Operations | Technology | Defect detection / product inspection | Convolutional neural networks

# Deep Learning Models of Scanner/Vision Tunnel Performance in Sortation Subsystems

Amazon

## BUSINESS PROBLEM

Scanner/Vision tunnel performance at Amazon's large crossbelt sorter sites tends to average around 80-90% read rate success, contributing to a large amount of manual rework and recirculation impacting sorter utilization. Amazon is well away from their target of 98% scanner performance for these sites. Furthermore, the mechanisms to deep-dive scanner issues make it extremely difficult to categorize no-reads (unsuccessful scans) into operational or actual equipment issues. As a result, Amazon has very little visibility as to no-read causes across sites and cannot properly put together a plan to improve the situation.

## APPROACH

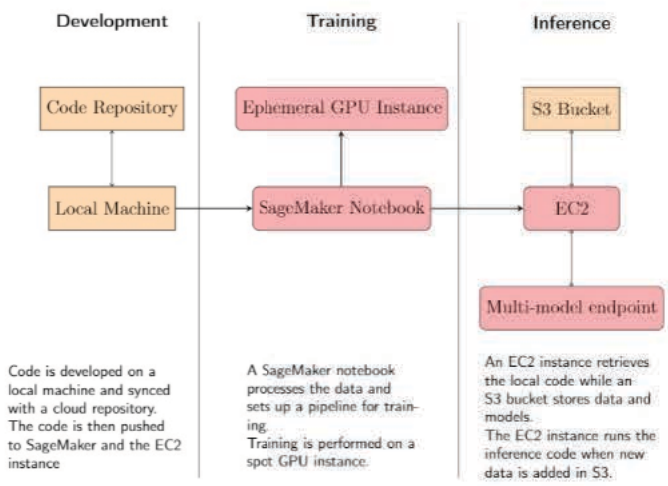
To address the no-read issues we witnessed across the fulfillment network, we build a pipeline on AWS to process scanner images. Then, we develop a deep learning ResNet model through AWS SageMaker to assign fault reasons for each image. A user interface finally allows operations managers to see which sites are lagging behind, launch deep-dives and test operational or equipment fixes.

## DATA SOURCES

We have access to hundreds of thousands of daily images of no-reads (unsuccessful scans) across most of the North American fulfilment network. We then manually label over 2300 images across ten sites in scope for our supervised models. We mostly use images from large cross-belt sorters, as these are the sorters handling the most volume and where issues are the most common.

### Data Types and Format

We use grayscale images with a resolution depending on the scanner settings but most commonly being 2048x1088.



## IMPACT

Our solution finally allows engineers and operations managers to understand the cause of no-reads at their respective sites and empowers them to address the issues. Fulfillment centers can see hundreds of thousands of packages every day and finding the root cause of tens of thousands of no-reads often cannot be done manually. Each package with a scanner no-read ends up in a separate chute, where associates need to manually determine what has happened and re-induct the package on the conveyor belt. As such, no-reads generate unnecessary manual labor, heavily impact sorter utilization and can lead to missed delivery windows. Despite the subjective nature of multiple labels, our models show less than 2% aggregated error across our validation set and less than 5% error across previously unseen scanners or sites, effectively correctly reporting all of the site-specific trends and issues. We further confirm the user experience through multiple pilots and tool demonstrations across North American sites. A conservative entitlement is approximately $2.2MM for the pilot sites in annual savings excluding customer impact, although the tool has the potential to save significantly more if it is actively adopted across the network. Our tool also further opens the door to a variety of other initiatives. Ultimately, the pipeline we deploy can be used for other purposes such as damage detection or sorter analytics.

### DRIVERS

With the rise of one and two day guaranteed deliveries, fulfillment centers are under increasing pressure to meet their targets and reduce operational issues. Given the recent improvements in computer vision models and their deployment over the cloud, there are now countless opportunities to leverage deep learning to automatically gather insights and act on supply chain faults at scale.

### BARRIERS

There are two main challenges with this project. First, the data collection can be much harder for certain sites and the format of images can greatly vary depending on the scanner vendors or the type of site. Second, some of our classifications are inherently subjective, such as determining if a package is centered, which creates uncertainty for machine learning models.

### ENABLERS

Amazon's fast-pace environment was critical to achieve all of this in such a short timeframe. Blockers were promptly addressed and we were able to focus on activities delivering value. There is also a great amount of expertise and support on both the operational and technical side available for such initiatives.

### ACTIONS

We first visited multiple sites to have a good understanding of the scanner issues they face. Then, we implemented the end-to-end deep learning tool while seeking constant feedback from our end-users. Once the tool was ready, we piloted it across a few sites to analyze its efficiency and perform final tweaks. Finally, we wrote detailed technical documents and an internal press release to launch it across the network.

### INNOVATION

While the algorithms we used are well-known to deep learning experts, they had yet to be used in the context of fault classification at Amazon. Furthermore, we leverage the red and blue channels of our grayscale images to contain additional statistical processing rather than repeat the gray intensity. This novel technique greatly improved our accuracy for certain models.

### IMPROVEMENT

The cost of missed deliveries in fulfilment centers, of which no-reads are a major cause, is estimated to be between $20M and $50M annually. In addition, rough estimates of the cost of no-reads for a single inbound cross dock in our pilot are over $9M annually. While time will tell how much of this cost will be reduced through the use of the tool, our conservative estimate of the annual savings is around $2.2M.

### BEST PRACTICES

Anyone replicating this process should carefully design a scalable and flexible pipeline. Such a tool requires several iterations of datasets, models and processing and it is critical to take the time to design the end-to-end process robustly before jumping into the deep learning component. Then, we need to give great care in determining when a model is good enough. Several great models can often deliver more value than single exceptional one.

### OTHER APPLICATIONS

A major component of this thesis relates to the use of AWS SageMaker to build and deploy machine learning models. Most of the architecture we describe could easily be adjusted for a variety of contexts such as product tracking, manufacturing defect detection or forecasting. Furthermore, the architecture we built at Amazon could be used for other applications such as package damage detection with only very minor changes.