OPEN ACCESS
International Journal of Innovations in Science & Technology
RESEARCH & INNOVATION DIVISION
IJIST

Tablet Guard: Load Cell based Quality Assurance with Image Processing

Sana Arshad*[1], Zainab Mustafa[2], Noman Mansoor[2], Shazia Kamran[2], Mariam Syed[2]
[1]Electronic Design Center, Department of Electronic Engineering, NED University of Engineering & Technology, Karachi, 75270, Pakistan
[2]Department of Electronic Engineering, NED University of Engineering & Technology, Karachi, 75270, Pakistan
*** Correspondence:** Sana Arshad sana@neduet.edu.pk

Citation | Arshad. S, Mustafa. Z, Mansoor. N, Kamran. S, Syed. M, "Tablet Guard: Load Cell based Quality Assurance with Image Processing", IJIST, Vol. 7 Issue. 1 pp 581-602 June 2025
DOI | https://doi.org/10.33411/ijist/202571581602
Received | Feb 28, 2025 **Revised |** March 22, 2025 **Accepted |** March 26, 2025 **Published |** March 28, 2024.

Every week, the pharmaceutical business manufactures thousands of pills, each of which must be thoroughly checked before being distributed to customers. The proposed Tablet-Guard project addresses this issue through innovative integration of multiple advanced technologies including load cell technology, artificial intelligence, and a servo motor-based removal mechanism for pharmaceutical quality assurance. The system incorporates deep learning-based image processing, coupled with a load cell using an HX711 module to inspect and assess the quality of each tablet in a blister strip as it moves along the conveyor belt. It inspects defects including irregular shapes and incomplete blister strips. The utilization of YOLOv8 enables real-time defect detection with high accuracy (mAP of 0.995), enhancing efficiency and minimizing production line disruptions. By accurately detecting and addressing defects such as broken, missing, or cracked tablets within blister strips, the system significantly minimizes the likelihood of substandard products being distributed to consumers.

You Only Look Once	YOLO
Artificial Intelligence	AI
Convolutional Neural Networks	CNNs
Discrete Fourier Transform	DFT
Residual Neural Network	ResNet
Region-Based Convolutional Neural Network	R-CNN
Liquid Crystal Display	LCD
Serial Data Pin	SDA
Serial Clock Pin	SCL
Pulse Width Modulation	PWM
Direct Current	DC
Inter-Integrated Circuit	I2C
Programmable Gain Amplifier	PGA
Analog to Digital Converter	ADC
Single Shot MultiBox Detector	SSD
Feature Pyramid Network	FPN
Mean Average Precision	mAP

Keywords: Artificial Intelligence; Yolo; Pharma; Deep Learning and Arduino.

Introduction:

Pharmaceutical drugs are produced on a large scale to meet industry demands, with certain disease treatments requiring precise formulations and rigorous quality control in tablet manufacturing. Throughout the production process, drug manufacturers must ensure that the correct dosage, chemical composition, and weight are accurately maintained and properly packaged [1]. However, drug quality can be compromised due to contamination, degradation, or defects during the production process. The consumption of substandard medications can lead to severe adverse effects in patients. Currently, most tablet and blister pack inspections are performed manually, which is a labor-intensive process prone to errors [2]. Human inspectors, despite their meticulous efforts, are susceptible to fatigue and subjectivity, leading to the oversight of defects and inconsistencies in medication packaging. Common concerns include counting errors, labeling mistakes, and variations in tablet weight, all of which can potentially jeopardize patient safety [3].

The pharmaceutical sector has faced challenges in meeting stringent regulatory requirements, and the limitations of manual inspection raised concerns about the reliability and efficiency of quality control. A load cell, as the name implies, is a device mainly a transducer that generates an electrical signal proportionate to the force being measured. This instrument detects strain and subsequently turns force into energy, serving as a standard for researchers and professionals. [4]. A load cell can be utilized to automatically detect changes in the weight of a defective blister strip. Image processing plays a crucial role in visual inspection automation. AI and Machine Learning technologies have revolutionized pharmaceutical quality assurance, providing advanced tools for defect detection and anomaly identification in blister packaging [5]-[6]. Machine Learning models, particularly those based on Convolutional Neural Networks (CNNs), enable the analysis of vast image datasets, identifying subtle variations and irregularities in tablet appearance and packaging with high precision.

Literature Review:

Several studies on defect detection in blister strips have been reported in the literature, some of which are discussed below. In [7], the authors have utilized various techniques including sectorization of Discrete Fourier Transform images for feature extraction, counting method, density distribution, Gray-level co-occurrence matrix, and area-based identification of capsule boundary. Using each of these computer vision techniques, Kekra et al. presented and compared the detection of defects and their types in images of curative tablet strips having multiple kinds of capsules. Another defect detection algorithm has been proposed in [8], which includes an image preprocessing process combining median filtering and threshold segmentation. However, since the system is based on predefined rules and features, it may face challenges in identifying novel or unexpected defects that fall outside its programmed parameters. Additionally, real-time applicability is not mentioned. In [9], the authors have proposed a method based on machine vision and deep learning for defect identification. However, they have utilized ResNet, which is mainly designed for image classification, not object detection. If defects occur in varying locations, YOLO or Faster R-CNN may be more effective. The design of a deep learning system that identifies faulty tablets during the manufacturing process is presented in [10]. This system does not classify or address various defect types but rather compares different detection techniques. Another paper presents the sorting of pills as tablets and capsules. They further classify tablets based on their shapes and capsules based on their colors [11]. Moreover, they have also focused on the detection of defects in count, cracks, and variations in size and shape. However, since template matching and morphological operations are used, the system might be highly sensitive to lighting conditions, shadows, and noise and they also do not incorporate real-time inspection. In [12], R-CNN is used for the detection of a certain pill among different pills. However, R-CNN is an older and slower deep-learning approach that involves multiple steps. Additionally, the study does not address complex tablet defects. Another work that presents the detection of tablets is carried out using a capsule neural network [13]. This method is slow due to the overhead introduced by the routing protocol algorithm. Most of the reported studies rely on machine learning and deep learning techniques for automating pill inspection. However, many of these studies utilize older deep-learning techniques, resulting in slower processing speeds. Additionally, some studies did not implement a real-time system but instead relied on static images for their experiments. The current study, on the other hand, utilizes the latest deep learning method, YOLOv8, for identifying defects in tablet strips. It also shows real-time monitoring of the subject over a conveyor belt which further proves its efficacy for automated industries.

Objective:

The proposed project "Tablet Guard System" involves the design of an image processing system to find the defects in tablets. A camera system was arranged that captures high-resolution images of each tablet, and a sophisticated image processing algorithm to analyze these images in real-time. A load cell mechanism is integrated with the microcontroller to test the weight of the blister strips and alert on LCD when an undefined weight occurs. The blister strips move on a self-designed simple mechanical system including a conveyor belt. Defective pieces will be removed by an automatic system. This project significantly enhances the accuracy of quality control by improving product quality while reducing production cost as manual labor needs to be paid while this system will be implemented only once. The idea aims to modernize quality assurance practices and reduce human interference in the manufacturing and production industries.

Novelty:

The novelty of the proposed Tablet-Guard project stems from its innovative integration of cutting-edge technologies. It combines load cell technology for precise weight measurement, artificial intelligence for intelligent decision-making, and a servo motor-driven mechanism for efficient tablet removal, all working together to enhance pharmaceutical quality assurance.

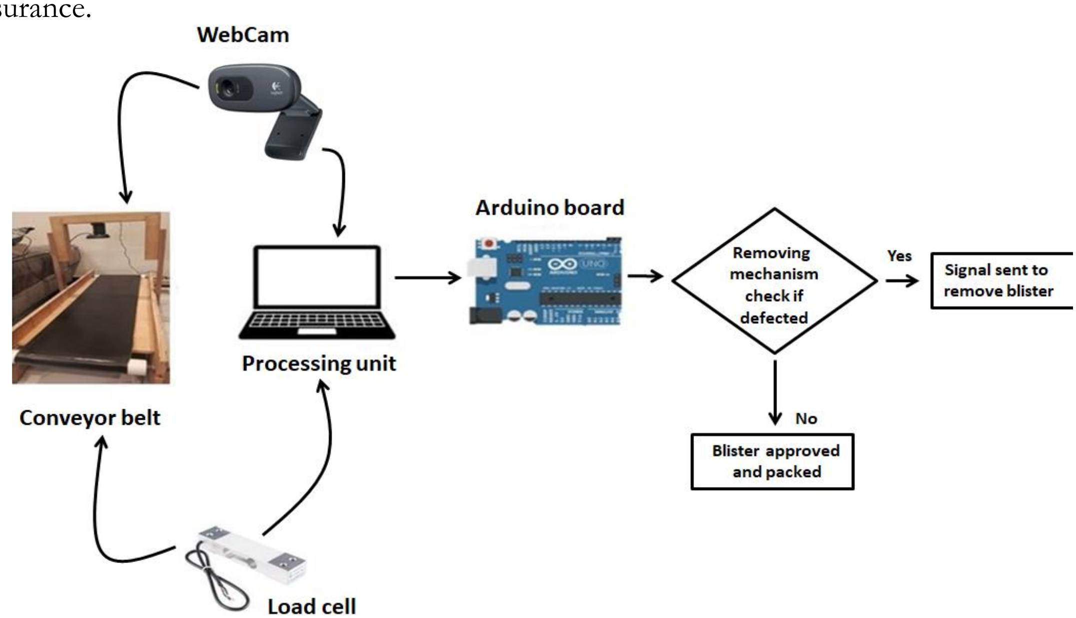


Figure 1. Block diagram of the Tablet Guard System

Simplified Working of the Tablet-Guard system:

The block diagram in Figure 1 illustrates the workflow of the Tablet Guard quality assurance system. The blister strips are initially placed on a conveyor belt system equipped with a mounted camera. As the belt moves, they pass through the inspection system for analysis. The load cell weighs each strip and sends the data to the processing unit, which determines whether any tablets are missing or wrongly weighed. Simultaneously, the camera captures images of the blister strips, which are evaluated by an algorithm running on the processing unit to detect broken or cracked tablets.

The Arduino board acts as the controller, receiving signals from the processing unit based on analysis results. If the algorithm finds a defect or if the microcontroller identifies inaccurate weight measurement, the microcontroller sends a signal to the Arduino, which then activates the servo motor-based removal mechanism. This ensures faulty blister strips are automatically rejected from the conveyor belt.

If no faults are found, the blister strips continue along the conveyor belt to the packing stage, where they are authorized and packaged for distribution. This automated procedure ensures that only high-quality pills reach the final packing stage, adhering to strict quality control criteria.

Methodology:

Arduino Connections:

For the proposed project, the Arduino Uno R3 was selected as the required microcontroller. The Uno has 14 input output digital pins and 6 input output analog pins. The HX711 load cell amplifier's excitation and output lines are connected to the Zemic Load Cell, while the supply and ground lines are connected to the Arduino's 5V and GND pins, respectively. The HX711's data and clock pins are wired to Arduino pins D4 and D5 as shown in Figure 2. A standard 16 x 2, I2C LCD (HD44780) is connected to the Arduino. The Arduino powers the I2C LCD using its 5V pin. The operating voltage of this display is 5V and it draws around 20-30 mA of current, which is well within the Arduino Uno's 5V pin capacity. The LCD's SDA and SCL pins draw around 100 µA - 1 mA each and consume very low power and thus are suitably connected to the A4 and A5 pins of Arduino respectively. An external regulator regulates the speed of the conveyor belt, which is driven by a DC gear motor and a DC adaptor.

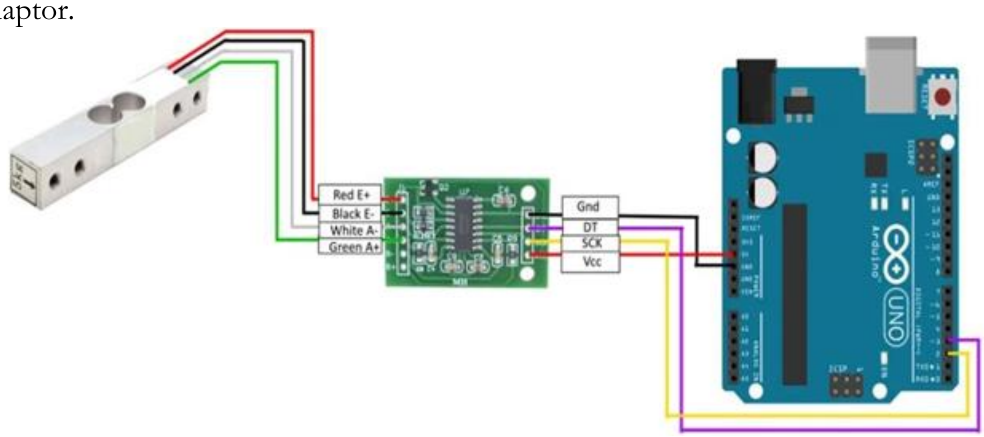


Figure 2. Load cell connections with HX711 & Arduino

A servo motor (SG90 9g), utilized for a removal mechanism, is controlled by an Arduino PWM pin (for example, D9). The Arduino integrates all components, including the load cell, servo motor, and I2C LED, and establishes power and ground connections to ensure correct performance. This arrangement ensures a well-organized system for monitoring and operating the conveyor belt and accompanying mechanisms as shown in Figure 3.

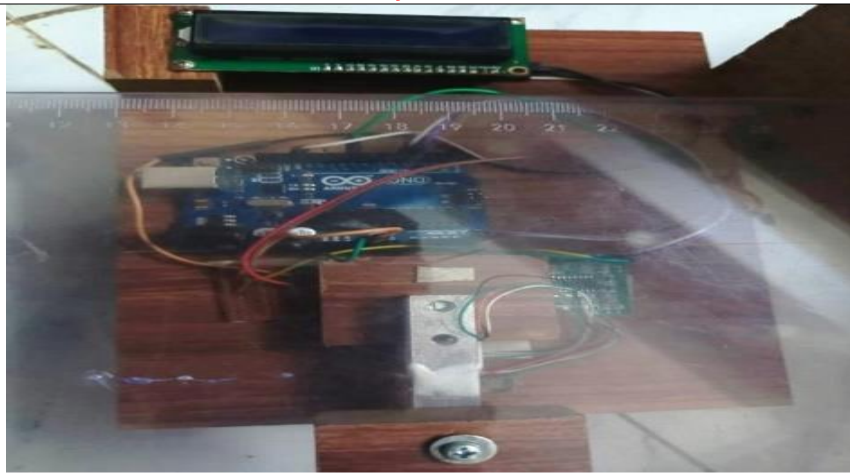


Figure 3. Overall Arduino connections

The Conveyor Belt System:

A conveyor belt system is a mechanical setup typically used in industrial settings for automated and systematic movement of items, from one point to another. As shown in Figure 4, our system uses two pulleys: the head pulley, which drives the belt and connects to the motor shaft via the head bearing, and the tail pulley, which redirects the belt, maintains tension, and houses internal tail bearings. Both pulleys feature lagging to prevent slippage. The belt, made of leather, measures 90cm long and 20cm wide, is driven by a 12V DC gear motor, whose shaft is coupled with the head pulley. A Mini PWM DC speed controller module having DC-DC voltage = 4.5V-35V, 5A current with 90W power consumption is used to regulate the gear motor at a constant speed (shown in Figure 5). The PWM speed controller is powered via a 12 V, 1A adaptor as its operating voltage (12 V) and current requirements (5A) are well beyond the limits of Arduino. The output terminals of the controller connect to the DC gear motor powering it up and operating it at the desired constant speed. The speed is adjusted using a knob (potentiometer). This constant speed is carefully determined by noting the processing time of load cell measurement, image processing, and servo motor response. A potentiometer in the motor control circuit allows for manual speed adjustments. By matching the speed to the slowest processing stage, we ensured smooth operation without any bottlenecks. The system's frame has a height of 20cm, a length of 100cm, and a width of 30cm. A 12V- 1A DC adaptor supplies power. A C270 HD Logitech HD Webcam provides sharp and smooth video quality of 720p/30fps. The camera captures real-time video of blister strips moving on the conveyor belt.

Working of Loadcell with HX-711 Module and Arduino:

The HX711 module employs a built-in PGA, which amplifies the small analog signal received from the load cell. The amplified signal is then converted into a 24-bit digital value using its ADC [14]. In our project, nuts and bolts were used to secure the load cell to the base, accounting for the slight bending that could occur when weight was applied. The setup was part of an Arduino Weighing Machine utilizing the HX711 Load Cell. For the circuit assembly, refer to Figure 2, which provides the interfacing details of the HX711 Load Cell with the Arduino Uno microcontroller. To make the entire path operational, it is essential to standardize the load cell with the HX-711 ADC and Arduino. Calibration involves placing a 100g weight when prompted on the LCD during setup. Once the 100g weight was applied, the calibration was completed, enabling accurate weight measurements with 99.9% precision. The load cell detects the load and transmits an analog voltage to the HX711 Module, which then converts it into a digital signal. This value is sent to the microcontroller, where the ADC result is translated into load terms. The final output is displayed on a 16x2 LCD screen, providing an accurate measurement of the applied weight. Additionally, the load cell detects weight inconsistencies, while the object detection algorithm identifies and counts defective pieces.

The HX711 needs 5 V and approximately 1.5 mA to operate. Thus, it is easily powered by Arduino. The load cell is then powered up by HX711. Thus, the load cell is also indirectly powered by Arduino via HX711.

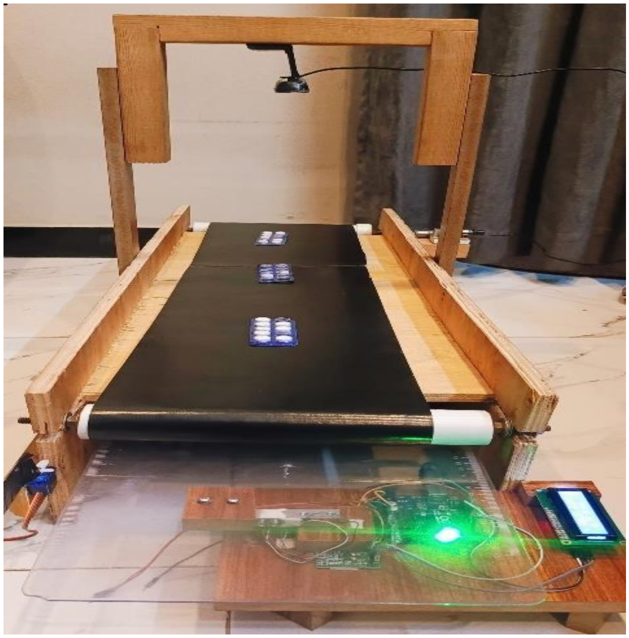


Figure 4. Conveyor Belt System

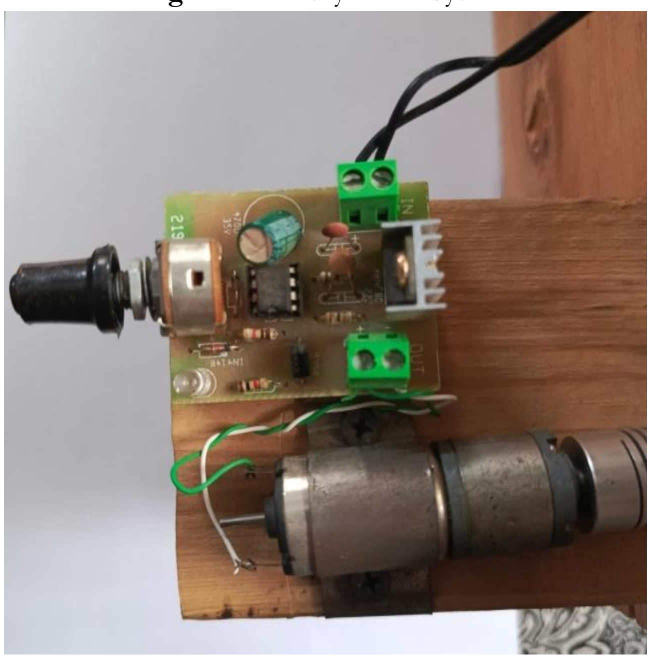


Figure 5. DC gear motor and speed controller

Servo-Motor Connections:

Since our removal mechanism is a simple plastic bag and not a heavy load, the required torque is low. Therefore, the simplest and most common SG90 servo is used. It operates at 4.8-6V and draws around 100-500 mA of current. However, Arduino Uno's 5 V pin (powered via USB or onboard regulator) can provide around 400-500 mA in total. If the servo motor takes too much current, it may reset the Arduino or can even damage it. Therefore, the power supply setup of the project includes a 5V battery dedicated to the servo motor. This leads to a stable connection for both the servo motor and the Arduino board. The VCC connection of the servo is connected to the 5V battery and its ground is connected to the power supply ground as well as to the Arduino ground to make a common ground. The servo signal pin is connected to an appropriate PWM pin on the Arduino, such as D9, as mentioned earlier.

Selection of Algorithm and Tool:

A Comparison of YOLOv8, SSD, Faster R-CNN and RetinaNet:

There are several advanced object detection models available today including YOLO, SSD, RetinaNet, and Faster R-CNN each with different strengths in speed, accuracy, and complexity. This section provides an overview of these models highlighting their key features and differences to understand their suitability for various applications.

YOLO has emerged as an important model for object detection [15]. It is a single-stage object detection model in which class probabilities and bounding boxes are predicted in a single step. Their outstanding speed allows these models to work for real-time inference; however, they sometimes sacrifice accuracy in comparison to other models [16].

Another single-stage object detection model is **SSD** which performs detection at multiple scales within a single network. It is an intermediate option between the high speed of YOLO and the high accuracy of Faster R-CNNs, which makes it a reasonable choice for several applications [16].

Faster R-CNN is a two-stage object detection model that identifies the regions of interest first and then classifies them. Although the accuracy of Faster R-CNNs is high, the two-stage process limits their speed [16].

RetinaNet is also a single-stage object detection model that offers high accuracy and efficiency. Its primary novelty is the use of Focal Loss with FPN which is a type of architecture used in object detection models to handle objects of different sizes more effectively. Moreover, it uses two subnetworks to address the issues of class imbalances [17]. Similar to YOLOv8, it has fast inference time [17][18]. A modified version of RetinaNet with a ResNet-50 backbone achieved 24 frames per second (FPS) on an NVIDIA RTX 2080 Ti GPU. It's important to note that actual FPS can differ depending on specific configurations and computational resources [19].

A more detailed comparison of the above models can be found in [16]. A comparison of the above models including their FPS values is provided in Table 1 [16]. The latest YOLOv8 model, with its capability to process 40-155 frames per second, enhanced accuracy, high speed, reduced background errors, ability to analyze the entire image at once, user-friendly interface, and streamlined usability via a command-line interface, perfectly aligns with our project's requirements.

Thus, this model was employed for real-time defect detection in our case. The model's efficient grid-based approach, which predicts class probabilities and bounding boxes within each cell, further enhances its overall performance.

Table 1. Performance comparison of different object detection models [16][19]

Model	Frames Per Second (FPS)
YOLOv8	40-155
SSD	22-46
Faster R-CNN	5-7
RetinaNet	24

Object detection:

Creating a Yaml File:

A YAML (Yet Another Markup Language) file contains details such as the dataset's path, classes, and other relevant information. In our case, the YAML file was created by the name "data_custom".

As shown in Figure 6, the "train" and "val" sections specify the paths to the training and validation datasets, respectively. The "nc" section lists the number of classes in the model, which in our case is set to three. The "names" section represents the labels of the classes displayed in the model's output. We have used "broken", "cracked" and "missing" as our class names.

As speed is significant in real-world applications, therefore, we have selected the YOLOv8 model, which offers a balance between speed and accuracy. The Ultralytics library, commonly linked with YOLO models for object identification, uses these dependencies to perform a variety of computer vision and machine learning tasks. Figure 7, shows its installation. For labeling our images, we selected the Roboflow software. Ultralytics and Roboflow are used together in the machine learning workflow as shown in [20]. Therefore, Roboflow was installed simultaneously. Its successful installation ensures that the required libraries and dependencies are available, enabling users to perform tasks such as data annotation, model training, and deployment within the notebook. The installation process is shown in Figure 8.

Dataset:

Collecting datasets is a crucial step in training AI models because these models learn patterns and make predictions based on the information they are exposed to during training. In our scenario, various blister strips were collected, including damaged pieces (such as empty pockets and irregular shapes) as well as undamaged ones. The dataset was generated by capturing snapshots of these strips. A total of approximately 800 images were captured and randomly divided into two data sets: one for defect detection and the other for strip detection. For defect detection, the training set included 736 images, the validation set included 38 images, and the remaining images were used for testing. Similarly, for strip detection, 640 images were used for training, 28 for validation, and the remaining images were reserved for testing.

data_custom.yaml X

```
names:
['broken' , 'cracked' , 'missing']
nc: 3
roboflow:
  license: CC BY 4.0
  project: blister_strip_detect
  url: https://universe.roboflow.com/blister-strips/blister_strip_detect/dataset/1
  version: 1
  workspace: blister-strips
test: /content/blister_strip_detect-1/test/images
train: /content/blister_strip_detect-1/train/images
val: /content/blister_strip_detect-1/valid/images
```

Figure 6. YAML File

Annotating Data Set on Roboflow:

Figure 9(a) shows the process of labeling defects which includes a yellow box for broken, a red box for cracked, and a purple box for a missing tablet. Figure 9(b) illustrates the labeling process for detecting blister strips by drawing a bounding box around the strip.

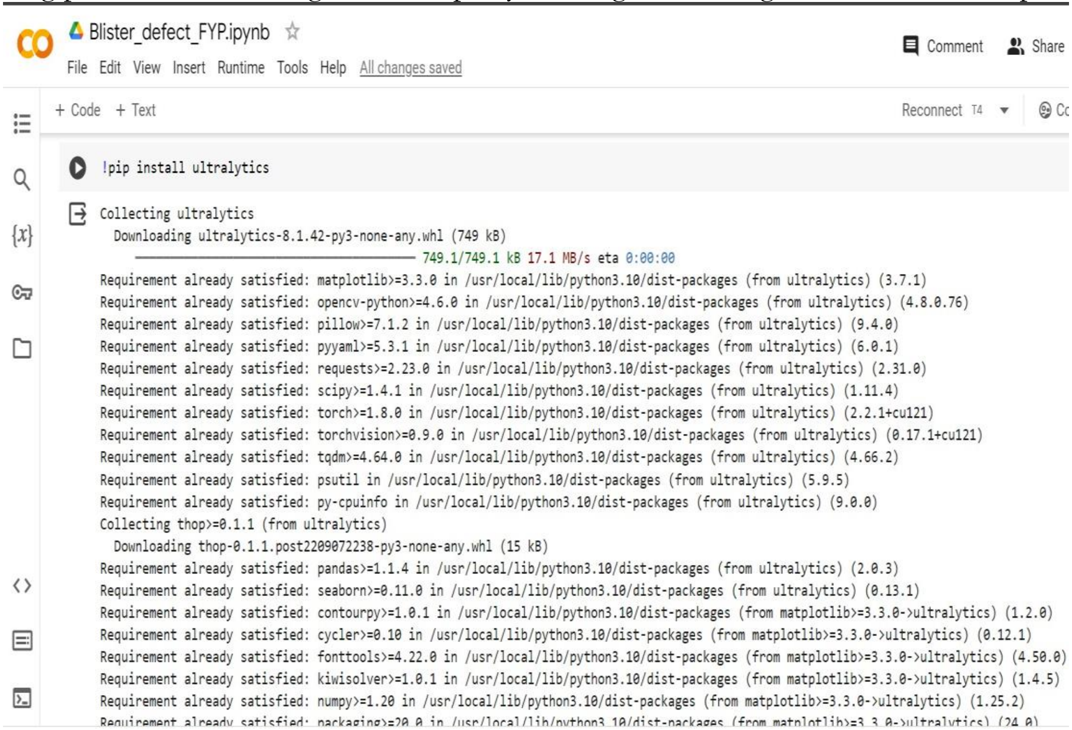


Figure 7. Installation of Ultralytics library

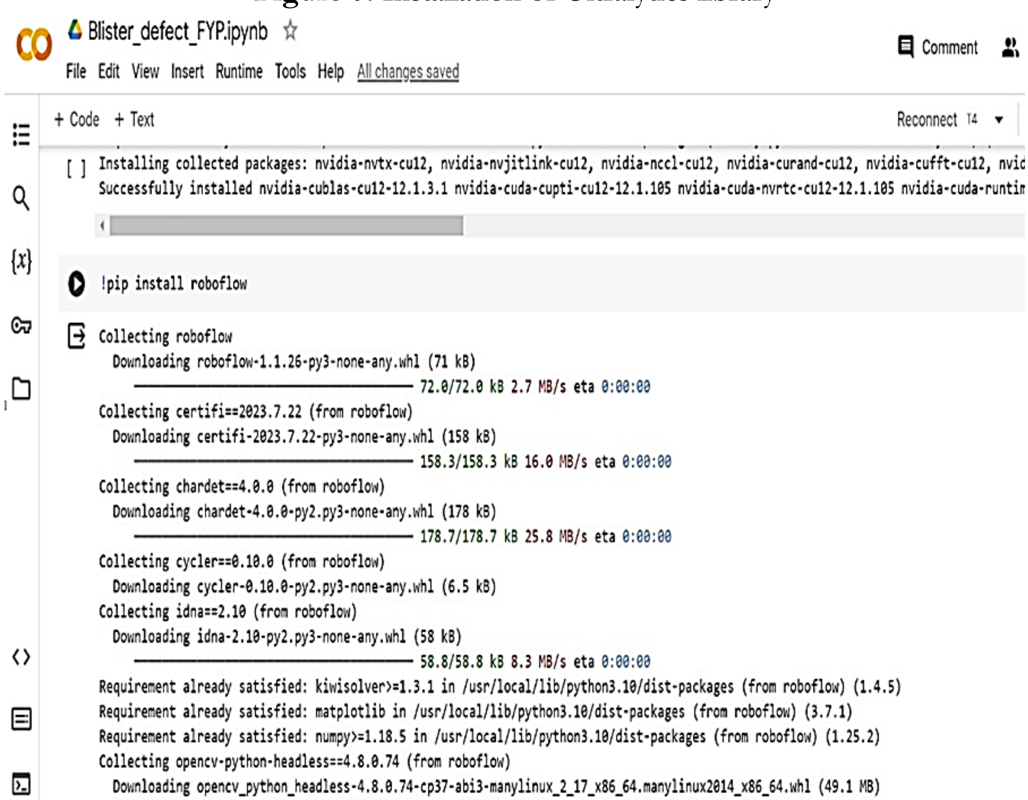
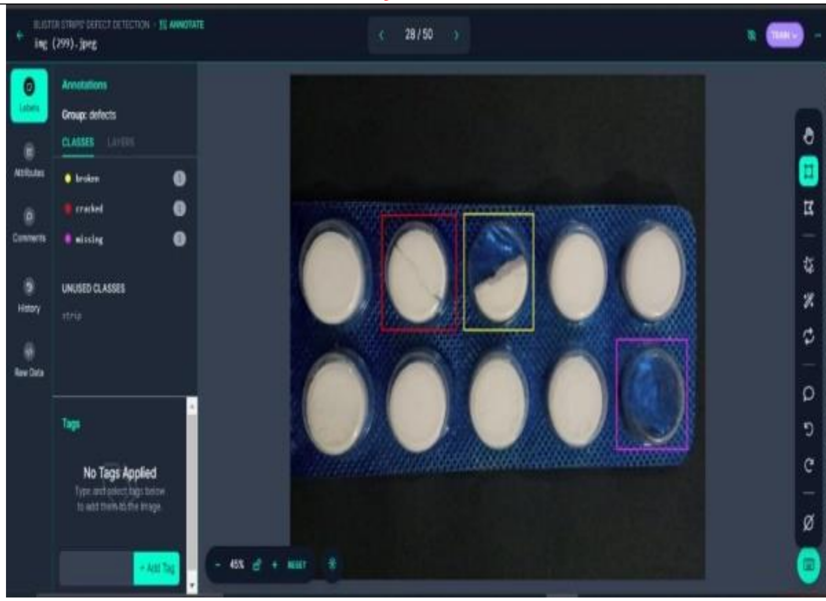
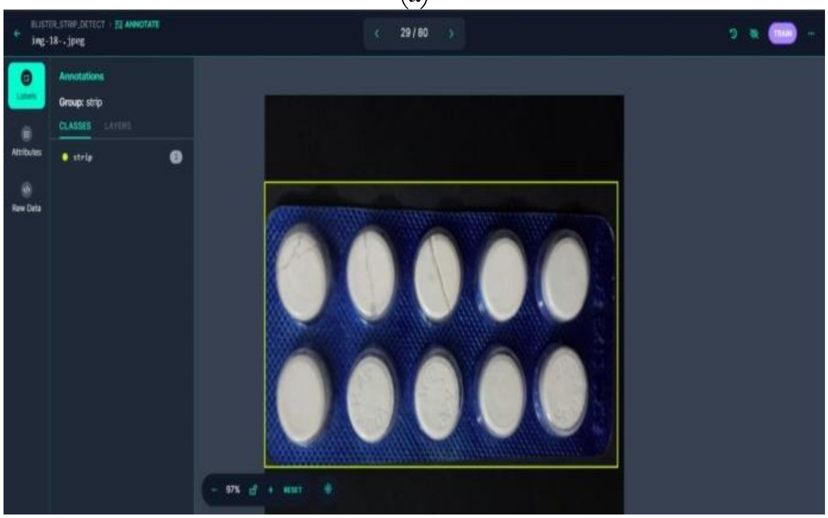


Figure 8. Roboflow installation for data training



(a)



(b)

Figure 9. (a) Annotating images for defects (b) Annotating images for strip detection

Training Process for Defect Detection:

Figure 10 depicts the training process for a YOLOv8 model on the defect dataset. We utilized the command "!yolo task=detect mode=train model=yolov8m.pt data=/content/blister_strip_detect/data_custom .yaml epochs=70 imgsz=640 plots=True" for training process.

Following are the descriptions of all the parameters in the above command:

- task=detect: Designates the task type as object detection.
- mode=train: Changes the mode to training.
- model=yolov8m.pt: This is the medium version of the YOLOv8 model.
- data=/content/Blister-Strips-defect-detections/data.yaml': Defines the location of the dataset configuration file.
- epochs=70: This sets the number of training epochs to 70.
- imgsz=640: Restricts the image size for training to 640 pixels.
- plots=True: Allows graphing of training outcomes.

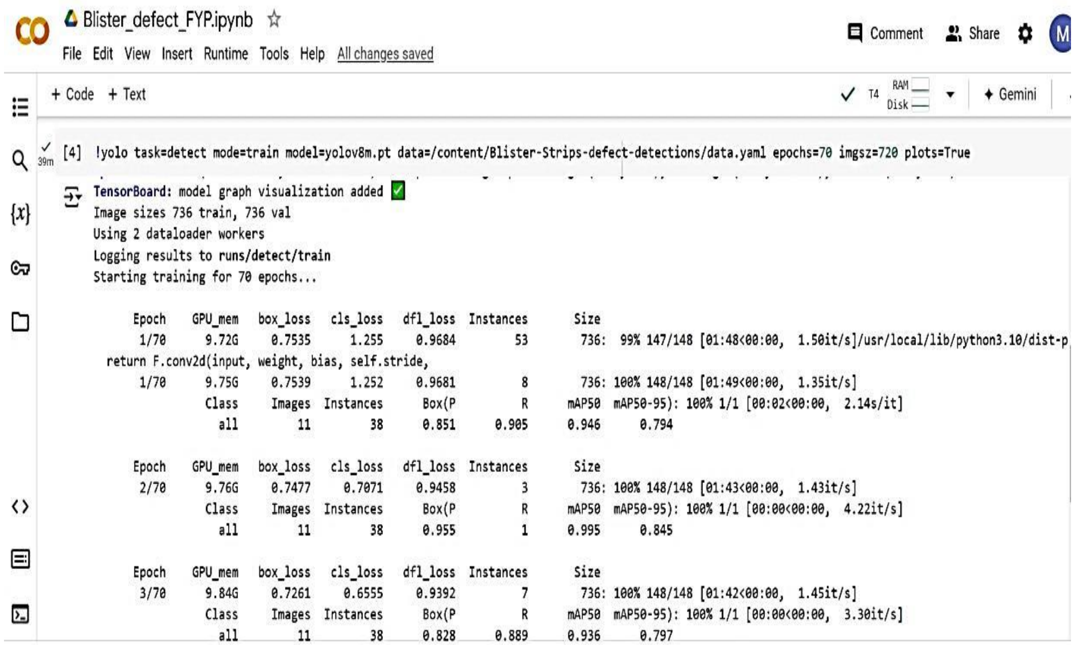


Figure 10. Training of defect detection model

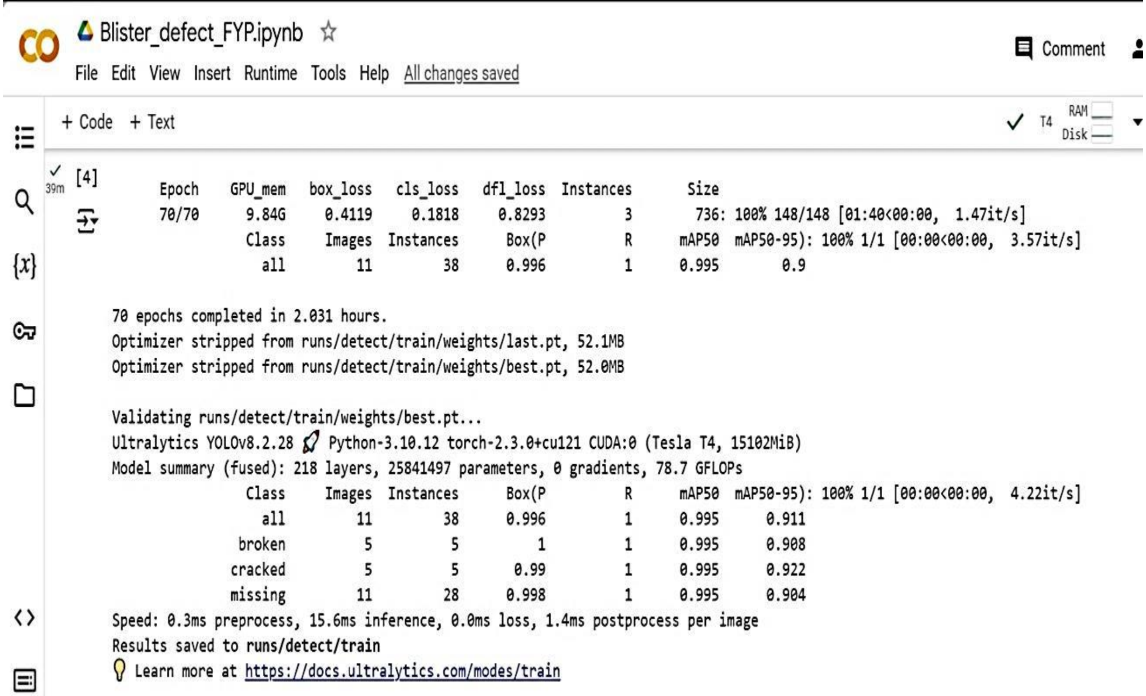
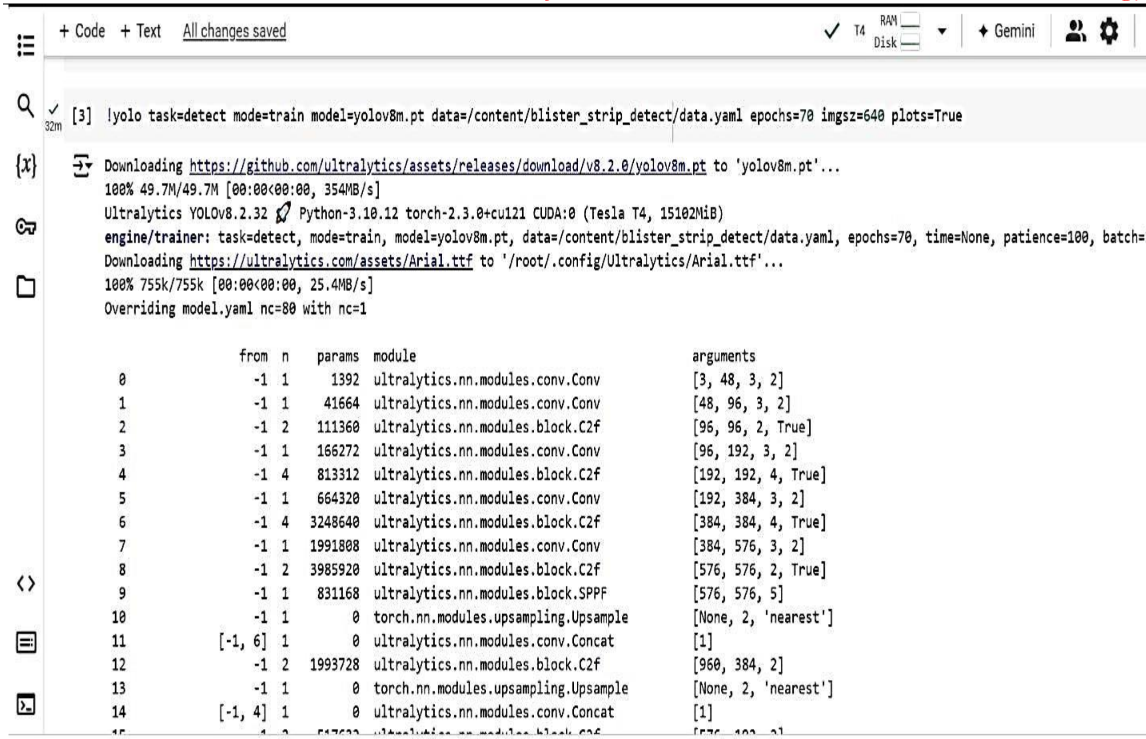


Figure 11. Summary of defect detection model training

As shown in Figure 11, the machine learning model training session using YOLOv8 for defect detection was completed in around 2.031 hours over 70 epochs, with a GPU memory consumption of 9.846 GB. The model was evaluated on 38 images, achieving a mean average precision (mAP) of 0.995 at an IoU threshold of 0.5 (mAP50) and 0.9 across multiple thresholds from 0.5 to 0.95 (mAP50-95). The dataset was trained to detect three classes broken, missing, and cracked, the model demonstrated high accuracy across all categories. Additionally, the processing times were efficient, with a preprocessing speed of 0.3 ms, an inference time of 15.6 ms, and a postprocessing time of 1.4 ms per image.



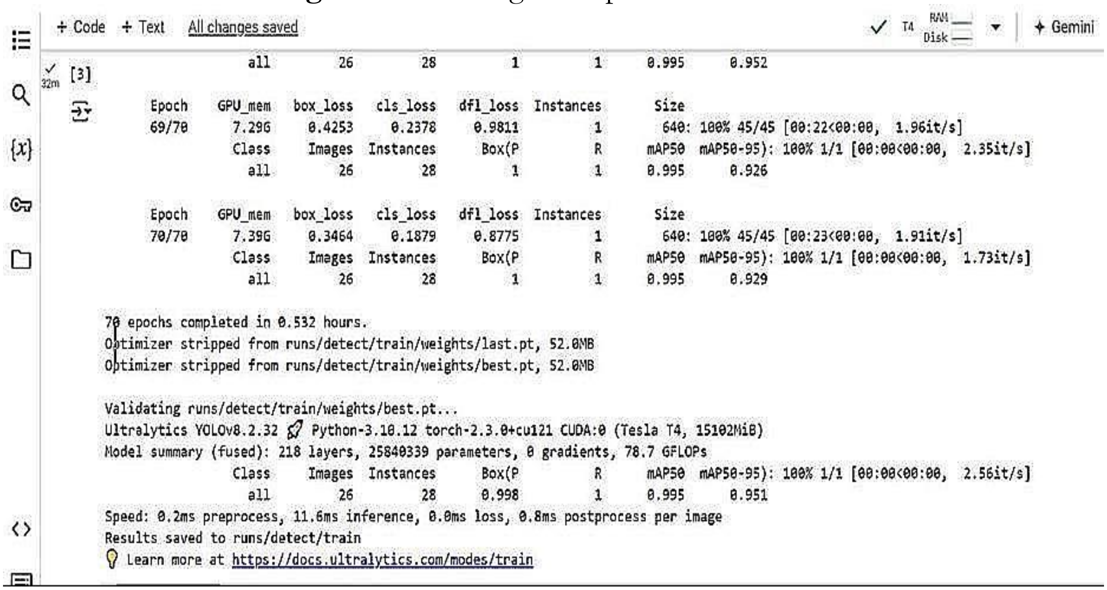
```
+ Code  + Text  All changes saved                                    T4 RAM Disk   Gemini

[3] !yolo task=detect mode=train model=yolov8m.pt data=/content/blister_strip_detect/data.yaml epochs=70 imgsz=640 plots=True

Downloading https://github.com/ultralytics/assets/releases/download/v8.2.0/yolov8m.pt to 'yolov8m.pt'...
100% 49.7M/49.7M [00:00<00:00, 354MB/s]
Ultralytics YOLOv8.2.32 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
engine/trainer: task=detect, mode=train, model=yolov8m.pt, data=/content/blister_strip_detect/data.yaml, epochs=70, time=None, patience=100, batch=1
Downloading https://ultralytics.com/assets/Arial.ttf to '/root/.config/Ultralytics/Arial.ttf'...
100% 755k/755k [00:00<00:00, 25.4MB/s]
Overriding model.yaml nc=80 with nc=1

                   from  n    params  module                                       arguments
  0                  -1  1      1392  ultralytics.nn.modules.conv.Conv             [3, 48, 3, 2]
  1                  -1  1     41664  ultralytics.nn.modules.conv.Conv             [48, 96, 3, 2]
  2                  -1  2    111360  ultralytics.nn.modules.block.C2f             [96, 96, 2, True]
  3                  -1  1    166272  ultralytics.nn.modules.conv.Conv             [96, 192, 3, 2]
  4                  -1  4    813312  ultralytics.nn.modules.block.C2f             [192, 192, 4, True]
  5                  -1  1    664320  ultralytics.nn.modules.conv.Conv             [192, 384, 3, 2]
  6                  -1  4   3248640  ultralytics.nn.modules.block.C2f             [384, 384, 4, True]
  7                  -1  1   1991808  ultralytics.nn.modules.conv.Conv             [384, 576, 3, 2]
  8                  -1  2   3985920  ultralytics.nn.modules.block.C2f             [576, 576, 2, True]
  9                  -1  1    831168  ultralytics.nn.modules.block.SPPF            [576, 576, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  2   1993728  ultralytics.nn.modules.block.C2f             [960, 384, 2]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
```

Figure 12. Training of strip detection model



```
+ Code  + Text  All changes saved                                    T4 RAM Disk   Gemini
[3]                   all         26         28          1          1      0.995      0.952

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      69/70      7.29G     0.4253     0.2378     0.9811          1        640: 100% 45/45 [00:22<00:00,  1.96it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.35it/s]
                   all         26         28          1          1      0.995      0.926

      Epoch    GPU_mem   box_loss   cls_loss   dfl_loss  Instances       Size
      70/70      7.39G     0.3464     0.1879     0.8775          1        640: 100% 45/45 [00:23<00:00,  1.91it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  1.73it/s]
                   all         26         28          1          1      0.995      0.929

70 epochs completed in 0.532 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 52.0MB
Optimizer stripped from runs/detect/train/weights/best.pt, 52.0MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.32 🚀 Python-3.10.12 torch-2.3.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 218 layers, 25840339 parameters, 0 gradients, 78.7 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  2.56it/s]
                   all         26         28      0.998          1      0.995      0.951
Speed: 0.2ms preprocess, 11.6ms inference, 0.0ms loss, 0.8ms postprocess per image
Results saved to runs/detect/train
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Figure 13. Summary of strip detection model training

Training Process for Strip Detection:

Figure 12 depicts the training process for a YOLOv8 model on the dataset of blister strips. The initial setup, configuration for training a YOLOv8 object detection model, and initialization of training were done by the following command:

"!yolo task=detect mode=train model=yolov8m.pt data=content/blister_strip_detect/data.yaml epochs=70 imgsz=640 plots=True".

The yolov8m model and dataset mentioned in `data.yaml` were used, with 70 epochs and an image size of 640 pixels. Plotting is enabled. This setup enabled the model for training on the specified dataset with detailed configuration logging.

Figure 13 presents the summary of the training and validation results for a YOLOv8 model for strip detection. A different data set was used for the training of strip detection. The model completed 70 epochs in 0.532 hours, using around 7.39 GB of GPU memory in the final epoch. Validation results indicated that the model was evaluated on 26 images, achieving a mean average precision of 0.995 at a threshold of 0.5 and 0.951 at thresholds from 0.5 to 0.95 of IoU. The training summary shows 218 layers, 258.4 million parameters, and 78.7 GFLOPs. The processing speeds are 0.2 ms for preprocessing, 11.6 ms for inference, and 0.8 ms for post-processing per image.

Overfitting Prevention Techniques:

In this study, three overfitting prevention techniques were utilized to ensure robustness. Firstly, overfitting was prevented by changing the direction and tilt of tablet strips in the dataset. This enabled the model to recognize tablets in various orientations rather than memorizing specific locations, allowing it to identify tablets even when their orientation changes. This approach aligns with real-life scenarios where the direction of the tablet strip may not always be the same. Secondly, the model was trained on images captured under different indoor lighting conditions. This helped the model to focus on key features such as defect detection, rather than being biased by light variations. Lastly, the training was stopped when the validation loss ceased to improve, ensuring that the model simply did not memorize the training data, including noise, which could lead to poor performance on unseen data.

Results and Discussion:

Figures 14 and 15 display graphs illustrating various metrics during the training and validation phases of our two trained models. It was observed that training losses for box, classification, and dfl decrease over epochs, while recall and precision increase with time, stabilizing at around 50 epochs. These results indicated that the model is efficiently learning from the training of data, adjusting its parameters, and refining its predictions, leading to better performance across several metrics.

Testing:

Various blister strips were tested after completing the training process. As shown in Figure 16, the system successfully identified both a single missing tablet and four missing tablets. Similarly, several blister strips with varying defects were tested. Identification of missing, broken, and cracked defects is presented in Figure 17.

Real-Time Testing of the Blister Strips:

The system fully integrated is shown in Figure 18. To assess the effectiveness of the designed system, faulty tablets were detected in real-time by inspecting them as they moved along the conveyor belt. The camera mounted on the stand above the conveyor belt took real-time images of the strips. The data was processed on the spot, and defects were identified.

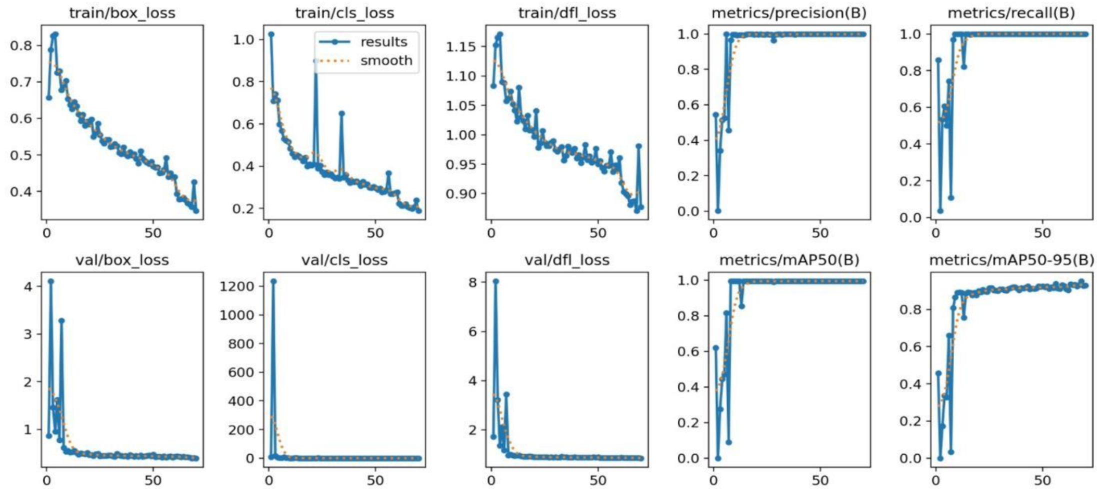


Figure 14. Strip Detection Training and Validation Graphs

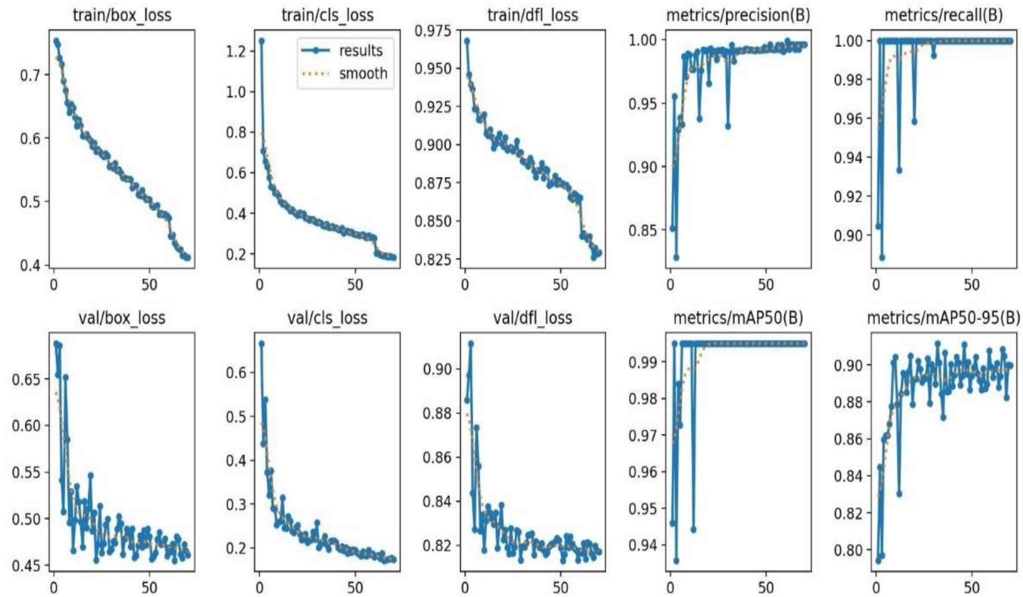


Figure 15. Defect Detection Training and Validation Graphs

Figure 16. (a) Results show a missing tablet with a Rectangular Red Box. (b) Results showing four missing Tablets in the Blister Strip

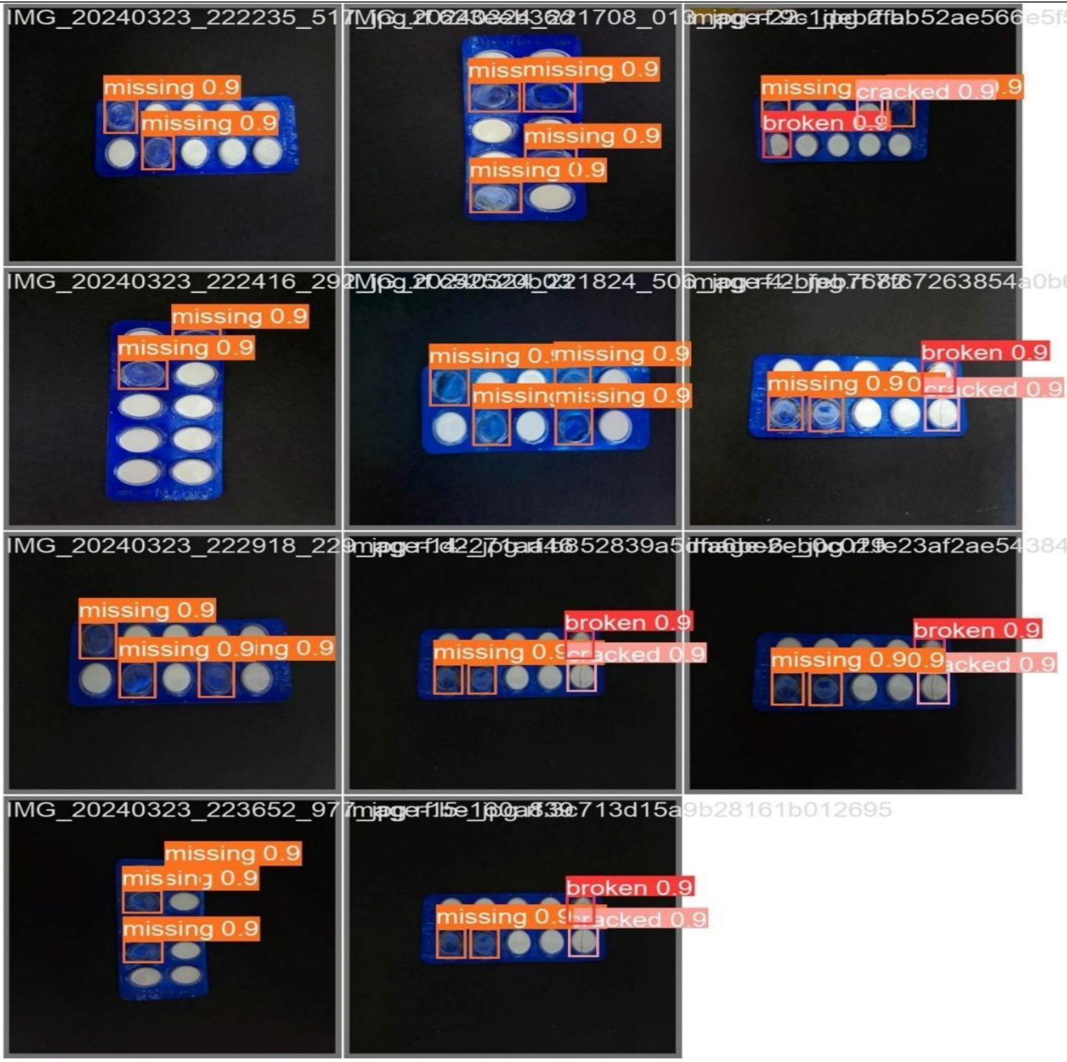


Figure 17. Results showing defects in different strips

Figure 18. Real-time detection of defects by the Tablet Guard System

The defect in weight was identified by the load cell as shown in Figure 19. The webcam feed results shown in Figures 20 and 21 are the final detected outcome of the project. These figures indicate how the strip is manipulated under the camera, and the algorithm works efficiently to detect broken, cracked, or missing tablets in a single strip.

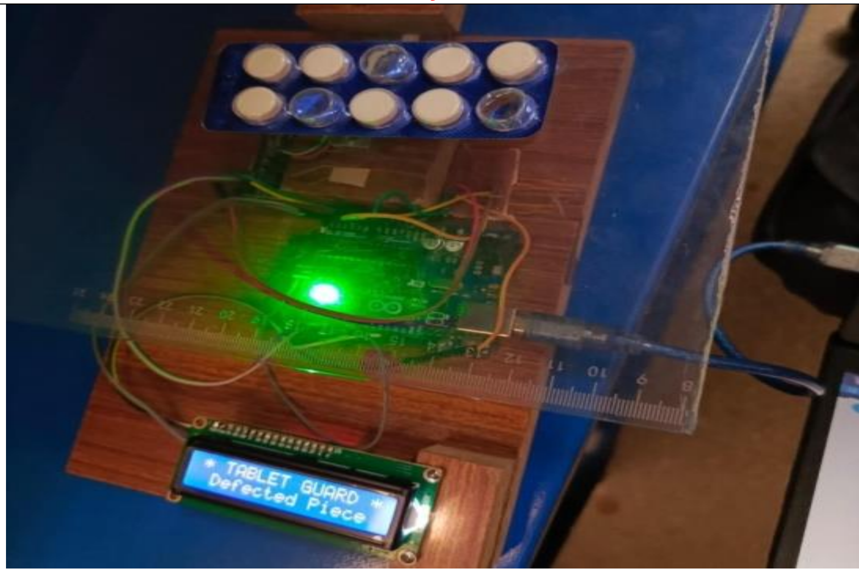
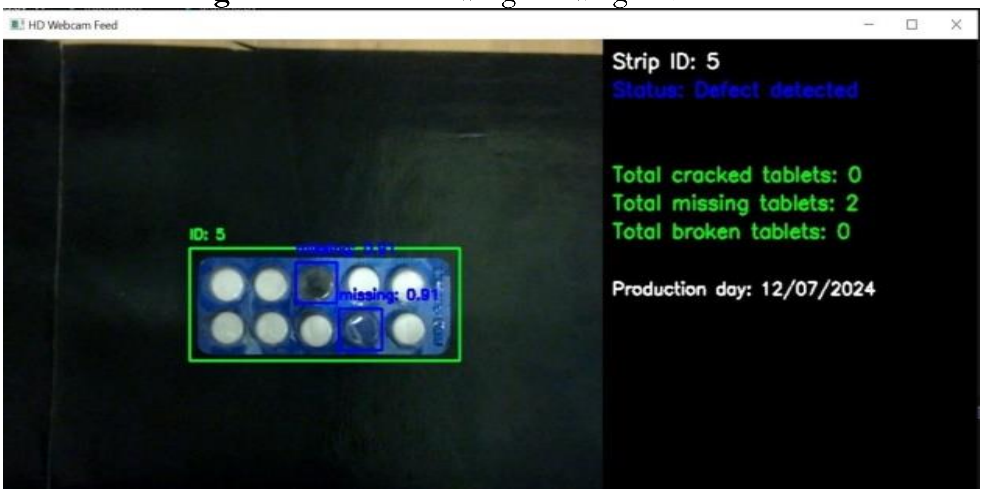
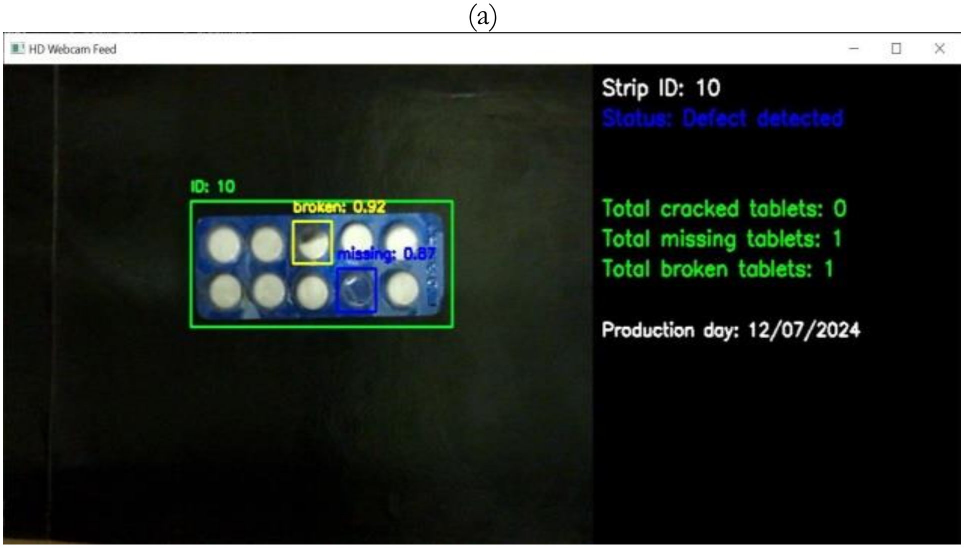


Figure 19. Result showing the weight defect

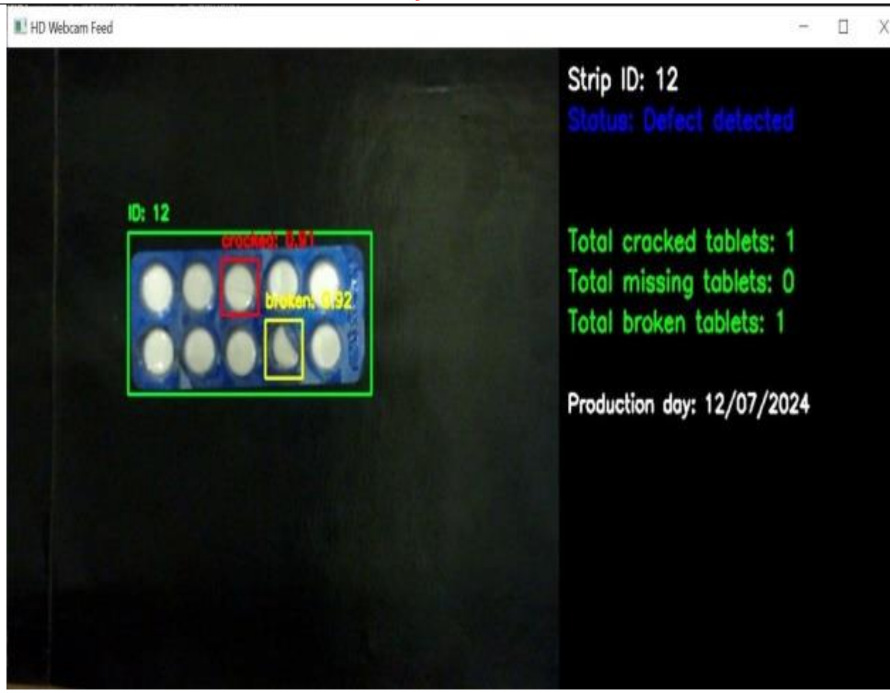


(a)

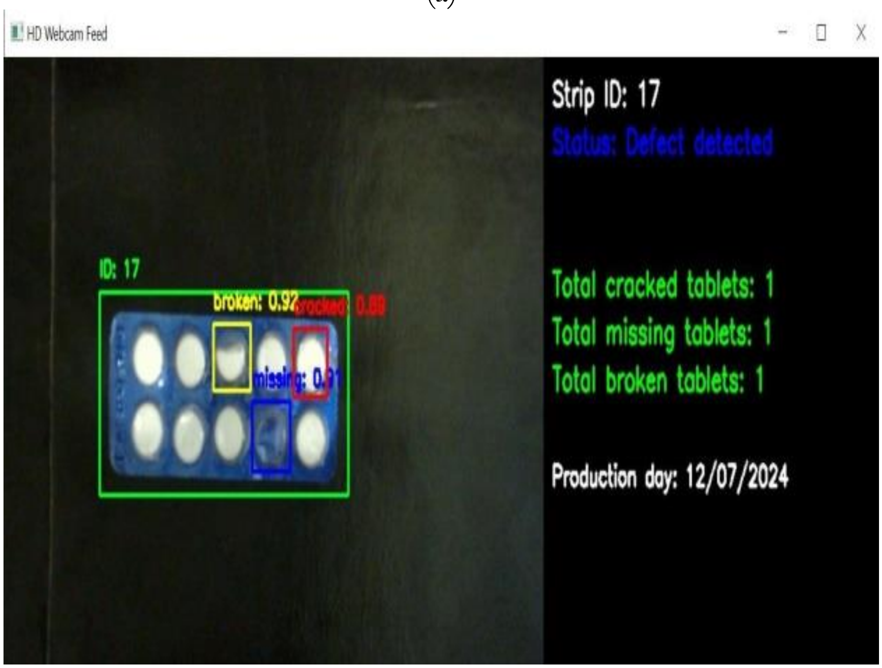


(b)

Figure 20. (a)Detection result and its strip ID (b) Detection result showing the missing and broken defect



(a)



(b)

Figure 21. (a)Result showing the cracked and broken defect (b) Result showing all the defects

The employment of YOLOv8 allows for real-time defect identification with an accuracy of up to 90%, thereby improving efficiency and reducing manufacturing line disorders. The significance of the proposed system lies in its ability to modernize the quality control practice, decreasing the possibility of defective products reaching customers, and ensuring safety while also strengthening the reputation of manufacturing companies.

Finally, the defective piece was removed with the help of removing mechanism connected to a servo motor as shown in Figure 22.

Figure 22. Servo Motor with a removing bar

Error Handling and Accuracy Improvement:

In our proposed system, handling false positives and error correction is essential for reliable defect detection. The combination of load cell-based weight analysis and the deep learning technique allows the system to cross-check for defective tablets. If a tablet is misidentified due to noise or lighting conditions, the dual check approach helps to reduce errors. In addition, the defect detection algorithm has been fine-tuned to minimize misclassifications. During the process, the errors were manually reviewed and logged for further model improvement.

Comparison of Results with Existing Studies:

In this section, a detailed comparison of our work with existing studies reported in the literature is presented. Table 2 summarizes this comparison. Although several studies on tablet defect detection exist, only a few focus on deep learning. Therefore, the comparison includes not only papers on blister defect detection but also those applying deep learning techniques to other related applications.

As shown in Table 2, the proposed Tablet-Guard project achieves an impressive mAP of 0.995 using YOLOv8 for real-time defect detection in blister strips. Its integration of a load cell ensures both physical and visual inspection, enhancing reliability.

While the design in [21] achieved an impressive 99.7% classification accuracy, it requires manual placement of tablets in a 3D-printed tray. In addition, its reliance on structured placement introduces segmentation challenges such as a shift in the tablets, misalignment in the tray, or the camera angle, which can mess up the segmentation, leading to inaccurate defect detection. Our proposed Tablet-Guard's conveyor system and deep learning-based defect detection eliminate such limitations by enabling continuous, high-speed inspection without positional constraints.

An extended model of Faster R-CNN, known as Mask R-CNN has been used by the authors in [12]. This approach shows potential for small-scale pill detection with a 0.916 mAP but struggles with efficiency due to its two-step processing and limited dataset adaptability, which is a disadvantage for high-speed pharmaceutical production lines.

Another notable comparison is with the design in [22], which uses a YOLOv5-based defect detection system and achieved 99.2% accuracy while incorporating additional quality parameters like crushing strength and disintegration time. While this approach enhances tablet integrity assessment, it lacks a physical removal mechanism like the one in our proposed Tablet Guard, which is crucial for automated production lines. Additionally, Tablet-Guard's YOLOv8 implementation surpasses YOLOv5 in accuracy and speed, making it a more efficient solution.

In the realm of blister package inspection, the YOLOv7-based system presented in [23], focused on detecting broken pills, cracks, foreign objects, and color mismatches, achieving an mAP of 0.962. However, its reliance on image uploads for defect identification makes it less effective for real-time inline production monitoring compared to Tablet-Guard's conveyor-based automation.

Comparing broader applications, the paper in [24] tackled steel surface defects with Faster R-CNN, SSD512, and improved YOLOv7, achieving 0.711, 0.724, and 0.812 mAP respectively. Though the methods enhance speed and feature extraction, their lower accuracy highlights challenges in adapting these approaches to pharmaceutical-grade precision.

Other applications, [25] (YOLOv8 for tomato leaf disease detection, 98.9% mAP) and [26] (YOLOv8 for Bangus freshness assessment, 98.5% mAP), demonstrate the versatility of deep learning in quality assurance across different domains. While these systems showcase high accuracy, they work well mostly in controlled, predictable setups. In contrast, in Tablet-Guard, defects must be detected while everything is in motion. This makes Tablet-Guard more flexible and adaptable to real-world manufacturing environments.

In summary, the proposed Tablet-Guard achieves high accuracy with a practical implementation that combines deep learning, load cell technology, and a removal mechanism, making it a robust, real-world solution for pharmaceutical quality assurance.

Conclusion:

Throughout this study, the particulars of the proposed Tablet-Guard project were thoroughly examined, with a prime focus on automating quality control in pharmaceutical tablet manufacturing. This is necessary for the safety and well-being of the patients. The workflow involves capturing images of blister strips moving on a conveyor belt, creating a dataset for training machine learning models, and employing CNN for defect detection. The system effectively detects broken, missing, and cracked tablets within blister strips. The load cell accurately measures the weight of each blister strip, while the camera, in conjunction with AI algorithms, identifies any visible defects. Upon detection of a fault, the Arduino board sends a signal to activate the servo motor, which removes the damaged blister strip from the production line. The training of a machine-learning model for defect detection and the design of a conveyor belt system for moving blister strips have been presented in detail. The results demonstrate that the proposed Tablet Guard System is both effective and reliable in maintaining high-quality control standards, significantly reducing the risk of defective products reaching consumers.

Future Improvement:

In our current scenario, the conveyor belt operates at a constant speed, which was carefully determined by noting the processing time of load cell measurement, image processing, and servo motor response. The speed was synchronized to ensure that defect detection and tablet removal occurred without delays. A potential future improvement could be the integration of a feedback-based speed control mechanism, allowing the conveyor belt to adjust its speed based on real-time processing delays. By doing this, the system will ensure that defect detection remains reliable and does not degrade in accuracy due to changes in processing time. Future work will explore this approach to improve adaptability and responsiveness to real-life scenarios.

Table 2. Summary of comparison of the proposed design with existing studies

	Application	Model used	Epoch	mAP@0.5	Year
This work	Defect detection in blister strips	YOLOv8	70	0.995	2025
[21]	Defect detection in film-coated tablets.	CNN (VGG16)	17	0.997 accuracy	2025
[12]	Medicine inspection	Mask R-CNN	300	0.916*	2022
[22]	Tablet defect detection	YOLOv5	200	0.992 accuracy	2024
[23]	Defect detection in blister packages	YOLOv7	50	0.962	2024
[24]	Steel strip defect detection	Faster R-CNN	NA	0.711	2022
		SSD512		0.724	
		YOLOv7		0.812	
[25]	Tomato leaf disease detection	YOLOv8	200	NA	2023
[26]	Freshness of Bangus	YOLOv8	~272	0.985	2023

Acknowledgement:

The authors express their gratitude for the facilities provided by the Sultan Qaboos Oman IT Chair office at the NED University of Engineering and Technology, Pakistan.

Author's Contribution:

Sana Arshad conceptualized the idea and composed the final draft. Zainab Mustafa investigated Convolutional Neural Networks (CNN) and load cells, delving into their applications and optimizations. Noman Mansoor focused on the hardware aspect, ensuring the conveyor belt functionality, motors alignment, and making data sets. Shaiza Kamran outlined the literature review, objectives, methodology, and intelligence applications within the pharmaceutical domain. Mariam Syed provided a comprehensive analysis of various object detection algorithms and trained the batches and datasets to detect the defects in blisters.

Conflict of interest:

The authors declare that there is no conflict of interest.

References:

[1] & P. S. A. Kamaraj, "Automation in Pharmaceutical Industry and Global Regulatory Compliance," JCPR. Accessed: Mar. 09, 2025. [Online]. Available: https://jcpr.humanjournals.com/automation-in-pharmaceutical-industry-and-global-regulatory-compliance/

[2] Deepti and R. Bansal, "Enhanced Feature Extraction Technique for Detection of Pharmaceutical Drugs," *Intl. J. Engg. Res. and Gen. Sci.*, 2015.

[3] A. M. S. Abdur Rehman Riaz, M.U. Farooq, "Tablet Inspection and Sorting Machine," *Int. J. Comput. Appl. Technol.*, vol. 126, no. 9, 2015, [Online]. Available:

https://www.ijcaonline.org/research/volume126/number9/riaz-2015-ijca-906191.pdf
[4] V. Itikala, "Arduino Weighing Machine Using Load Cell and HX711 Module," *SSRN Electron. J.*, Jul. 2021, doi: 10.2139/SSRN.3918720.
[5] N. S. Arden, A. C. Fisher, K. Tyner, L. X. Yu, S. L. Lee, and M. Kopcha, "Industry 4.0 for pharmaceutical manufacturing: Preparing for the smart factories of the future," *Int. J. Pharm.*, vol. 602, p. 120554, 2021, doi: https://doi.org/10.1016/j.ijpharm.2021.120554.
[6] and M. A. V K. T. S., C. Prasad, P. V., "Anomaly detection in pharmaceutical pills using image processing," International Journal of Advance Research, Ideas and Innovations in Technology. Accessed: Mar. 09, 2025. [Online]. Available: https://www.researchgate.net/publication/359993017_Anomaly_detection_in_pharmaceutical_pills_using_image_processing
[7] H. B. Kekre, D. Mishra, and V. Desai, "Detection of defective pharmaceutical capsules and its types of defect using image processing techniques," *2014 Int. Conf. Circuits, Power Comput. Technol. ICCPCT 2014*, pp. 1190–1195, Mar. 2014, doi: 10.1109/ICCPCT.2014.7055049.
[8] L. Zhang, R. Zhou, Y. Sun, X. Chen, and Z. Yang, "A Defect Detection Algorithm of Round Tablets Based on Image Information," *2024 9th Int. Conf. Intell. Comput. Signal Process. ICSP 2024*, pp. 625–633, 2024, doi: 10.1109/ICSP62122.2024.10743238.
[9] J. Sa, Z. Li, Q. Yang, and X. Chen, "Packaging defect detection system based on machine vision and deep learning," *2020 5th Int. Conf. Comput. Commun. Syst. ICCCS 2020*, pp. 404–408, May 2020, doi: 10.1109/ICCCS49078.2020.9118413.
[10] H. T. Quan, D. D. Huy, N. T. Hoan, and N. T. Duc, "Deep Learning-Based Automatic Detection of Defective Tablets in Pharmaceutical Manufacturing," *IFMBE Proc.*, vol. 85, pp. 789–801, 2022, doi: 10.1007/978-3-030-75506-5_64.
[11] Prajwala N B, "Defect Detection in Pharma Pills Using Image Processing," *Int. J. Eng. Technol.*, 2018, doi: 10.14419/IJET.V7I3.3.14497.
[12] S.-H. L. Hyuk-Ju Kwon, Hwi-Gang Kim, "Pill Detection Model for Medicine Inspection Based on Deep Learning," *Chemosensors*, vol. 10, no. 1, p. 4, 2022, doi: https://doi.org/10.3390/chemosensors10010004.
[13] J. Liu, M. Zhu, T. Cai, D. Kong, and Y. Xing, "Intelligent Tablet Detection System Based on Capsule Neural Network," *2022 IEEE 2nd Int. Conf. Power, Electron. Comput. Appl. ICPECA 2022*, pp. 665–669, 2022, doi: 10.1109/ICPECA53709.2022.9719182.
[14] D. Yendri, Adrizal, Derisma, and H. Afif, "Design of cow cattle weighing system technology and automatic giving feed," *2020 Int. Conf. Inf. Technol. Syst. Innov. ICITSI 2020 - Proc.*, pp. 185–191, Oct. 2020, doi: 10.1109/ICITSI50517.2020.9264977.
[15] R. G. and A. F. J. Redmon, S. Divvala, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA. Accessed: Mar. 09, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/7780460
[16] "YOLOv8 vs SSD: Choosing the Right Object Detection Model," Keylabs. Accessed: Mar. 23, 2025. [Online]. Available: https://keylabs.ai/blog/yolov8-vs-ssd-choosing-the-right-object-detection-model/
[17] N. Klingler, "RetinaNet: Single-Stage Object Detector with Accuracy Focus," viso.ai. Accessed: Mar. 23, 2025. [Online]. Available: https://viso.ai/deep-learning/retinanet/
[18] "Achieve Faster Inference Speeds with Ultralytics YOLOv8 and Intel's OpenVINO," Ultralytics, [Online]. Available: https://www.ultralytics.com/blog/achieve-faster-

inference-speeds-ultralytics-yolov8-openvino.
[19] "Rotated-RetinaNet," GitHub, [Online]. Available: https://github.com/ming71/Rotated-RetinaNet.
[20] S. G. E. Brucal, L. C. M. de Jesus, and L. A. Samaniego, "Development of a Localized Tomato Leaf Disease Detection using YoloV9 Model via RoboFlow 3.0," *2024 IEEE 13th Glob. Conf. Consum. Electron.*, pp. 601–603, Oct. 2024, doi: 10.1109/GCCE62371.2024.10760343.
[21] A. V. Kabir A. Pathak, Prapti Kafle, "Deep learning-based defect detection in film-coated tablets using a convolutional neural network," *Int. J. Pharm.*, vol. 671, p. 125220, 2025, doi: https://doi.org/10.1016/j.ijpharm.2025.125220.
[22] Z. K. N. Anna Diószegi, Máté Ficzere, Mészáros, Lilla Alexandra, Péterfi, Orsolya, Farkas, Attila, Galata, Dorián László, "Automated tablet defect detection and the prediction of disintegration time and crushing strength with deep learning based on tablet surface images," *Int. J. Pharm.*, vol. 667, p. 124896, 2024, doi: https://doi.org/10.1016/j.ijpharm.2024.124896.
[23] R. Patgiri, V. Ajantha, S. Bhuvaneswari, and V. Subramaniyaswamy, "Intelligent Defect Detection System in Pharmaceutical Blisters Using YOLOv7," *2nd Int. Conf. Emerg. Trends Inf. Technol. Eng. ic-ETITE 2024*, 2024, doi: 10.1109/IC-ETITE58242.2024.10493735.
[24] H. W. and Z. X. Y. Wang, "Efficient Detection Model of Steel Strip Surface Defects Based on YOLO-V7," *IEEE Access*, vol. 10, pp. 133936–133944, 2022, doi: 10.1109/ACCESS.2022.3230894.
[25] S. G. E. Brucal, L. C. M. De Jesus, S. R. Peruda, L. A. Samaniego, and E. D. Yong, "Development of Tomato Leaf Disease Detection using YoloV8 Model via RoboFlow 2.0," *GCCE 2023 - 2023 IEEE 12th Glob. Conf. Consum. Electron.*, pp. 692–694, 2023, doi: 10.1109/GCCE59613.2023.10315251.
[26] L. A. Samaniego, S. R. Peruda, S. G. E. Brucal, E. D. Yong, and L. C. M. De Jesus, "Image Processing Model for Classification of Stages of Freshness of Bangus using YOLOv8 Algorithm," *GCCE 2023 - 2023 IEEE 12th Glob. Conf. Consum. Electron.*, pp. 401–403, 2023, doi: 10.1109/GCCE59613.2023.10315381.