International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-15 Issue-1, October 2025

Optimizing the Object using Real-Time Computer Vision and Neural Network

Roberto Aguero, Noah Olson, Justus Selwyn

Abstract*: Poultry and food processing manufacturing units have several automated and monitoring processes of the food that go into packaging. However, at the packaging section, manual human intervention is needed to ensure that the correct amount of food, in terms of count and weight, is placed into every package. This is not without error. Hence, an accurate real-time measurement of sample object counts and weight is critical for optimizing processing efficiency and automating workflows in the production chain. The system identifies the food object, counts it, and weighs it before packaging the batch. In this work, we present a novel approach that integrates an Ultralytics You Only Look Once (YOLO) v10 model, a convolutional neural network (CNN)-based object detection framework, with an automated weighing system and dashboard to optimize quality control.

***Keywords*: Machine Learning, Object Detection, Computer Vision, Neural Networks, Ignition SCADA.**

Abbreviations:
CVML: Computer Vision and Machine Learning
WDR: Wide Dynamic Range
R-CNN: Region-based Convolutional Neural Network
GPUs: Graphical Processing Units
TN: True Negatives
TP: True Positives
FP: False Positives
OPCUA: Open Platform Communications Unified Architecture

I. INTRODUCTION

Among the various products processed at poultry plants, chicken nuggets are a high-demand item. These companies not only supply fast-food chains and other customers, but also produce their own branded products. The focus of this study is specifically on chicken nuggets, a product that undergoes a complex, multi-step production process. The nuggets are first cooked and breaded before being fried. After frying, they are frozen, bagged and cased. At each stage of this process, counting and weighing are performed to ensure the correct quantity is displayed on the packaging. This necessitates human involvement at every step, leaving significant room for error and highlighting the need for automation.

Manuscript received on 07 July 2025 | First Revised Manuscript received on 12 July 2025 | Second Revised Manuscript received on 20 September 2025 | Manuscript Accepted on 15 October 2025 | Manuscript published on 30 October 2025.
*Correspondence Author(s)
Roberto Aguero, Department of Computer Science, John Brown University, Siloam Springs, USA. Email ID: agueror@jbu.edu.
Noah Olson, Department of Computer Science, John Brown University, Siloam Springs, USA. Emai ID: nolson@jbu.edu.
Justus Selwyn*, Professor, Department of Computer Science, John Brown University, Siloam Springs, USA. Email ID: jselwyn@jbu.edu, ORCID ID: 0000-0002-9267-0305.

A. Motivation

Industry-leading companies in the meat and protein sector are continuously seeking ways to integrate modern technologies into their workflows. Despite these advancements, a significant portion of their processes still relies on manual labour, which not only proves to be inefficient but also costly. If even the most prominent companies in the industry face these challenges, it is reasonable to assume that smaller or less technologically advanced plants experience even greater inefficiencies. While various previous solutions, such as conveyor belts and automatic packaging systems, have been introduced, these methods still require human intervention, which prevents a fully autonomous workflow.

B. Background & Overview

To address this issue, we propose an advanced computer vision and machine learning (CVML) system capable of automating the identification, counting and weighing of chicken nuggets in an industrial setting. Currently, the manual intervention leads to inconsistencies in the product counts and slower processing times. Our approach differs from existing methods by utilising OpenCV and deep learning-based object detection to identify and track individual nuggets in real-time, even in complex scenarios such as those involving overlapping or irregularly shaped sample pieces. The system leverages CNNs trained on diverse sample images to improve performance under various lighting conditions. Unlike traditional solutions that still rely on human oversight, we aim to fully automate the process entirely, ensuring high accuracy and reliability in packaging. While human verification may still be necessary to validate the system's results, the need for their intervention is almost eliminated.

II. SYSTEM OVERVIEW

Industry leaders currently employ a conveyor belt-based sample processing mechanism. In this process, an operator selects a random number of nugget samples, counts them, and weighs them on a scale. They then record these numbers, which can be altered at any moment due to various external factors. The counting process can be addressed using a camera and the model. For weighing, a standard scale can be integrated to measure the weight simultaneously as the nuggets are being counted

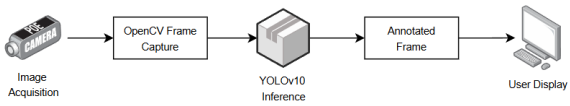
A. Vision Architecture

To accurately count the nuggets, it is essential to obtain a clear and reliable view of the samples. This was achieved using a camera, specifically an AXIS FA1125 Sensor Unit. This camera was

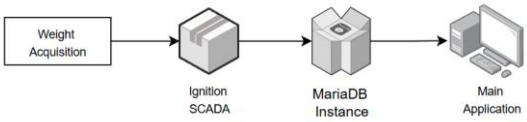
chosen due to its small size, lightweight design and compatibility with the minimum software requirements for OpenCV. It features a 91-degree horizontal field of view with 1080p resolution and a Wide Dynamic Range (WDR) feature, allowing for maximum usability in irregular lighting conditions. Although it does not have the highest quality image, its range is sufficient for the prototype of this system.

For the software components of the system's vision module, the OpenCV Python library cv2 was chosen in conjunction with Roboflow's Supervision. The cv2 library enables real-time, continuous video capture, allowing frames to be fed directly to the YOLOv10 model for processing. This ensures consistent monitoring of the nugget samples on their tray. Roboflow's Supervision complements this by accurately annotating the detected objects in each frame, providing precise information about the count and position of the nuggets. The reason for implementing this system using the YOLO model is that, in recent years, CNN-based object detectors have become the norm for real-world applications that rely on real-time detection [1]. Besides just being able to carry out the task, YOLO provides speed and accuracy, which are crucial given the system requirements. It stands out within this category due to its unique architecture, which allows it to simultaneously predict multiple bounding boxes and class probabilities in a single forward pass [1]. This design minimizes computational overhead while maximizing detection efficiency. YOLOv10 also introduces optimised network layers and enhanced feature extraction techniques, resulting in faster predictions without compromising accuracy. This enhancement is particularly beneficial in this project's use case, where the system must quickly and reliably detect nuggets in real-time with minimal delay.

Other alternatives considered for the CNN-based model included the Single Shot Multi-Box Detector (SSD) and the Faster Region-based Convolutional Neural Network (Faster R-CNN). Ultimately, after researching the trade-offs of each model, the team decided to move forward with YOLO, given its rigorous classification on a live video feed, as well as its reputation as the fastest image detection algorithm among the three choices [2]. Additionally, YOLO is actively being contributed to by the open-source community, keeping it updated and adjusting to modern needs. By balancing speed, precision, and efficiency, YOLOv10 proved to be the most suitable choice for achieving the project's performance goals. This combination streamlines the counting process and reduces the need for manual intervention. The overall flow of the vision module is illustrated in Fig. 1.



[Fig.1: Vision Module Flow]



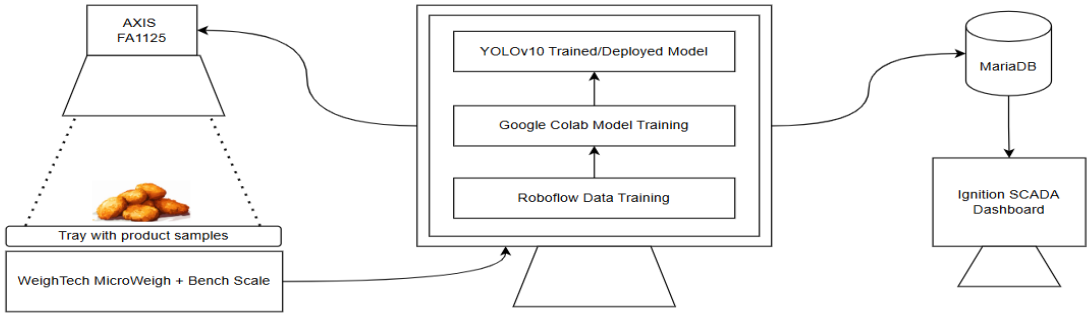
[Fig.2: Weighing Module Flow]

B. Weighing Architecture

To ensure accurate weight measurements, a MicroWeigh Digital Indicator Model MW2001 scale is integrated into the system, operating in tandem with the vision module. This scale is positioned beneath the tray carrying the nugget samples, allowing for real-time weight acquisition as the counting process takes place. By synchronising the weight measurements with the captured frames, the system ensures that every time a sample is saved to the database, all data is saved concurrently. The data read from the scale is transmitted to the Ignition dashboard using the Open Platform Communications Unified Architecture (OPC UA) protocol. This setup enables seamless communication between the scale and our system, providing a comprehensive and automated approach to sample processing. The overall flow of the weighing module is illustrated in Fig. 2.

C. Overall Architecture

The overall system integrates the two previous components into a unified and automated sample processing pipeline. This architecture employs a modular design, which enhances the system's error resilience and facilitates easy development and distribution across the team. The complete integration of these modules is illustrated in Fig. 3.



[Fig.3: Overall System Architecture]

III. NUGGET IDENTIFICATION

To correctly identify nuggets in real-time, the system requires a competent and fast model that can distinguish nugget samples from other sample objects. The first step in developing the automation for the identification was to gather

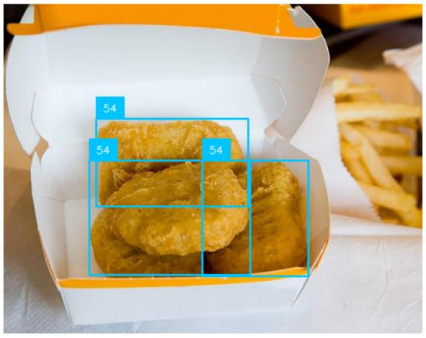
and pre-process the data required for training the machine learning model. This involved identifying and collecting images of nuggets that could be grouped as the dataset for training. Around 600 images were collected, with an attempt to obtain ranging nugget samples, such as tightly coupled nuggets or lone-standing nuggets, in all possible positions. To ensure a diverse and representative dataset, the lighting and positioning features of Roboflow were used in combination to simulate real-world scenarios where the nuggets may appear differently on the tray. Once these images were collected, the next step was to annotate them, where each nugget in the dataset was labelled to define its boundaries. Annotation was performed using Roboflow's labelling tool, ensuring that the model could accurately distinguish nuggets from other objects while ensuring the labels are converted to a YOLO-compatible format.

A. Model Training

These labelled images were then processed and split into three sets: training, testing and validation. The training set was used to teach the model to recognise the nuggets, the validation set was used to fine-tune the hyperparameters and prevent overfitting, and the testing set was used to evaluate the model's performance. With the dataset prepared, the next step was to train the model. A Google Colab environment was utilized due to its availability of cloud-based Graphical Processing Units (GPUs), which accelerated the computation process. The dataset was uploaded to the environment, and pre-trained YOLOv10 weights were downloaded to fine-tune said model. Using these weights, our custom model was trained using 25 epochs and a batch size of 8, ensuring a balance between total training time and model accuracy. This configuration enabled effective learning of the model while maintaining computational efficiency, ultimately resulting in a responsive model suitable for real-time inference use cases [3].

B. Nugget Counting

Before moving on to real-time inference, it was necessary to verify whether the model could detect chicken nugget samples in regular images. Thus, after proper training, the model was tested using sample images from the dataset to verify its detection accuracy. If it were capable of detecting all the nuggets in a picture, then it would be fully capable of counting, given that in this use case, counting just refers to detecting all the possible samples in the frame.



[Fig.4: Sample Nugget Identification Image]

This verification step ensured that the model's object detection performance translated to the accurate count estimation. One of the test images used for this verification is shown in Fig. 4.

IV. EXPERIMENTS AND RESULTS

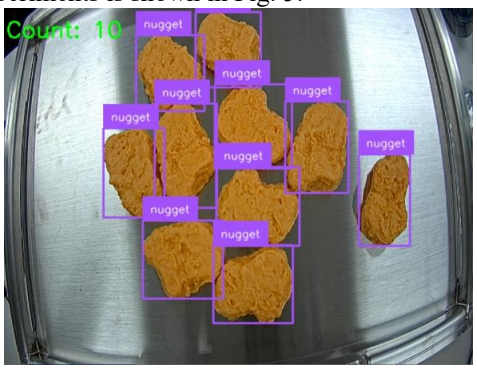
After it was determined that the model was capable of identifying the nuggets based on given images, it was put to the test with the vision portion of the system. The primary goal of these experiments is to accurately identify the number of nuggets visible to the camera, adapting to any changes in real-time. Additionally, the model should not confuse any non-nugget objects as nuggets throughout the entire process.

A. Experiment Setup

Since a real production line would ideally process thousands of nugget samples, which is far more than the system could realistically process during development, 3D-printed nugget replicas were used as substitutes, providing a consistent and practical solution for testing. To simulate a real-time production line environment, the trained YOLOv10 model was integrated into a Python-based application that continuously processed frames from a top-down camera feed, as described in the architecture. As the camera captured each frame, the application ran inference on the image and returned bounding box predictions for each detected nugget. These detections were then used to count the nuggets in real time. This experimental environment enabled rigorous testing across a range of configurations, including ideal conditions and edge cases, allowing for comprehensive evaluation. By controlling placement and lighting, the system's detection capabilities can be evaluated correctly in varying levels of complexity.

B. Initial Experiments

Before introducing the more complex edge cases, a series of initial experiments was conducted to validate the model's fundamental object detection performance. The primary focus of this phase was to ensure that the trained model could consistently detect individual nugget samples under standard conditions, without significant occlusion or interference [4]. During these tests, nuggets were arranged in straightforward layouts to confirm that the model correctly identified the presence of the sample and returned accurate bounding boxes. This stage served as a critical checkpoint to verify that the model was functioning as intended, and also to determine whether it required further training [4]. An example of the initial experiments is shown in Fig. 5.



[Fig.5: Initial Experiment Regular Layout]

Results from the initial experiments confirmed that the model was capable of reliably detecting nugget samples in clean scenarios,

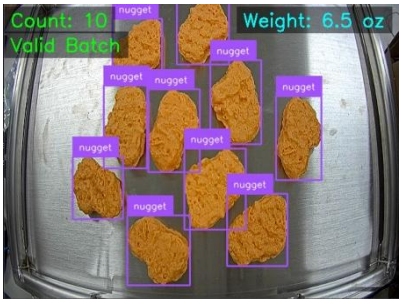
producing high-confidence detections with minimal false positives or missed samples. These outcomes established a performance baseline and validated that the model had been successfully trained for its intended purpose. Only after this baseline was confirmed were more challenging tests introduced in the main experiment phase.

C. Main Experiments

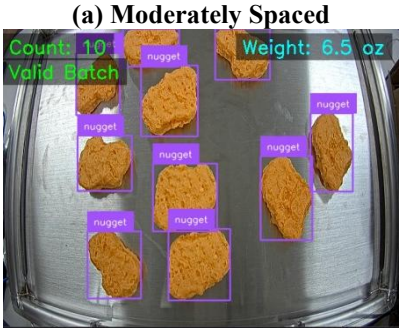
To evaluate the performance of the trained YOLOv10 model under various sample presentation conditions (mirroring an imperfect production-line environment), multiple test configurations were designed. Each configuration used a consistent count of ten nuggets arranged in distinct spatial patterns. The goal was to assess the reliability of detection across different nugget placements, densities, and orientations. It is worth noting that, under these experiments, the validity of the nugget sample batch was not the primary focus; therefore, even if some configurations turn out to be invalid, it does not affect the model's accuracy or performance.

i. Spread-Out Configurations

The first placement to be experimented on was the spread-out nuggets, basically ensuring that they were not touching each other. While this is unrealistic for a production line environment, it is essential to verify whether the model operates properly under ideal, yet practical, conditions. The first results of the experiment are presented in Figs. 6(a) and 6(b).



(a) Moderately Spaced

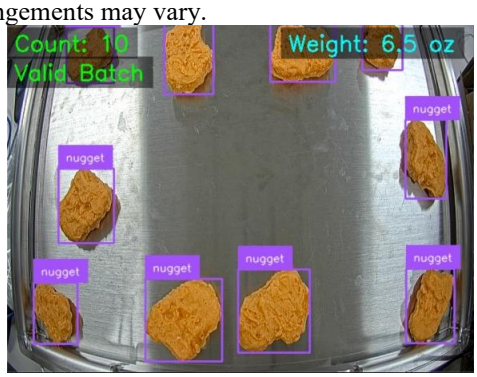


(b) Widely Spaced

[Fig.6: Nugget Detection Results Under Varying Spacing Conditions]

As shown in Figs. 6(a) and 6(b), the model successfully detects all the nuggets in both moderately and widely spaced configurations. These results align with expectations, as the clear separation between nuggets in both cases minimised occlusion, allowing the model to perform with high accuracy and confidence. One more configuration of sparse nuggets was used to test the camera's strength. The following can be found in Fig. 7.

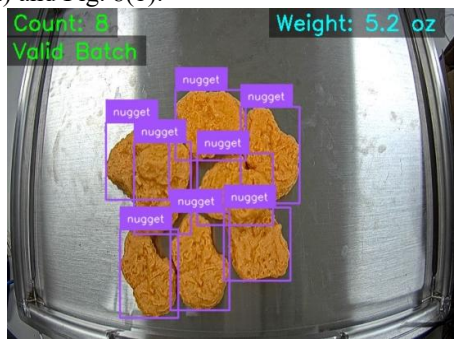
Despite the increased distance between the individual samples and a challenging camera angle that cuts off a few of the samples from being fully in the frame, the model accurately detected all nuggets without any false negatives. This configuration confirms the robustness of the model across spatially diverse layouts. Across all three spacing conditions, the model consistently demonstrated a reliable detection performance. As expected, having a distinct visual separation between nuggets facilitated the accuracy of the object identification. These results confirm the model's capability to generalize across the sort of distributions where tray arrangements may vary.



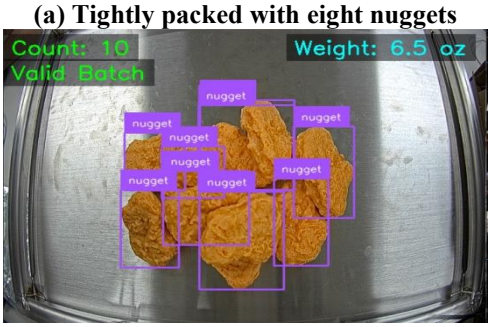
[Fig.7: Nugget Detection Under Extremely Dispersed Configuration]

ii. Tightly-packed Configurations

Another possible layout that the model might encounter in the production line environment is a tightly packed nugget configuration. Under these conditions, nuggets were placed near one another, touching or overlapping at the edges. This setup introduces greater visual complexity due to occlusion, merging colours, and a reduced bounding space. These challenges are commonly encountered in high-throughput environments [5]. An example of this configuration is shown in Fig. 8(a) and Fig. 8(b).



(a) Tightly packed with eight nuggets

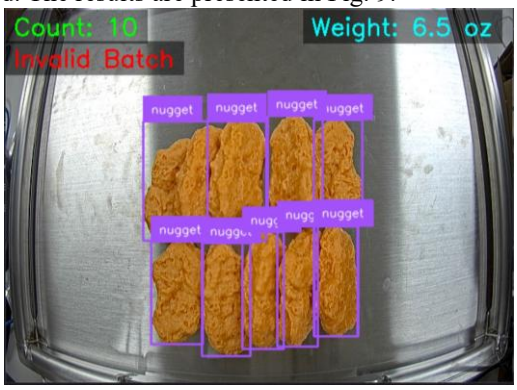


(b) Tightly packed with 10 nuggets

[Fig.8: Nugget Detection Under Tightly Packed Configurations]

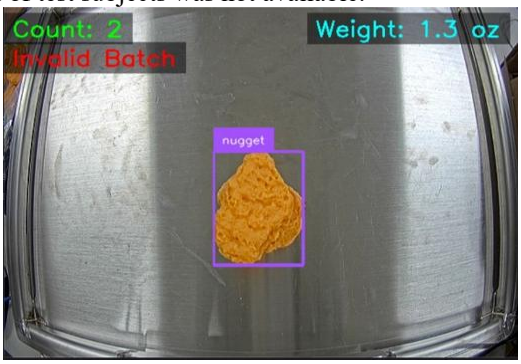
Despite the increased density and overlapping regions between the nuggets, the model was able to detect the eight and ten samples,

respectively. Although several bounding boxes were placed closer together and had minor overlaps, no duplicate detections or false positives occurred. This performance indicates that the model maintains high precision even under visually congested arrangements, effectively distinguishing each object even with minimal spacing. To further verify the model's accuracy, a configuration of ten nuggets stacked on top of each other in a contiguous, horizontal manner was arranged. The results are presented in Fig. 9.

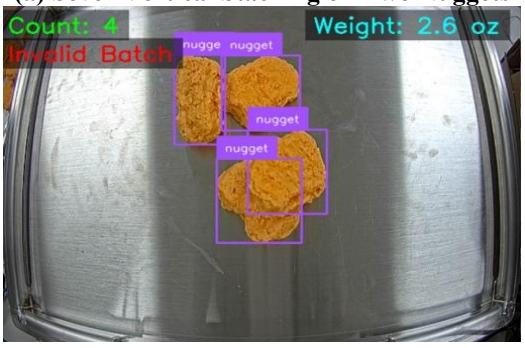


[Fig.9: Nugget Detection Results in a Highly Dense Vertical Stacking Configuration]

The reasoning for this arrangement was to verify whether the model would detect the contiguous nuggets as just two nugget samples, but it correctly recognized the ten nuggets. The frame reveals numerous overlapping bounding boxes, but this did not ultimately affect the counting. Overall, these results validate the model's accuracy in handling realistic sample conditions where perfect spacing cannot be guaranteed [5]. Its ability to detect all nuggets in these tightly spaced scenarios reinforces its potential suitability for the intended use case. One possible improvement in this area is with a larger number of nugget samples (close to 30), but this amount of test subjects was not available.



(a) Sever Vertical Stacking on Two Nuggets



(b) Partial Vertical Stacking on Four Samples

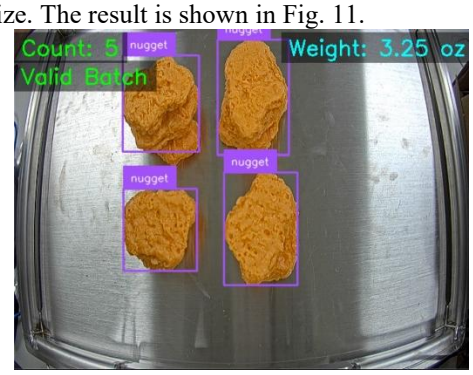
[Fig.10: Nugget Detections under Vertically Stacked Configurations]

iii. Stacked Configurations

In addition to the tightly condensed arrangements, one of the most challenging configurations tested involved nuggets stacked vertically, where one nugget partially or fully obscures the view of another. This test was designed to simulate conditions where nuggets are not only closely spaced but also layered, a widespread occurrence in production line environments due to clumping. Examples of these configurations and the model's performance can be seen in Figs. 10(a) and 10(b).

As shown in Fig. 10(a), the model correctly detected both nuggets, despite the image appearing as if it were just one. This demonstrates the model's ability to identify the colours or shadows that determine where a nugget collision or coupling occurs, which is impressive given its limited field of view. On Fig. 10(b), the model is similarly able to identify all four samples, despite their partial vertical stacking. Given that the camera is mounted directly above the tray, it would be normal for the model to struggle with these detections due to its lack of depth visibility. These configurations demonstrate how the model can overcome the limitations of top-down detection, particularly when vertical layering can obscure objects in lower counts of nugget samples.

The exact configuration was attempted with ten nugget samples to assess the model's performance with a larger sample size. The result is shown in Fig. 11.



[Fig.11: Detection Results for the Vertically Stacked Configuration of Ten Samples]

As seen in Fig. 11, the model was only able to detect five of the ten samples due to significant occlusion. The way the nugget samples were stacked was challenging for the model, given that from a top-down view, they do not appear abnormally sized and thus are harder to distinguish. This type of configuration shows the challenge of maintaining accuracy with increased sample density from a limited perspective. Despite the challenging nature of vertically stacked nuggets, the model performed above expectations under these configurations, given that it was not trained with any data that resembled the way these samples were arranged. To further improve performance in stacked scenarios, future iterations of this system may benefit from incorporating multi-angle camera inputs to improve unpredictable tray layouts.

D. Experiment Observations

Across all test configurations, the YOLOv10 model exhibited strong detection performance, particularly in scenarios with minimal occlusion and separated nuggets. While this is not

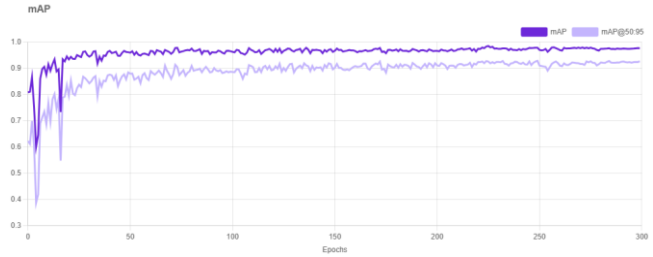
always the case in a production line environment, it confirms that the model is adequately trained for the intended task. In the more challenging cases, the model still performed above expectations, demonstrating its ability to generalise beyond ideal conditions. Despite not being trained on such complex arrangements, it was able to overcome these lower-density stacking scenarios. These results demonstrate the importance of considering camera perspective and 3D occlusion in future iterations, potentially through depth sensing, to improve detection under high-density, real-world conditions [5].

E. Testing Results

The chosen training splits proved to be effective, as the model accurately identified and counted the intended objects. When tested in a live environment, the model demonstrated impressive accuracy and performed well in most edge cases, quickly identifying objects with minimal delay, even when presented in poor lighting or difficult-to-count layouts. While its performance was strong overall, it was not flawless; in some instances, the model mistakenly identified non-nugget objects as nuggets. Despite these occasional errors, the model's speed and reliability made it well-suited for the intended task of counting. To correct for any one-off errors, a special algorithm using a hash table is employed. The most frequent count (almost always the correct one) would be retrieved from the data structure and then output to the database/dashboard.

i. Model Accuracy Metrics

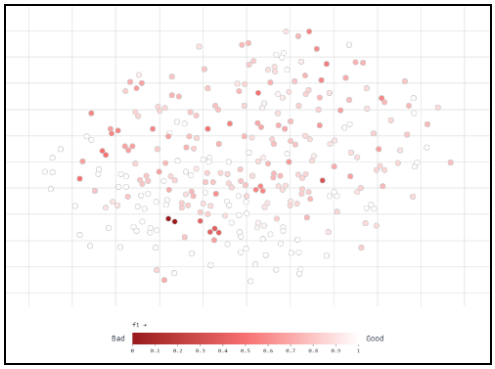
To properly visualise the model's performance, training graphs were generated. These splits resulted in a mean average precision (mAP) of 97.8%. The training graph for this metric is shown in Fig. 12. The model achieved a total precision of 95.2%. Finally, the recall of the model turned out to be 94.5%. This is a valuable metric for our use case because it measures the percentage of relevant labels that were successfully identified.



[Fig.12: Model's Mean Average Precision (mAP)]

ii. F1 Score Analysis

To further validate the model's accuracy, F1 scores were calculated. The F1 score is another helpful measure in this use case, given that it factors in both the model's accuracy and recall. This is particularly important for our use case, where both avoiding false positives (misidentifying non-nugget objects) and minimizing false negatives (failing to detect actual nuggets) are crucial. By incorporating both metrics, the F1 score provides a more accurate assessment of the model's reliability in real-world conditions. The vector analysis performed on the model yielded the scatter plot shown in Fig. 13.

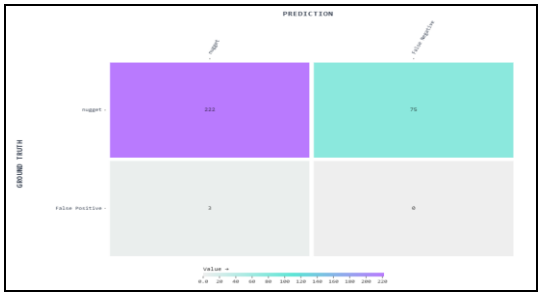


[Fig.13: Model's F1 Score]

To provide context, a dot in this scatter plot represents one image within the dataset. The colour of the dot determines the F1 score of that image, and thus the higher that metric is (on a scale from 0.0 to 1.0), the more opaque or close to white the colour of the dot will be. On the contrary, the lower the F1 score is, the more the colour of the dot will gravitate towards red. Additionally, the dots on the left side of the scatter plot represent data points from singular or very few individual nugget samples. In contrast, the dots closer to the right border represent images with a larger number of samples. Dots closer to the top of the scatter plot represent images with clustered nugget samples. In contrast, towards the bottom of the scatter plot, images with more spread-out nugget samples can be found. His distribution aligns closely with the observed F1 scores, as most of the points near the top achieved an F1 score of 1.0. This clear pattern highlights how the model's performance is influenced by the spatial arrangement of objects within the images. By analyzing this behaviour, the scatter plot proved to be a valuable tool for understanding the model's decision-making process and identifying the conditions under which it performs optimally [6].

iii. Confusion Matrix Evaluation

Finally, a confusion matrix was also used for evaluation, given its natural value for classification tasks. The confusion matrix achieves this by comparing the predicted labels generated by the model against the actual ground-truth labels from the dataset. There are four grids, which represent true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). The precision of the model is calculated as TP/(TP+FP). The confusion matrix for this study is presented in Fig. 14.



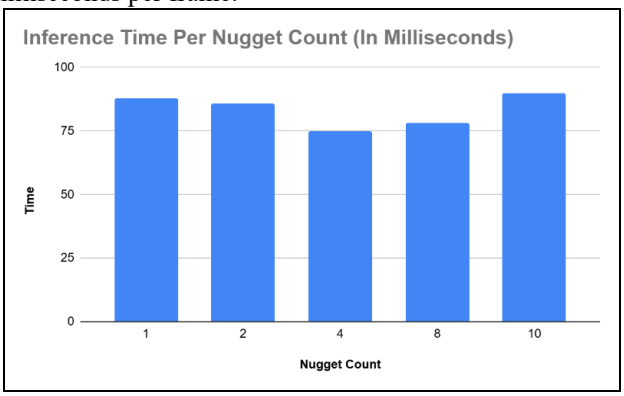
[Fig.14: Model's Confusion Matrix]

In the context of our model, the precision was calculated to be approximately 99%, based on the results of the confusion matrix. At first

glance, this value appears very high, indicating that the model has overfitted to the training data. This might affect the intended use-case of the model, given that the nugget samples it will be evaluating in real-time are highly variant. These insights emphasise the importance of using different metrics to assess the model's performance, as they provide various perspectives on the model's strengths and weaknesses.

iv. Inference Speed and Efficiency

In addition to accuracy, the model's speed and responsiveness are critical for real-time deployment in a production-line environment. To evaluate this, inference time was measured across varying nugget counts within the frame. As shown in Fig. 15, the system maintained a consistent inference time, ranging from approximately 75 to 90 milliseconds per frame.



[Fig.15: Inference Time Per Nugget Count (in Milliseconds)]

Interestingly, the inference time did not increase linearly with the number of detected nuggets. Although it fluctuated, it remained constant on average, despite possibly facing image complexity, bounding box post-processing overhead, and other factors. Even at the highest density of 10 nuggets, the average inference time remained close to the rest, confirming that the system operates well under real-time constraints. This consistent performance ensures that the application can keep up with live production-line speeds, enabling accurate and timely sample processing.

V. CONCLUSION

This study demonstrates the successful development and implementation of an automated system for real-time counting and weighing of chicken nuggets using computer vision and machine learning techniques. By integrating a YOLOv10 model with a synchronized weighing module and dashboard interface, we effectively addressed key challenges in poultry plant sample processing. The proposed approach reduces manual labor requirements, minimizes errors in sample counting, and enhances operational efficiency. Our comprehensive testing process revealed that the system performs reliably in diverse real-world scenarios, making it suitable for large-scale industrial deployment. Future improvements may involve refining the model's robustness to further reduce misclassifications rates and enhancing the dashboard interface to deliver more intuitive insights to operators. Overall, this system offers a scalable and practical solution to improve quality control in poultry processing plants, paving the way for broader automation across the food production industry.

ACKNOWLEDGMENT

The authors would like to thank the Department of Computer Science at John Brown University for providing facilities and infrastructure to carry out this project, as well as the Food Processing Company.

DECLARATION STATEMENT

Authors are required to include a declaration of accountability in the article, including review-type articles, that stipulates the involvement of each author. The level of detail differs; some subjects yield articles that consist of isolated efforts that can be easily detailed, while other areas function as group efforts at all stages. It should be after the conclusion and before the references. After aggregating input from all authors, I must verify the accuracy of the following information as the article's author.

- **Conflicts of Interest/ Competing Interests:** Based on my understanding, this article has no conflicts of interest.
- **Funding Support:** This article has not been funded by any organizations or agencies. This independence ensures that the research is conducted with objectivity and without any external influence.
- **Ethical Approval and Consent to Participate:** The content of this article does not necessitate ethical approval or consent to participate with supporting documentation.
- **Data Access Statement and Material Availability:** The adequate resources of this article are publicly accessible.
- **Author's Contributions:** All authors have individual partnerships in this article. Roberto Aguero has worked on YOLOv10 models and object detection. Both Roberto Aguero and Noah Olson conducted the testing part and obtained the test results. Justus Selwyn has provided the problem statement for the food processing company, offered guidance and feedback on the work throughout the project, and contributed to the writing of this paper.

REFERENCES

1. A. Wang, H. Chen, L. Liu, K. Chen, Z. Lin, J. Han, and G. Ding. YOLOv10: Real-Time End-to-End Object Detection. arXiv preprint arXiv:2405.14458, 2024. DOI: https://doi.org/10.48550/arXiv.2405.14458
2. Srivastava, S., Divekar, A.V., Anilkumar, C. et al., "Comparative analysis of deep learning image detection algorithms", J Big Data 8, Vol. 66, 2021. DOI: https://doi.org/10.1186/s40537-021-00434-w
3. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition, Pearson Series, ISBN-13: 978901246109 93, 2021. https://people.engr.tamu.edu/guni/csce625/slides/AI.pdf
4. J. Howse, J. Minichino, *Learning OpenCV 4 Computer Vision with Python 3*, Packt Publishing, 3rd Edition, ISBN-13: 978-1789531619, 2020. https://www.amazon.in/Learning-OpenCV-Computer-Vision-Python/dp/1789531616
5. A.Torralba, P.Isola, W. T. Freeman, *Foundations of Computer Vision*, The MIT Press, ISBN-13: 978-0262048972, 2024.

https://mitpress.mit.edu/9780262048972/foundations-of-computer-vision/

6. Christopher M. Bishop, Hugh Bishop, *Deep Learning: Foundations and Concepts*, Springer, ISBN-13: 978-3031454677, 2024. DOI: https://doi.org/10.1007/978-3-031-45468-4

AUTHOR'S PROFILE

Roberto Aguero was a computer science student at John Brown University. His areas of interest are Machine learning, computer vision, and applied mathematics. While an undergraduate at JBU, he worked on several academic projects in Machine Learning, Java, and Web Technologies, and contributed to the Collaborative Design Lab by building an accident handling system for bikers. Along with his major in computer science, he also did a minor in Mathematics. He has also presented his project work at conferences and made poster presentations.

Noah Olson was a computer science student at John Brown University. His areas of interest include computer vision, software engineering and development. He did several academic projects for his Java, Python and Web Technologies courses. He also completed a game project for the Collaborative Design Lab. He presented his work as a poster presentation at conferences. Currently, he is pursuing his master's in computer science. He is also passionate about music production and performance.

Justus Selwyn is a professor & chair in the Department of Computer Science at John Brown University. With a doctorate in computer science, he specializes in software engineering, artificial intelligence, machine learning, software analysis & design. He has several of his research works published in conferences and journals. Currently, he is working on projects that design and build AI systems and algorithms for small and medium-sized businesses, as well as virtual visual assistants. He also provides consultancy services to several countries in the NWA region. He is also a member of the IEEE Computer Society and ACM.