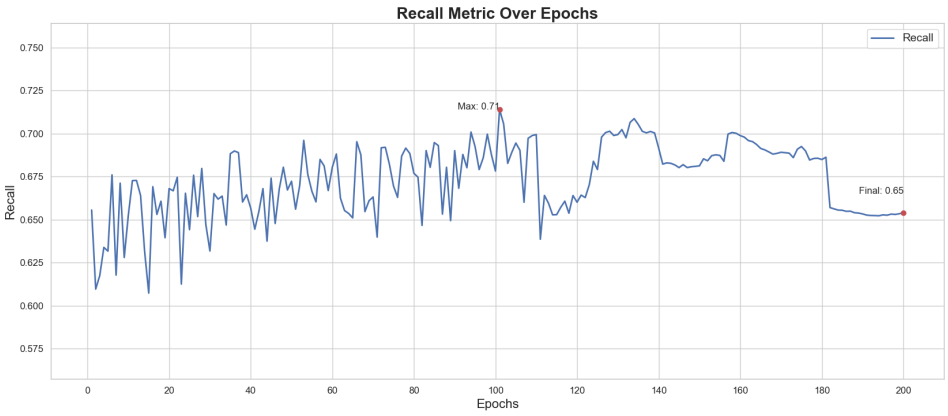


Figure 4.6: Recall curve of the best training. Source: Author.

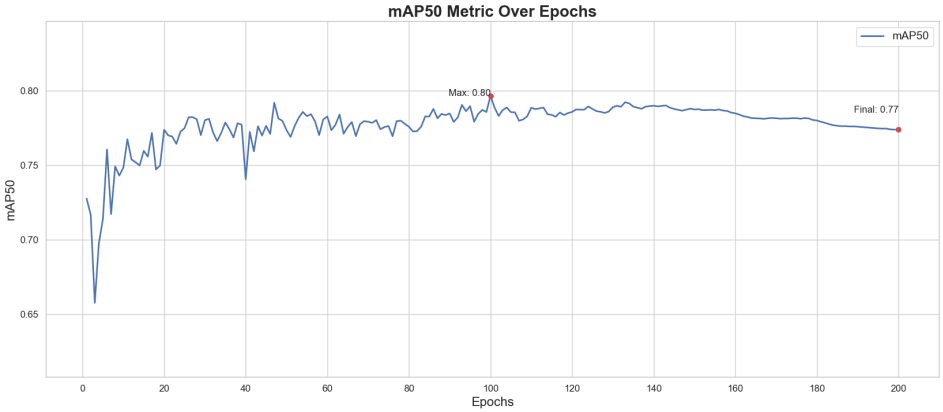


Figure 4.7: mAP50 curve of the best training. Source: Author.

The primary issue in this project was the challenge posed in dataset construction, in the means of having a reduced availability of terminals necessary to conduct training. Since this issue persisted during the greater part of the project's duration, different approaches were considered to address this problem. The dataset merging approach, cited in table 4.2, was used as a method for generalizing the best performance from well-defined, but not context-specific data, such as BI-400 dataset, and features from real specific data for the project, as in Parts-SA dataset. This approach was not as successful, once the validation stage exposed an undermining performance when the ratio of context-specific data for this project increased. From this training, the model might not have learned the characteristics from the real scratches as much as the non-context-specific scratches, especially when removing the BI-400 dataset, for example, retraining and obtaining a worse performance.

On the other hand, the approach using video frames, cited in table 4.2 as well, asserted the issue involving the small dataset, once the quantity of data was defined by the quantity of terminals, and securing new terminals would be a more complex and unlikely task. When capturing the images from the credit card machines, videos with a maximum length of 10 seconds and 30 *FPS* were recorded, generating a new and artificial dataset, containing the same characteristics from the unique terminal images, now incremented with more frames from the same terminals. After recording all these videos, an annotation pipeline started, followed by replicating these annotations based on the first frame of each video. At the end of this pipeline, 57 videos composed the dataset, along with 6306 images and their labels.

For this project, it was implemented a desktop application that leverages the power of neural networks by integrating our neural network model. By incorporating this model into the application, users can benefit from the capabilities of deep learning for scratch detection. Also, this application features a user-friendly interface designed to be intuitive and easy to use. The interface includes a range of tools and features that enable users to

capture and inspect images, as well as view and analyze detected scratches. For example, users can adjust the sensitivity of the scratch detection algorithm, change the camera settings, and other features, as seen in Figures 4.8 and 4.9. This level of customization shows that the application can be effective in a wide range of scenarios.

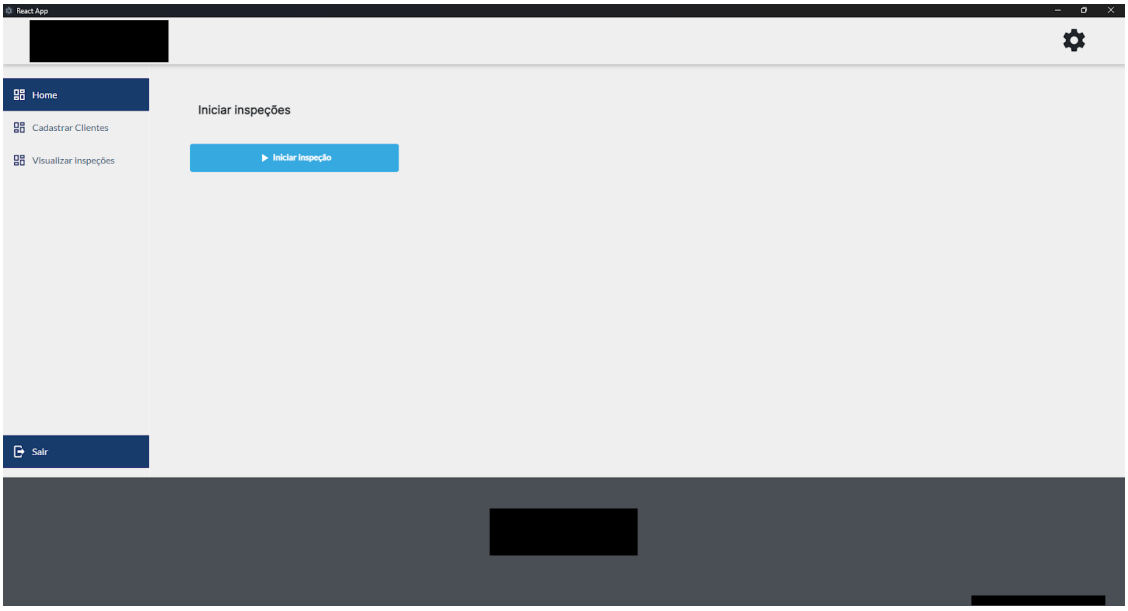


Figure 4.8: Home screen of application. Source: Author.

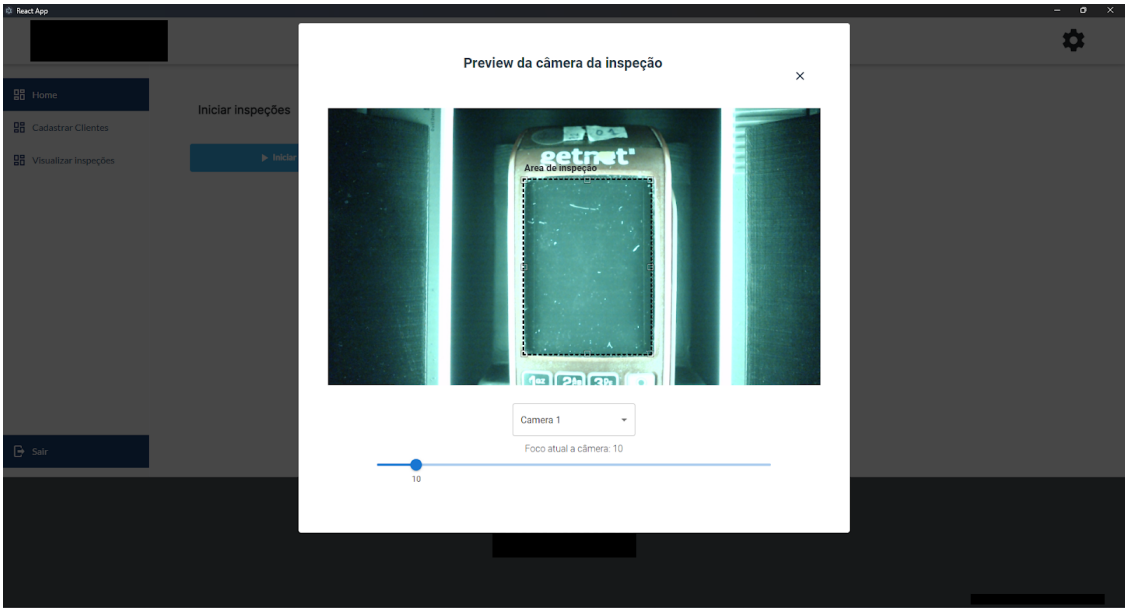


Figure 4.9: Camera settings screen on application. Source: Author.

The YOLO model is effective in detecting scratches due to its ability to accurately process images in real time and identify objects. This aspect with the selection of this model is essential, since this application should generate accurate responses in real-time. The speed of inference directly impacts the user's experience, as delays may lead to frustration and decreased productivity. Also, the reliability of the detections generated by the

model is essential for making decisions before generating a report. In summary, real-time functionality further enhances the application's value by providing as close as immediate results. Therefore, the combination of speed, accuracy, and real-time performance is imperative to meet user expectations and ensure the application's practical utility in demanding environments. An example of one inspection's result can be seen in Figure 4.10.

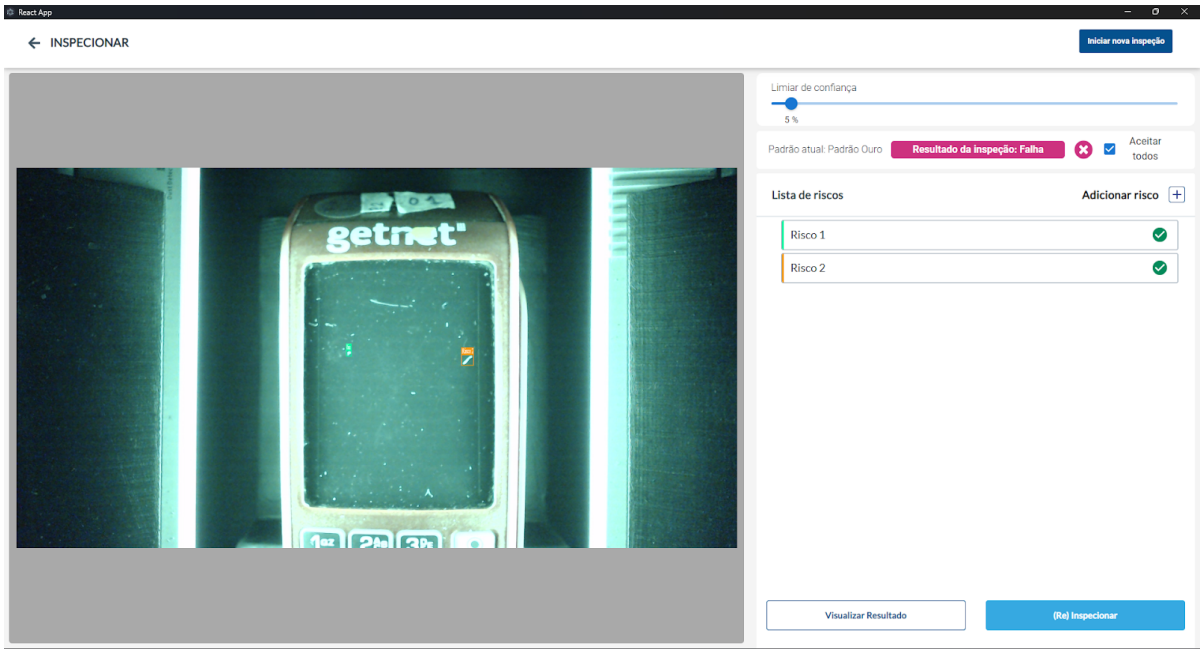


Figure 4.10: Example of inspection from application. Source: Author.

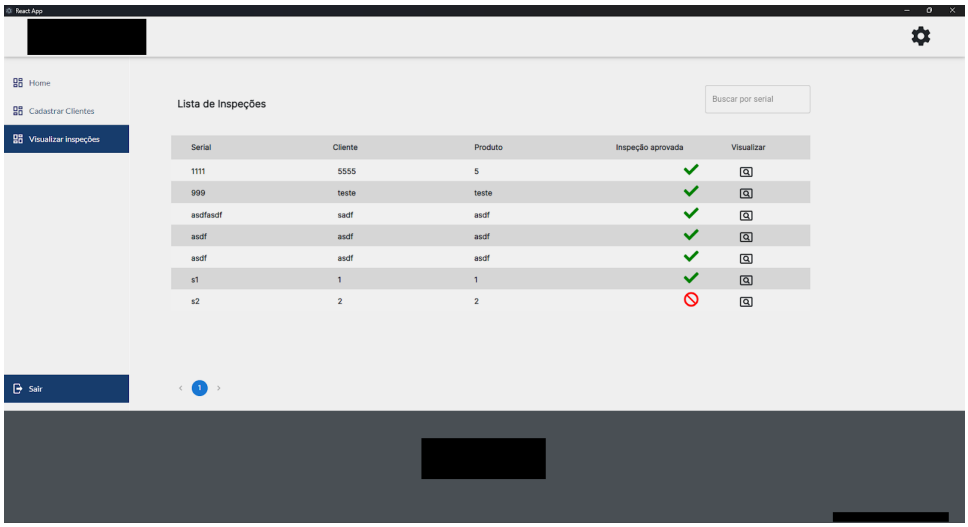


Figure 4.11: Screen with old inspections on application. Source: Author.

After the application obtains an inference from the model and the user populates the application with data from both the client and the recently inspected device, such as the client's name and type, terminal diagnostics, final evaluation, and terminal condition, a

PDF report is generated containing this data (check Figure 4.12). The visualization of the report might also be explored and analyzed later on the other page, incorporating old inspections and reports, as seen in Figure 4.11.

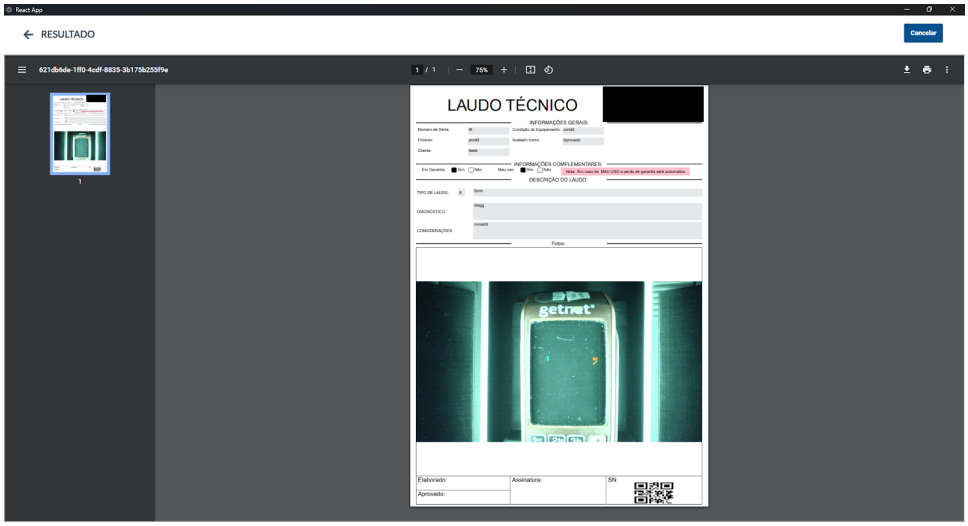


Figure 4.12: Example of a report generated by the application. Source: Author.

In the field of computer vision, one of the persistent challenges is achieving accurate object detection in video streams, especially when dealing with intermittence—where objects temporarily disappear or get occluded. This issue is critical in many applications, especially in real-time applications, where continuous and reliable tracking is primordial. After many trainings, it was detected that our model had a certain intermittence on specific occasions, from tilted terminal positioning to differences in lighting. After many image preprocessing operations, such as white balancing and morphology treatment, this intermittence would not be reduced.

In this project, this challenge is addressed with BoT-SORT [Aharon et al. (2022)], a state-of-the-art tracker designed to enhance neural network models by reducing intermittence. BoT-SORT employs sophisticated algorithms that predict and maintain object trajectories, even through periods of occlusion or temporary disappearance. By integrating BoT-SORT with the predictions of our model, it was possible to achieve robust bounding box tracking. This not only improves the reliability and accuracy of the neural network's performance but also ensures that the model can be effectively utilized in real-world scenarios where uninterrupted tracking might be necessary.

Upon further analysis, the results reveal the notable impact of the dataset containing screens with artificial scratches during training, as well as the effect of incorporating the BI-400 dataset, as its omission leads to a considerable decrease in performance. After using recordings of the terminal screens, the performance of the models increased significantly in the precision, recall, and mAP metrics. For the integration of the final model into the application, the tracker amplified the model's consistency and robustness, achieving a

refined model for the user in the final version of the application. One example of inspection from the application after these treatments can be seen in Figure 4.13.

Figure 4.13: Resulting image and detection obtained from the model. Source: Author.

Conclusion

This work involves the creation of a solution capable of cosmetically inspecting equipment for screen scratches. This project aims to assist in the inspection process of such equipment in repair centers through computer vision tools. The data obtained during inspections is useful for approving or rejecting a specific piece of equipment, thereby providing valuable information to support decision-making regarding the inspected equipment.

From a technical standpoint, alterations in architectures are observed that can enhance the application's stability and performance, contributing to making the continuous development process more sustainable. Furthermore, new state-of-the-art models in the field of computer vision are constantly emerging, introducing a new quality standard to be achieved. On the other hand, the continuous collection of data during the use of the application in production can be employed to retrain the models, resulting in a greater specialization of the tool in its domain of operation.

Finally, it was necessary to conduct tests and consider expanding the hardware capacity to determine the conditions for deploying the application in a future version that can handle a larger number of scenarios and equipment. In other words, analyzing the scalability of the product to apply it not only to a single piece of equipment but to an entire equipment portfolio.

Since this is a cosmetic inspection of the screens of these devices, a limiting factor in implementing this type of solution was the difficulty in collecting data to train the vision models that met our requirements, requiring incremental data collection to improve the system's stability.

In addition to providing a commercial application for computer vision tools, research of this kind plays a crucial role in establishing connections between industry and scientific-technological production. Given the broad current relevance of this topic, the application of artificial intelligence to practical problems addresses one of the fundamental missions of the research and innovation field: integrating intellectual production into people's daily lives. Following the completion of this work, the content was revised and adapted for publication as a scientific article, submitted and accepted in the 2024 XIV Brazilian Symposium on Computing Systems Engineering (SBESC) conference [Cavalcante et al. (2024)], reinforcing the relevance of this project, as much as contributing to the field of computer vision and anomaly detection. At the same time, the practical implementation

of knowledge developed in academia provides invaluable experience, often resulting in new ideas and indicating promising directions and opportunities for future research.

Bibliography

Aharon, N., Orfaig, R., and Bobrovsky, B.-Z. (2022). Bot-sort: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*.

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., and Farhan, L. (2021). Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of big Data*, 8:1–74.

Bottou, L. (2012). Stochastic gradient descent tricks. pages 421–436.

Cavalcante, H., Alves, D., Amaral, L., and Vieira, T. (2024). Screen scratch detection on credit card payment terminals for logistics optimization. In *2024 XIV Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 1–6.

CVAT.ai Corporation (2023). Computer Vision Annotation Tool (CVAT).

Goodfellow, I., Bengio, Y., and Courville, A. (2017). *Deep learning*, volume 1. MIT press Cambridge, MA, USA.

Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J., and Chen, T. (2015). Recent advances in convolutional neural networks. *ArXiv*, abs/1512.07108.

Hwang, T. (2018). Computational power and the social impact of artificial intelligence. *arXiv preprint arXiv:1803.08971*.

Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., et al. (2022). Yolov6: A single-stage object detection framework for industrial applications. *arXiv preprint arXiv:2209.02976*.

Li, T.-Y., Tsai, J.-Z., Chang, R.-S., Ho, L.-W., and Yang, C.-F. (2012). Pretest gap mura on tft lcds using the optical interference pattern sensing method and neural network classification. *IEEE Transactions on Industrial Electronics*, 60(9):3976–3982.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.

Ultralytics (2023). YOLOv8: A state-of-the-art real-time object detection system. `https://docs.ultralytics.com`. Accessed: 7th August 2023.

Wagh, A. (2022). Gradient descent and its types.

Wang, H., Raj, B., and Xing, E. (2017). On the origin of deep learning. *ArXiv*, abs/1702.07800.

Wunsch, L., Anding, K., Polte, G., Liu, K., and Notni, G. (2023). Data augmentation for solving industrial recognition tasks with underrepresented defect classes. *Acta IMEKO*, 12(44):1–5.

Yanling, Z., Bimin, D., and Zhanrong, W. (2002). Analysis and study of perceptron to solve xor problem. *The 2nd International Workshop on Autonomous Decentralized System, 2002.*, pages 168–173.

Yuste, R. (2015). From the neuron doctrine to neural networks. *Nature Reviews Neuroscience*, 16:487–497.

Zhang, J., Ding, R., Ban, M., and Guo, T. (2022). Fdsnet: An accurate real-time surface defect segmentation network. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3803–3807. IEEE.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., and Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232.

Zou, Z., Chen, K., Shi, Z., Guo, Y., and Ye, J. (2023). Object detection in 20 years: A survey. *Proceedings of the IEEE*, 111(3):257–276.