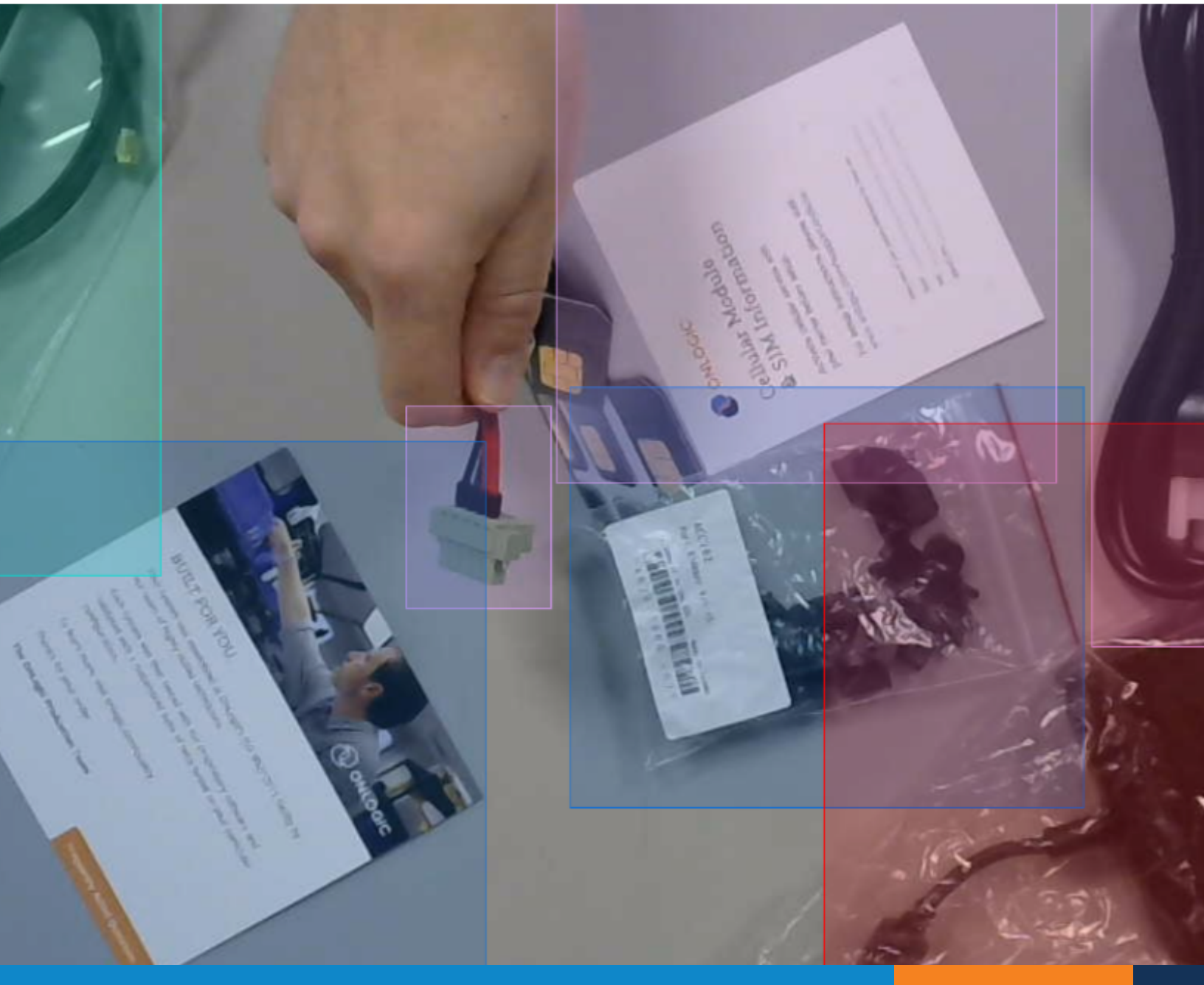
ONLOGIC

Using Synthetic Data to Fine-tune YOLOv10 for Order Accuracy Validation



ONLOGIC

Table of Contents

Abstract

In today's fast-paced manufacturing environments, ensuring the accuracy of outgoing orders is a critical task. Traditionally, this process has relied on manual verification, which can be time-consuming, error-prone, and subject to human fatigue. To address these challenges, this white paper explores a vision-based solution built by the team at OnLogic. The solution outlined below leverages a fine-tuned YOLOv10 real-time object detection model to automatically detect and verify components within an assembly cell. By augmenting the assembly line with cameras, and integrating synthetic data into the training process, this solution can improve the accuracy and efficiency of order fulfillment.

This paper describes the technical details of this approach, including data collection, model training, and testing infrastructure, as well as the potential benefits of this solution for manufacturing operations.

The practical application of machine vision for quality assurance

Validating outgoing order shipments is an issue commonly backed by human verification. However, this task can be prone to error due to volume of orders, time constraints, and simply human error. Further exacerbating the time and volume constraints, when the team of verifiers are the same individuals assembling the items for shipment, it is less likely that a person, or a colleague, will catch their own mistakes.

These types of tasks are often well suited for machine learning solutions, where humans are tasked with repeatedly and quickly performing a task that a human may be capable of performing accurately given sufficient time. The challenge is not necessarily one of difficulty, but of speed and consistency over an extended time frame.

As such, we looked to leverage a computer vision-backed verifier to aid in ensuring the accuracy of outgoing orders using our internal assembly line as a test case for the technical solution. By augmenting assembly cells with multiple cameras at different angles, we are able to capture a complete view of the assembly table. This view is then fed to a fine-tuned YOLOv10 model that attempts to detect the items present at the time of assembly. These item detections are then compared against an expected parts list, derived from a bill of materials, to check for any missing, extraneous, or mis-picked parts. As an additional benefit, this process allows for easy logging of all outgoing orders, allowing for both an objective checkpoint in the assembly process and a review source if a customer reaches out regarding a shipping error.

As part of building this system, there were several key parts to address. First, we needed an accurate and detailed dataset that captured the possible views of our components during assembly. Additionally, we needed a method for verifying the results of our detection model in a real-world scenario. Finally, this process augmentation would need to be completed with minimal disruption that might affect current processes, both while building the system and after introduction.

Approach

At a high level, there are three parts that need to be designed and built in order to support order accuracy validation:

- Data collection
- Model training
- Test infrastructure

Data collection and labeling

Data collection is vital for training models to accurately interpret raw data. A comprehensive dataset will enable neural network architectures to learn diverse visual patterns, enhancing their ability to identify objects and scenes. The quality, quantity, and diversity of collected data will directly impact the effectiveness of a computer vision model for any task.

The data for the project consists of image files. The images are labeled with information about what the image contains and where, information called its object class. The specific format for labeling the images is defined by the expectation of the model, but in general, images are often labeled by drawing a bounding box around the object(s) of interest. The coordinates of these bounding boxes, along with the object class, are then stored in a separate file, and each image has its own corresponding label file.

We accomplished this task by setting up a simulated assembly cell and mounting an overhead camera. Items were then placed on the assembly mat while the camera took images, which were then stored in a cloud bucket for easier retrieval.

However, once the images were stored,they still needed to be labeled. Due to the volume of images, we chose to leverage Label Studio, an open-source platform that offers a web-interface for labeling data to train ML models. Figure 1 shows an example of the assembly cell, while figure 2 shows an example of Label Studio.

Fig. 1. Assembly cell used for simulations

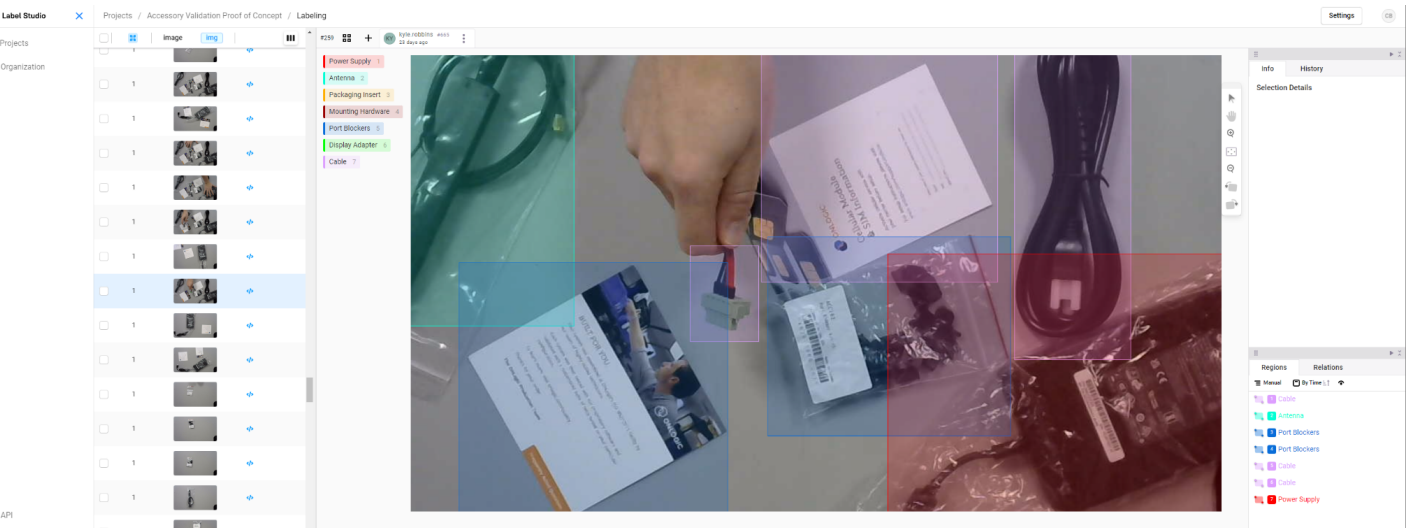


Fig. 2. Web-interface for Label Studio

Synthetic data

The process for collecting and labeling images is incredibly labor and time intensive. Thus, in order to augment the real-world data, we leveraged synthetic data created by Data Monsters for this effort. Data Monsters modeled a problematic part, a display adapter, in a 3D software and produced 500 example images of the part in varying orientations, placements, backgrounds, and light conditions. Additionally, a subset of these images were of the display adapter in a plastic bag, which could cause issues during training due to occluding the object (an example is shown in Figure 3). Lighting differences leading to random reflections off the bad cause particular difficulties for the model.

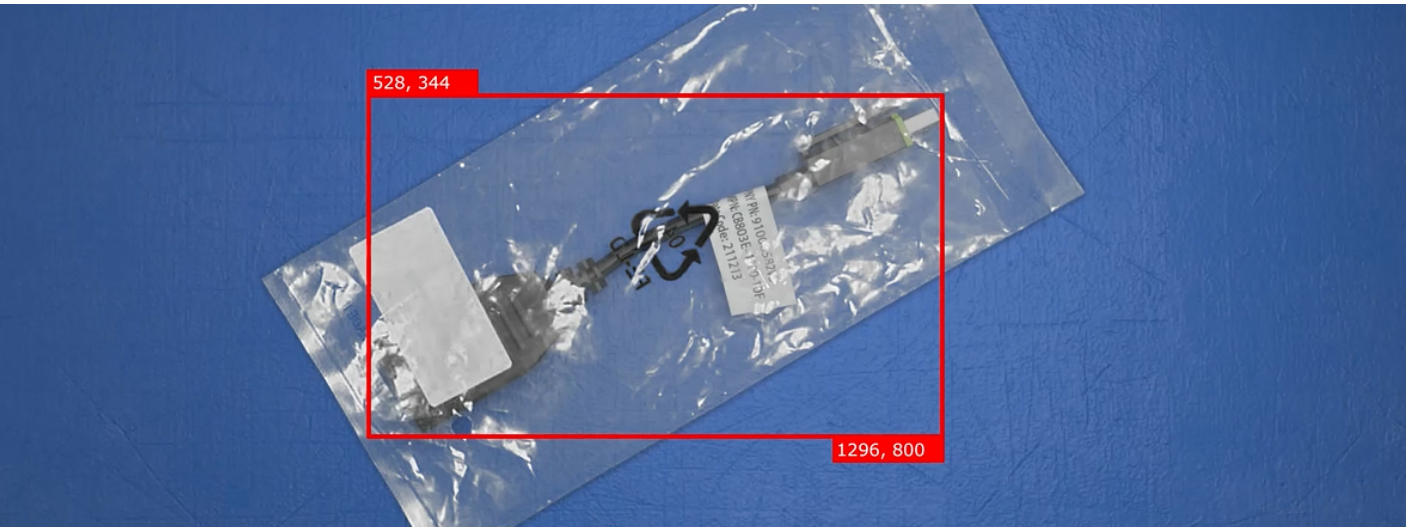


Fig. 3. Synthetic image of a display adapter in a plastic bag with glare, rendered in a 3D environment

A further benefit of synthetic data generation is that the objects did not have to be labeled manually. Instead, the images provided by Data Monsters were rendered with bounding boxes and programmatically-generated labels.

Model training

Rather than training an architecture from scratch, we used transfer learning in which a pretrained model is fine-tuned. This approach is considerably faster and more energy efficient than building and training a neural network from scratch. Additionally, leveraging a pre-trained model will often outperform a model only trained on data specific to the task as it will better resist overtraining to the dataset and better adapt to any out-of-distribution features that did not appear in the training set. The YOLOv10 model architecture was selected to verify that computer vision is able to fit to the training data.

Along with training, we tuned the model hyperparameters using an evolutionary-based tuner, an algorithm that iterates over time to optimize performance. The tuner creates a genome based on the hyperparameters that most affect the performance of the model, and each successive generation mutates the current genome slightly. Due to the computational and memory requirement of training a model, each generation only had a population size of n=1, which reduces the genetic diversity the evolutionary algorithm is able to leverage while optimizing the hyperparameters, reducing the efficiency of the tuning process on a generational scale.

The goal is not to find a set of best-performing hyperparameters; we care more about training a well-performing model rather than fitting a model to the optimal hyperparameters. Thus, at the end of tuning, we need only to extract the best trained model.

The tuning process was run for 1500 generations, where each generation was allowed to run for 50 iterations, with an early stopping lag of 4 iterations. That is, if during a training run, the model loss did not improve over the past 4 training passes, the current generation is terminated and the next generation begins.

Results

Separate YOLOv10 models were fine-tuned using the tuner described previously (1) using only real images and (2) using a dataset composed of both real and synthetic data. The results are captured in figures 4-7, where we find that the use of targeted synthetic data is able to drastically fill gaps in model performance, specifically for the display adapter. Furthermore, we find that including the targeted synthetic data not only improves the performance of the selected components, but leads to carry-over improvements when detecting other components as well.

The model architecture was the smallest size of 2.3M parameters, and the batch size was set to 32. Training occurred on an A4000.

Non-synthetic data

The first model was trained using solely the real-world data collected from the simulated assembly cell. Each training pass took 4 seconds on average.

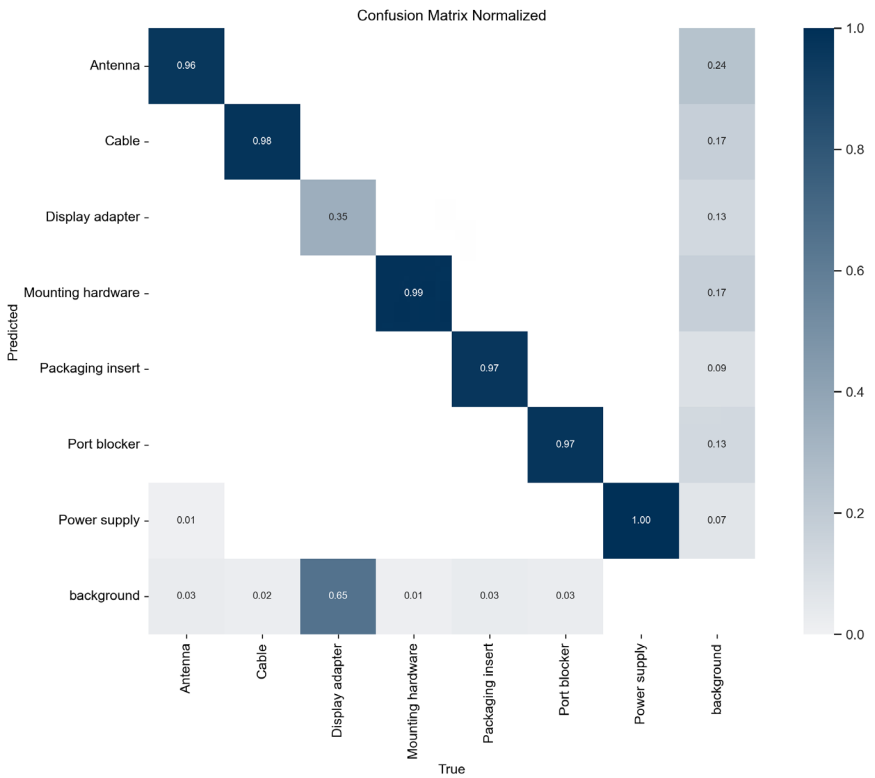


Fig. 4 Normalized confusion matrix: real data

This model struggles with the display adapters significantly more compared to the other object classes. Notably, it does not often confuse it with other components, but misses detecting it completely. This is likely due to the fact that this component is often in a bag with glare obscuring a large part of the component. While other components are sometimes bagged, the display adapter has a larger intra-class distribution due to the variety of types of display adapters that may be present.

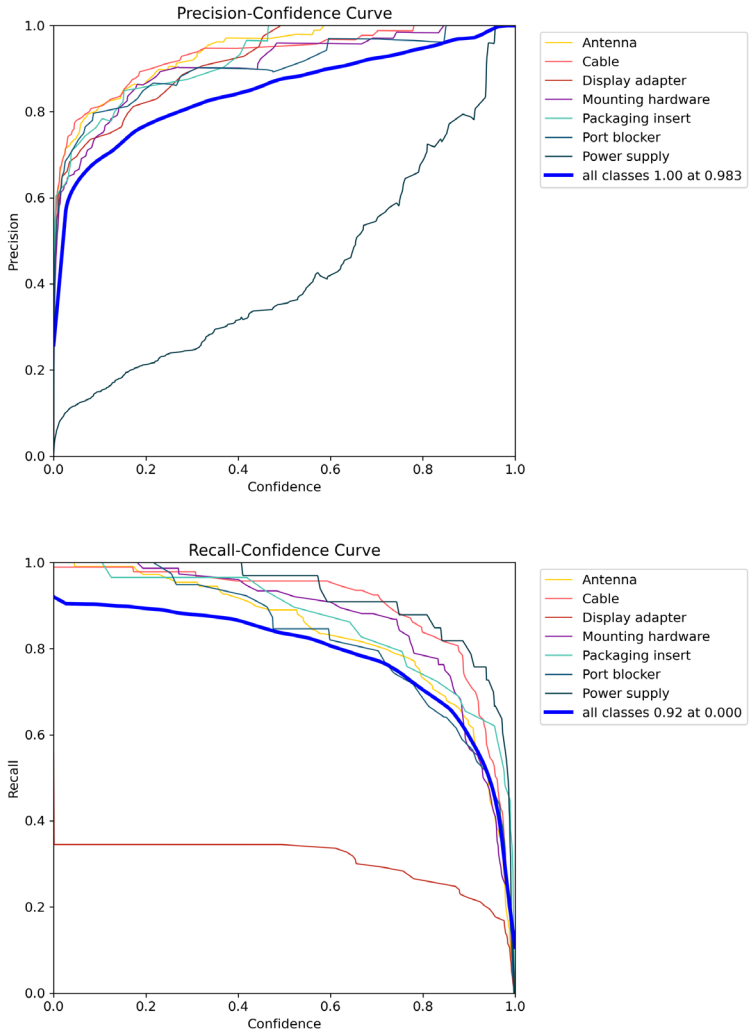


Fig. 5: (A) Precision and (B) recall curves for real data.

There are a few methods available to us to improve model performance in this situation, as informed by the precision and recall curves shown in Figure 5. The dropoff in the recall curve suggests it had trouble learning distinguishing features for a large percentage of the training examples, possibly due to a large inter-class variance among the examples provided. Furthermore, the low precision suggests a higher number of false positives. Combined with the confusion matrix, these data suggest the model may often be misclassifying the background as a display adapter.

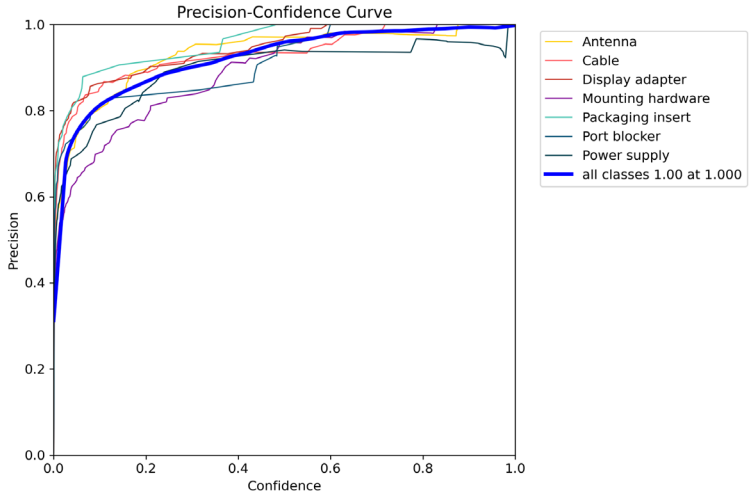
The model performance could likely be improved further by targeted training of this component or by increasing the number of training examples of this class specifically. In our case, we looked to examine how including targeted synthetic data could help remedy these difficulties.

Incorporating synthetic data

Another model was then trained by combining real-world data with synthetic data. Each training pass took 6 seconds on average, 2 seconds longer than with the real-only dataset as this dataset was 66% larger with the addition of the synthetic images.

Fig. 6. Normalized confusion matrix: combined real and synthetic data

Incorporating both datasets during training largely eliminated the confusion the model had with the background and the display adapter, as can be seen in the normalized confusion matrix shown in figure 6. Additionally, this performance gain carried over to the precision and recall curves as well, where we see the display adapter performing on par with the other component classes.



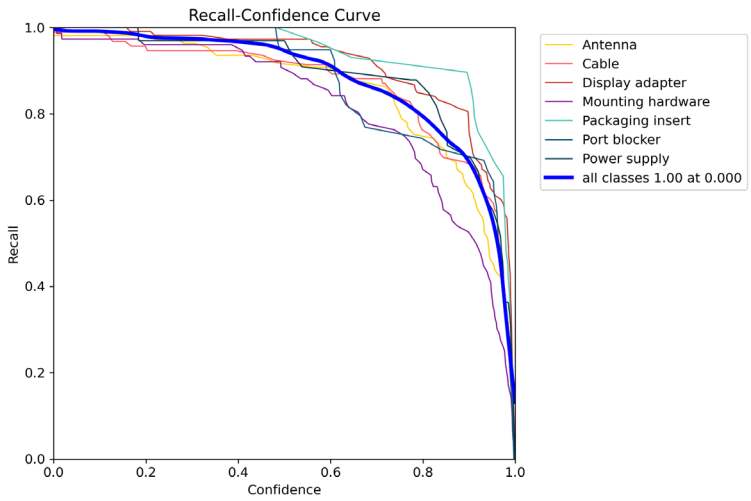


Fig. 7: (A) Precision and (B) recall curves for combined real and synthetic data

As a further check on the model performance, both false positives and false negatives want to be minimized, with a slight priority given to false positives. In the assembly process, it is more common that a component will be missed rather than an extra component included as assemblers add components during assembly rather than removing unneeded ones. This priority to false positives is aligned with the performance of the model in the precision and recall curves.

Model comparison

Comparing the performance metrics of using only real data alongside incorporating synthetic data shows the drastic performance difference this results in. Figure 8 shows this comparison using mean average-precision at the intersection of unions threshold of 0.50 (mAP50). Effectively, this captures how well the model performs on the "simple" examples, which will likely account for the majority of cases since we can expect the components to be separated out on the assembly platform while they are being gathered.

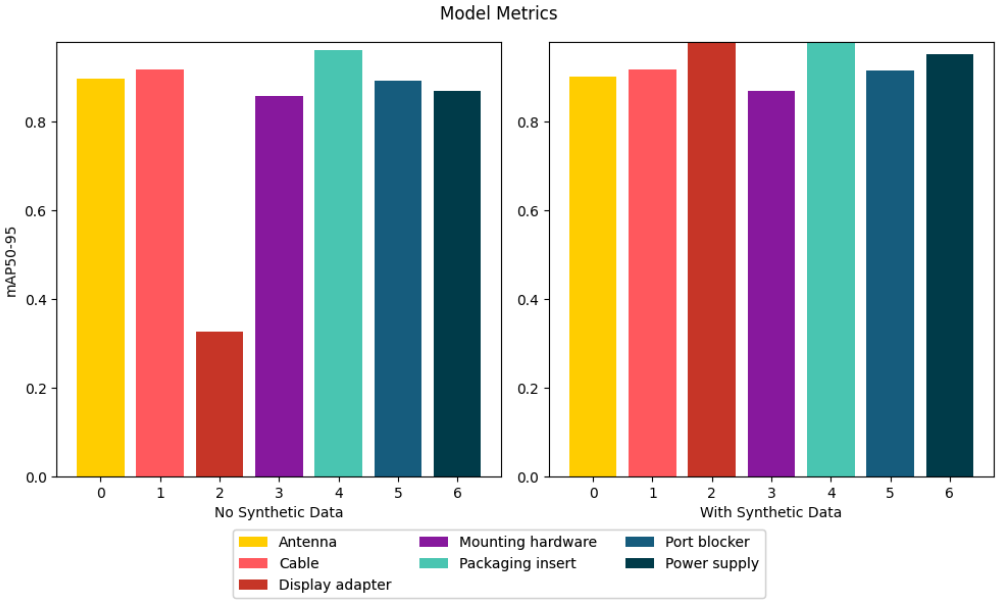


Fig 8: mAP50 results for data trained without synthetic data vs trained with synthetic dataset, evaluated on only real images. Synthetic data was only used for the display adapter (class 2). Including the synthetic data also improved the mAP50 of other classes.

Notably, while the synthetic data only included images of display adapters, including this data during training also improved the performance when considering power supplies and port blockers. This result may be due to a common feature between the separate components, possibly the power supply having the attached power cord that interferes with the model's ability to distinguish between the power cord and the adapter. This relation likely holds true for the power supply and port blocker as well: An improvement in distinguishing the power cable from the display adapter has carry-over effects for the power supply and port blocker.

Conclusion

The goal of this proof of concept was to verify the feasibility of automatically validating outgoing shipments using computer vision, along with exploring the benefits of incorporating targeted synthetic data during the training process. Using data collected from a real-world environment, the model was able to achieve a measure of success of object-level detection. However, there were initially issues in detecting some classes, namely display adapters. Specifically, the model would be prone to more false positives than might be useful for deployment.

By incorporating targeted synthetic data, we were able to iterate on the model performance much more rapidly than solely using data collected from the real world, enabling us to reduce the mAP50, precision, and recall errors drastically when measured on the display adapter. Furthermore, this improvement applied not only to the class captured in the synthetic data, but carried over to improvements in performance for other classes as well.

Future work

On the modeling side, performance of the model could likely be further improved by incorporating a varying set of expected backgrounds. This addition could help with the precision issues of the initial model when detecting display adapters and would also likely cut down on false positives of the model in general. Additionally, data could be further synthesized in an automated manner solely from the real-world data through crop and splice techniques. For this, we would extract the regions of interest of non-overlapping bounding boxes, yielding a set of images of each class individually with a tight boundary. These images could then be stitched back together in random assortments while applying additional random data augmentation techniques such as skew and rotation. Not only could this be used to target weak points of the model, but could also help break unintended associations the model may make due to how we laid out the components while gathering the training set (e.g. a power supply is always accompanied by a port blocker).

On the business side, simple, repetitive tasks are common in the manufacturing process, such as with fault detection and inventory management. With the increasing ease of training computer vision models, these tools may be well-suited as a fill-in for human monitors for parts of the process, allowing for humans to handle more challenging aspects of the manufacturing process.

About the author

Andrew Festa is a machine learning engineer at OnLogic who assists the company and its customers in turning AI-based ideas into tangible solutions. Andrew's experience includes research into reinforcement and evolutionary learning in a multi-agent context, as well as time-series classification of electroencephalographic signals. In his previous work, Andrew served as a software engineer on the verification and validation team at UTC Aerospace, and an AI software engineer at IOMAXIS. Andrew finds fulfillment in developing solutions that leverage cutting-edge technology to overcome complex challenges.

Computers you can depend on

Here at OnLogic, we create highly-configurable small form factor computers that thrive where others fail. We collaborate with innovative companies around the world, working together to solve their most complex technology challenges. Our consultative approach combined with a powerful online platform, our quick-turn production capabilities, and modular hardware design, can help make the impossible possible.

Industrial Automation | **Machine Learning** | **Edge Computing** | **Embedded Applications** | **Product Security**

Ready to get started? Contact us!

Our team of experienced hardware specialists are here to help you select and customize your ideal computing platform. Reach out for a free project consultation.

North America

Call: +1 (802) 861 2300
Email: info@onlogic.com
www.onlogic.com

Europe

Call: +31 88 5200 700
Email: info.eu@onlogic.com
www.onlogic.com/eu-en/