Final Report

AUSTRALIAN MEAT PROCESSOR CORPORATION

Final Report

Identification of Naked Loins Using Machine Learning Algorithms

Project Code	Prepared by	Date Submitted
2021-1274	MEQ Probe	29/03/2022
	Published by	**Date Published**
	AMPC	29/03/2022

Contents

1.0 Executive Summary

1.1 Project objectives

This project concerns the development of algorithms, software and machine learning models to identify naked lamb loin primals on an abattoir conveyor belt at chain speed. The automated identification of these primals has the potential to offer significant advantages to processors in terms of product reliability and reduced labor costs, and would additionally represent a strong foundation on which the development of related models and systems could be based for other applications relevant to meat processors.

The scope of the research includes targeted data collection, data labelling, machine learning model development, algorithm design and development, system evaluation and testing. The research was conducted in cooperation with Gundagai Meat Processors (GMP). The specific objectives addressed by this project are:

- To produce a labelled dataset consisting of 30,000 images of naked lamb loins.
- To develop a machine learning model that is able to identify a naked lamb loin on the GMP boning belt with 99.9% accuracy at chain speed.
- To develop a machine learning system that can alert a nearby user if the model is unsure, allowing for manual override.

1.2 Data collection and labelling

In order to develop a machine learning model to recognize objects, it is necessary to capture raw data (in this case: images of primals) and then label those data with relevant information (in this case: the locations and types of primals in the images) to build a high quality dataset which can be used for model training, analysis and evaluation. Data collection was carried out in two phases:

1. In the initial phase, data was collected manually by human operators using both fixed and handheld cameras.
2. In the second phase, with an initial model, software, and fixed camera hardware in place, automated heuristics were used to identify and record relevant data based on identified weaknesses in the in-development model, as part of an active learning regime.

In both phases, the collected raw data were then subject to a four-part labelling process:

1. The in-development machine learning model was used to make provisional object identifications in the images.
2. Using a specialized computer vision annotation platform, human annotators familiar with lamb primals corrected the model's output.
3. These human-corrected labels were reviewed by a highly experienced member of the team to ensure the accuracy of the object identifications.
4. Finally, some automated processes were carried out on the reviewed labels to render the data suitable for use in training the machine learning model.

1.3 Machine learning model development

The machine learning model architecture chosen for the object identification task was YOLOv4, which is both highly accurate at object detection, and optimized to allow for real-time predictions. The process of model development involved data collection and labelling as described above, data augmentation to increase the effectiveness of training, and hyperparameter optimization for both accuracy and speed of prediction.

1.4 Model output post-processing

Aside from the development of the core machine learning model which is able to recognize primals within an image, the overall system also incorporates several algorithmic post-processing steps which are able to take advantage of the ability of the system to capture multiple images of a given primal across time and from different angles across several cameras in order to further increase the overall accuracy of primal identification. The primary post-processing algorithms applied to the raw machine learning predictions are as follows:

1. An object tracking algorithm to track individual primals through time as they cross each individual camera's field of view.
2. A cross-camera matching algorithm to match detections of the same primal from different angles across multiple cameras at a given moment in time.
3. An overall classification algorithm which uses the outputs of the machine learning model along with the temporal tracking and cross-camera matching information produced by the above algorithms to produce highly robust object classifications, which do not depend solely on any single captured image.
4. An object counting algorithm which uses the output of the overall classification algorithm to count the number of each type of primal passing the camera station.

Although the counting of primals represents an additional degree of complexity beyond the formal scope of the project, a counting algorithm was included in the system both because of its utility in system evaluation, and as a demonstration of a potential application of the technologies and methods developed and employed in the project.

1.5 System validation and evaluation

While the initial project proposal included an intention to evaluate the system in by having a human evaluator monitor the system during operation in the plant in real-time, during the development of the project it became clear that it was not feasible for a human to accurately monitor up to six cameras and the belt itself to confirm the accuracy of the predictions given the rate at which primals appear on the belt. To solve this problem, a more sophisticated evaluation regime was developed, in which data was recorded at the system during plant operation to be played back later using specialized simulation software, allowing the human evaluator to review the system's predictions in slow motion, frame-by-frame, or even by rewinding and replaying sections to ensure that they are able to accurately check the system's predictions. Furthermore, the evaluations were based on the counts of primals generated by the system, which demands not only that the system correctly identify each primal, as required by the project specification, but that it is additionally able to ensure that each primal is counted exactly once.

1.6 Results and limitations

For naked loin primals, the system was evaluated against captured data containing over 1000 individual naked loins captured over a week of operation, and the system was shown to have a 100% rate of accuracy at counting the loins in that dataset.

1.7 Findings, recommendations and conclusions

The success of this project provides strong evidence for the applicability of computer vision and machine learning techniques in the abattoir boning room, and we would recommend that further research and development be undertaken to bring these technologies to bear in further applications which have direct and specific benefit to processors. In our opinion, these technologies could have immediate value in the following areas:

- Automatic Sorting: a machine learning/computer vision system could be developed to control mechanical systems such as belt diverters to enable automated sorting of meat cuts in certain situations.
- Supply Chain Traceability: with some expansion of scope, potentially including more cameras and processing nodes, the technologies used in this project could be applied to the problem of boning room traceability to track an individual carcase as it is transformed into primals and individual cuts.
- Quality Assurance: a machine learning/computer vision system could be developed to identify and count objects once boxed, and alert users to the presence of foreign objects

Based on the results of this project, we conclude that machine learning and computer vision technologies can be employed in an abattoir environment to detect naked lamb loin primals with a high degree of accuracy, and further that these methods show strong promise in terms of applicability to boning room automation, analysis, and quality assurance tasks generally, with the potential to increase the reliability of abattoir processes and reduce labor costs.

2.0 Introduction

In the red meat industry a consistent product is key. The problem is there are a multitude of end-point products that can be delivered by a processor. Output from a single plant can be complex and diverse, with product specifications changing day-to-day to meet customer requirements. There is a certain level of skill required in identifying a piece of meat correctly and efficiently on the production line, and ensuring the product is meeting specifications, biosecurity and quality control measures. As the number of potential options for an object increases, so do the potential errors in classification. This is an issue in the red meat industry, as identification tasks such as packaging products and final checks for boxed meat are often delegated to unskilled labour, with the least experience in the red meat industry. This lack of skill and high rate of labour turnover often results in a high risk of human error occurring, disrupting work flow and having the potential to create quality and biosecurity issues for the processor. If products do not meet specifications, biosecurity requirements or are incorrectly packaged, there is the potential for reputational damage both domestically and internationally. If the wrong product is in the wrong box, the costs can be very high.

The aim of this project was to deliver a machine learning based computer vision system specifically designed to identify naked loins in Australian Abattoir boning rooms, helping to ensure the right product ends up in the right box. This development positively impacts the red meat industry because it proves that artificial intelligence algorithms can be used to successfully identify primals with extremely high accuracy, which reduces the risk of having incorrectly packed products. In addition, the machine learning model built at the GMP abattoir is solid enough that it can then act as a baseline for future models. This is because the model was trained with more than 30,000 images collected during different days and times from a variety of camera angles, which means that adapting this model for other similar processing plants will only require a short development iteration.

A variety of data collection, data labelling, software development, algorithm design, statistical, and machine learning techniques were applied and tested to successfully identify specific cuts of meat. This artificial intelligence system has the capabilities and flexibility to be applied across a wide variety of use cases throughout processing plants of the red meat industry. The ability to capture images that are processed and analysed at chain speed will enable output to be used in real-time decision making. Unskilled labour requirements will be reduced, and a greater level of quality assurance and biosecurity guarantee will be achieved consistently and efficiently across the variety of end-point products manufactured by processors.

3.0 Project Objectives

The objectives of this projects are outlined as follows:

- To produce a labelled dataset consisting of 30,000 images of naked lamb loins.
- To develop a machine learning model that is able to identify a naked lamb loin on the GMP boning belt with 99.9% accuracy at chain speed.
- To develop a machine learning system that can alert a nearby user if the model is unsure, allowing for manual override.

4.0 Methodology

3.1 Data collection

3.1.1 Initial data collection

Using a modular camera system prototype, images were collected in plant over a series of days. This system undertook initial testing with a simple software application and one Baumer camera for the capture of images at two predetermined locations in the abattoir. After assessing several viewpoints, camera settings, space for the mounted system, belt position, focal length, exposure and lighting conditions, one location on the boning belt was deemed suitable for the collection of naked lamb loins. These images were collected over a series of days in plant at the predetermined location on the boning belt, which allowed for different angles and types of naked loins to be captured. Both a handheld camera and a fixed camera system were used for the collection of images. See *Appendix 1* for a sample of the initial images collected in the abattoir.

3.1.2 Bulk data collection

Using a more advanced version of the modular camera system that included a total of six cameras with different lens types and angles, the rest of the images were captured simultaneously from fixed mounting points over a series of months. See *Appendix 2* for a sample of the new types of frames the new system was able to capture.

In parallel to the bulk data collection step, images were uploaded to an interactive labelling platform and annotated with rectangular bounding boxes that locate and identify primals. These were then submitted for quality review to an experienced member of the team to ensure maximum quality and consistency. Each primal dataset was converted to a csv file that contains information about the images captured including the objects located in the frame and of its respective coordinates.

3.2 Data labelling

Collected images were manually labelled for primal identification using bounding boxes. These are rectangular boxes used to define the location of the target object. Bounding boxes are determined by the x and y axis coordinates in the upper-left corner and the x and y axis coordinates in the lower-right corner of the rectangle. They are represented by two coordinates (xmin, ymin) and (xmax, ymax). One frame may contain multiple images of primals, referred to as objects in AI terms. Each object has its own object-class and bounding box ((xmin, ymin) and (xmax, ymax)), creating a filename that contains multiple, and sometimes overlapping bounding boxes.

A Computer Vision tool was used to annotate images, as it provides an interactive user interface and allows easy tracking and reviewing of bounding box annotations. See *Appendix 3* for sample images with bounding box annotations created in the Computer Vision tool.

Once all the collected images were labelled, a quality review process was initiated. All labels were reviewed and corrected by a single member of the team with expertise in identifying differences between primals. This ensures the dataset is high quality and consistent. All labels were then combined to a final csv file that contains one row per object identified. Each row has one value for the following columns:

a) Filename. E.g: gmp_20210930_202107.png

b) Frame width and height. Eg: 768 * 768

c) Object: E.g: ‘naked_loin’

d) Bounding box coordinates as xmin, ymin, xmax, ymax. E.g: 418.35, 131.13, 570.88, 216.76

See *Appendix 5* for a sample of the output

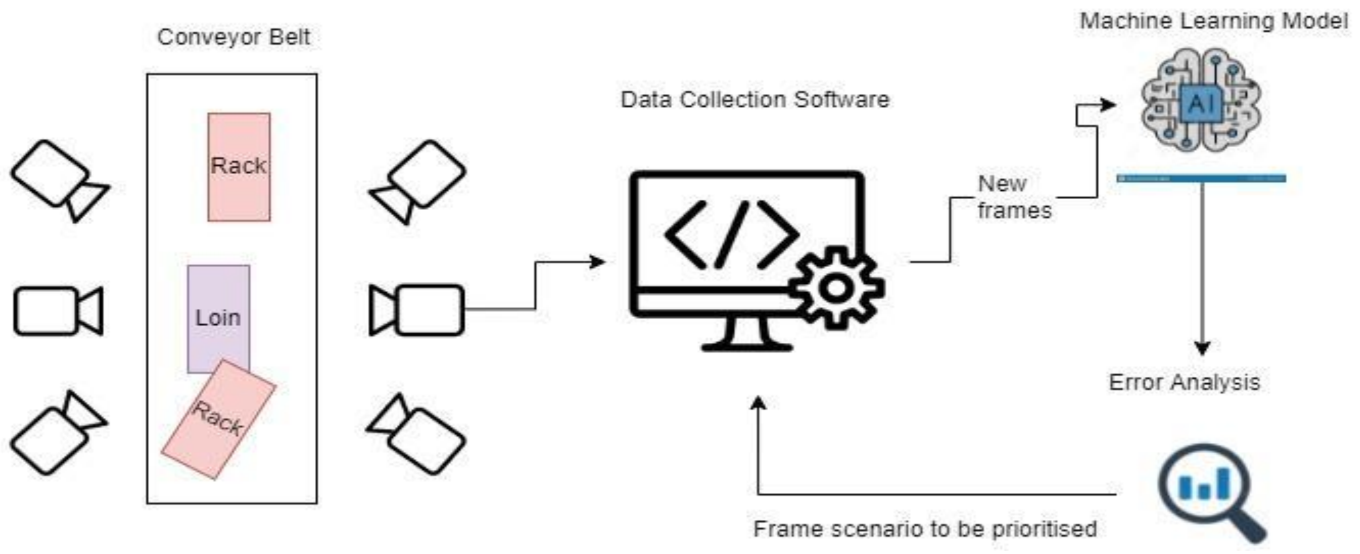
3.3 Improving data collection using active learning

Active learning (AL) is a subfield of machine learning that focuses on the study of data sets with the goal of obtaining performance gains by actively selecting the most useful samples to be labeled. In contrast to active learning, passive learning refers to when all data is given at once to the annotator (Ren *et al.*, 2021). AL has important applications in the machine learning community as it is proven that proper selection of samples for training the model can reduce the probability of mistakenly predicting the response variable for an unknown explanatory variable (Hino 2020).

The active learning technique was implemented by continuously analysing the model’s top errors and creating potential error detection mechanisms for the camera system. This technique allowed us to achieve the project’s objectives ahead of schedule because it allowed us to maximise performance with limited human intervention. Data was captured by the camera system, fed into the machine learning model for predictions, and then analysed for error detection. The output of the analysis would suggest a couple of the scenarios where we thought the model was struggling that then would be visualised by a member of the field team and confirmed as being model errors. The camera software system would then be updated to capture data in real-time that could simulate a similar scenario.

Figure 1: Active learning cycle implemented for this project.

Figure 1 shows how data was collected in real-time, fed into the machine learning model for predictions, and then to error analysis. One or more frame scenarios were selected as a priority for data that replicated a similar scenario to be collected in the coming weeks.



For example, after performing the first the data analysis on the model's errors it became clear that for some scenarios the model detected an object more than once as being in more than one class in a given image. In order to update the model to avoid this type of error, it was necessary to obtain many samples of images for which the model makes such double-detections, annotate them with the correct classifications, and then train a new model version with the resulting image-annotation pairs. To achieve this, an automated heuristic was implemented to identify frames where two bounding boxes are closely aligned with each other (within a small tolerance), and then these frames were stored and prioritized for human annotation.

While not all images gathered using heuristic methods necessarily produce the targeted type of error, careful design of the heuristic algorithms used to identify the images to store and prioritize allowed the automated production of sets of images with a high concentration of error-producing samples, and so by using this automated method of prioritization, we were able to rapidly and automatically target various identified error types for correction, allowing the model to be improved much more rapidly than if the images to be annotated were selected manually.

Table 1 describes the various types of errors we identified during model development, along with a description of the heuristic algorithms used to identify, store and prioritize images for annotation to target each error type.

Table 1: Types of errors identified during model development and heuristics used to prioritize images for annotation to target each error type.

Error Type Description	Automated Heuristic Used
Object is detected more than once as being in more than one class in a given image.	Checking for frames containing two or more bounding boxes that closely align (with a small tolerance). If two predicted bounding boxes overlap almost exactly, one of those predictions may be in error.
An incorrect class is predicted for an object on a particular camera, or an object is not detected on a particular camera.	Multiple heuristics were used: 1. Checking agreement between cameras as to the number of objects of each class within a specified region (with some tolerance) of each frame which views the same part of the belt. If cameras disagree, predictions for the image from the camera that is in the minority may contain an error. 2. Using the geometric multi-camera object matching algorithm described in the section below to match objects between camera frames based on their position within the frames. If a single object is detected across multiple cameras, but some cameras disagree on the correct classification of an object, the prediction(s) from the camera(s) that are in the minority may be in error.
Object is detected correctly in images from a particular camera, except for one frame in which it briefly receives an incorrect classification.	Checking pairs of consecutive frames for bounding boxes which are of different predicted classes, but which match closely in size and location. If this occurs, the object may have been incorrectly classified in one of the two frames.
Object is detected, but the predicted class is known not to be present on the belt at time of detection.	Checking all predicted classifications against a configurable list of object classes which are known a priori not to be physically present on the belt for a given recording session and location.
Object is incorrectly classified by the model	Checking all predictions from the model against a specified confidence threshold. Because a low confidence-value output from the model is indicative of high uncertainty in the classification, predictions with a very low corresponding confidence value may be in error.
Objects are incorrectly classified or not detected due to occlusion by other objects.	Checking number of detected objects present in each frame against a specified threshold. Images containing a high number of detected objects are more likely to cause the model to make errors due to occlusion than other images.
Object is incorrectly classified due to distortion of fish-eye lens	Checking number and position of detected objects in fish-eye camera images to find images where a single object is present in the higher-distortion edge regions of the image. These images are more likely to cause the model to make errors due to lens distortion than typical images, while also allowing rapid annotation as only the single object of interest is present in the frame.

3.4 Data cleaning and processing

After successful collection of images of naked lamb loins and the production of prioritised labels in the form of bounding boxes, a final dataset for the primal was created. Prior to feeding that dataset into an initial Machine Learning (ML) algorithm, data cleaning and preprocessing was required to remove corrupt and irrelevant records, and to transform data into an appropriate ML format. The steps for data cleaning and processing that were applied programmatically using Python were:

1. Adding empty labels for images where there is no primal.
2. Enforcing consistency in labelling names.
3. Clipping bounding boxes that fall outside the edges of the image
4. Normalizing coordinates to allow easier re-scaling
5. Resizing images and bounding boxes to fit model architecture.

This process ensured only the highest quality data was used to train the models.

3.5 Machine learning model development:

3.5.1 Model selection

The Computer Vision task required for this project is referred to as Object Detection in Artificial Intelligence terms. The developed model needs to be able to locate the presence of objects with a bounding box and predict the classes (naked rack, naked loin, and bagged rack) of the located objects in an image. Amongst the various models available to solve problems like this, You Only Look Once (YOLO) was selected as the preferred base architecture as it proves to be predicting faster than Region-Based Convolutional Neural Networks models like R-CNN, and more accurately than other single-shot models like Single Shot Detector (SSD) (Kim *et al.*, 2020). Multiple versions of YOLO architectures have been released, but the fourth version (YOLOv4) was specifically designed to produce extremely accurate results in real-time, which is why it was chosen for this project (Bochkovskiy *et al.*, 2020).

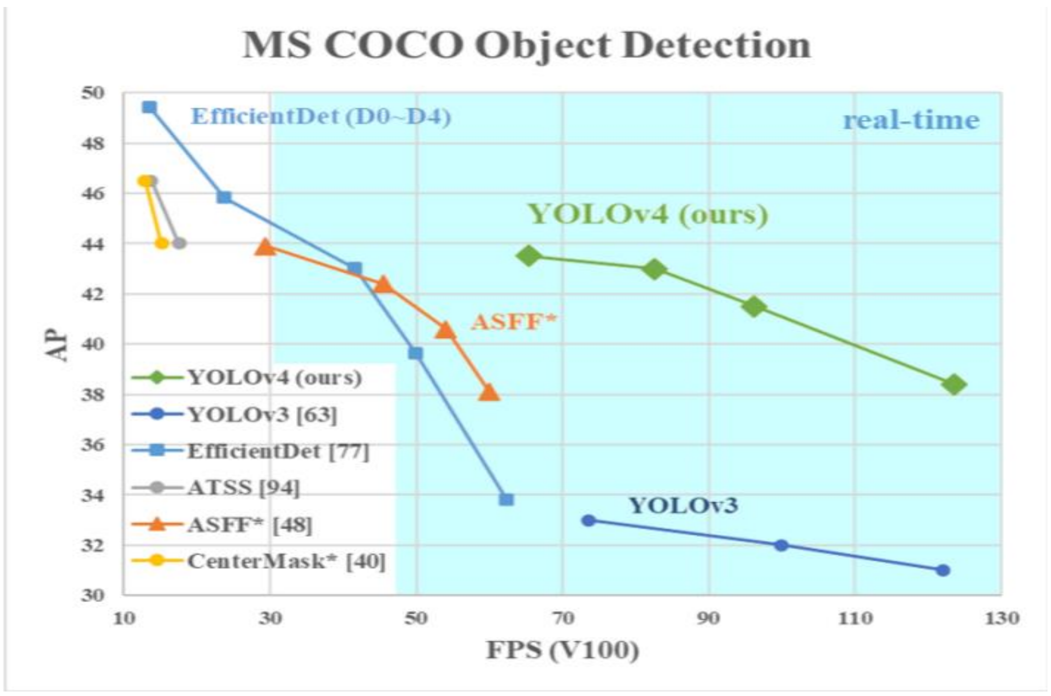
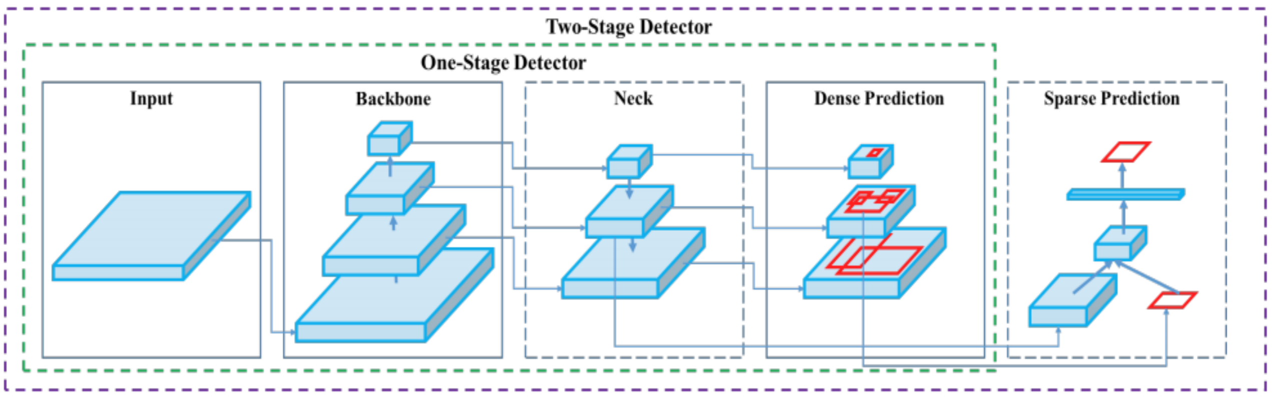


Figure 2: Comparison of the proposed v4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively (Bochkovskiy et al., 2020).



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

Figure 3: YOLOv4's architecture for object detection (Bochkovskiy *et al.*, 2020).

For this project, YOLOv4's architecture had to be modified to fit the requirements and trained from scratch on samples from the dataset delivered for the previous milestone.

3.5.2 Parameter configuration and hyperparameter tuning

Hyperparameter are those parameters whose value controls the learning process of an algorithm. In Machine learning, optimization or tuning is the process of selecting the best combination of hyperparameters that minimizes the predefined loss function on a validation dataset. The hyperparameters that achieved the best performance for the final model are the following ones:

Table 2: Hyperparameters chosen to train the final model.

batch	subdivisions	momentum	decay	learning rate	burn in	max batches	policy steps
64	16	0.949	0.0005	0.001	20000	98000	80000, 87000,93000

Below there’s a summary of what each parameter respresents (Redmon, 2016):

- The subdivision parameter controls how many samples will fit into RAM (minibatch/subdivision = samples loaded simultaneously per pass).
- Momentum refers to accumulation of movement, and it controls how much the history affects the further change of weights (optimizer).
- The decay parameter is a weaker updating of the weights for typical features, it eliminates dysbalance in the dataset (optimizer).
- The learning rate is the one used initially.
- Burn_in is the number of iterations that the initial burn_in that will be processed for.
- Max_batches is the number of iterations that the training will be processed for.
- Policy is the method for changing learning rate: constant (by default).
- Steps are the numbers of iterations at which the learning rate will be multiplied by the scale factor.
- Scales is the factor used to multiply learning rate. if policy=steps - f.e. if steps=8000,9000,12000, scales=.1,.1,.1 and the current iteration number is 10000 then current_learning_rate = learning_rate * scales[0] * scales[1] = 0.001 * 0.1 * 0.1 = 0.00001.

3.5.3 Data augmentation

Data augmentation is a common technique used in Machine Learning to increase the size and quality of samples by adding slightly modified copies of already existing data. This approach can help reduce overfitting, when a network learns to model the training data with a high variance, which can fail to fit future observations reliably. Augmentation procedures that were applied to this project include image rotation, shifting, cropping, resizing, blurring, scaling, and modifying saturation levels, exposure, and hue. The image augmentation algorithms that were used in this project to reduce overfitting had to be applied to both the images and its respective bounding boxes, which added another layer of complexity to the task as linear algebra had to be used to ensure bounding boxes were correctly modified. See *Appendix 4* for an example of before and after random cropping transformations used for data augmentation. After successful application of multiple data augmentation techniques, an improvement of up to 6pp was recorded for the model's accuracy.

3.5.4 Model size vs inference speed trade-off

Although developing a model that can achieve 99.9% accuracy is crucial to the success of this project, keeping up with the speed of the chain is as important as the model's quality. Building a very accurate system that is too slow to predict in real-time might not be of value to the plant. For that reason we performed an experimental analysis to select the ideal model size to maximize model's accuracy without sacrificing speed. The goal of this analysis was to understand the impact that model size has on the algorithm performance. Model version 12 (v12) was trained on an image size of 768*768*3 while model version 13(v13) was trained on 416*416*3. Both models had the same train/eval split to ensure the mAP comparison is fair. Results of the experiment can be seen below. Please note that mAP refers to the mean Average Precision (mAP) for all classes of primals on a frame by frame basis, and IoU refers to the Intersection Over Union that describes the extent of overlap between the predicted and the real bounding box. The greater the region of overlap, the greater the IOU. An IoU value of 1 indicates perfect overlap, whereas a 0 indicates no overlap.

Table 3: Results of an experiment to compare object detection performance between two different model sizes.

	Model Size	**Samples**	**mAP@0.5**	**IoU**	**Precision**	**Recall**	**F1 Score**	**False Negative**
V12	768*768	5952	95.52	84.73	0.96	89	92	1942
V13	416*416	5952	94.73	82.81	0.94	88	91	2179
Difference (+pp)			0.79	1.92	2	1	1	-237
Difference (+%)			0.83%	2.29%	2.11%	1.13%	1.09%	-11.50%

Regarding model performance by size, it is clear that a bigger size model is more accurate, as it is capable of providing improvements in Precision and IoU, and significantly reducing the number of FalseNegatives. On the other hand, the gains in model's quality are not directly proportional to the losses in speed. Speed was measured in frames per second (FPS), and as *Figure 4* shows, a smaller image size allows the model to predict more frames per second. To find a balance between speed and accuracy, the final model size selected for this project was 608*608*3.

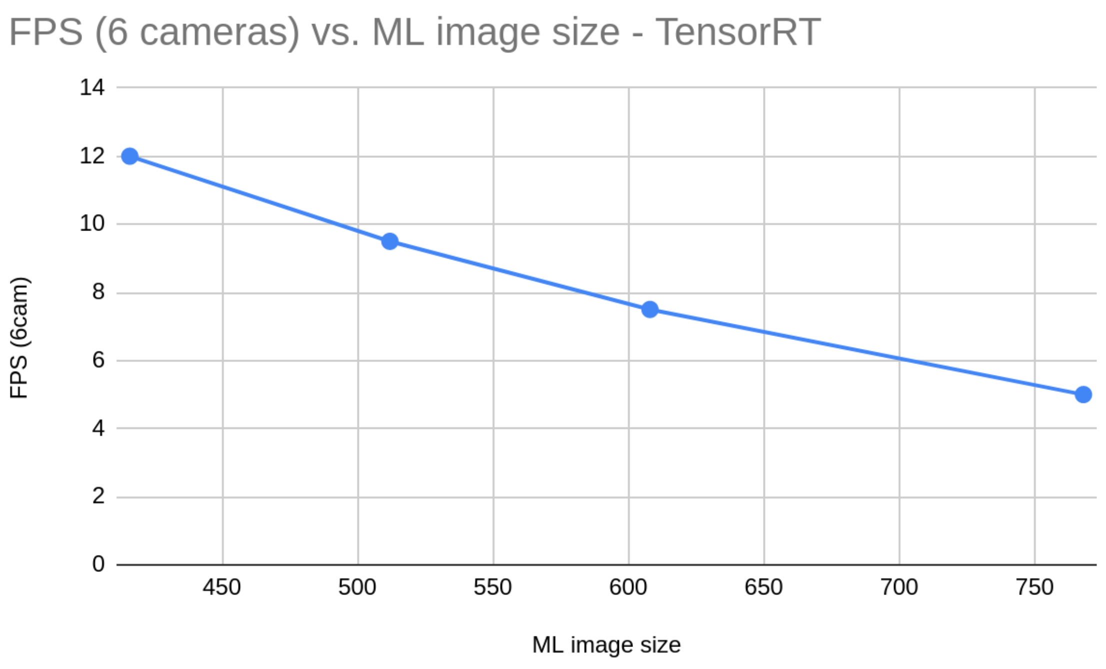


Figure 4: Comparison of ML inference processing frame rate with six cameras vs size of model input image.

3.6 Model output post-processing

This section describes the various types of processing that the system uses to enrich and contextualize the output of the core machine learning model in order to improve overall accuracy and produce additional useful results.

3.6.1 Cross-camera object matching

Selection of approach

While the core machine learning model developed for this project is able to detect and classify primals within a single image, the system itself incorporates multiple cameras which are in many cases able to simultaneously view the same primal. As such, if the system can determine which detections from different cameras correspond to the same actual object, then the accuracy of classification for that object can be improved, because incorporating predicted classifications from multiple images captured from different angles can significantly reduce the impact of single-frame errors.

However, in order to match detected objects between cameras, some additional processing must be performed. There are a number of possible ways to approach this, but the choice of algorithm was informed by the limitations introduced by the intended application of the system:

- Primals within a class tend to be visually similar to each other. This reduces the applicability of cross-camera object matching methods which rely on the visual appearance of objects in the calculation of predicted matches.
- The system is designed to be relocatable. This makes it less feasible to use methods which carefully measure the position and optical characteristics of the cameras to calculate the intrinsic and extrinsic camera parameters that would allow a direct theoretical mapping from real-world coordinates to pixel coordinates within each camera using a camera matrix, as these parameters would need to be re-measured each time a camera is moved or the system relocated.
- The system is designed to be used in various abattoir environments and in close integration with running meat processing apparatus, which limits the practicality of methods which require a specialized calibration object to determine the relative position of each camera. Because there may be limited space, or because it may not be possible to stop operating plant systems, it may in some cases be difficult to introduce a calibration object into the view of all cameras.

Given these concerns, the ideal cross-camera frame matching approach would be one that requires no special physical calibration, but which does not rely on the appearance of each individual object to match that object between cameras. While this could be achieved in numerous different ways, we chose to use a method which uses the outputs of the core machine learning model as landmarks to induce a direct geometric relationship between coordinates within each camera frame. This approach has the advantage of not requiring either explicit calibration, or the development and maintenance of a separate specialized machine learning model solely for cross-camera object matching. It does, however, require that the system be set up and left to gather data for several hours or days before it is able to learn enough information to start calculating reliable cross-camera object matches.

Description of algorithm

The cross-camera object matching algorithm developed for use in this project has two components: pre-calculation of frame-mapping information, and real-time use of the pre-calculated frame mappings to predict which detections across multiple cameras should be matched with each other.

Pre-calculation of the frame mapping data is carried out as follows:

1. The cameras are mounted securely and the system is left in place to collect images and make single-image detections of objects for some period of time.
2. After a set number of frames have been collected, all of the classifications and bounding box coordinates produced by the machine-learning model are read from the database.
3. Each camera's frame area is divided into small rectangular cells, which will be the regions used for geometric matching. The size of the cells is a configurable parameter: if the cells are set to be smaller and more numerous, a greater degree of spatial precision will be possible, but more frames of data will be required to properly induce the relationships between the cells.
4. For every pair of cameras i and j, a four-dimensional array $A_{(i,j)}$ is initialized to all zeros, with the four axes corresponding to the horizontal and vertical coordinates of the cells within the frame area of each camera, and the values at each coordinate intended to describe the degree of association between each pair of frame-area cells in terms of object centroids appearing in those cells. That is to say, once the below steps have been

carried out, $A_{(i,j)}[x_i, y_i, x_j, y_j]$ should contain an estimate of the relative extent to which a detected object appearing in cell (x_i, y_i) in the frame area of camera i is expected to correspond to the same physical object as a detection found in each cell (x_J, y_j) in the frame area of camera j.

5. For each pair of cameras i and j, every pair of images from those cameras taken at the same moment in time is iterated over once for each class of objects, as follows:
   a. If either of the images contain more than some threshold number of objects of a particular object class, processing of the current pair of images using that class is skipped. The reason for this thresholding step is that (assuming that the model's classifications are generally correct and that the two cameras' views overlap in the relevant areas) in the case where there is only one object of a given class present in two different cameras at the same moment in time, those detections can be matched with each other unambiguously to induce a geometric relationship, but as the number of detections of that class increases, the number of ways in which they could potentially be matched into corresponding pairs increases combinatorially, so the information that can be extracted from those detections is significantly reduced. However, even if a pair of images is skipped for some class of objects because there are too many objects of that class, that same pair of images may be used with another class which has fewer objects present in the cameras' views at that moment.
   b. If the images are not skipped for the current object class, the centroid coordinates for each object within each camera's frame area are calculated, and the corresponding cells (x_i, y_i) and (x_J, y_j) within each frame area are identified.
   c. For each cell identified in this way, the corresponding value in the four-dimensional array $A_{(i,j)}$ for the current camera pair (i.e. $A_{(i,j)}[x_i, y_i, x_j, y_j]$) is increased by 1 divided by the square root of the number of possible ways the object could be matched (i.e. if there is only one object across the two cameras, the single known-correct value is increased by 1, but with more objects, the value is split amongst the corresponding coordinates for each possible candidate matched pair). For a given camera pair, the same array $A_{(i,j)}$ is used for each object class, so the final values for each coordinate in $A_{(i,j)}$ may be composed of contributions from several object classes.
6. Once this process is carried out for each pair of cameras and each object class, the cross-camera mappings are recorded to data storage as the arrays $A_{(i,j)}$ for each possible camera pair i and j (with $i \neq j$)
7. Steps 2-6 are repeated periodically to update the mappings with newly-available geometric information from newly-captured frames.

Figure 5 provides a visual representation of this pre-calculation process. The system was generally configured to schedule this occasional processing during known break times so as not to interfere with performance during active operation.

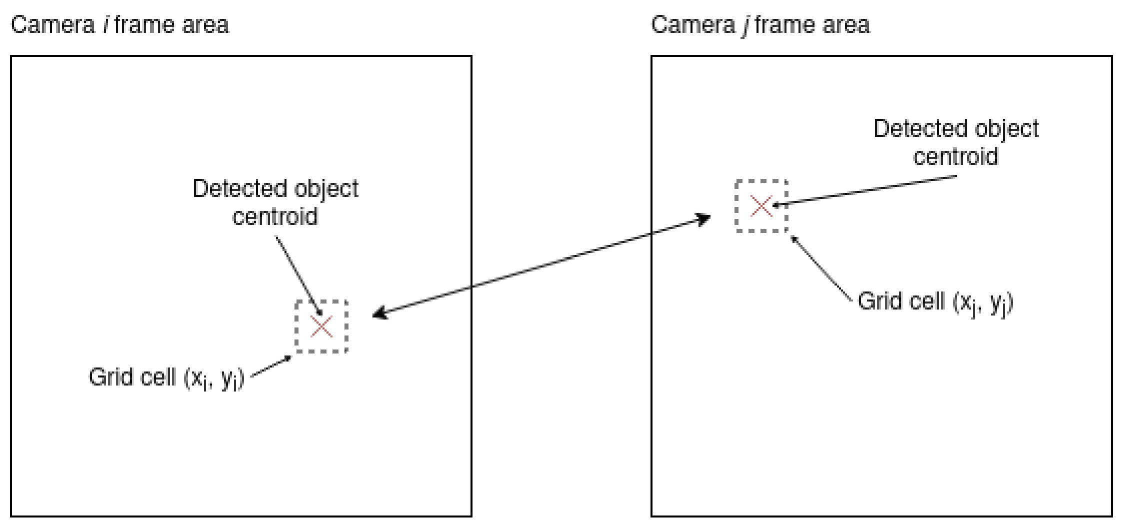


Figure 5: Illustration of a simple case for cross-camera mapping pre-calculation. Suppose that a single object is seen on both cameras i and j, is assigned the same class by the ML model, and is the only object of that class appearing in those frames. If the object appears in grid cell (x_i, y_i) on camera i, and cell (x_J, y_j) on camera j, this is taken as evidence of a relationship between those cells, and the corresponding value $A_{(i,j)}[x_i, y_i, x_j, y_j]$ is increased by 1 in the cross-camera map $A_{(i,j)}$ corresponding to this camera pair.

Once the pre-calculated frame mapping data has been recorded, the system is then able to use the data to generate matches between objects during real-time operation. For each frame (at a particular moment in time, across all cameras), the process is as follows:

1. For each detected object in each camera's image for the current frame, the confidence value, classification and the centroid coordinates of the bounding box for that detection are extracted from the output of the core machine learning model.
2. The product of the sets of detections for each camera is taken, to produce all possible candidate combinations of one detection from each camera which could potentially be matched with each other as corresponding to the same physical object. Additionally, all possible candidate combinations of detections from a subset of cameras are added, to account for situations in which an object is visible in some cameras but not in others.
3. For each candidate combination of detections from multiple cameras, each pair of objects across two cameras is taken, and the 'match score' for that pair is obtained by looking up the corresponding value in the pre-calculated array for that camera pair, from above. That is, for a pair of detected objects with centroid coordinates appearing in cells (x_i, y_i) and (x_J, y_j) of the frame areas of cameras i and j, the match score for that pair of detections is the value $A_{(i,j)}[x_i, y_i, x_j, y_j]$. If any pair in the candidate combination has a match score lower than a certain configurable threshold, processing of that combination is skipped, to reduce computation time.
4. For each candidate combination, a combined match score is calculated as follows:
   a. The average of the match scores for each pair in the combination is taken
   b. The average model confidence value of the detections in the combination is taken
   c. The average match score and average confidence are linearly combined with configurable scaling

coefficients to produce the unpenalized combined match score

d. A penalty is applied for each camera that does not contribute a detection to the combination by
e. multiplying the unpenalized match score by a configured coefficient between 0 and 1 for each missing camera.
f. The resulting value is the combined match score for the entire candidate combination.

5. All candidate combinations for which all component detections have the same object classification are ranked in descending order of combined match score, the highest-ranked candidate is taken as a confirmed match, and the system considers the component detections of that candidate to correspond to the same physical object. Those detections are removed from consideration and the process is repeated, with the next-highest ranked candidate combination that does not include any of the detections that have already been matched by higher-ranking candidate being taken as the next confirmed match. This continues until the process reaches a candidate which has a combined match score less than some specified minimum threshold.
6. Step 5 is repeated, but this time additionally considering combinations for which not all component detections have the same classification, and excluding detections which are already part of a confirmed match from the first pass in Step 5. These matches may indicate classification errors by the model and are used for model error detection and model development.
7. All of the confirmed matches from Steps 5 and 6 are recorded as the cross-camera object matches for the current frame, and, by construction, each detection will appear in at most one confirmed match set.

Figure 6 shows an example of the output of this process for a particular frame across several cameras. The colored diamonds drawn around the centroids of each bounding box indicate the cross-camera match set that each object belongs to.

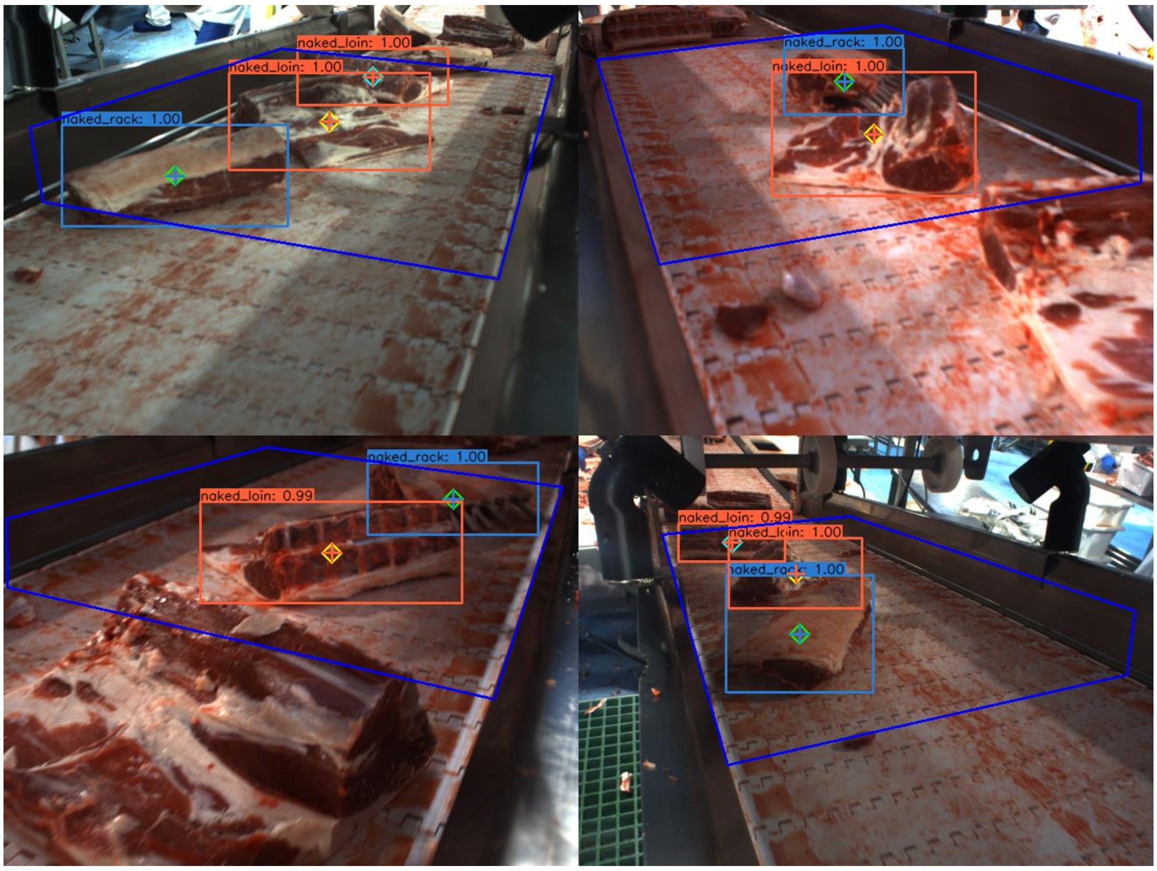


Figure 6: Illustration of cross-frame object matching. The colored diamonds surrounding the center of each object indicate which objects in other cameras it has been matched with: objects with green diamonds are matched with other objects with green diamonds, and similarly for yellow and cyan. Note that objects whose centers are outside the blue region in each frame are not processed by the ML model, and that two of the cameras are pointing in the opposite direction as the other two.

3.6.2 Object tracker

The machine learning algorithm developed for this project can predict a list of bounding boxes from an input image in real-time, but it can't relate one frame to another through time. Although the object tracker was not a project objective, it is clear that the addition of a tracker that could combine predictions from all frames and find associations between objects through time brings a lot of value.

An object tracker was developed to find associations between naked loins' bounding boxes through time. The goal of this tracker is to identify the path that naked loins follow through the conveyor belt. The tracker was based on the Simple Online Realtime Tracker algorithm adapting the Hungarian Algorithm, also known as Kuhn Munkres algorithm, to find associations between objects from sequential frames, and the Kalman Filter to predict future positions based on current ones (Bewley *et al.*, 2016). By combining the input of the YOLOv4 model with these two algorithms and adapting some parameters to fit this specific task, we're able to accurately identify the track that naked loins follow though the conveyor belt in real-time. This then enables us to count the total number of objects that go through the conveyor belt.

3.6.3 Combined overall classification

While the machine-learning model itself is able to produce highly-accurate classifications of objects using only a single image from a single camera, the accuracy of these classifications can be further improved using the outputs of the object tracking and cross-camera object matching algorithms by combining the model's predicted classifications of the same object across multiple images across time and from multiple angles across different cameras. In this way, a classification error which occurs for a single image or even several images can be corrected by reference to other images of the same object, because the model's single-image classification is correct in the vast majority of cases.

For a given object detection in a given camera, the overall classification of the object using both intertemporal information from the object tracking system and and cross-camera matching information from the object matching system is obtained as follows:

1. First, the object tracking system is used to find the corresponding track of previous detections of the same object in the same camera.
2. For each of those previous detections (and the actual detection of interest), the frame matching system is used to find the corresponding detections in other cameras.
3. For each of the corresponding detections in other cameras, the tracking system is used to identify the temporal track to which it belongs.
4. For each other-camera track, and for each corresponding matched detection found in Step 2, a 'track weight' is assigned to the track based on the number of matched detections in that track, with tracks closer in time to the present contributing a higher value to the total track weight than tracks further away in time (this is because the tracking system is more reliable for detections made more recently to the current frame). The track weight represents the degree to which that track (in another camera) is expected to correspond with the original detection's track (in the original camera). The track weight of the original detection's track is assigned a configurable constant value.
5. For each possible class that the object could belong to, a 'class weight' is initialized to zero.
6. For each track that has a non-zero track weight, each detection in that track is iterated over. The model confidence of the detection, the track weight for the track to which the detection belongs, and a time penalty reflecting how much time has elapsed since the detection, are used to form a value which is then added to the accumulated class weight for that detection's predicted object class. That is, each detection which is related to the original detection of interest by either intertemporal tracking or cross-camera matching contributes a vote towards the corresponding class weight value, and the weight of that vote is determined by the confidence of the detection, and the degree of relatedness to the original detection.
7. The object class with the highest class weight after all relevant detections have contributed their values is selected as the combined overall classification of the object corresponding to the original detection, which may or may not be the same as the predicted classification of the original detection.

Figure 7 shows a diagrammatic illustration of the overall combined classification process for a given detection.

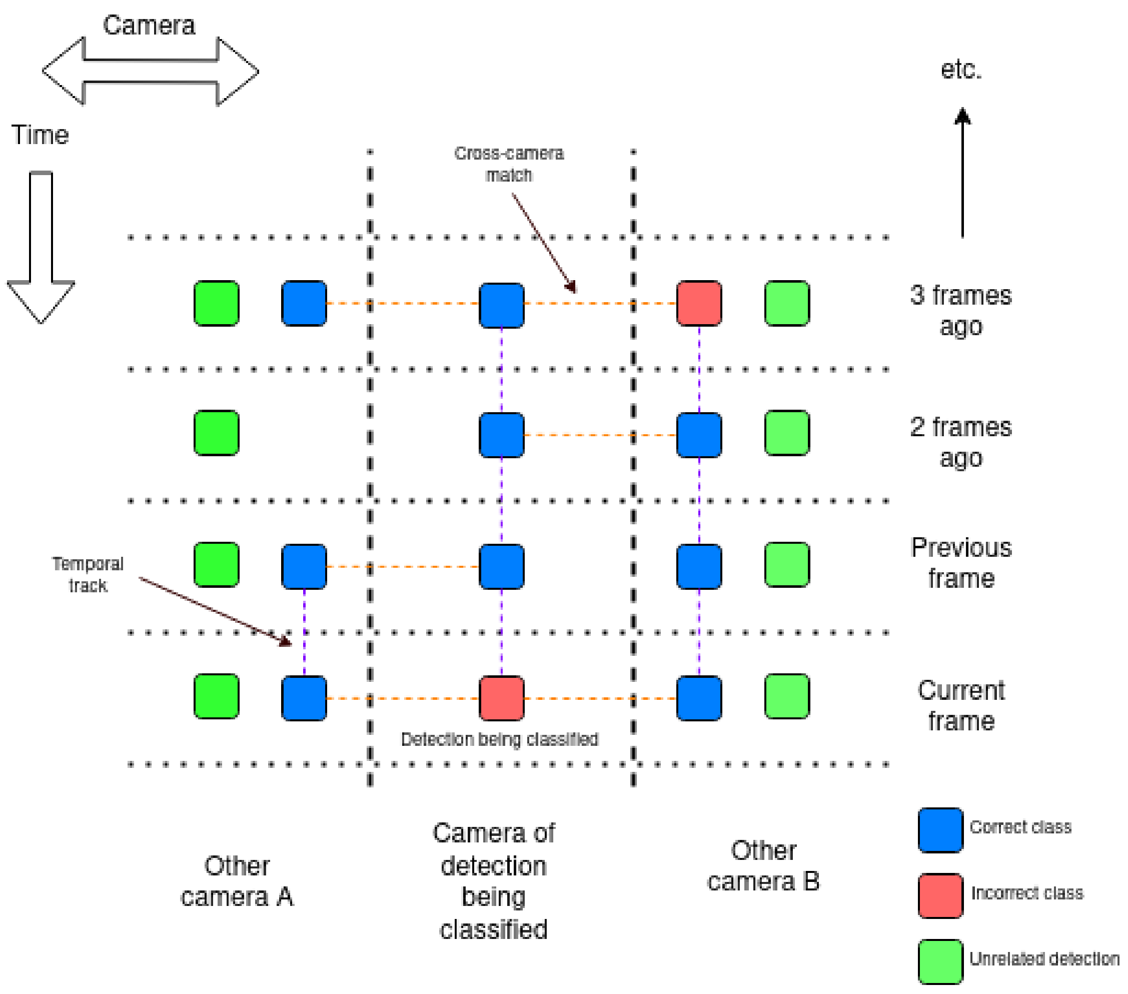


Figure 7: Illustrative example of overall classification process using both cross-camera matching and temporal tracking. Although the detection at the center bottom is incorrectly classified (red square) if considered on its own, the overall counting system is able to use the cross-frame matching information to find detections at other times and from other cameras which are likely to be the same physical object which can be used to correct the classification.

All of the red and blue squares are connected to the detection being classified by a path of orange dotted lines (cross-camera matches) and purple dotted lines (temporal tracks), and so each will contribute a weighted vote to the overall classification of the object. In this case, the incorrect detection (red) would be overridden, as the detections in the connected network are overwhelmingly correctly classified (blue), despite a missing detection and resulting lost track (2 frames ago on the left), and a missing cross-camera match (previous frame, to the right). Unrelated detections in the center camera are omitted for clarity.

3.6.4 Counting mechanism

To demonstrate and evaluate the capabilities of the camera system, in addition to producing classifications of objects appearing on the belt, the software was also equipped with the option to attempt to count the number of objects of each class passing by the camera station as the system operates. This adds another layer of complexity to the processing of the model outputs, as it not only requires that objects be correctly identified, but also requires that each object is counted exactly once. The counting system comprises an algorithm that attempts to count objects seen by each camera individually, and a secondary algorithm that processes those single-camera counts to produce an overall count which attempts to correct for situations in which an object is not detected in a particular camera either due to occlusion by another object or due to the angle at which the object is seen.

Prediction zone calibration

To carry out object counting using the implemented method, it is necessary to first configure a ‘prediction zone’ within each camera’s frame. This is a defined area of the frame which all objects will pass through; when an object first enters the zone, the system attempts to count it. It is important for the overall multi-camera counting algorithm that the prediction zones within each camera frame are fairly closely aligned, in that objects entering the prediction zone in a particular camera should also enter the prediction zones in other cameras within a short period of time. To achieve this, the software includes two methods of prediction zone calibration: automatic and manual.

Automatic calibration of the prediction zones involves placing a calibration object of a specific color on the belt in the target location. The system can then detect the region occupied by the calibration object in each camera’s frame area and store that region as the prediction zone for that camera.

Manual calibration is achieved by interactively drawing the edge of the region with the computer mouse on a sample image taken from each camera. This requires some human judgement to ensure that the drawn regions are well-aligned, but can be used in cases where it is not feasible to use the automatic calibration method, for example while the belt is running, and can also be used remotely when no user is physically present at the camera station.

Figure 8 shows an example of calibrated prediction zones across multiple cameras.

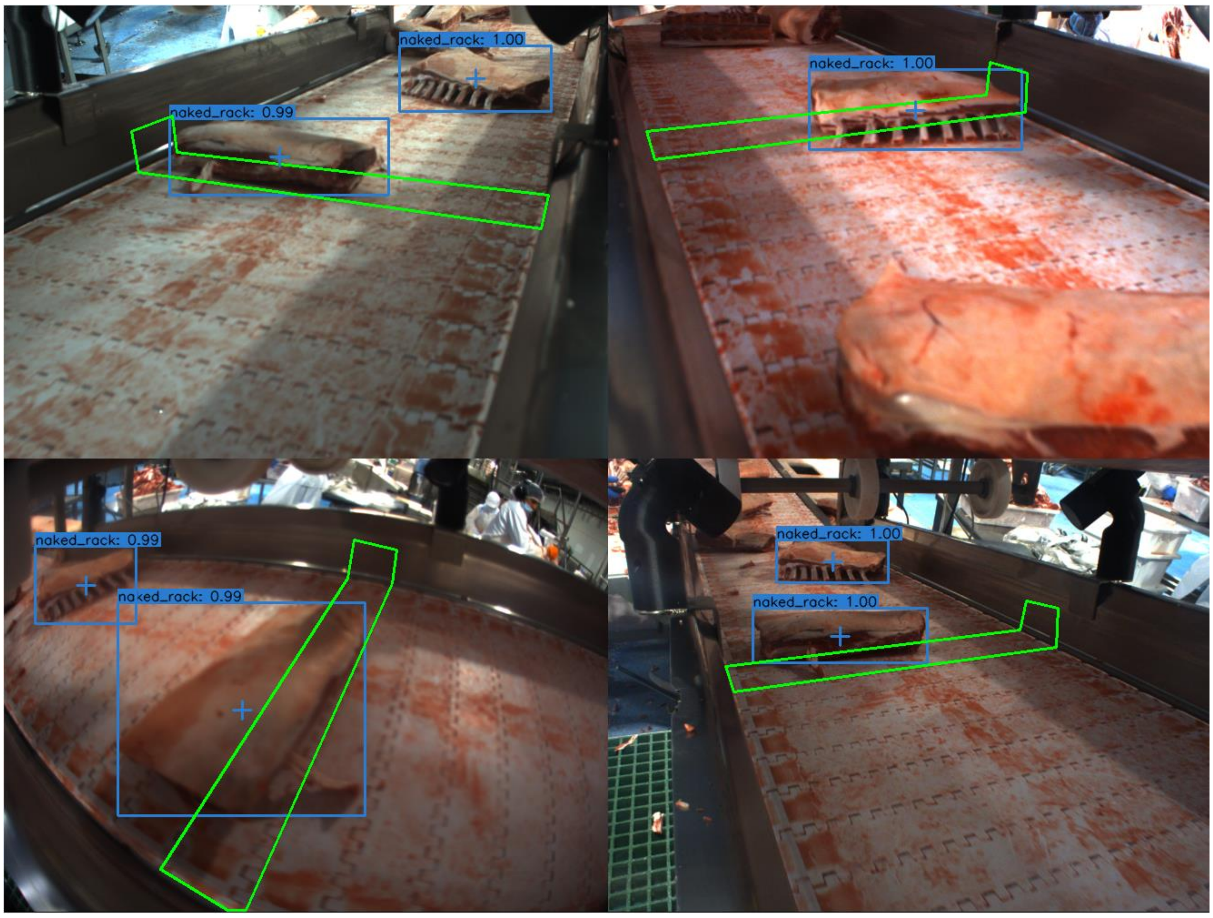


Figure 8: Example of calibrated prediction zones across multiple cameras. The green regions are the prediction zones, and note that the naked rack near the center of the frame is either inside, just about to enter, or has just exited the region, indicating that the prediction zones are well-aligned between the cameras.

Single-camera counting

To perform counting of objects by classification for a single camera, each frame is processed as follows:

1. All detections for the current frame are checked to determine whether the centroid of their bounding box is inside the camera's configured prediction zone.
2. For each object which has its centroid inside the prediction zone, the temporal object tracking system is used to find centroids for that object in previous frames. If the object has at least a configurable minimum number of previous detections, and their centroids are all outside the prediction zone, then the system considers that object to have 'just entered' the prediction zone for the first time and attempts to count it.
3. When an object is found to have entered the prediction zone for the first time, the combined class prediction algorithm described above is used to determine a best overall classification for the object, and the corresponding count (for the current camera) of objects of that class is incremented.

Multi-camera counting

Because not all objects are necessarily seen or detected by every camera, particularly as they may be hidden from view by other objects, a secondary system is used to combine the results of the single-camera counting procedure described above to produce the system's overall count of objects from each class. This multi-camera counting procedure is carried out as follows, for each frame:

1. The single-camera counting procedure described above is carried out for each camera to update each single-camera count value for each class of object.
2. For each class, the maximum count across all cameras is taken as the current value of the overall count for that class
3. For each class, if the overall count for that class has not changed in a certain threshold number of frames (see explanation below), a reconciliation step is triggered where each camera's single-camera count for that class is set to the overall count value for that class.

The reason that the maximum across all cameras is used is because overcounts in a single camera are very rare, while undercounts are common, as objects can often be out-of-view of a given camera due to occlusion by another object.

The threshold number of frames used to trigger the reconciliation step for each object class is intended to account for the fact that objects may enter the prediction zone in different cameras at different times. If the prediction zones are carefully aligned with each other between cameras, this time difference can be minimized, but it can't be eliminated entirely, as objects can differ significantly in size, shape and orientation. As such, this threshold should be set to ensure that all objects which have already been counted on any camera will not be counted any later than the threshold number of frames on any other camera. In this case, once the number of frames since the last overall count increment for that class has passed, the system can assume that any future count increment on any camera will correspond to a new object, and so the single-camera counts are reconciled in preparation, so that the next count increment on any camera for that object class will immediately cause the overall count for that class to be updated.

Even with this reconciliation step, an undercount is still possible, but only in situations where many objects of the same class pass by the camera station in a consistently rapid enough succession as to not permit reconciliation at any point, and no single camera is able to detect all of the objects. In practice, we found that when the prediction zones are properly calibrated and the reconciliation frame threshold is correctly set, this is an extremely rare occurrence.

4.0 Results and limitations

4.1 Validation method

The original plan for validation was an in-plant evaluation of the system, that consisted of counting primals as they passed a certain point on the belt for a period of two hours per day over a one week period. This validation method was proposed to ensure that the model is as accurate as reported. However, as the system was developed, it became apparent that it would be very difficult for a human to accurately validate the model in real-time. The speed at which the belt is running, combined with the high number of primals present on the belt at any given time, made the task of counting and identifying naked loins in real-time impossible to undertake, even for an expert in primal identification.

Instead, the final validation method consisted of recording multiple runs over a week, allowing for model analysis to occur over a 6 hour collection period per day. An individual expert in primal identification was then able to review the 190 real-time simulations collected through slow motion replays in order to determine if the number of naked loins identified by the system was correct. This allowed for the team to more effectively allocate time and resources to ensure validation was conducted to a high level of accuracy.

4.2 Validation results and limitations

The human expert in primal identification evaluated 190 real-time simulation runs that were collected from the system. Both the human and the system counted 1018 naked loins that passed a certain point on the boning belt, with the expert not detecting any errors in the system's naked loin count. This indicates that, as far as these specific simulation results are concerned, our proposed system is 100% accurate at identifying the number of naked loins that passed a certain point on the boning belt.

3.8.1 Limitations

Stacked primals occur on a semi-regular occasion and we have observed a rare phenomenon where pieces are stacked in such a way which completely obscures the view of a particular piece. This is not a problem specific to the cameras and algorithms used by this project as in these situations a human also would not be able to identify the piece (since it is completely covered). We have spent a great deal of time learning about this rare problem and we think there are some workflow optimisations that can solve this. We expect to do some more work technically to solve this for future projects. For clarity when primals are stacked we have no issue in identifying them outside of this rare occurrence. Figure 9 shows multiple back straps piled which were correctly identified, which shows the robustness of the system.

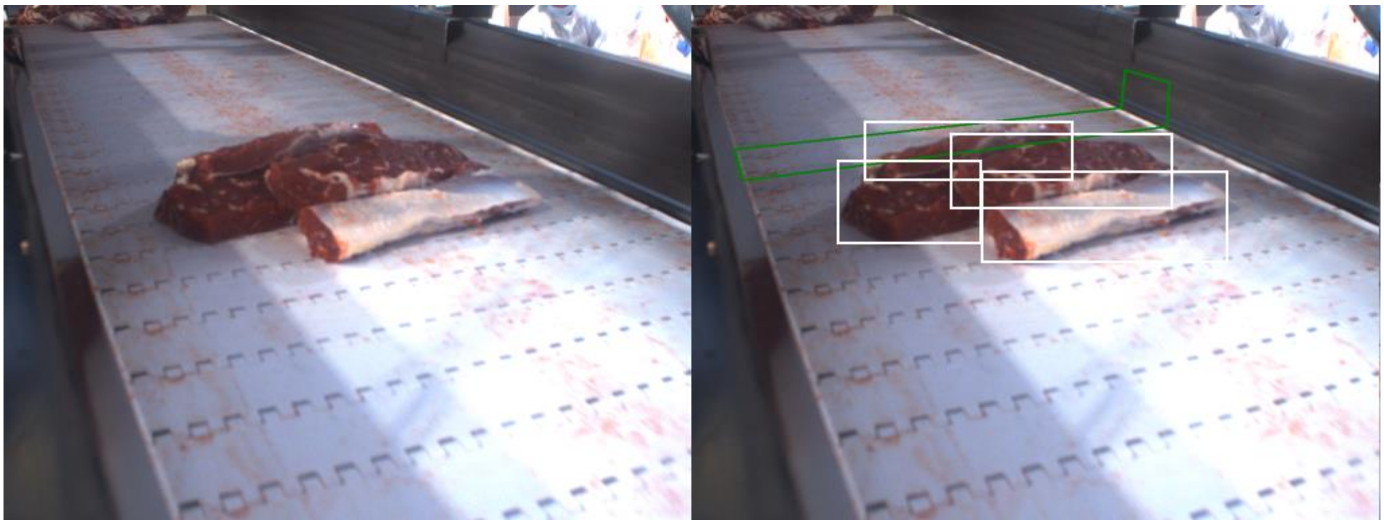


Figure 9: A pile of backstraps, which are correctly identified by the system

5.0 Project Outcomes

5.1 Labelled dataset consisting of 30,000 images of naked lamb loins

Over a period of six months, 30,000 images of naked lamb loins were captured in an abattoir setting. A wide variation of loin images were captured by handheld camera and a mounted system of six cameras. These images were reviewed, annotated with bounding boxes, generating a dataset of 30,000 labelled naked loins. This dataset was formated to act as an input for a machine learning model. The complete cvs for these 30,000 labelled images is attached to this report.

5.2 Machine learning model that is able to identify a naked lamb loin on the GMP boning belt with 99.9% accuracy at chain speed

The proposed Machine Learning Model was able to identify the 100% of 1018 naked loins that passed through the conveyer belt of a processing plant in 190 real-time simulations.

5.3 Machine learning model that can alert a nearby user if the model is unsure, allowing for manual override

While the system is able to produce highly accurate results, it also incorporates functionality to allow the user to directly override the system's outputs. The touchscreen GUI, when run in the mode designed to display the system's detections and overall counts for each class, is able to alert the user to a detection about which the model indicates lower confidence, by highlighting its bounding box in red, and the user is able to override the counts manually using touchscreen controls if a primal is incorrectly classified by the system. An illustrative screen capture of the GUI running in this mode is shown in *Figure 10*.

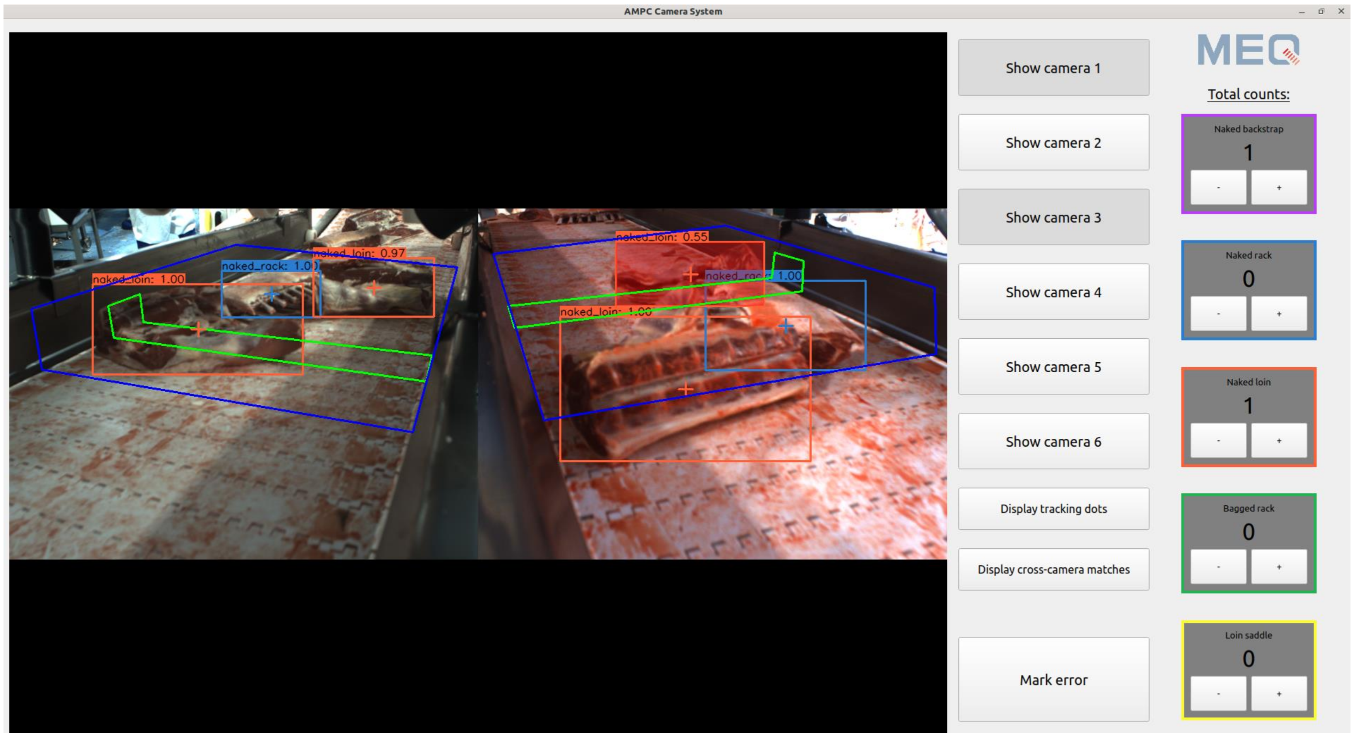


Figure 10: Screen capture of GUI showing bounding boxes and current-session counts for each object class. Note the red bounding box indicating a low-confidence detection (in this case, nevertheless a correct one), and the '+' and '-' buttons below each count, which allow the user to manually override the system's automatic counts.

6.0 Discussion

The validation results described in Section 4 demonstrate that the computer vision system is successful at identifying naked loins with greater than 99.9% accuracy at chain-speed. There is evidence that the system is more accurate than a human expert trying to identify primals in real-time, as it is a very complex task for a single person to detect multiple types of primals simultaneously at chain-speed, and furthermore the system is able to view the belt from multiple angles simultaneously and cross-reference between those angles in a way that would be impossible for a human.

Moreover, the success of this development corroborates the value that computer vision and machine learning techniques are able to provide to the operations of an abattoir boning room. Processors can benefit from these technologies as they bring objective measurement, alleviate risks, and help reduce labour costs.

7.0 Conclusions / Recommendations

In conclusion, machine learning and computer vision technologies can be employed in an abattoir environment to detect naked lamb loin primals with a high degree of accuracy. These methods show strong evidence in terms of applicability to boning room automation, analysis, and quality assurance tasks generally, with the potential to increase the reliability of abattoir processes and reduce labor costs.

We recommend that further research and development be undertaken to bring these technologies to further applications which have direct and specific benefit to processors. In our opinion, these technologies could have immediate value in the following areas:

- Automatic Sorting: a machine learning/computer vision system could be developed to control mechanical systems such as belt diverters to enable automated sorting of meat cuts in certain situations.
- Supply Chain Traceability: with some expansion of scope, potentially including more cameras and processing nodes, the technologies used in this project could be applied to the problem of boning room traceability to track an individual carcase as it is transformed into primals and individual cuts.
- Quality Assurance: a machine learning/computer vision system could be developed to identify and count objects once boxed, and alert users to the presence of foreign objects

8.0 Bibliography

Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B., 2016, September. Simple online and realtime tracking. In 2016 IEEE international conference on image processing (ICIP) (pp. 3464-3468). IEEE.

Bochkovskiy, A., Wang, C.Y. and Liao, H.Y.M., 2020. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934.

Hino, H., 2020. Active learning: Problem settings and recent developments. arXiv preprint arXiv:2012.04225.

Kim, J.A., Sung, J.Y. and Park, S.H., 2020, November. Comparison of Faster-RCNN, YOLO, and SSD for real-time vehicle type recognition. In 2020 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia) (pp. 1-4). IEEE.

Redmon, J. 2016. Darknet: Open Source Neural Networks in C. Available at: http://pjreddie.com/darknet/.

Ren, P., Xiao, Y., Chang, X., Huang, P.Y., Li, Z., Gupta, B.B., Chen, X. and Wang, X., 2021. A survey of deep active learning. ACM Computing Surveys (CSUR), 54(9), pp.1-40.

9.0 Appendices

Appendix 1 - Initial Sample of Images

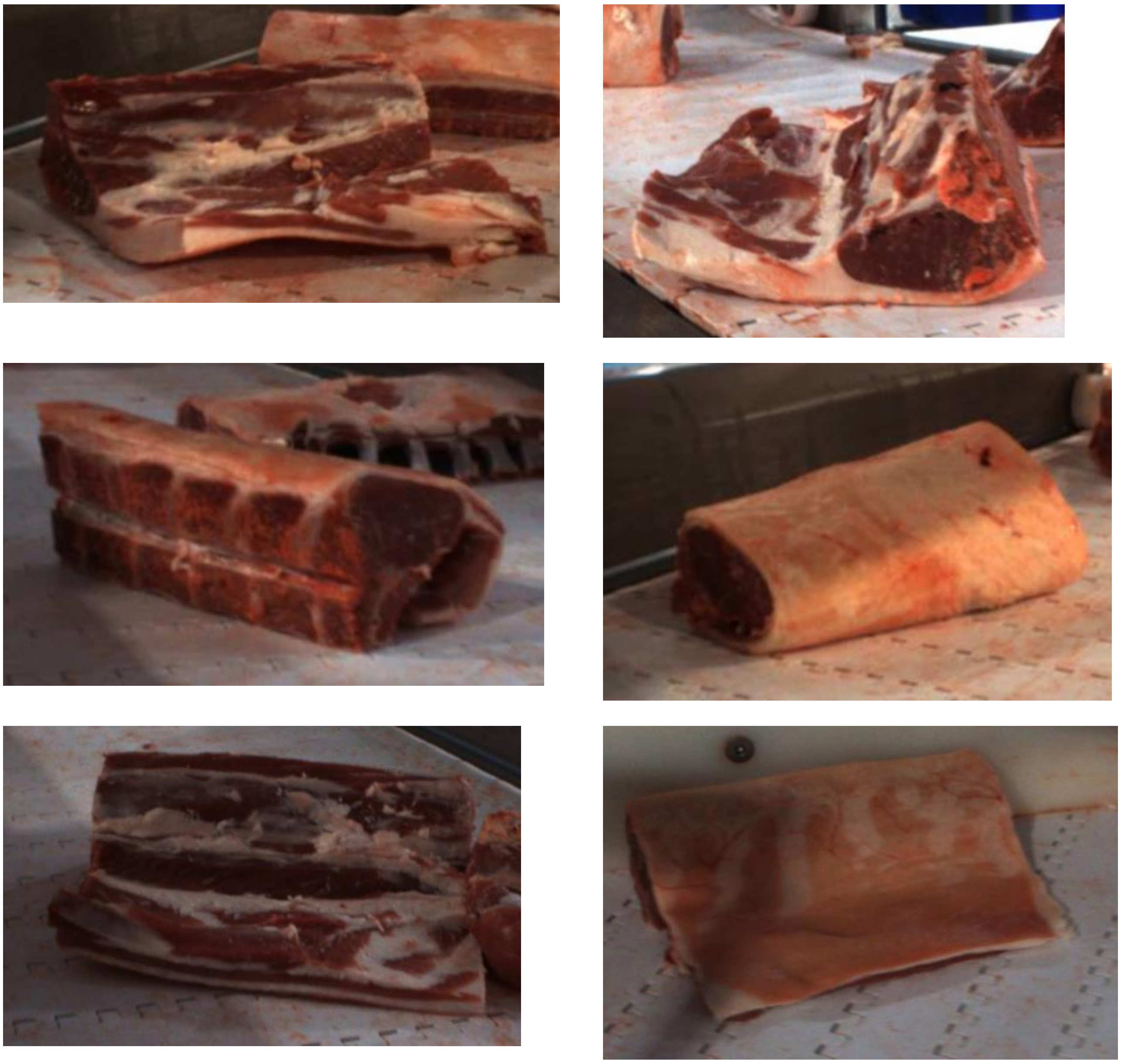


Figure 11: Six examples of naked lamb loin images collected in the plant throughout different days on one conveyor

Appendix 2 - Sample Images with Bounding Box Annotations

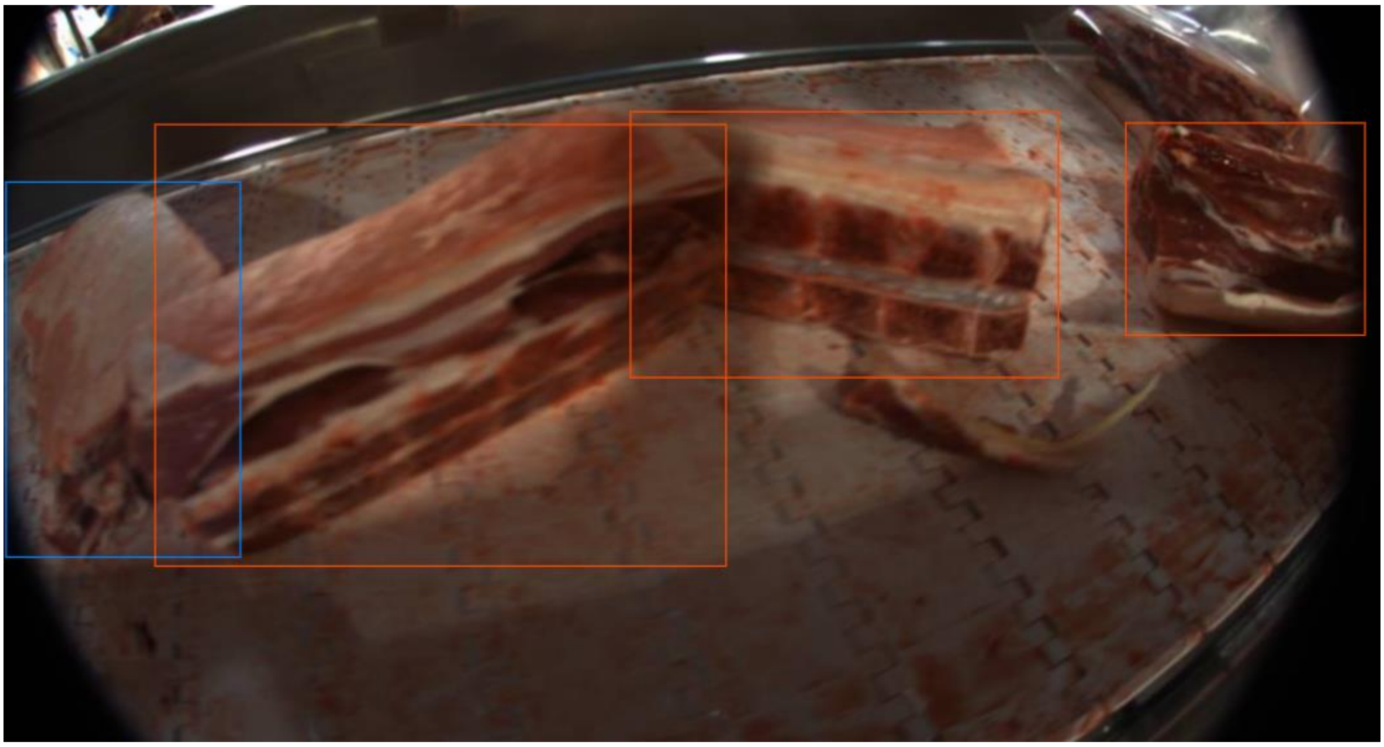


Figure 12: Image gmp_20211123_132448_730_eFNN_cam2_main_belt_class_change.png has four objects annotated - three naked_loins (red rectangles) and one naked_rack (blue rectangle).

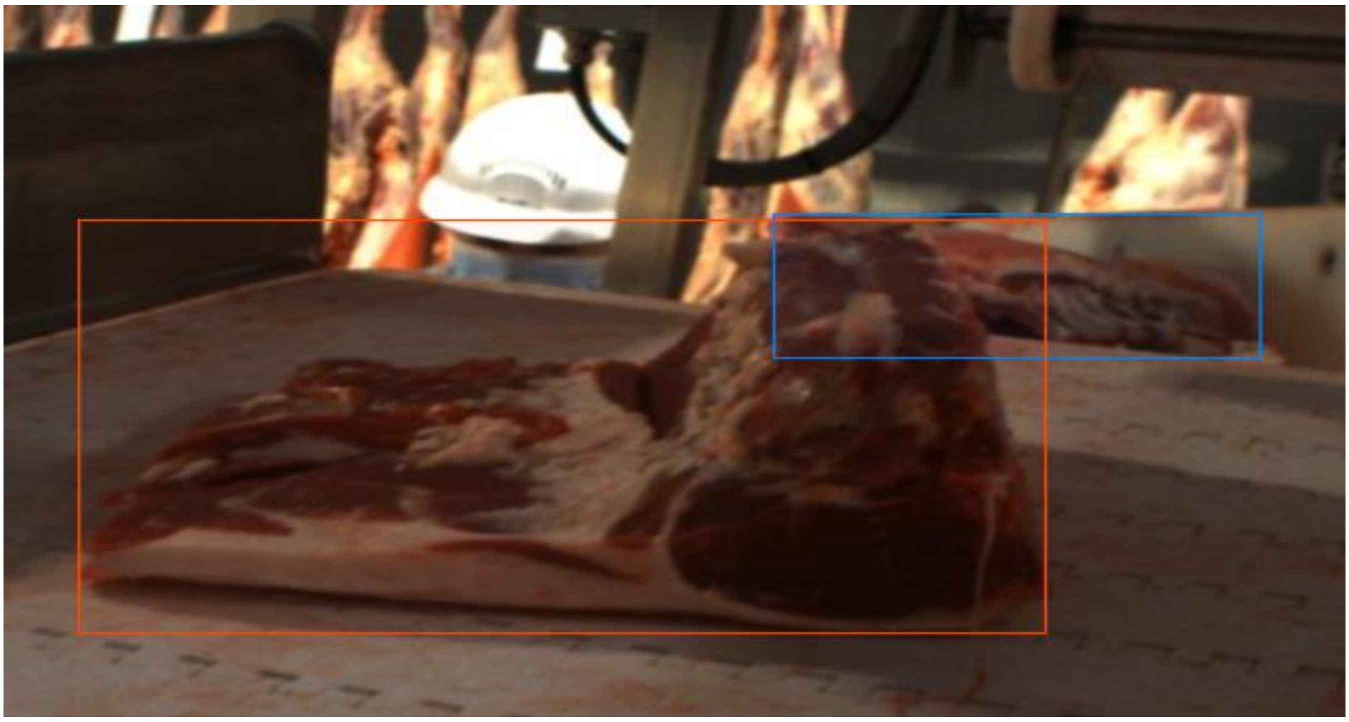
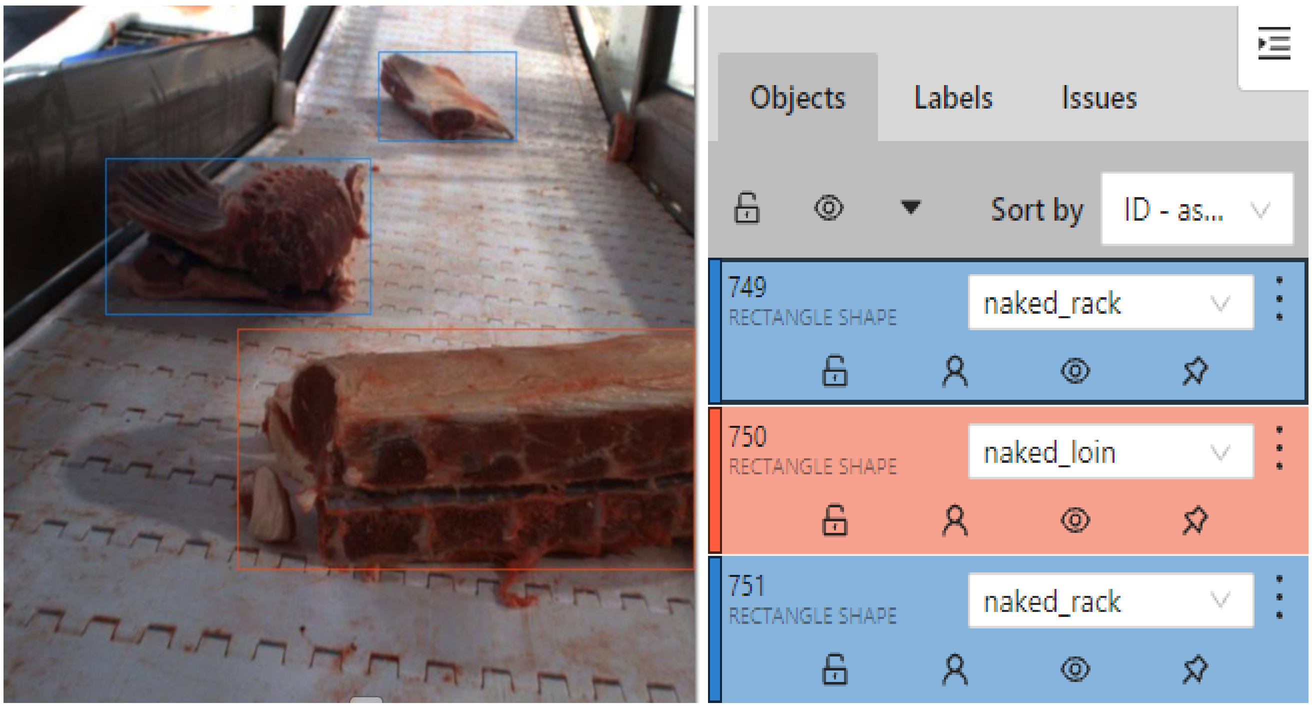


Figure 13: Image gmp_20210810_114108_9Z8j.png has two objects annotated - one naked_rack (blue rectangle) and one naked_loin (red rectangle).

Appendix 3 - Sample Images with Bounding Box Annotations



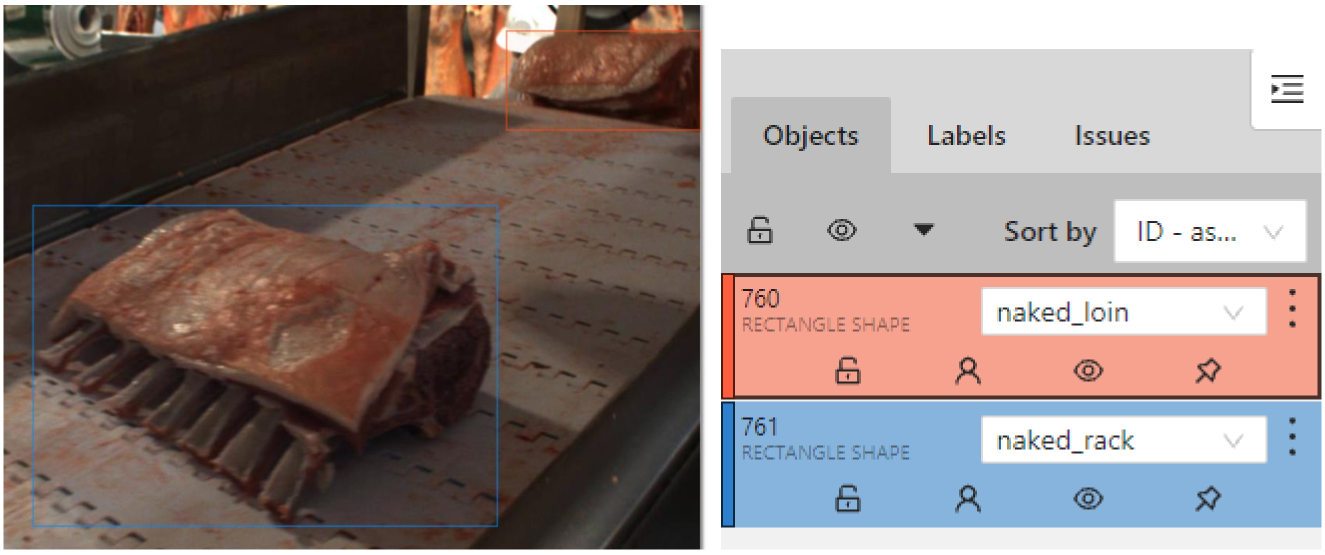
Sort by: ID - as...

749 RECTANGLE SHAPE naked_rack

750 RECTANGLE SHAPE naked_loin

751 RECTANGLE SHAPE naked_rack

Figure 14: Image gmp_20210810_103518_Zr47.png has three objects annotated - two naked_racks (blue rectangles) and one naked_loin (red rectangle).



Sort by: ID - as...

760 RECTANGLE SHAPE naked_loin

761 RECTANGLE SHAPE naked_rack

Figure 15: Image gmp_20210810_113658_o6Vy.png has two objects annotated - one naked_rack (blue rectangle) and one naked_loin (red rectangle).

Appendix 4 - Example of data augmentation method: randomly cropped copies of the same image.

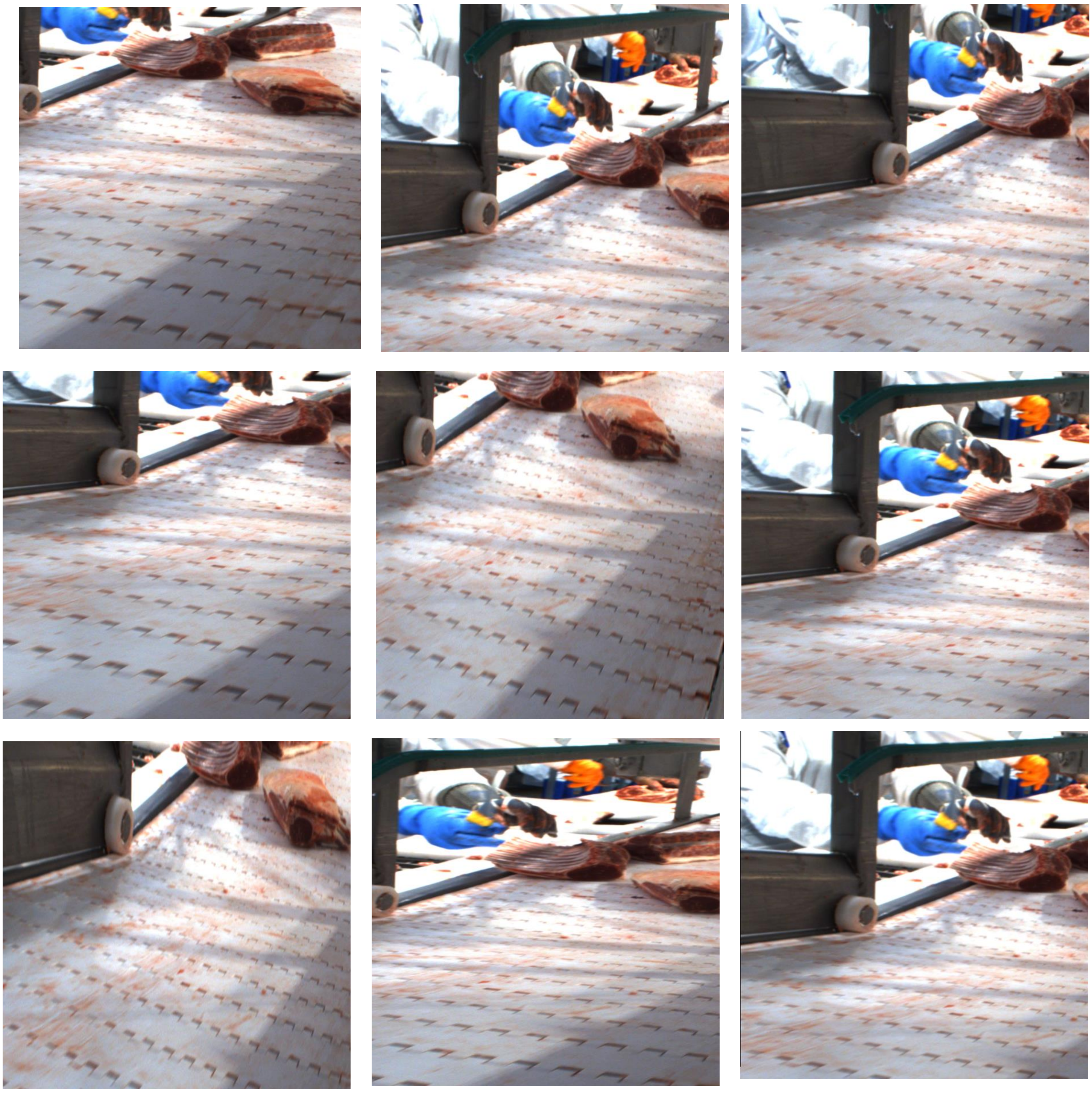


Figure 16: Nine different augmented copies of image gmp_20210810_102513_Dxp7.png

Appendix 5 - Sample of csv with labels

Filename	Width	Height	Object	xmin	ymin	xmax	ymax
gmp_20210810_102513_Dxp7.png	768	768	naked_loin	657.96	94.5	758.3	143.8
gmp_20210810_102513_Dxp7.png	768	768	naked_loin	495.3	177.2	660.4	247.4
gmp_20210810_102514_cfqj.png	768	768	naked_loin	445.91	210.04	629.17	288.44
gmp_20210810_102514_cfqj.png	768	768	naked_loin	644.2	109.03	748.27	165.6
gmp_20210810_102514_FSjB.png	768	768	naked_loin	474.42	196.3	646.3	269.5
gmp_20210810_102514_FSjB.png	768	768	naked_loin	653.5	102.2	753	158.27
gmp_20210810_102515_3EcE.png	768	768	naked_loin	422.86	227.92	612.83	310.24
gmp_20210810_102515_3EcE.png	768	768	naked_loin	635.44	117.56	742.18	176.36
gmp_20210810_102515_hQSA.png	768	768	naked_loin	624.58	122.99	738.56	185.41
gmp_20210810_102515_hQSA.png	768	768	naked_loin	417.43	240.58	601.07	329.24
gmp_20210810_102516_ALVn.png	768	768	naked_loin	438.24	249.63	590.21	358.18
gmp_20210810_102516_ALVn.png	768	768	naked_loin	607.4	124.8	724.99	191.74
gmp_20210810_102516_f7id.png	768	768	naked_loin	416.35	266.46	569.31	384.05
gmp_20210810_102516_f7id.png	768	768	naked_loin	595.55	132.58	718.57	202.23
gmp_20210810_102712_BH9f.png	768	768	naked_loin	558.55	111.23	762.99	322
gmp_20210810_102713_DevT.png	768	768	naked_loin	551.31	141.08	768	383.51

gmp_20210810_102713_Pjvr.png	768	768	naked_loin	524.17	135.65	763.89	356.37
gmp_20210810_102714_ejuH.png	768	768	naked_loin	574.83	159.17	768	422.41
gmp_20210810_102714_ekh2.png	768	768	naked_loin	537.74	177.26	768	461.31
gmp_20210810_102715_cNNW.png	768	768	naked_loin	455.43	223.4	768	553.57