University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

TailSafe: A Pig Head-to-Rear Contact Detection System using Convolutional Neural Networks

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science
Submitted by:

Romwell Joackin O. Santos

June 2023

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "TailSafe: A Pig Head-to-Rear Contact Detection System using Convolutional Neural Networks" prepared and submitted by Romwell Joackin O. Santos in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Vincent Peter C. Magboo, M.D., M.Sc.
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Avegail D. Carpio, M.Sc.		
2. Richard Bryann L. Chua, Ph.D. (*cand.*)		
3. Perlita E. Gasmen, M.Sc. (*cand.*)		
4. Ma. Sheila A. Magboo, Ph.D. (*cand.*)		
5. Marbert John C. Marasigan, M.Sc. (*cand.*)		
6. Geoffrey A. Solano, Ph.D.		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Vio Jianu C. Mojica, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

Marie Josephine M. De Luna, Ph.D.
Chair
Department of Physical Sciences and Mathematics

Maria Constancia O. Carrillo, Ph.D.
Dean
College of Arts and Sciences

Abstract

Pig tail biting poses significant challenges in pig farm monitoring, serving as an indicator of underlying pen issues. Existing monitoring methods are limited in their scalability and invasiveness. This study introduces TailSafe, a web-based decision support tool utilizing YOLOv5 and convolutional neural networks. TailSafe enables farmers to diagnose pig pen issues through potential tail biting outbreaks. Users upload pig pen images for processing, and the system provides results for contact presence classification and counts. TailSafe comprises two components: a detection method to identify pig heads and rears, and an interaction method to compute head-to-rear IoUs for contact identification.

Keywords: tail biting, decision support tool, YOLOv5, convolutional neural network, pig pen images, contact presence, detection method, interaction method

Contents

List of Figures

List of Tables

I. Introduction

A. Background of the Study

Livestock monitoring is essential in maintaining animal health and welfare. It is crucial in assessing the performance and growth of animals. Moreover, it also provides timely feedback regarding animal health to diagnose early and prevent illness. Present day technology allows for automated monitoring using transponder attachments or invasive equipment attached to livestock for data gathering, usually for behavioral data [5]. However, with the increasing volume of livestock and decreasing number of farmers, it remains increasingly difficult to maintain consistent monitoring, even with existing options for automation.

In order to provide a more proficient way of monitoring livestock, several studies on camera-based monitoring are being developed. The computer vision approach becomes the main area of focus to advance livestock monitoring. There are existing studies regarding movement tracking, weight estimation, thermal analysis, posture detection, and more that are capable and beneficial for the industry [5]. This field is now known as Precision Livestock Farming (PLF), which automates the detection and monitoring of livestock to promote and maintain animal health and welfare. Additionally, this also includes real-time analysis of sounds and biological metrics using visual data. Consistency and efficiency in monitoring are important to farmers because poor health management and presence of damaging behavior (like tail biting) negatively affect the growth of pigs. Consequently, it also increases production cost where farmers become compelled to spend more to address these problems - for example needing to buy medication such as antibiotics. In terms of consumers, this is also important as monitoring preserves product quality and maintains willingness to purchase livestock [5].

Contemporary research shows that the use of pig-related data over a temporal space evaluates behavioral changes and determines the pig's state of health and welfare. In a study conducted by [6], pig behavior is analyzed and classified into

various categories, including lying, eating, drinking, walking, and standing. To track the behavior, multiple object detection techniques using the YOLO algorithm are being employed. The tracking process utilizes the Kuhn-Munkres and MOSSE algorithms, resulting in an average precision rate of 95%. Behavioral classifications are accomplished through the use of the ResNet18 model. Long-term observations in [6] determine patterns such as how ventilation affects drinking frequency, how moving and standing frequencies decrease over time, or how lying drastically increases as the pigs age.

One of the branches that emerges from the venture into PLF technologies is pig contact behavior, which can be used as an indicator for changes in a pig's physiological status or health status [1]. One study develops the means to automatically detect and quantify pig heads and pig rears using the YOLO algorithm with ResNet50 as its backbone [1]. It utilizes the bounding boxes created in the detection method to compute interaction scores and determine the presence of pig head-to-rear contact. For this method, the average accuracy for detecting the parts and identifying interaction scores is about 92.56% $\pm$ 3.74% confidence. Upon analysis of the results, it is evident that the occurrence of abnormal behavior, such as tail biting, affects pig contact behavior. Pigs tend to be contact avoidant when experiencing discomfort, which is unusual, as pigs are social animals and inquisitive in nature.

Pig contact behavior plays a vital role in monitoring the well-being and health of pigs. Automated techniques, such as pig part detection, have been developed to address this aspect. However, further advancements are needed as there is currently no consensus on the most effective model for this task, and it is not yet suitable for commercial implementation. Additionally, challenges remain, including occlusions between pigs, camera orientation considerations, and the detection of abnormal behaviors like flank chewing and ear biting, which require accurate detection of head-to-flank and head-to-head contact.

B. Statement of the Problem

Maintaining profitable pig farms while adhering to animal health and welfare standards becomes increasingly challenging due to current livestock demands. However, existing monitoring practices prove inadequate and inefficient in meeting these demands. The growing gap between the animal-to-staff ratio further complicates matters, rendering expansions and larger scale farming impractical [7]. Consequently, the development of an alternative monitoring tool that is automatic, low-cost, real-time, and non-contact becomes imperative. Timely monitoring is particularly crucial in identifying sub-optimal conditions in pigs [5]. Several studies focus on monitoring pigs using various parameters such as behavior, weight, and thermal indicators. One such approach is contact behavior monitoring, which helps identify instances of tail biting. However, research on this type of monitoring for pigs remains limited, resulting in a scarcity of trained models for contact detection. Therefore, this study aims to identify the optimal model and hyperparameters for contact detection. The trained model integrates into a system designed to serve as a decision support tool for screening pig pens, enabling the suggestive identification of health-related diseases or abnormal behaviors through tail biting.

C. Objectives of the Study

The objectives of this study entails evaluating the capability of the YOLOv5 (You Only Live Once) algorithm in detecting pig heads and pig rears using CSPDarknet53 as the backbone. Additionally, identifying interactions, or the presence of head-to-rear contact utilizes pig part detection in pig pen images for the computation of the part's Interaction over Union (IoU) that is compared to a calibrated interaction threshold. Furthermore, developing the web-based system integrates and implements the detection and interaction methods which farmers can use as a decision support tool for diagnosing potential pen problems or pig-health related concerns through tail biting.

1. Determining the optimum YOLOv5 model size for detecting pig parts using Image size [400x400], Batch size [8], and Epochs [50] as base parameters:
   (a) YOLOv5s
   (b) YOLOv5m
   (c) YOLOv5l
2. Determining the optimum hyperparameters for the selected model size that uses CSPDarknet53 in the YOLOv5 algorithm for detecting pig parts:
   (a) Image size (400 x 400, 500 x 500, 600 x 600)
   (b) Batch size (8, 16, 32)
   (c) Epochs (50, 100)
3. Evaluating the performance of the selected model size and the different hyperparameter configurations in the YOLOv5 algorithm for detecting of pig parts:
   (a) Mean Average Precision
   (b) Precision-Recall Curve
   (c) F1 Curve
4. Extracting the bounding box coordinates from the detected pig parts in pig pen images and computing each head-to-rear IoU.
5. Determining the optimum image size for detection and IoU interaction threshold for identifying presence of pig head-to-rear contact in pig pen images:
   (a) Image size (400, 500, 600)
   (b) Interaction Threshold (0.05, 0.07, 0.08, 0.09, 0.1, 0.3, 0.5)
6. Evaluating the performance of the interaction method in identifying pig head-to-rear contact in pig pen images:
   (a) Accuracy

(b) Precision

(c) Recall

(d) F1-Score

(e) Normalized Matthew's Correlation Coefficient

7. Comparing the results of the two best trained models to previous network architectures for detecting pig head-to-rear contact in [1].

8. Integrating the detection method and interaction method in a simple website that has the following functionalities:

   (a) The user can upload an image

   (b) The user can view the results of the processed image.

9. The website displays the following results:

   (a) The image with detected head and rear parts with confidence scores for each

   (b) The classification if with head-to-rear contact or without head-to-rear contact

   (c) The number of contact count

10. Applying data augmentation techniques to the dataset to add data variety for detection training:

    (a) Rotation

    (b) Horizontal Flip

    (c) Crop

    (d) Brightness

D. Significance of the Project

Pig monitoring is essential for maintaining animal condition and the overall productivity of a farm. The responsibility proves difficult because of the growing gap in ratios between farmers and livestock demand. Moreover, larger farms are tedious to maintain, especially with a small number of farmers working for them. The increasing demand for livestock requires farmers to operate at maximum or higher capacities than manageable. This problem persists with consideration of existing methods for livestock monitoring, such as the use of pedometer collars, ear tags, RFID tags, and transponder attachments. An issue with this solution is that it is invasive in nature, which raises public concerns for animal health and welfare. Thus, an alternative solution that is more efficient and non-invasive is most desirable. This emphasizes the importance of developing Precision Livestock Farming (PLF), which offers automatic and non-invasive solutions that typically employ computer vision and deep learning techniques. In PLF, strides have already been made to advance this objective in terms of measuring behavior, movement, weight, and more. Interestingly, a branch of this is the study of contact behavior, which can be indicative of health-related concerns [1]. These solutions are also inexpensive, as they only require low-tech hardware such as simple cameras and do not need any extra sensors or transponders to be used.

An abnormal behavior that occurs in pig pens is tail biting. This type of behavior can be indicative of environmental hindrances such as lack of occupation material, limited feed or water and high stocking density [8]. Another indication of this type of behavior is possible health-related concerns. Therefore, the presence of tail biting in a pig pen can serve as evidence for necessary action. With that, developing a model that can detect the presence and quantify the severity of this behavior proves to be beneficial to pig monitoring. This is accomplished through head-to-rear contact detection, which is the model that this study develops. The web-based system implements this model for the contact presence task which pig farmers can use as a decision support tool for diagnosing pig pens, and determining

necessary actions for preventing pig pen health and welfare deterioration, and increased production costs.

E. Scope and Limitations

The following are the scope and limitations of this study:

1. Training the models use images sourced from [1], which implies processing and combining the AUF and AFBI datasets.

2. Selecting the optimum YOLOv5 model size for detecting pig parts using the base parameters is constrained to the following set of sizes:

   (a) YOLOv5s

   (b) YOLOv5m

   (c) YOLOv5l

3. Selecting the optimum hyperparameters for the selected model size that uses CSPDarknet53 in the YOLOv5 algorithm for detecting pig parts is limited to the following hyperparameters:

   (a) Image size (400 x 400, 500 x 500, 600 x 600)

   (b) Batch size (8, 16, 32)

   (c) Epochs (50, 100)

4. Selecting the optimum image size for detection and IoU interaction threshold for identifying presence of pig head-to-rear contact in pig pen images is limited to the following parameters:

   (a) Image size (400, 500, 600)

   (b) Interaction Threshold (0.05, 0.07, 0.08, 0.09, 0.1, 0.3, 0.5)

5. Performance comparisons for detecting pig head-to-rear contact with models in [1] is only conducted for the two best trained models.

6. The study only utilizes and tests the effects of the following data augmentation techniques:
    (a) Rotation
    (b) Horizontal Flip
    (c) Crop
    (d) Brightness
7. The website application has the following limitations:
    (a) It only accepts PNG, and JPEG images as input.
    (b) It only accepts file sizes up to 5MB.
    (c) It only processes a single image at a time.
    (d) It does not indicate where in the image the interaction occurred.
    (e) Uploaded images does not persist in the system, and has to be reuploaded for redetection.

F. Assumptions

1. The system is used only as a decision support tool to assist in diagnosing for potential pen problems, pig health-related concerns, or tail biting outbreaks.
2. The user of the web application has the proper training and knowledge to analyze and understand the output results of the system.
3. The uploaded image is preprocessed into the acceptable format.
4. The uploaded image is a pig pen image.

II. Review of Related Literature

A. Precision Livestock Farming Significance

With the influx of demands for livestock production, jointly, interests in advancements of precision livestock farming (PLF) becomes progressively relevant. The object of discussion lies in the development of practices for modern pig farming. For practical and commercially available applications, there are automations on environmental temperature, airflow, supply of feed and water, and waste removal [6]. Moreover, methods for monitoring are also currently being used in the field such as RFIDs, ear tags, manual marking, CCTV (manual surveillance), and so on. However, these methods are not well-suited for larger scale commercial farms because the complexity for monitoring is directly proportional to the scale. In a sense, larger scale commercial farms become impractical without monitoring automation because of the animal-to-staff ratio [7].

The main concern with current practices is that it is ineffective for scaling farms due to requirement of manual labor, and it is invasive in nature. The value in PLF is that it accomplishes automatic real-time monitoring and diminishes human and animal interaction [5]. Livestock interaction and invasive methods might cause unnecessary stress, specially with unfamiliar situations. Additionally, it can also cause inaccurate monitoring due to changes in behavior caused by interaction which propagates cost. Proper monitoring is crucial in farm supervision and maintenance. Having PLF provides economic solutions and maintains livestock health and welfare.

In that regard, further investigations on PLF are of interest. Previous work on pig monitoring, specific to behavioral analysis, contact behavior, and posture recognition, have been considered. This study focuses on contact detection and contact behavior of pigs utilizing CNNs.

B. Behavioral Analysis

Pig diurnal activity can reveal state of health and welfare [6]. Assessment of long-term trends and deviations can serve as indicators for determining pig status. The suggested pipeline to extract long-term behavior changes in pigs is through detection, tracking and behavior classification [6]. The behavior classifications observed in the study are lying, eating, drinking, walking, and standing. This study uses video data of growing pigs recorded only during the daytime. Multiple object detection is conducted with the use of YOLOv3 that is pre- trained over the ImageNet dataset, and then fine-tuned for pig detection. For tracking, Kuhn-Munkres algorithm and MOSSE algorithm is utilized. The associations in the annotated bounding boxes are defined as tracklets and is used for the tracking stage. Finally, classifying the bounding boxes into behavior is left to ResNet18. For classification, the centroid location is used for identifying movement, while orientations relative to the feeder and water supply are used for consumption classifications. The computed average precision for detection is 95%. However, the average accuracy for individual frame-level label is only at 73%. Despite this, collective behavior statistics suggests that this is still accurate as the ground-truth and predicted behaviors have similar distributions with KL divergence value of 0.014 and average global prediction error of 0.14. In that sense, individual errors cancel out and return an accurate collective statistic [6]. Observations also led to recognizing that eating and drinking only occur 10% of the time and that active behavior are more common for earlier ages. Hence, older members accustomed to the environment tend to have extended lying behavior. Nonetheless, for detection, the study is limited, though having high accuracy. Due to pig's social behavior, where herds tend to stick closely together under normal circumstances, there were cases where the bounding boxes would count two pigs. The researchers recommend using ellipses because of its closeness to a pig's natural shape for detection ground-truth which might be more effective.

Farm or pen environment heavily influences pig behavior [9]. This was tested

on four different treatments: (1) temperature, (2) relative humidity, (3) volatile organic compounds (VOC), (4) illuminance. Under varying levels of these treatments, pigs were observed for feed intake, standing, lying, sitting, drinking, rooting, posture transitions, wallowing and biting for behavioral changes. This study utilized a general linear model using IBM SPSS statistic v.25 and Tukey's honest significance difference to determine how the relationship between the treatments and behaviors. In terms of temperature changes, pigs tend to stand longer on lower temperatures and lie longer when it is hotter. Moreover, pigs tend to drink more for higher temperatures. For humidity, it is directly proportional to the behaviors rooting and wallowing. VOC did not seem to have significance over pig behavior. Lastly, with low illuminance the pigs tend to lie longer and shorter with high illuminance. A limitation of this study is that gathering the observations and time were not automated. However, while this is the case, it gives an understanding of how the environment affects changes in pig behavior which is crucial for analysis and controlling independents.

Significant to behavioral analysis of livestock is identification, to generalize individualizing pigs with consideration of different circumstances such as age and environment. The use of individual detection and tracking in a pen was tested on different ages (young and old) of pigs and different environments (enriched and barren) [10]. The typical models for detection and tracking were used in this work, YOLOv3 and SORT. SORT was chosen for the use Hungarian algorithm and Kalman filter. The Hungarian algorithm was used to predict and detect the current frame and its relation to the previous adjacent frame. The Kalman filter is used to predict the future positions of bounding boxes (usually to compensate for occlusions). Additionally, the IoU score was used to check for overlaps between bounding boxes to say if the current and previous should be the same object. In cases that the bounding boxes generated do not identify pig, there are extra IDs generated or cases of false positives, these were filtered and removed to boost the detection and tracking process. Unfortunately, there needs to be a human

observer who reassigns the trajectory of the ID when the trajectory is lost [10]. In this study, finisher pig performed better than nursery pig. Additionally, the finisher pig were also better for barren environment because of the unoccupancy of surface in pens [10].

C. Pig Detection

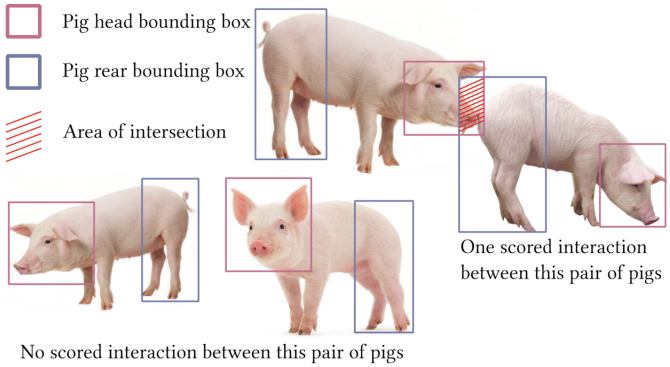


Figure 1: Interaction Scoring Sample [1]

An area of concern is pig contact behavior. Naturally, pigs engage in frequent contact, specially in better conditions [1]. Decrease in frequency and duration of contact can serve as indicators for changes in physiological or health status, which might lead to abnormal behaviors such as tail biting [1]. An automatic method for quantifying pig head and pig rear is accomplished using YOLO with ResNet50 as the backbone. This model is accompanied by K-means clustering to define anchor boxes to capture the head and rear accurately. Moreover, there are two additional sub-networks in the model to detect smaller and larger parts, for size differences that distance creates. To observe interactions, the Interaction over Union (IoU) between the detected head and rear is computed for an interaction score [1]. This method is computationally inexpensive when compared to recurrent neural networks. Together with this, using the interaction threshold calibration (ITC) an interaction threshold is pre-calibrated to identify if there are interactions. Observations used in this study are those that are 3 days before a tail-biting outbreak, as annotated by an animal behavior scientist. The interac-

tion score average accuracy is 92.56% ± 3.74%. It was observed that there is a decrease in contact during the outbreak, suggesting contact-avoidance behavior when experiencing discomfort. The model is affected by higher number of interactions because of pig-to-pig occlusions, and camera orientation. The study can be extended to observe head-to-head or head-to-flank contact for other abnormal behaviors such as flank chewing [1].

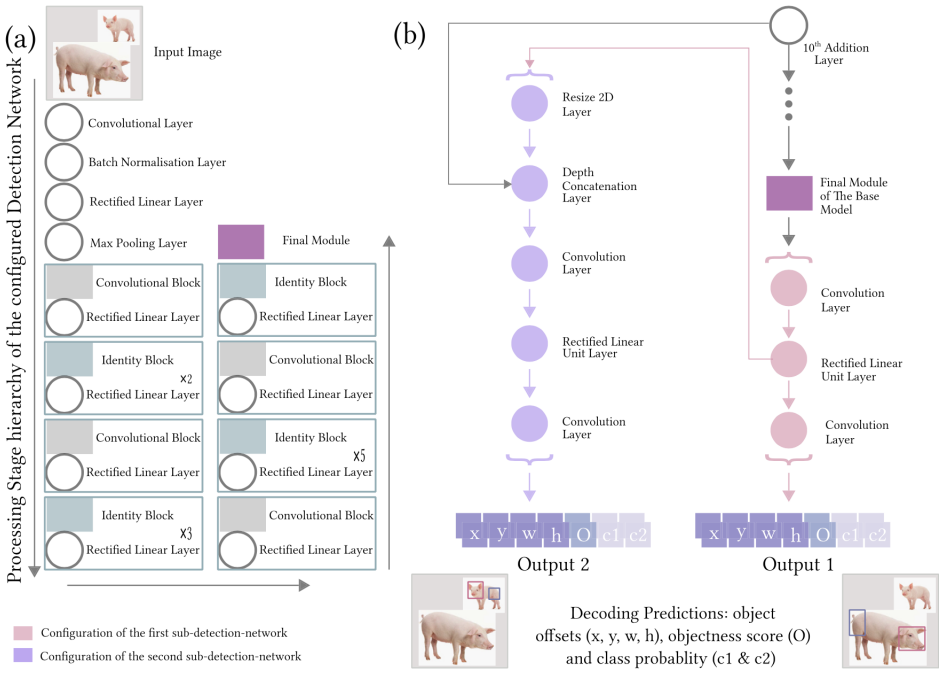


Figure 2: YOLO with ResNet50 Backbone and Sub-Networks [1]

Posture and locomotion activity detection, and tracking algorithms for the purpose of pig monitoring has been an object of interest for several studies. Commonly, the convolutional neural networks YOLO and Faster R-CNN has been used for pig detection [7]. In terms of tracking objects, a model commonly used for this task is the Simple Online and Realtime Tracking (SORT). For monitoring pig physico-temporal activities in different greenhouse gas concentrations, these models were utilized to understand behavioral changes in experimental environments. With specifics, for detection YOLOv4 and Faster R-CNN with ResNet50 as the backbone, and for tracking Deep-SORT. The work uses IoU to measure bounding box accuracy in the model. In cases of missing locations, the Kalman filter is used to estimate and provide corrections. The models for detection and tracking are

modeled separately and combined into a end-to-end pig posture scoring model. In using Faster R-CNN, for lateral lying, sternal lying, standing posture, the following mean average precision (mAP) are 97.21%, 96.83%, and 95.23% respectively. Additionally, the multi-object tracking accuracy (MOTA) is 93.86% and the multi-object tracking precision (MOTP) is 82.41%. On the other hand, YOLOv4 has mAP, in the same order, 98.52%, 98.33% and 99.18% with 93.31% MOTA and 81.23% MOTP. In this case, it would seem that YOLOv4 is a better model to use for detection. To support this further in [11] it is observed that Faster R-CNN are affected by occlusions more which means that the YOLO architecture is more robust for this use-case.

Posture recognition is an essential part of behavior analysis as it is utilized in studying pig behavior using image data or video data. The posture classifications of interest in [12] are standing, lying on stomach, lying on the side, and exploring. In [12] YOLOv5 is used as the detection network for pigs, it was chosen over the other versions of YOLO due to its improved speed and accuracy over the previous models. In conjunction, the DeepLabv3+ extraction network is used for the task of segmentation, with pre-trained weights ResNet, Xception and MobileNet. The goal is to extract the key frames from images, detect pigs, extract contours, and classify the posture [12]. This model achieved 92.45% accuracy for classifying behavior. The following measures were used to determine the effectiveness of the DeepLabv3+: mean IoU (MIoU) on Pascal VOC at 89% and Cityscape at 82.1%. The highest semantic-segmentation accuracy result was with the use of ResNet101 at 92.45% and classification accuracy for postures of 92.26%. The use of semantic segmentations provided more accurate detection because it effectively removed the background - it also helped for multi-target recognition. However, this study was conducted on only one observation angle which limits classification on non-favorable orientations. Moreover, movement classifications are difficult as images are static.

A lightweight convolutional neural network was developed for posture detection

of individual pigs called Light-SPD-YOLO. Important in this task are the low-level feature information, which is used for posture location, which is maintained in this model using feature fusion. Moreover, the model also uses a compression block ShuffleNetV2 and ECA-Net for parameter reduction. In light weight models, one of the main problems is feature loss which is mitigated with the use of ECA-Net. In turn, the dimensionality reduction operation should not prevent the feature learning [13]. The optimal parameters were obtained using YOLOv5's hyperparameter evolution that evaluates cost of fitness; pre-trained using COCO hyperparameters list. This was compared to the following models R-CNN ResNet50, YOLOv3 Darknet53, YOLOv5 Darknet53, YOLOv5 MobileNetv3-Large YOLOv5 MobileNetv3-Small. Upon testing, the average precision of the model for standing, lying on belly, lying on side, sitting, and mounting are 97.7%, 95.2%, 95.7%, 87.5%, and 84.1% respectively. Additionally, the speed of inference for this model is around 63ms. In comparison to the other existing models, the proposed lightweight model performs significantly well with minimal parameters needed. However, a limit to this model, as well as the previous models, is that it still has performance degradation for high-level behaviors such as mounting and exploring – typically this is explained by its temporal and spatial requirements [13]. There are also possible improvements on handling occlusions, specially for scaling farms, and illuminations which can be resolved by depth.

Another approach in pig detection used in [14] is through using a CNN-based detector and correlation filter-based tracker through a hierarchical data association algorithm. The tracking targets here are called tag-boxes. Alongside this, multiple object tracking is also implemented using key-points learning using correlation filters. The data association algorithm refines the detection hypothesis. In the case that tracks drift, it is corrected by probing the tracking failure and reinitializing the tracking [14]. With this, optimal trackers improve with the refined detections, while the tracking fragments integrated into their respective tracks and maintain identifications. Interestingly, this method does not need manual mark-

ing or physically identifying the pigs. However, this study is subject to effects of illumination, similar pig appearance, shape deformation and occlusions [14].

D. Synthesis

The relevance of PLF is increasing due to the growing demand of livestock production. Currently in the industry there are existing methods for monitoring such as RFIDs, ear tags, and CCTV which are not well-suited for larger scale commercial farms. The main advantage of employing PLF is that it allows automatic real-time monitoring, hence reducing the need for manual labor and minimizing human and animal interaction. This study focuses on using CNNs and IoU computations for head-to-rear contact detection as a step towards improving the efficiency of pig farming and welfare of pigs.

Discussed here are various methods of analyzing pig behavior in a farm setting. This is conducted using image and video data, and utilizing computer vision techniques and algorithms such as YOLO and deep learning architectures. Advances in classifying pig activity, effects of environmental factors such as temperature, humidity, and light, and pig identification are common studies that answer to this problem. Behavioral analysis is crucial in understanding how external factors such as the environment and changes in pig diurnal activity affects their welfare. Monitoring the behavior of pigs can provide useful information about their health and welfare, and as seen in previous work, the automation of these methods can improve the efficiency of this task.

Monitoring pig activity relies heavily on the accuracy of the detection method being used may it be for behavior classification, posture detection, or contact behavior tasks.

III. Theoretical Framework

A. Tail Biting

Pigs are known to be inquisitive in nature, often they would use their mouths to investigate the environment and have the natural tendency to chew [8]. This behavior can lead the pigs to exhibit signs of tail biting. The act can commonly occur when the environment does not allow the pigs to explore and be kept occupied through other means. There are several causes to tail biting such as lack of occupation material, limited feed or water, feed composition, high stocking density, environment, weather changes, genetic disposition, and health-related problems; therefore, tail biting could be used as an indicator for hindrances to maintain livestock health and welfare.

There are three known forms of tail biting. The first is due to limits in occupancy where the environment invites the pig's inquisitive nature to chew on other pig's tails because of frequent and extended contact of the head and rear. The second is due to competition in the pen, often caused by inadequacy of the feeder for the pig population. Factors like this cause aggression between pigs which leads to tail biting. The third is due to a single pig being a fanatical bitter which creates stress in the pen and might extend tail biting behavior to other pigs; therefore, these pigs must be separated immediately.

B. Convolutional Neural Network

A neural network is a composition of an input layer, one or more hidden layers, and an output layer. Each node in this network connects to another and is associated with specific weights and thresholds. Outputs that exceed the threshold value activates the node and sends data to the next layer of the network. Convolutional Neural Networks (CNN) is an extension of this architecture that provides an approach for image processing and object recognition using pixel data from image or video data. The CNN layers can extract simple features initially, such

as colors and edges, and is progressively able to recognize complex features to identify target objects. The CNN has three main layers [15]:

1. Convolution Layer

The core building block of a CNN is the convolution layer where most of the network's computation occurs. This layer consists of a few necessary components being the input data, a kernel or filter and a feature map [15]. The kernel is an array of weights used to represent parts of an image. The usual size of a kernel matrix is 3x3, 5x5 or 7x7, and is applied to areas of the image; this also determines the receptive field. After applying the kernel, the dot product of the input pixels within the area of the image and the kernel is calculated to produce the output array, which is also called the feature map. This process is done repeatedly, across the image, to get the entire feature map relative to the input data. In order to process the entire image, the kernel is shifted by a stride, which is the distance between the previous and successive kernel. However, shifting the kernel by stride does not guarantee that it will always fit the input data, which causes an inconsistency in the input and output matrix sizes. Hence, padding is applied to avoid this and produce larger or equally sized outputs.

2. Pooling Layer

The function of a pooling layer, essentially, is for downsampling. It is used for dimensionality reduction and reducing parameters in the input [15]. In terms of process, it conducts similar to the convolution layer, however without inclusion of weights in the kernel. Alternatively, it aggregates the area of the image currently being processed which would be the output. This helps reduce the spatial size of the feature map, in turn, reducing complexity, training time and risk of overfitting of the model. There are two types of pooling which are: max pooling which selects the maximum value and average pooling which calculates the average value in the receptive field.

3. Fully Connected Layer

The function of a fully connected layer is to conduct the classification task using the features extracted from the previous layers. This layer connects nodes of the output layer directly to the previous layer, in turn flattening it or compiling the extracted data to form a final output.

C. Detection Method

1. YOLO Algortihm

The YOLO (You-Only-Look-Once) algorithm is a real-time object detection algorithm. As the name suggests, this algorithm can detect objects with a single forward propagation and only requires a single CNN to perform its function [16]. It is a multi-scale network that combines a feature extraction network and a detection network to generate predictions. The feature extraction network that uses a CNN architecture divides the image into a grid of cells. With this, it is able to predict the probability of the presence of an object in the cell divisions. The detection network uses these predictions to produce labeled bounding boxes that can be used to locate the coordinates of objects in the image. The class probabilities and bounding boxes are computed simultaneously allowing for the model to detect numerous objects in images with high speed and precision making it suitable for real-time applications [17]. This algorithm makes use of three important concepts which are residual blocks, bounding box regression, and IoU.

This study implements and tests the YOLOv5 algorithm for object detection of pig parts, specifically locating the pig head and rear of pigs in pig pen images. Locating these parts accurately are crucial in determining the head-to-rear contact presence, and possibility of tail biting behavior. The YOLO algorithm uses CSP-Darknet53 as the backbone network, which is the algorithm's default backbone [4]. Additionally, it also utilizes the distance metric of IoU, which is a standard for YOLO algorithms [17].

2. YOLOv5

YOLOv5, developed by Glenn Jocher, CEO of Ultralytics in 2020 [18], has emerged as a prominent and highly effective solution for object detection tasks. Building upon the success of its predecessors, YOLOv5 introduces several advancements aimed at improving the accuracy and speed of real-time object detection. Notably, compared to YOLOv4, YOLOv5 offers faster training and inference times while maintaining the same mean average precision (mAP) [4] [19]. Furthermore, YOLOv5 stands out for its lightweight architecture and smaller model size compared to similar counterparts. This algorithm has demonstrated exceptional performance across diverse domains, including but not limited to person detection, vehicle detection, and animal detection. Its remarkable achievements make YOLOv5 a valuable tool for researchers and practitioners seeking efficient and accurate object detection capabilities in various applications.

Ultralytics is dedicated to continuously enhancing the YOLO model through their ongoing releases [20]. The YOLOv5 model, being open source and actively maintained in a GitHub repository [2], offers the advantage of easy replication and modification to suit specific needs and applications.

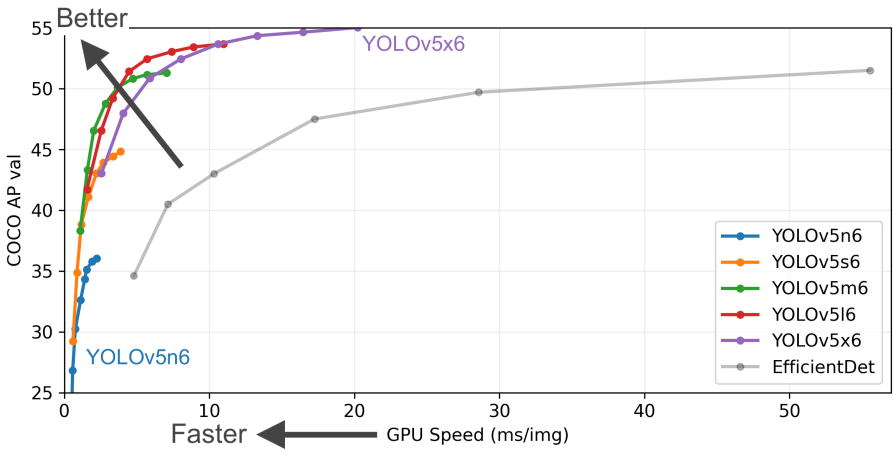


Figure 3: YOLOv5 Model Sizes Comparison [2]

The YOLOv5 algorithm offers various model sizes, allowing flexibility and customization to meet specific requirements. The different sizes of YOLOv5, namely YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, represent a trade-

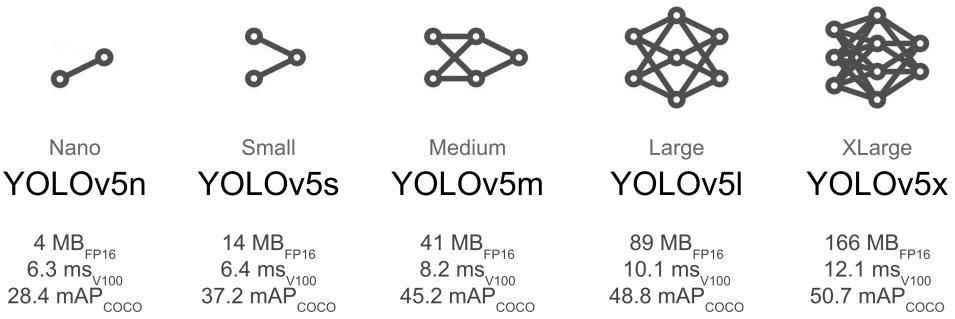


Figure 4: YOLOv5 Model Sizes [3]

off between model complexity, inference speed, and accuracy. YOLOv5n and YOLOv5s are smallest variants, featuring a lightweight architecture with fewer parameters, resulting in faster inference times but potentially sacrificing some detection accuracy. YOLOv5m strikes a balance between size and performance, offering moderate model complexity and achieving a reasonable trade-off between speed and accuracy. YOLOv5l and YOLOv5x are larger models, featuring increased network depth and capacity, which can yield higher accuracy but with slower inference times due to the additional computational requirements. The availability of different model sizes empowers users to choose the most suitable configuration based on their specific application needs, striking a balance between computational efficiency and detection performance.

3. Training YOLOv5

In the Ultralytics website, there are several best practices recommended to optimize training for YOLOv5. Firstly, it is advised to have a dataset with a minimum of 1500 images per class and at least 10,000 labeled instances per class to ensure sufficient training data. The dataset should also encompass image variety, including different times of day, seasons, weather conditions, lighting, angles, and sources. Labeling consistency and accuracy are crucial, with all instances of all classes correctly labeled and closely enclosed within their respective bounding boxes. Verifying labels by inspecting train batch images at the start of training is also suggested. Background images, which contain no objects, can be added to

the dataset to reduce false positives. The choice of model size should be tested to determine the most suitable one for the specific use case. During training, it is advisable to start with 300 epochs and adjust based on early overfitting. Higher image resolutions, such as 1280, can benefit datasets with small objects. The batch size should be maximized within hardware limits to avoid poor batch-norm statistics. Default hyperparameters should be used initially, and increasing augmentation hyperparameters can help reduce overfitting. The reduction of loss component gain hyperparameters can also mitigate overfitting. For automated hyperparameter optimization, the Hyperparameter Evolution can be utilized. More details on optimizing train can be found in [3].

4. YOLOv5 Architecture

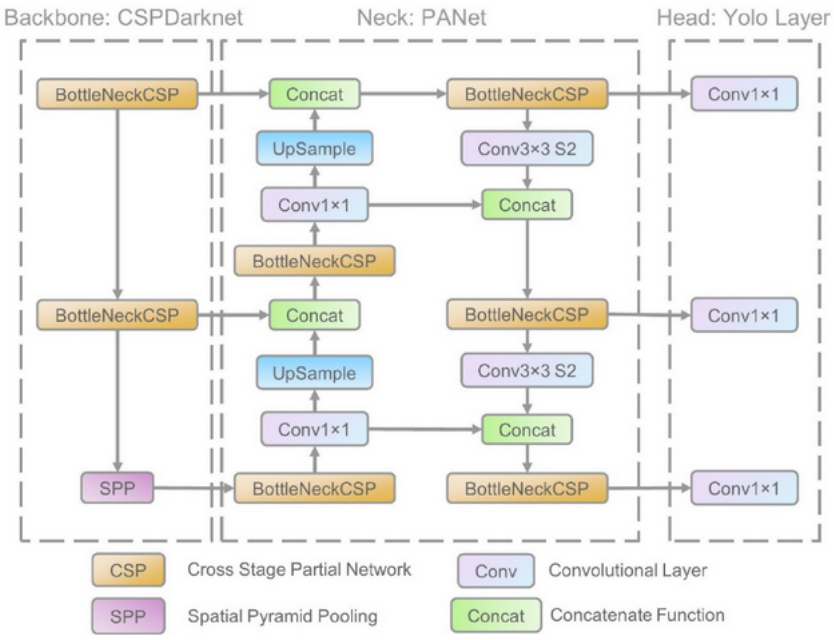


Figure 5: YOLOv5 Network Architecture [4]

CSP-Darknet53 serves as the backbone for YOLOv5, derived from Darknet53, which was the backbone for YOLOv3. The authors of YOLOv5 introduced the Cross Stage Partial (CSP) network strategy to enhance its performance. This strategy addresses the issue of redundant gradients in deep networks by truncating the gradient flow. By partitioning the feature map of the base layer and merging

it through a cross-stage hierarchy using the BottleNeckCSP module, YOLOv5 effectively reduces the number of parameters and computational requirements while increasing inference speed. This improvement in efficiency plays a crucial role in real-time object detection models, making YOLOv5 a powerful and efficient solution [4].

The neck of YOLOv5 incorporates two key changes. It utilizes a variant of Spatial Pyramid Pooling (SPP) to aggregate information and increase the receptive field without compromising network speed. Additionally, the Path Aggregation Network (PANet) is modified by integrating the BottleNeckCSP module, which applies the Cross Stage Partial (CSP) network strategy. This modification improves information flow, aids in accurate pixel localization for mask prediction, and enhances the overall performance of YOLOv5 [4].

In terms of the network's components, YOLOv5 maintains the same head structure as its predecessors, YOLOv3 and YOLOv4. This head consists of three convolution layers responsible for predicting bounding box coordinates, object scores, and object classes. Notably, the computation of target bounding box coordinates has been updated in YOLOv5, introducing differences from previous versions. Activation functions play a crucial role, and YOLOv5 employs SiLU (Sigmoid Linear Unit) and Sigmoid activations. SiLU is used in hidden layers to improve convolution operations, while Sigmoid is employed in the output layer [4].

Regarding the loss function, YOLOv5 utilizes Binary Cross Entropy (BCE) loss for class and objectness predictions, as well as Complete Intersection over Union (CIoU) loss for accurate location predictions [4]. The final loss is computed based on a specific equation. These components collectively contribute to the improved accuracy and performance of YOLOv5 in object detection tasks.

D. Interaction Method

The interaction method is conducted by computation of the IoU of the bounding boxes (Bbox) from the detected pig heads and pig rears and then comparing it

with the calibrated IoU interaction threshold. The IoU computation sums to getting the absolute value of the intersection between Bboxes over its union (see Figure 6). The IoU interaction threshold is calibrated through comparison of model performance using interaction metrics for different configurations over the interaction dataset. Head-to-rear IoU scores that are greater than or equal to the threshold is classified as with contact, otherwise without contact.

$$IoU = \frac{|Bbox_1 \cap Bbox_2|}{|Bbox_1 \cup Bbox_2|}$$

Figure 6: IoU formula [1]

E. Data Augmentation

The study applies the following image augmentation techniques to the dataset to simulate similar circumstances in pig pens. Data augmentations are conducted in Roboflow [21]:

1. Spatial Augmentation

1. **Rotation.** The image is rotated both clockwise (CW) and counter-clockwise(C-CW) by 90°, as well as within a range of ±35°. Each rotation produces a distinct image with unique characteristics.

2. **Horizontal Flipping.** The image is horizontally flipped, creating a mirrored version that retains its key features and characteristics.

3. **Crop.** The image is cropped to a minimum and maximum zoom range of 0-50%, capturing a smaller portion with enlarged and focused details.

2. Pixel Augmentation

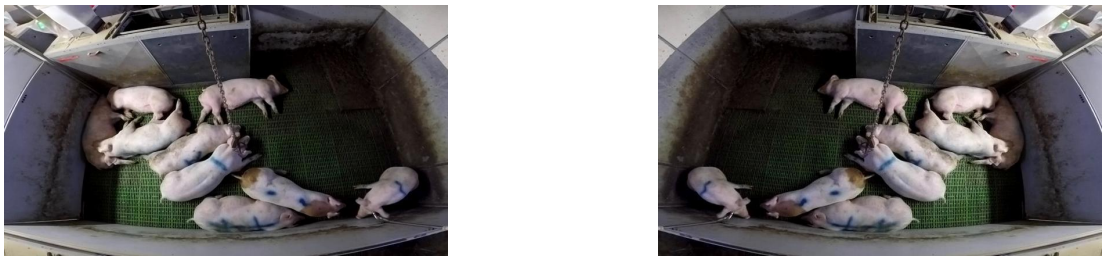
1. **Brightness.** The image brightness is increased or decreased by the within the range of ± 20%, where each adjustment creates a unique image.

(a) Before Rotation (b) Rotated 90° CW (c) Rotated 90° C-CW

Figure 7: Spatial Augmentation: 90° Rotations

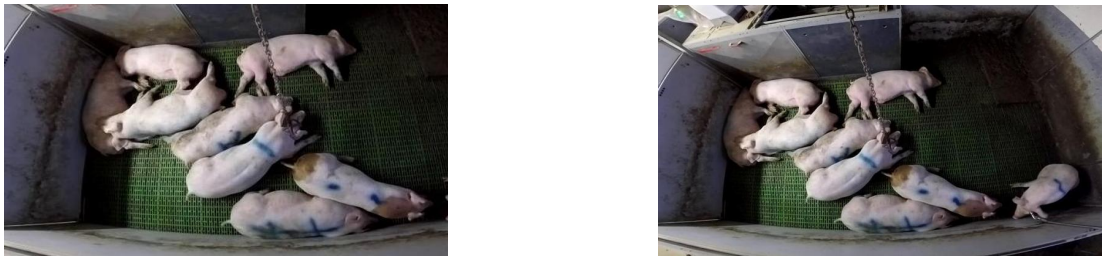
(a) Before Rotation (b) Positive Rotation (c) Negative Rotation

Figure 8: Spatial Augmentation: ± 0-35° Rotations



(a) Before Flipping (b) After Flipping

Figure 9: Spatial Augmentation: Horizontal Flipping



(a) Cropped Sample 1 (b) Cropped Sample 2

Figure 10: Spatial Augmentation: Crop

(a) Brightness to +20% (b) Original Image (c) Brightness to −20%

Figure 11: Pixel Augmentation: Brightness

F. Detection Performance Metrics

1. **Mean Average Precision (mAP).** This metric is commonly used for evaluating the performance of object detection models and is the main metric for the detection method. It is calculated by finding the average precision of each object class and then taking the mean of those values. The higher the mAP score, the better it is at detecting objects of different classes in an image and determining their locations. This metric also uses an IoU threshold usually set at 0.5, however, there are instances where ranges of IoU are observed such as from 0.5 to 0.95 denoted as *mAP@0.5:0.95*.

$$mAP = \frac{\sum_{n=i}^{C} AP_i}{C}$$

   (a) C is the number of object classes

   (b) AP_i is the average precision for the class

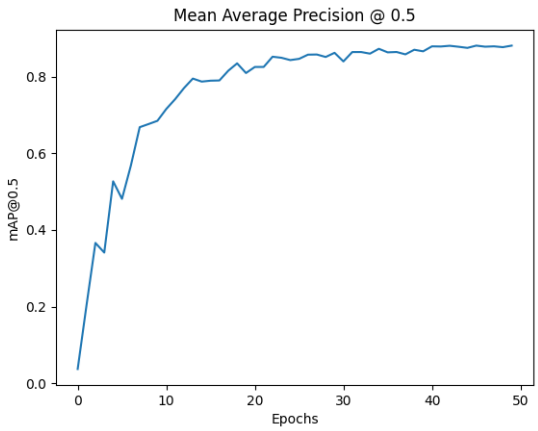
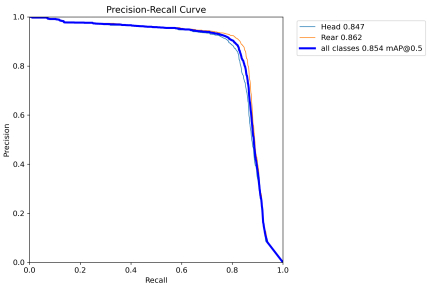


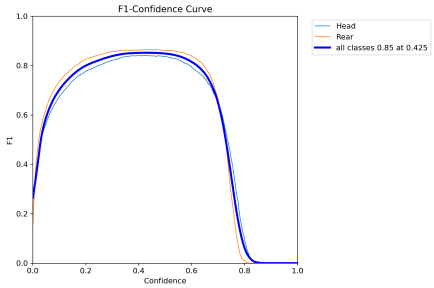
Figure 12: Sample mAP Curve Graph

2. **Precision-Recall Curve (PR Curve).** The PR curve is a graphical representation of the trade-off between precision and recall. It is generated by plotting the precision (y-axis) and the recall (x-axis) at different threshold settings. The area under the PR curve (AUC-PR) is a metric that ranges from 0 to 1 used to evaluate the performance of object detection models.

3. **F1 Curve.** The F1 curve summarizes the balance between precision and recall in a concise graphical form. By plotting the F1 score (y-axis) against different threshold settings or decision boundaries (x-axis), the curve showcases the model's performance across the range of possible classification thresholds. The F1 score is a single metric that captures the harmonic mean of precision and recall, providing an overall assessment of the model's effectiveness in object detection tasks.



(a) Sample PR Curve Graph



(b) Sample F1 Curve Graph

Figure 13: Detection Metrics

G. Interaction Performance Metrics

1. Confusion Matrix

A confusion matrix is an N x N matrix, where N is the number of classes, used to evaluate the performance of classifiers. It compares over the actual classification (known as the true class) and the classification predicted by the model. This matrix is sub-divided into four individual components used to define the classification measures [22]:

1. **True Positive (TP).** A true positive defines an outcome where the model predicts the true positive class correctly.

2. **True Negative (TN).** A true negative defines an outcome where the model predicts the true negative class correctly.

3. **False Positive (FP).** A false positive defines an outcome where the model

predicts positive when it belongs to the true negative class. It is an incorrect classification.

4. **False Negative (FN).** A false negative defines an outcome where the model predicts negative when it belongs to the true positive class. It is an incorrect classification.

2. Classification Measure

1. **Accuracy.** This metric measures the rate at which the classifier makes a correct prediction. Therefore, it is the ratio of the correct predictions over the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

2. **Precision.** This metric measures the rate of correct positive predictions in terms of the total positive predictions. Therefore, it is the ratio of the true positive over predicted positives.

$$Precision = \frac{TP}{TP + FP}$$

3. **Recall.** This metric measures the rate of correct positive predictions in terms of the true positive class. Therefore, it is the ratio of the true positive over the actual positives. This is also known as the "true positive rate" or "sensitivity."

$$Recall = \frac{TP}{TP + FN}$$

4. **F1-Score.** This metric is the harmonic mean of precision and recall which provides a scale that maintains the balance between these two classification measures.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall}$$

5. **Normalized Matthew's Correlation Coefficient (nMCC).** This metric that encapsulates the balance between true positives, true negatives, false positives, and false negatives. It offers a scaled evaluation of the overall classification performance, considering the distribution of all prediction outcomes. The MCC by itself ranges from $[-1, 1]$ to measure agreement between the predicted and actual values, it is normalized to adjust the range to $[0, 1]$. This is the main metric to decide the best interaction threshold.

$$MCC = \frac{TN \times TP - FN \times FP}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

$$nMCC = \frac{MCC + 1}{2}$$

IV. Design and Implementation

A. Pig Tail Biting Dataset

The dataset being used in this study is acquired from the research conducted by [1]. on pig contact behavior. This dataset is a combination of selected images from two video datasets: AFBI and AUF. The AFBI dataset was obtained from an experiment conducted at the Agri-Food and Biosciences Institute in Northern Ireland. The videos were captured in a controlled environment with maintained ventilation, artificial lighting, and daily health checks. The selected images from AFBI dataset have a resolution of 765 x 432 pixels. On the other hand, the AUF dataset was obtained from an experiment conducted at the Department of Animal Science, Aarhus University in Denmark. The data involved finisher pigs in a controlled environment with different treatments applied to the pens. These treatments include variations in tail docking, provision of straw, and stocking density, aimed at assessing risk factors for tail biting. The selected images from the AUF dataset have a resolution of 689 x 474 pixels. Both datasets were manually annotated by two trained individuals with a background in animal behavior, specifically for the head and rear parts of the pigs. Additionally, another dataset was created for validation of interactions, which consists of images from both base datasets. This validation dataset was annotated by an animal behavior scientist, assigning an interaction score to each image. The images in the validation dataset were classified into categories of "(0) no contact," "1 contact," "2 contacts," or "3 or more contacts."

B. Research Approach

This study takes an exploratory approach to determine the optimal YOLO model size and training hyperparameters for detecting head-to-rear contact in pen images using the YOLOv5 algorithm with CSPDarknet53 as the backbone. The max number of epochs being tested is 50 and 100. The image size also varies from 400

x 400, 500 x 500, and 600 x 600 pixels. The parameter batch size is manipulated by batches of 8, 16, and 32. Other hyperparameters for training are being kept the same and constant as employed in [1] as these are also standard values used in previous studies on pig detection. The learning rate is being retained at 1x10^{-}3. Moreover, the overlap threshold is also being retained at 0.5, which has been established in [1]. Data augmentation techniques are being applied to the detection dataset to increase the variety of the dataset and improve the performance of the model for general purposes. The following data augmentation techniques are: rotations, horizontal flipping, cropping, and brightness. The IoU between the head and rear of pigs detected by the algorithm is integral in determining contact between pigs. Indication of contact is determined using an interaction threshold that is being calibrated using a held-out dataset from the annotated interaction dataset. IoU that exceeds the interaction threshold is indicative of interaction between pigs. Performance evaluations of the detection method using YOLOv5 is facilitated by using mAP, PR curve, and F1 Score as metrics. For the interaction method, prediction evaluations are faciliated by using the confusion matrix, and metrics such as accuracy, precision, recall, F1-score, and nMCC. The model configuration that outputs the best results in the detection method and interaction method metrics is the optimum model for pig part detection.

C. Use Cases

The website application, TailSafe, integrates the optimum model chosen from model performance evaluations. This application is capable of detecting head-to-rear contact between pigs in a given pig pen image. The target users of this application are pig farmers. The pig farmer can upload pig pen images to the system, which are processed to determine the presence of head-to-rear contact, classified as with contact or without contact. Furthermore, the system also provides information on how many times contact occurs in the uploaded image. A use case diagram of this system is shown below:

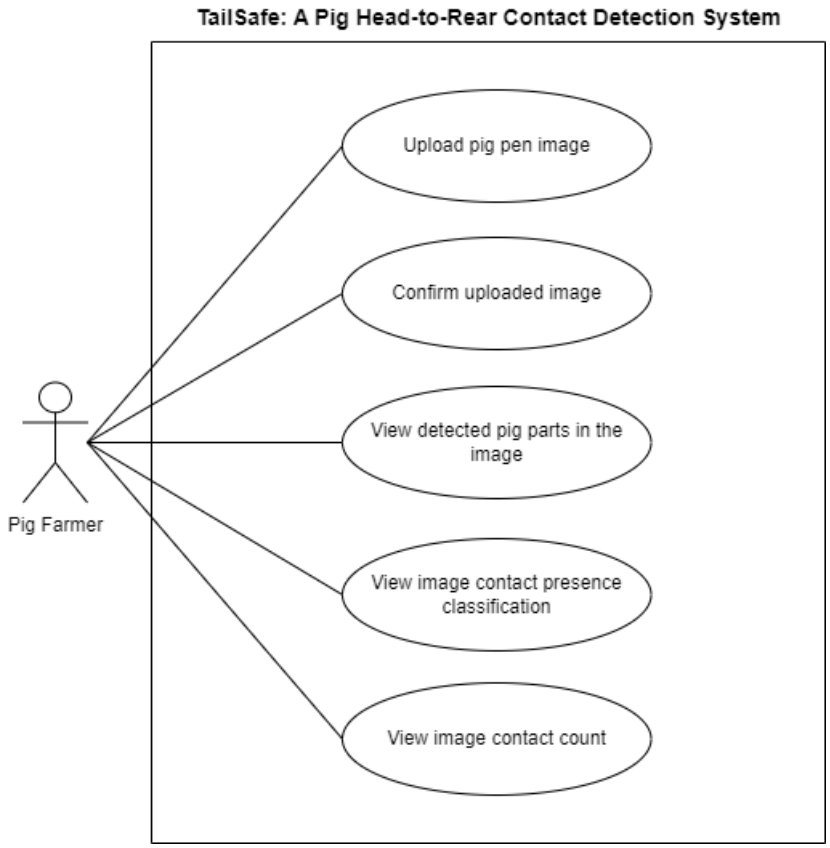


Figure 14: Use Case Diagram of TailSafe

D. Technical Architecture

The main programming language being used to develop and implement the detection and interaction method is Python. The models and methods are run in Google Colab. To support faster training times and heavier training workloads, GPU accelerated runtimes in Colab Pro are being used. The available GPUs in Colab Pro include Nvidia T4, V100, or A100 Tensor Core with high-ram configurations. The development of the methods are with support of open-source libraries available to Python such as PyTorch and TensorFlow. PyTorch is a deep learning framework that provides a flexible and efficient platform for building and training neural networks. This library also supports the object detection model YOLOv5. With that, the researcher leverages the combination of this framework and model support to build, train, and test for this model. TensorFlow is a machine learning and deep learning library developed by Google. This library and its set of tools are useful for metric outputs leveraging the TensorBoard. The main compo-

nents and utilities for YOLOv5 are available at the Ultralytics Github repository in [2] which is being used in this study. Modifications to the YOLOv5 resource are being made, especially for model configurations, to suit the specific detection task. When it comes to the datasets, all data preprocessing, compiling, splitting, annotation, and augmentation tasks are being handled in Roboflow.

Development and implementation of the website application are being conducted in Python using the Django framework. Django is a free and open-source web framework that follows the Model-View-Controller architecture and features intelligent development capabilities for building website applications.

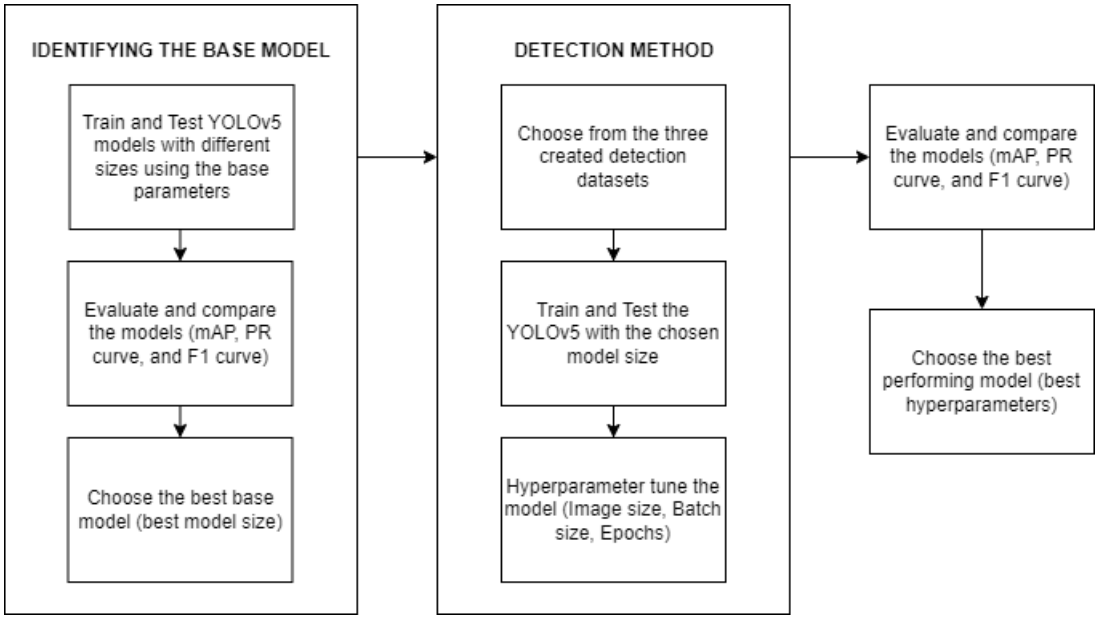


Figure 15: Detection Method Training Pipeline

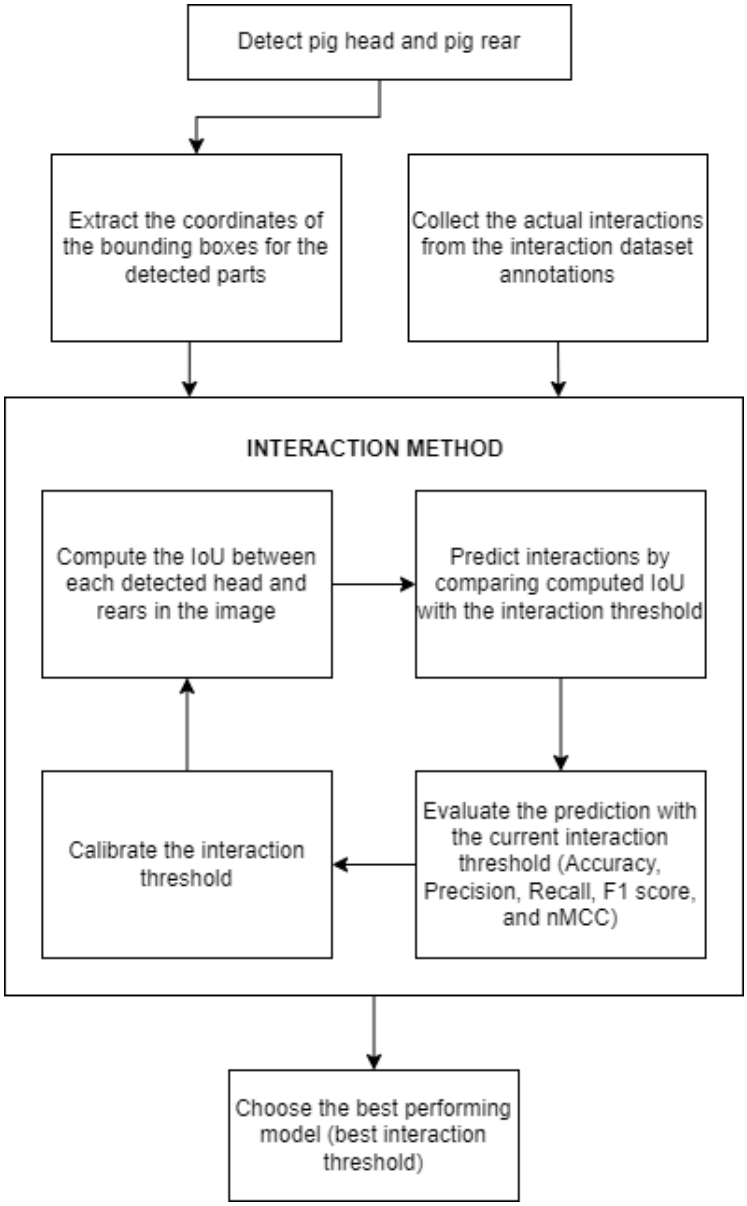


Figure 16: Interaction Method Training Pipeline

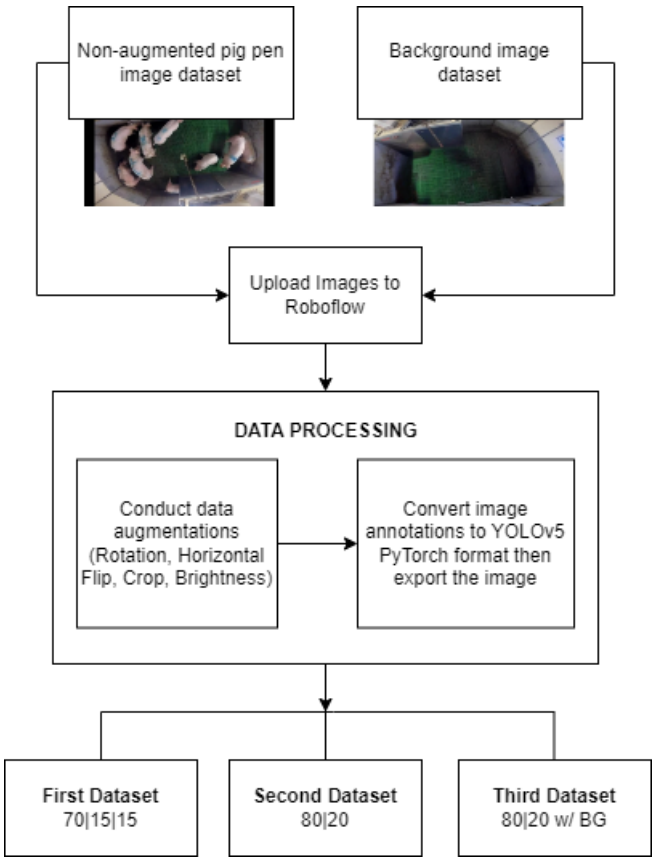


Figure 17: Data Processing Pipeline: Detection Dataset

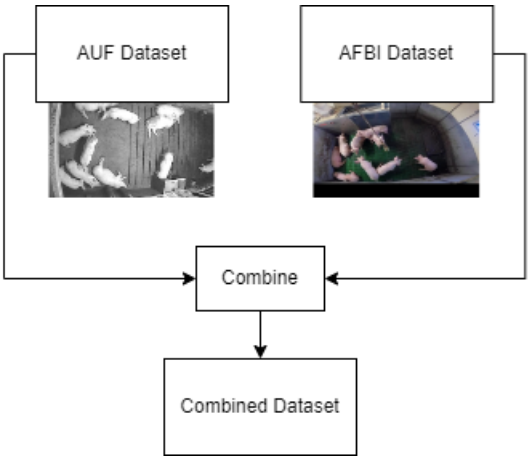


Figure 18: Data Processing Pipeline: Interaction Dataset

V. Results

A. Dataset

The dataset utilized in this research study was obtained from a readily available source, from [1]'s research on pig contact behavior. The dataset consists of combined images from two sources, namely the AFBI and AUF datasets as previously mentioned.

Regarding the detection dataset, the AFBI dataset comprises 1,556 pig pen images with a resolution of 765 x 432, while the AUF dataset contains 1,225 pig pen images with a resolution of 689 x 464. After application of data augmentations, the detection dataset used in [1] consisted of a total of 51,193 instances, combining 26,533 instances from AFBI and 24,660 instances from AUF. It is worth noting that these datasets were captured from different pens on the farm, resulting in varying pig quantities per pen. The specific number of pig heads or pig rears per image was not mentioned. For this study, the non-augmented dataset was used, which also includes annotations for pig head and pig rear in COCO, CSV, TFRecord, and VOC formats, with the COCO format being utilized in this study.

Furthermore, there was another dataset created from combining the AFBI and AUF datasets, comprising 433 images from AFBI and 305 images from AUF, resulting in a total of 738 images. This interaction dataset contains annotations related to the count of pig tail biting within the pens. Each image was classified into four categories: (0) no contact, 1 contact, 2 contacts, or 3 or more contacts. Notably, the AUF dataset only had a few instances with three or more contacts, which prompted the combination of datasets to achieve a more balanced distribution of classes [1]. Despite this approach, the classification of the dataset still remains imbalanced (refer to Table 2). Majority of images in the dataset depict instances without any contact, significantly outnumbering the images that capture contact between pigs.

B. Data Processing

In this research, the non-augmented data and COCO annotations were uploaded to Roboflow (see [21]) for data processing. The dataset was processed using the generate function in Roboflow, and no data preprocessing was applied. The dataset was then split into a 70-15-15 ratio without any further modifications. In the data augmentation phase, several techniques were utilized, including rotation by 90°, rotation by ± 35°, horizontal flip, cropping between 0% and 50%, and brightness adjustment of ± 20%. These techniques were applied to generate a new dataset specifically for training the detection model. This dataset is referred to as "Primary" in Table 1, and served as the primary dataset for training in this study. After generation, the dataset was exported from Roboflow in YOLOv5 PyTorch format to follow compatibility with YOLOv5 training and detection.

Image Count of Detection Datasets					
Dataset	**Split**	**Train**	**Valid**	**Test**	**Total**
Original [1]	90-10-00	2502	0	279	2781
Original (BG)	80-20-00	2446	610	0	3056
Primary	70-15-15	5847	417	417	6681
Secondary	80-20-00	6684	555	0	7239
Secondary (BG)	80-20-00	7338	610	0	7948

Table 1: Detection Datasets

In the table provided, there are datasets labeled "Original" that consist of non-augmented images from [1]. To improve the model's performance, a new dataset was created by combining background images, as suggested in [3]. This dataset, labeled "Original (BG)," includes backgrounds. The "Secondary" dataset used the Original dataset and is split to 80-20, which considers the interaction dataset as a separate test set for the detection model. Similarly, the "Secondary (BG)" dataset, derived from the Original (BG) dataset, was also split 80-20. These dataset configurations focused on the exploration of different data compositions

and its effects on the model's performance. In the Split section of the table, the order is as follows: train, valid, and test splits. If there are only two values present in the Split section, it indicates that the dataset does not include a separate test split.

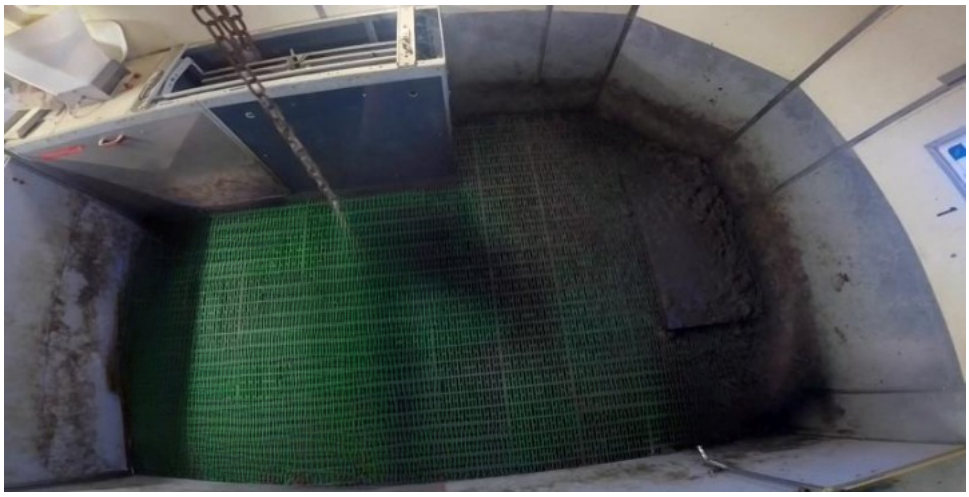


Figure 19: Sample Background Image

For the interaction dataset, the AUF and AFBI sets were combined into one whole dataset called "Combined" which was used for the interaction method to compute for IoU scores and identify contact presence.

Interaction Dataset			
Dataset	**Image Count**	**w\ Contact**	**w\o Contact**
Combined	738	282	456

Table 2: Interaction Datasets

C. Detection Method

1. Determining the Best Model Size

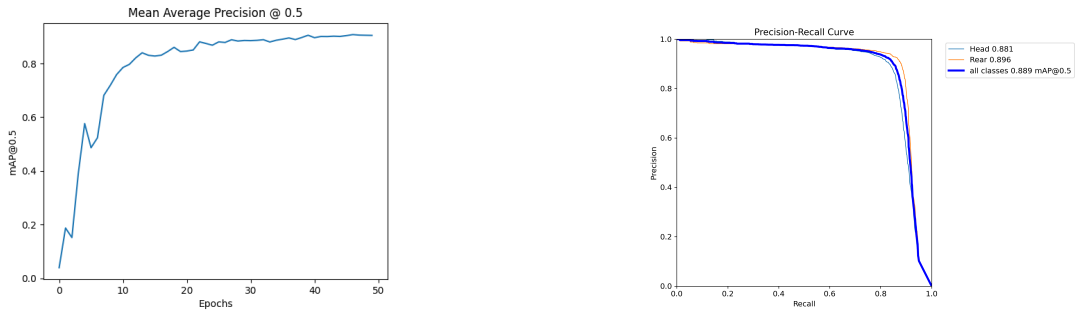
The models were trained using the primary dataset, considering three different sizes: small, medium, and large. All three models were trained with the base hyperparameters, including an image size of 400, batch size of 8, and 50 epochs.

This approach served as a control to evaluate and determine the model size that's most suitable for the objective of detecting pig head and pig rear.

Base Model Results				
Model	**Precision**	**Recall**	**mAP@0.5**	**Speed (hrs)**
Small	0.897	0.840	0.880	0.908
Medium	0.911	0.870	0.908	1.247
Large	0.912	0.884	0.918	1.832

Table 3: Base Model Results

After analysis of the training results, it was observed that larger model sizes outperform smaller ones in detecting pig heads and pig rears. However, there was minimal discrepancy between the performance of the medium and large models, as indicated by the metrics presented in Table 3. Conversely, the small model exhibits the poorest performance among the three sizes. In terms of training speed, it is noteworthy that the medium model demonstrates a 58.5% faster training rate compared to the large model. This speed advantage allows for quicker conclusions to be drawn. Taking these factors into consideration, the medium model size was selected as the base model.



(a) Base Model mAP Curve Graph (b) Base Model PR Curve Graph

Figure 20: Base Model (Medium-Sized) Metrics

2. Training the Base Model and Hyperparameter Tuning

The base model was trained on the primary dataset using various hyperparameter combinations, including different image sizes (400, 500, 600), batch sizes (8, 16, 32), and epochs (50, 100). In total, 18 models were trained, taking into account all metrics. The table presented in Table 4 highlights the top three models that emerged from this training process. In the Model section of the table, the format represents the model size, image size, batch size, and epochs used for each model.

Best Detection Model Results [Primary Dataset]			
Model	**Precision**	**Recall**	**mAP@0.5**
m-600-8-100	0.929	0.895	0.930
m-600-16-100	0.926	0.892	0.930
m-500-8-100	0.926	0.895	0.929

Table 4: Best Detection Model Results [Primary Dataset]

3. Evaluating Medium-Sized Model Results

The top three models, as presented in Table 4, demonstrate promising results. Among them, m-600-8-100 is the best-performing model with the following results 92.9% Precision, 89.5% Recall, and 93% mAP. Although, there are only slight difference in the results of these three models. Hence, choice between the models can be guided by considering the hyperparameters used per model. Based on training results, higher batch sizes and smaller image sizes developed cost-effective models that are less resource-intensive to train. The trade-off between these parameters are not exaggerated. For the purpose of this study, these three models were selected to be integrated with the interaction method.

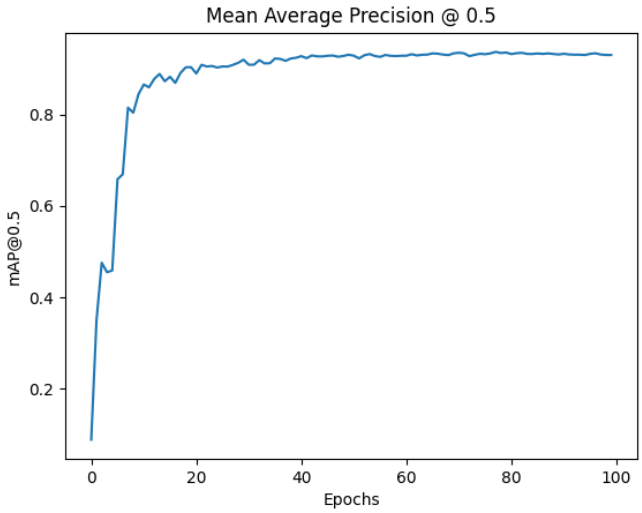


Figure 21: Best Detection Model (m-600-8-100) mAP Curve [Primary Dataset]

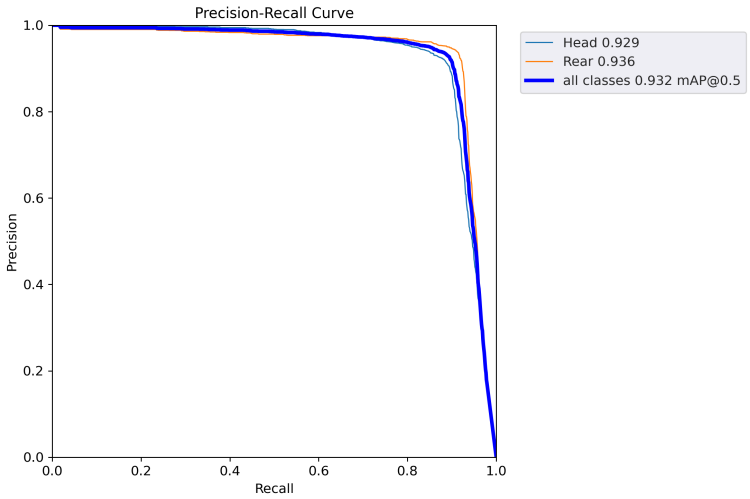


Figure 22: Best Detection Model (m-600-8-100) PR Curve [Primary Dataset]

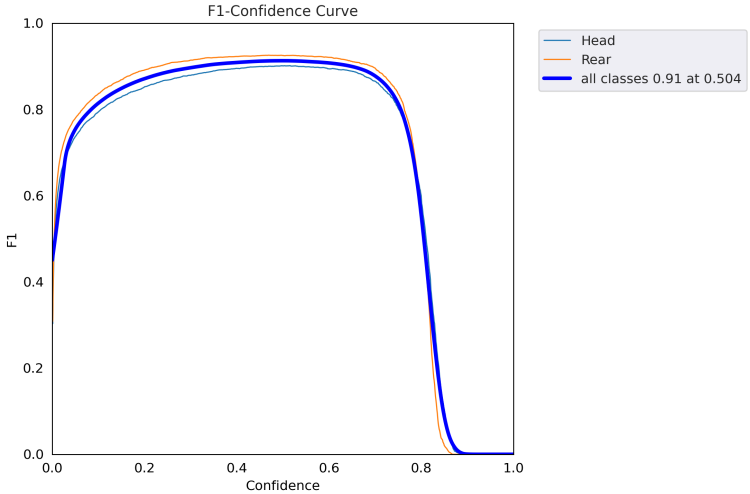


Figure 23: Best Detection Model (m-600-8-100) F1 Curve [Primary Dataset]

The model m-600-8-100 exhibits a notable increase in mAP during the initial epochs of the training process. The performance reached a stable point at approximately 40 epochs, and attained a commendable score of 0.9. Throughout the entire 100 epochs of training, there is no observable downward trend in performance, indicating that the model does not suffer from overfitting. This suggests that the model maintains its effectiveness and generalization capabilities throughout the training duration. Fig. 21.

Analyzing the PR curve, it is evident that the area under the curve is substantial, indicating a strong performance characterized by high precision and recall values. This signifies a balanced trade-off between accurately identifying positive instances and minimizing false positives. Additionally, the F1 curve exhibits a prominent peak at 0.9, signifying a favorable harmonic mean between precision and recall. The confidence level of the F1 score is slightly above 0.8, further reinforcing the model's robustness and ability to maintain consistent performance across different thresholds. These results highlight the model's effectiveness in achieving reliable and accurate predictions. Fig. 22 & 23.

D. Interaction Method

1. Integrating Methods

The weights of the top three models, namely m-600-8-100, m-600-16-100, and m-500-8-100, were extracted from the training results. Furthermore, the weights for the small model and the large model with base hyperparameters were also acquired. This selection was motivated by the desire to assess the performance disparity between the small model, which exhibited the lowest performance, and the chosen models. Additionally, the weight acquisition for the large model was driven by its close performance proximity to the top three models, as indicated in Table 3.

The interaction dataset was utilized to detect pig heads and pig rears using the

selected models (sample Fig. 24). The resulting bounding boxes were extracted, and for every combination of head and rear coordinates, the IoU was computed. The detection process was performed across three different image sizes (400, 500, 600), while the presence of interaction was determined based on seven IoU interaction thresholds (0.05, 0.07, 0.08, 0.09, 0.1, 0.3, 0.5). With five models tested, a total of 105 tests were conducted for each combination of image size and IoU threshold for every model.

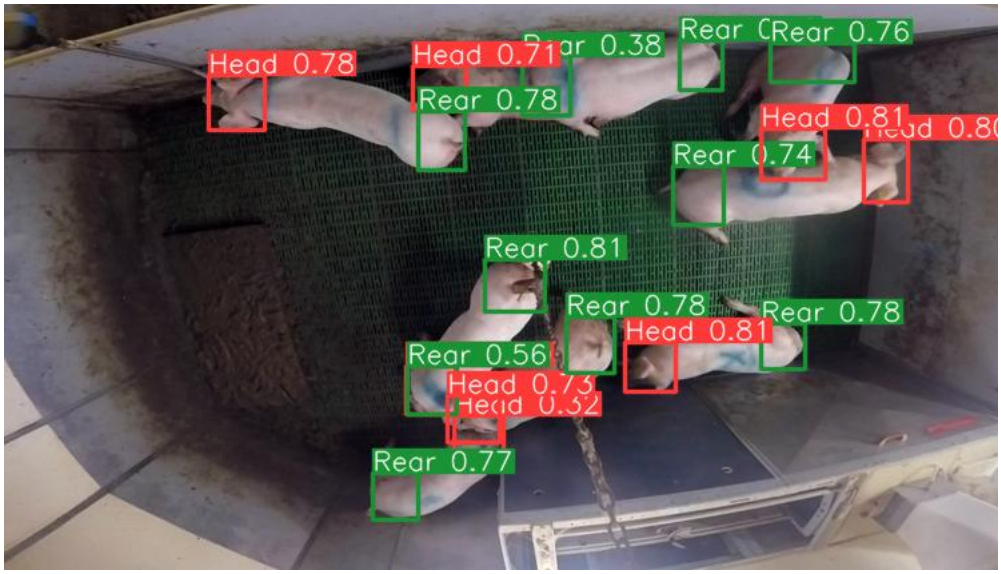


Figure 24: Sample Pig Head and Pig Rear Detection

2. Evaluating Interaction Results

In the Interaction Method Results Table (Table 5), the Model section uses the format model size, image size, batch size, and epochs. These are the models trained from the detection method. On the other hand, the Parameter (Param.) section is formatted to the image size and IoU interaction threshold used for the interaction method. The following abbreviations are also used for the tables: Accuracy (Acc.), Precision (Prec.), Recall (Rec.), and F1-Score (F1).

Best Interaction Method Results [Primary Dataset]						
Model	**Param.**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
s-400-8-50	600-0.07	0.725	0.608	0.787	0.686	0.730
l-400-8-50	400-0.07	0.760	0.647	0.816	0.723	0.764
m-500-8-100	600-0.07	0.755	0.642	0.809	0.716	0.758
m-600-8-100	500-0.05	0.755	0.624	0.901	0.737	0.776
m-600-16-100	600-0.07	0.743	0.623	0.826	0.710	0.751

Table 5: Best Interaction Method Results [Primary Dataset]

After testing the trained models on the interaction dataset, it appears that the overall performance is not particularly strong. Across all parameters, the average values hover around 75%. As anticipated, the small model s-400-8-50 performs the least favorably, with metrics of 72.5% accuracy, 60.8% precision, 78.7% recall, 68.6% F1 score, and 0.73 nMCC. On the other hand, the best-performing model in the interaction method results is the top detection model m-600-8-100 with metrics of 75.5% accuracy, 62.4% precision, 90.1% recall, 73.7% F1 score, and 0.776 nMCC.

Based on observations, the metrics for all models show a general trend when considering different image sizes and IoU thresholds:

- Increase in image sizes tend to improve performance for most models.
- Accuracy and Precision exhibit higher scores within the range of 0.07-0.09 IoU thresholds.
- At an IoU threshold of approximately 0.05, accuracy and precision experience a notable decrease, but recall significantly increases.

Considering the tight results obtained in the interaction method testing, it is important to note that even the worst-performing model is not significantly distant from the best-performing model. The nMCC score was computed to assess the balance of these metrics across models. Based on the nMCC, the model that

stands out as the best is still m-600-8-100 using an image size of 500 and IoU interaction threshold of 0.05 with an nMCC score of 0.776, accompanied by a high recall metric. This outcome was expected, given that this model utilizes a lower threshold of 0.05. Consequently, it follows that as the threshold decreases, the occurrence of false positives tend to increase.

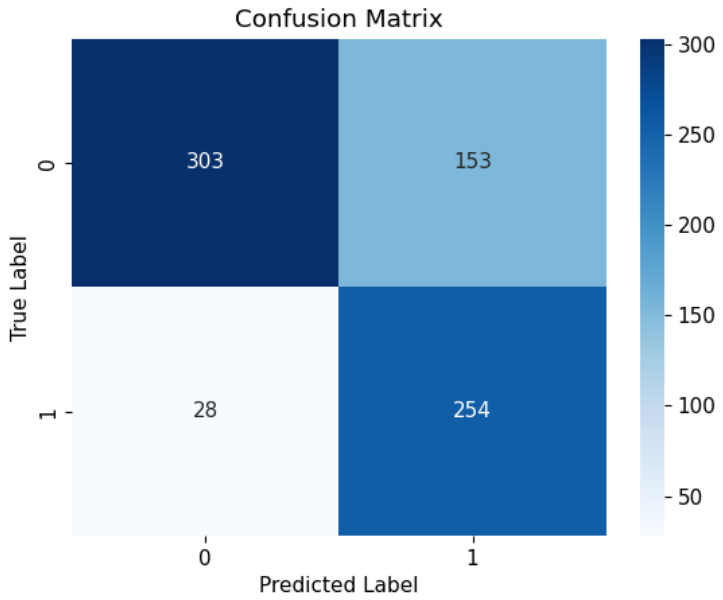


Figure 25: Confusion Matrix of The Best Model [Primary Dataset]

E. Model Improvements

1. Creating Different Datasets

Due to the unsatisfactory performance of the models in the interaction method, a suggestion from [3] was implemented to enhance the results. This involved utilizing 80-20 data splits and incorporating background images to reduce the false positive rates. Hence, the Secondary and Secondary (BG) datasets were created, with the target to improve the performance of the interaction method. The underlying assumption is that the effectiveness of the interaction method is closely tied to the performance of the detection model, as it relies on the bounding boxes generated by the detection model.

2. Evaluating Improved Datasets Results: Detection

Based on the previous test for the interaction method the best models are m-600-8-100 and l-400-8-50. With that, these models configurations were used to train with the Secondary and Secondart (BG) datasets to observe changes in model performance.

Detection Model Results [Secondary Datasets]				
Model	**Dataset**	**Precision**	**Recall**	**mAP@0.5**
m-600-8-100	Secondary	0.935	0.915	0.949
l-400-8-50		0.932	0.889	0.934
m-600-8-100	Secondary (BG)	0.938	0.909	0.947
l-400-8-50		0.930	0.895	0.935

Table 6: Detection Model Results [Secondary Datasets]

After training the models using the new datasets, there were slight improvements observed in the metrics. Comparing the Secondary and Secondary (BG) datasets across the models, no significant differences were evident. Among the models, m-600-8-100 stands out as the top performer for both datasets. However, the m-600-8-100 model without a background dataset holds a slight advantage. As these detection models exhibit relatively high performance, all four models were tested in the interaction method.

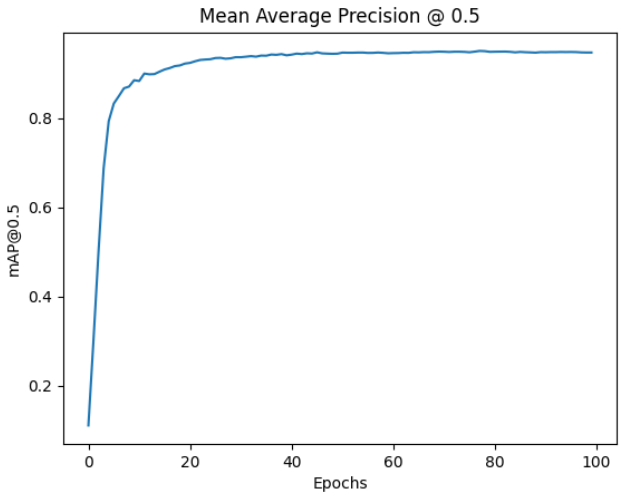


Figure 26: Best Detection Model (m-600-8-100) mAP Curve [Secondary Dataset]

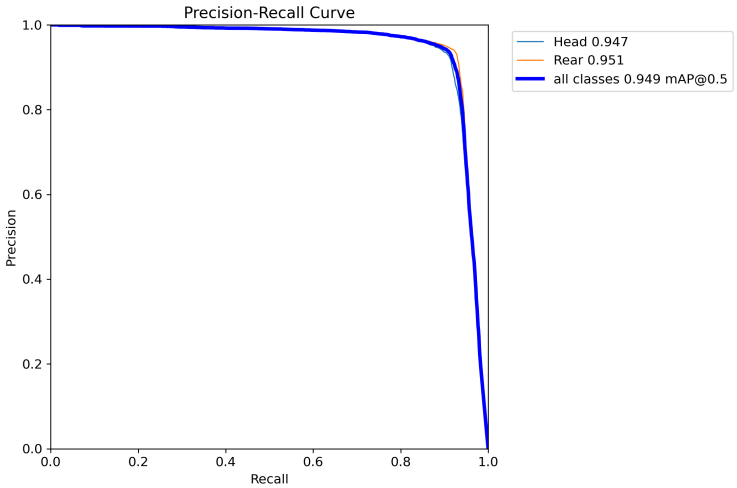


Figure 27: Best Detection Model (m-600-8-100) PR Curve [Secondary Dataset]

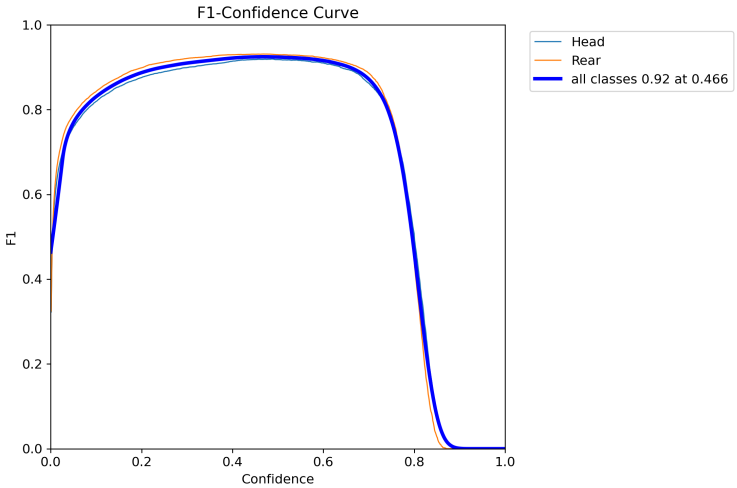


Figure 28: Best Detection Model (m-600-8-100) F1 Curve [Secondary Dataset]

The m-600-8-100 model trained with the Secondary dataset shows a noticeably smoother mAP curve during training. It reached its peak at approximately 0.9 and stabilized faster compared to the model trained with the Primary dataset (see Fig. 21). Additionally, the PR curve exhibits a larger area under the curve, indicating improved precision and recall. The F1 score for this model remains similar to the previous model. Fig. 26, 27, & 28.

3. Evaluating Improved Datasets Results: Interaction

Best Interaction Method Results [Secondary Datasets]						
Model	**Param.**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
m-600-8-100	500-0.07	0.774	0.681	0.766	0.721	0.767
l-400-8-50	600-0.05	0.738	0.621	0.809	0.703	0.745
m-600-8-100 (BG)	600-0.06	0.762	0.645	0.837	0.728	0.768
l-400-8-50 (BG)	500-0.07	0.757	0.647	0.805	0.720	0.769

Table 7: Best Interaction Method Results [Secondary Datasets]

In the interaction results of the models trained with the Secondary datasets, there is a slight improvement in accuracy and precision metrics compared to the models trained with the Primary dataset. However, the recall metric shows significantly lower values for lower interaction thresholds. The nMCC values are comparable to the previous results without significant breakthroughs. Nevertheless, what stands out in this model is the improved balance of metrics, indicating a more balanced performance. Therefore, if there is a need for balanced results, this model may be preferred.

Although the model l-400-8-50 (BG) achieves the highest nMCC score, indicating a stronger overall performance, the m-600-8-100 model exhibits better accuracy, precision, and balance in terms of metrics despite its slightly lower nMCC. Therefore, with the factors of accuracy, precision, and balance considered, the m-600-8-100 model with 500-0.07 parameters served as a more preferable choice.

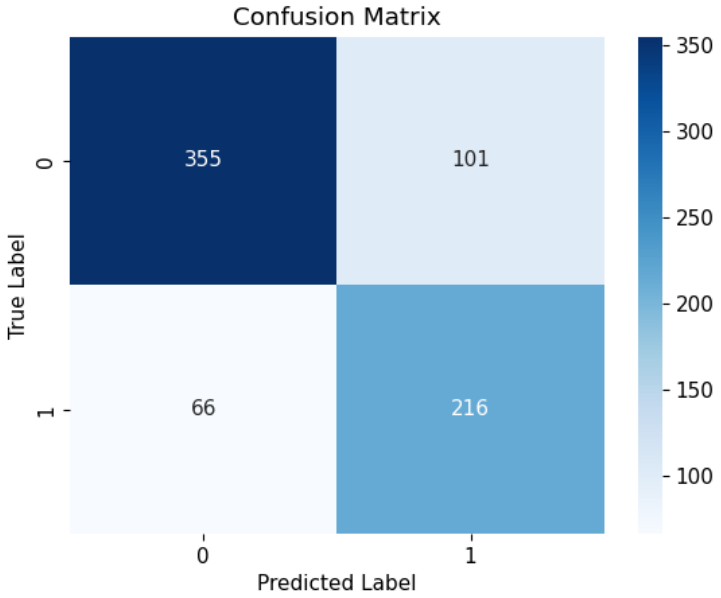


Figure 29: Confusion Matrix of The Best Model [Secondary Dataset]

F. Best End-to-End Models

In conclusion, after training all the detection models and testing in the interaction method, these are the two best-performing models for detecting pig head-to-rear contact in pig pen images. A summary of the performance graphs of these models are in Fig. 30 & Fig. 31. The table and figures provide a concise overview of the key metrics and results achieved by the two best models.

Best End-to-End Models				
Model	**Parameter**	**Dataset**	**nMCC**	**mAP@0.5**
m-600-8-100	500-0.05	Primary	0.776	0.930
m-600-8-100	500-0.07	Secondary	0.767	0.949

Table 8: Best End-to-End Models

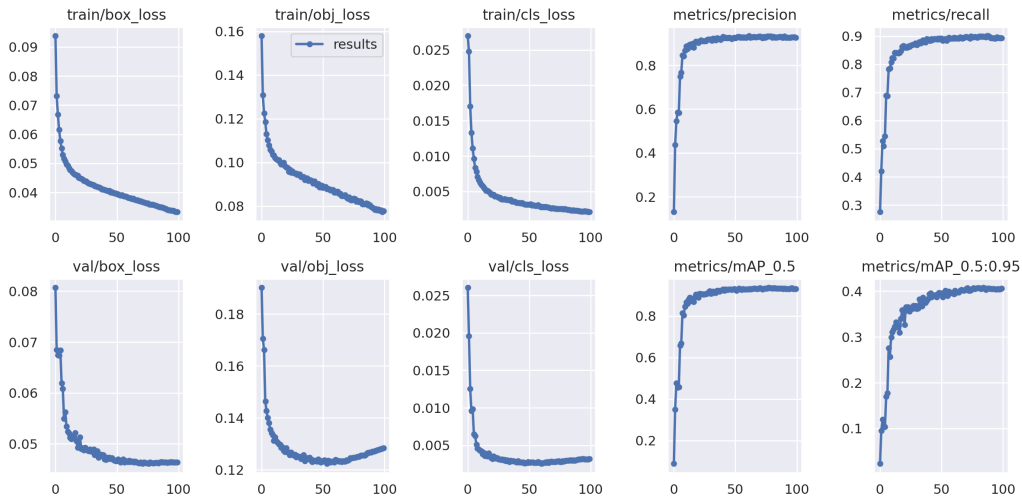


Figure 30: Summary of m-600-8-100 [Primary Dataset]

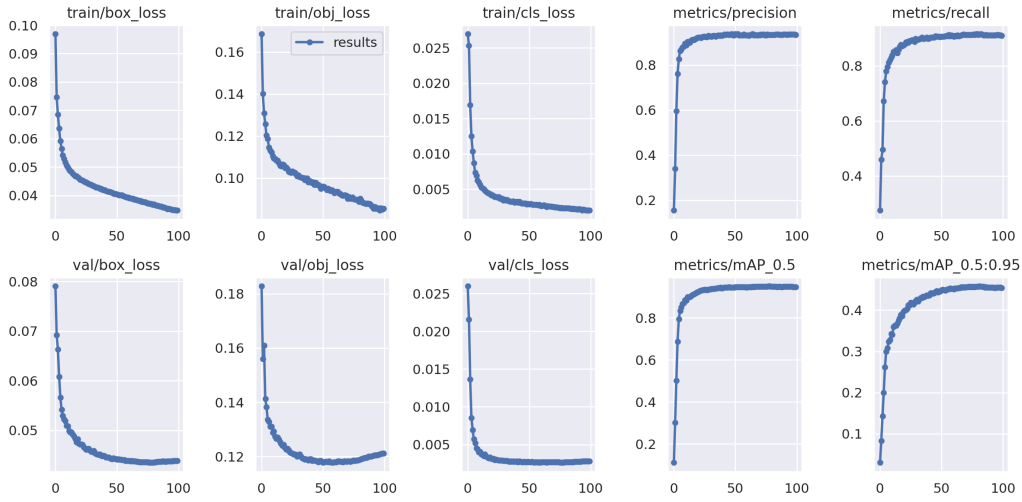


Figure 31: Summary of m-600-8-100 [Secondary Dataset]

G. Comparative Analysis

Detection Method Comparisons		
Model	**Base Network**	**mAP@50**
YOLO	Darknet53	0.840
YOLO	CSPDarknet53	0.850
Configured YOLO	ResNet50	0.863
Configured YOLO	CSPDarknet53	0.874
Configured YOLO [1]	ResNet50	0.863
YOLOv5 m-600-8-100 (Pri)	CSPDarknet53	0.930
YOLOv5 m-600-8-100 (Sec)	CSPDarknet53	0.949

Table 9: Detection Method Comparisons (Source: [1])

The trained detection models exhibit improved performance in detecting pig heads and rears compared to previous models reported in [1]. The average mAP scores for both models are high, reaching approximately 94%. Consistent with the findings in [1], the models demonstrate better detection capabilities for pig rears compared to pig heads, as illustrated in Fig. 22 and 27 where the mAP scores for heads and rears are presented.

Interaction Method Comparisons		
Model	**True Positive**	**True Negative**
Configured YOLO [1]	0.925	0.942
YOLOv5 m-600-8-100 (Pri)	0.901	0.664
YOLOv5 m-600-8-100 (Sec)	0.766	0.779

Table 10: Interaction Method Comparisons

The values presented in this table were derived from the confusion matrices of the interaction results. Specifically, for the Configured YOLO model, the confusion matrix used is the one reported in [1]. When considering the True Positive Rate

or the proportion of correct detections with contact, the first model achieves a commendable score of 90%, which is comparable to the performance reported in previous work. However, the second model falls short with a score of 76.6%. In terms of the True Negative Rate, both models do not demonstrate comparable performance to previous work, as they achieve relatively low scores. In terms of overall balance, the second model exhibits more tightly balanced metrics, while the first model exhibits a significant trade-off with a higher rate of false positives.

H. TailSafe: Website Application

1. Landing Page

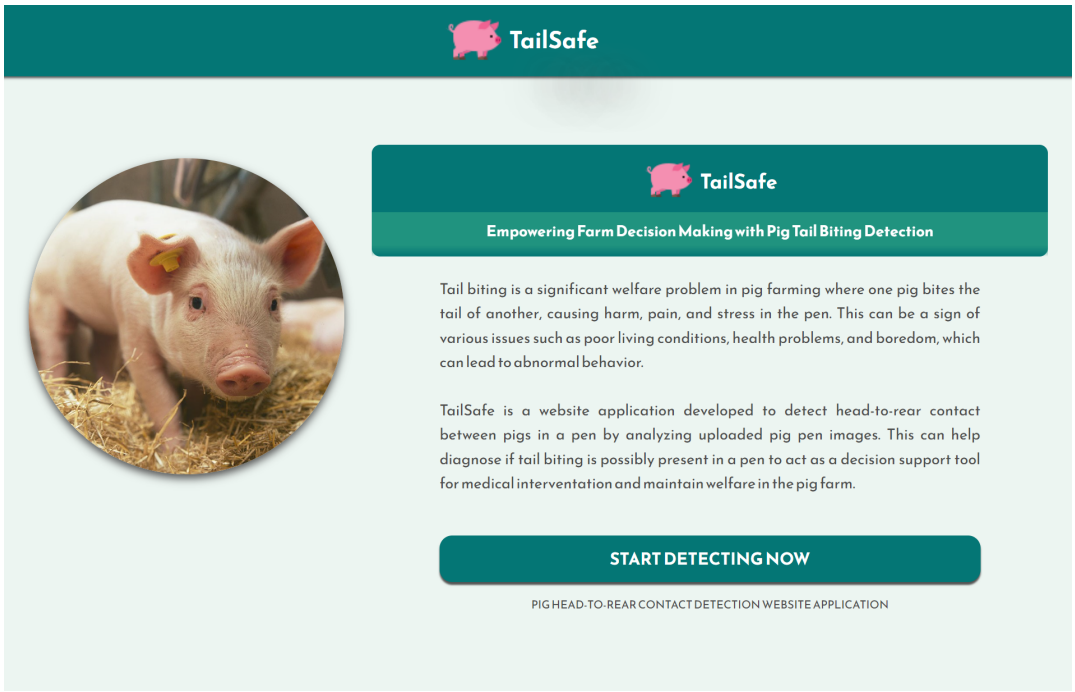


Figure 32: TailSafe Landing Page: Main Section

Upon accessing the website application, the user is welcomed by the landing page (Fig. 32) featuring the title "TailSafe" and an overview of its purpose. The landing page provides valuable information about tail biting and explains the role of TailSafe in addressing this issue. Positioned below is a button labeled "Start Detecting Now," clicking on it seamlessly redirects the user to the image upload section of the page.

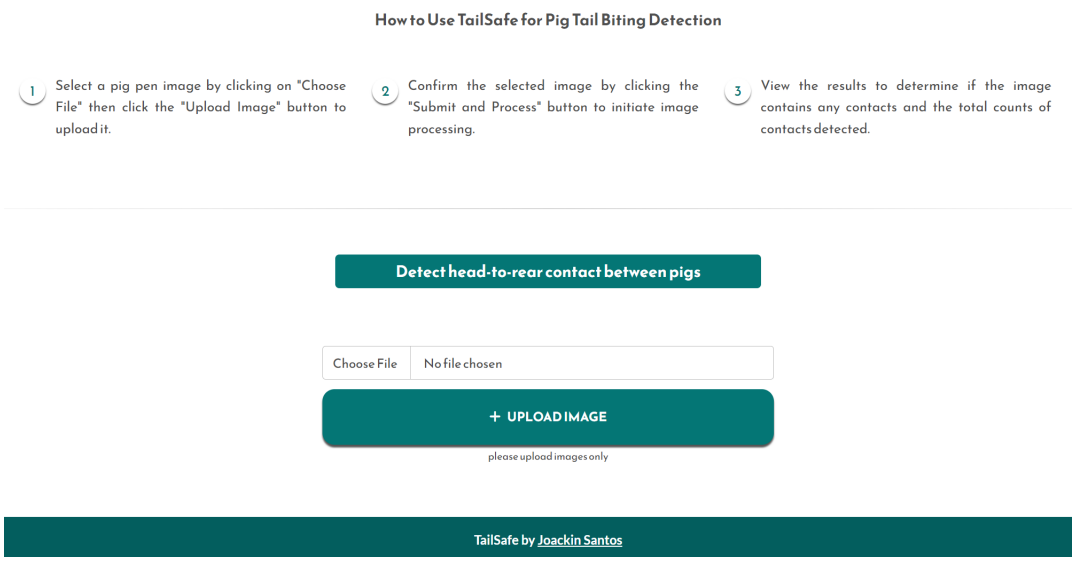


Figure 33: TailSafe Landing Page: Upload Section

Above the image upload section (Fig. 33), a helpful guide is presented for assistance in using the website application for upload of pig pen images and detection of contact presence. Directly below, users are prompted to select an image file.

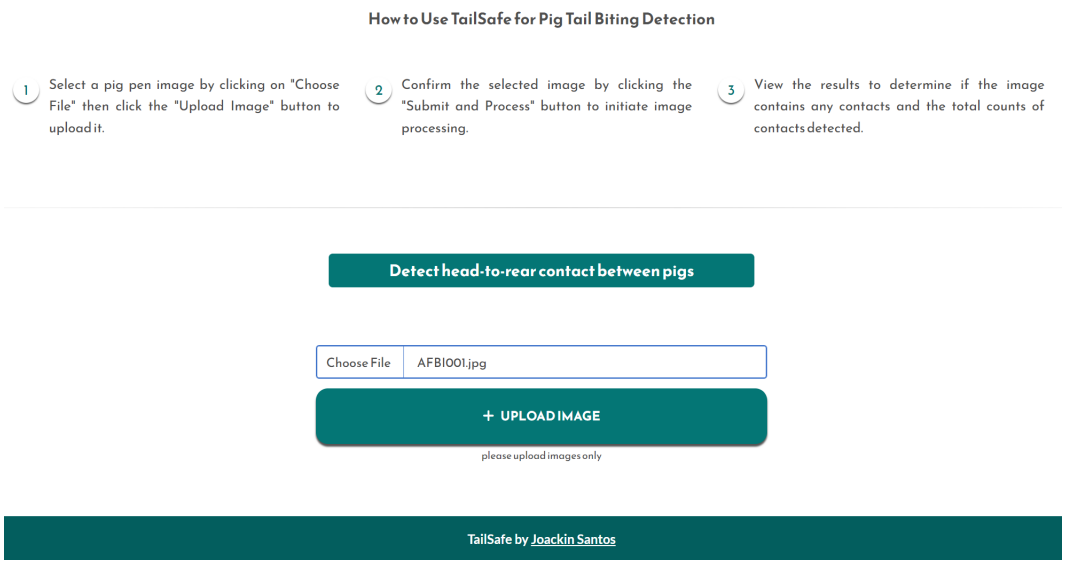


Figure 34: TailSafe Landing Page: Upload Section (after uploading)

Once the user uploads an image, the file name of the image will replace the default text and be displayed in the "Choose File" bar. The upload process is completed once the user clicks the "Upload Image" button, where the image will

be submitted to the system.

2. Image Confirmation Page

After an image is uploaded to the system, the user is redirected to the image confirmation page (Fig. 35). In this page, the user is able to check and confirm if the uploaded image is correct through the displayed image on the page. Upon confirmation, the user clicks the "Submit and Process" button which begins the processing of the image. Alternatively, the user may click the "Back" button to return to the landing page's upload section.

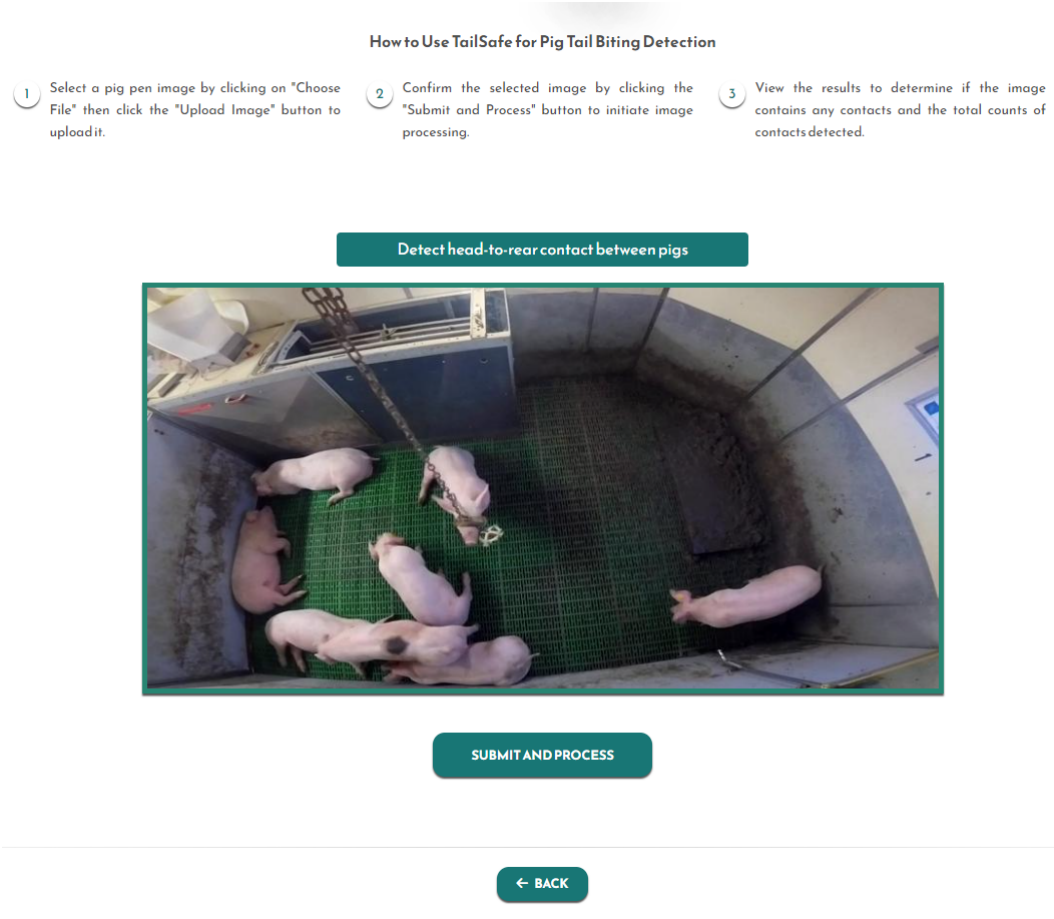


Figure 35: TailSafe Image Confirmation Page

Processing the image includes detection of the pig head and pig rears in the image. Consequently, it computes the IoU of pig head-to-rear bounding boxes, and its interaction classification and count.

3. Results Page

Upon completion of the process, the user is automatically redirected to the results page (Fig. 36 & 37). The results page displays the image indicating in highlighted boxes the pig heads (red box) and the pig rears (green boxes) for easy visualization. Moreover, the page displays the interaction classification, indicating whether the image is with contact or without contact, along with the corresponding contact count. Below this, brief explanations on what the results mean are provided for clarity.

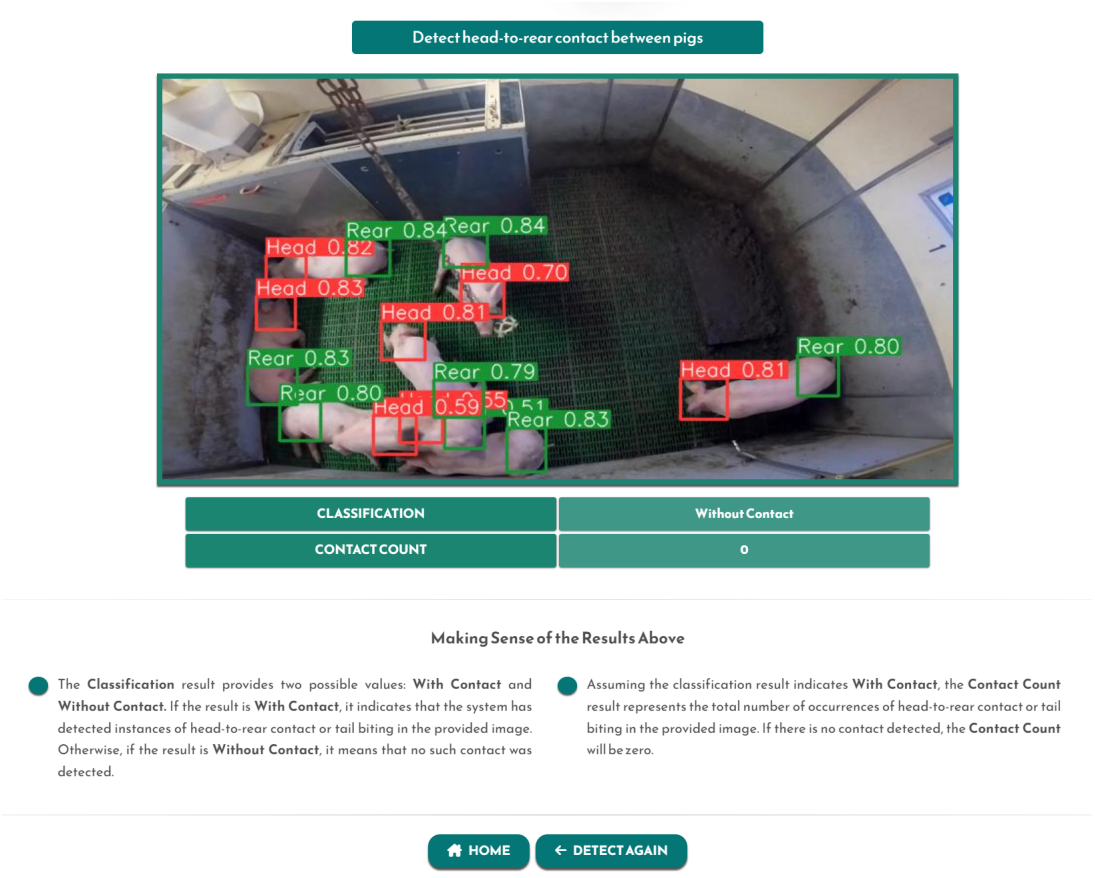


Figure 36: TailSafe Results Page: Without Contact Classification

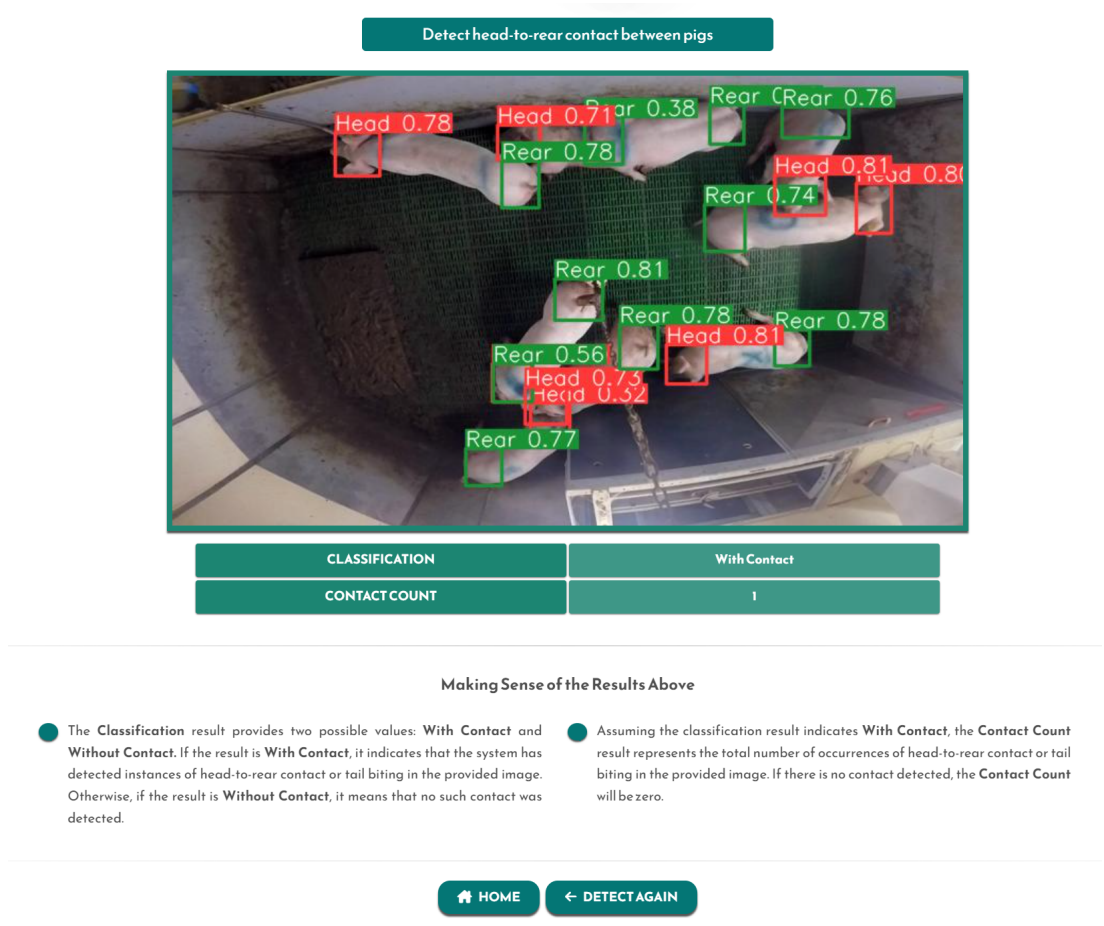


Figure 37: TailSafe Results Page: With Contact Classification

I. Summary of Detection and Interaction Method Results

1. Detection Models using Primary Dataset

Detection Model Results [Primary Dataset] @ 50 Epochs					
Model	**Image**	**Batch**	**Precision**	**Recall**	**mAP@0.5**
m-400-8-50	400	8	0.911	0.870	0.908
m-400-16-50		16	0.909	0.875	0.904
m-400-32-50		32	0.908	0.859	0.897
m-500-8-50	500	8	0.918	0.884	0.917
m-500-16-50		16	0.922	0.882	0.920
m-500-32-50		32	0.917	0.858	0.895
m-600-8-50	600	8	0.921	0.884	0.920
m-600-16-50		16	0.918	0.888	0.923
m-600-32-50		32	0.916	0.804	0.847

Table 11: Detection Model Results [Primary Dataset] @ 50 Epochs

Detection Model Results [Primary Dataset] @ 100 Epochs					
Model	**Image**	**Batch**	**Precision**	**Recall**	**mAP@0.5**
m-400-8-100	400	8	0.918	0.888	0.919
m-400-16-100		16	0.924	0.881	0.921
m-400-32-100		32	0.911	0.805	0.840
m-500-8-100	500	8	0.926	0.895	0.929
m-500-16-100		16	0.911	0.853	0.885
m-500-32-100		32	0.918	0.847	0.876
m-600-8-100	600	8	0.929	0.895	0.93
m-600-16-100		16	0.926	0.892	0.93
m-600-32-100		32	0.921	0.845	0.88

Table 12: Detection Model Results [Primary Dataset] @ 100 Epochs

2. Detection Models using Secondary Datasets

Refer to Table 6, these are the only models trained using the Secondary datasets.

3. Interaction Method Results using Primary Dataset

Interaction s-400-8-50 Results [Primary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.622	0.516	0.174	0.260	0.553
	0.30	0.631	0.541	0.234	0.327	0.573
	0.10	0.694	0.601	0.592	0.596	0.675
	0.09	0.701	0.600	0.649	0.624	0.688
	0.08	0.720	0.615	0.723	0.660	0.713
	0.07	0.714	0.602	0.741	0.665	0.713
	0.05	0.694	0.570	0.805	0.668	0.710
500	0.50	0.648	0.608	0.220	0.323	0.593
	0.30	0.655	0.615	0.255	0.361	0.604
	0.10	0.715	0.624	0.642	0.633	0.700
	0.09	0.717	0.615	0.691	0.651	0.708
	0.08	0.713	0.602	0.734	0.661	0.711
	0.07	0.715	0.600	0.766	0.673	0.719
	0.05	0.706	0.578	0.858	0.690	0.731
600	0.50	0.617	0.495	0.160	0.241	0.543
	0.30	0.623	0.518	0.209	0.298	0.560
	0.10	0.728	0.640	0.656	0.648	0.713
	0.09	0.718	0.616	0.695	0.653	0.710
	0.08	0.724	0.613	0.748	0.674	0.722
	0.07	0.725	0.608	0.787	0.686	0.730
	0.05	0.709	0.581	0.855	0.692	0.732

Table 13: Interaction s-400-8-50 Results [Primary Dataset]

Interaction l-400-8-50 Results [Primary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.629	0.569	0.117	0.194	0.556
	0.30	0.644	0.623	0.170	0.267	0.585
	0.10	0.760	0.691	0.674	0.682	0.745
	0.09	0.768	0.687	0.723	0.705	0.757
	0.08	0.763	0.663	0.773	0.714	0.759
	0.07	0.760	0.647	0.819	0.723	0.764
	0.05	0.738	0.608	0.887	0.722	0.761
500	0.50	0.641	0.635	0.142	0.232	0.579
	0.30	0.664	0.702	0.209	0.322	0.618
	0.10	0.747	0.667	0.674	0.670	0.732
	0.09	0.755	0.664	0.723	0.693	0.745
	0.08	0.751	0.646	0.770	0.702	0.748
	0.07	0.752	0.639	0.809	0.714	0.756
	0.05	0.729	0.601	0.865	0.709	0.749
600	0.50	0.627	0.574	0.096	0.164	0.552
	0.30	0.634	0.597	0.131	0.215	0.567
	0.10	0.749	0.678	0.656	0.667	0.733
	0.09	0.756	0.669	0.716	0.692	0.746
	0.08	0.756	0.659	0.748	0.701	0.750
	0.07	0.749	0.643	0.773	0.702	0.750
	0.05	0.745	0.617	0.879	0.725	0.764

Table 14: Interaction l-400-8-50 Results [Primary Dataset]

Interaction m-500-8-100 Results [Primary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.618	0.500	0.131	0.208	0.541
	0.30	0.634	0.565	0.184	0.278	0.571
	0.10	0.752	0.697	0.621	0.657	0.733
	0.09	0.762	0.695	0.670	0.680	0.746
	0.08	0.763	0.679	0.720	0.699	0.752
	0.07	0.755	0.656	0.752	0.701	0.749
	0.05	0.730	0.605	0.851	0.707	0.747
500	0.50	0.625	0.537	0.128	0.206	0.550
	0.30	0.634	0.570	0.174	0.266	0.570
	0.10	0.748	0.679	0.645	0.662	0.731
	0.09	0.757	0.677	0.699	0.688	0.745
	0.08	0.764	0.674	0.741	0.706	0.756
	0.07	0.762	0.659	0.780	0.714	0.759
	0.05	0.741	0.616	0.855	0.716	0.756
600	0.50	0.614	0.475	0.099	0.164	0.528
	0.30	0.627	0.548	0.142	0.225	0.557
	0.10	0.744	0.669	0.652	0.661	0.728
	0.09	0.745	0.660	0.688	0.674	0.733
	0.08	0.763	0.664	0.770	0.713	0.758
	0.07	0.755	0.642	0.809	0.716	0.758
	0.05	0.744	0.615	0.879	0.724	0.763

Table 15: Interaction m-500-8-100 Results [Primary Dataset]

Interaction m-600-8-100 Results [Primary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.642	0.596	0.199	0.298	0.584
	0.30	0.650	0.607	0.241	0.345	0.598
	0.10	0.753	0.681	0.667	0.674	0.738
	0.09	0.753	0.666	0.713	0.688	0.743
	0.08	0.762	0.661	0.773	0.712	0.758
	0.07	0.759	0.646	0.816	0.721	0.762
	0.05	0.738	0.610	0.876	0.719	0.758
500	0.50	0.648	0.638	0.181	0.282	0.592
	0.30	0.659	0.650	0.230	0.340	0.609
	0.10	0.759	0.684	0.684	0.684	0.745
	0.09	0.768	0.678	0.748	0.712	0.760
	0.08	0.768	0.678	0.748	0.712	0.760
	0.07	0.771	0.658	0.833	0.736	0.775
	0.05	0.755	0.624	0.901	0.737	0.776
600	0.50	0.630	0.562	0.145	0.231	0.561
	0.30	0.634	0.568	0.177	0.270	0.570
	0.10	0.751	0.676	0.667	0.671	0.735
	0.09	0.756	0.667	0.723	0.694	0.746
	0.08	0.749	0.645	0.766	0.700	0.746
	0.07	0.744	0.627	0.812	0.708	0.750
	0.05	0.737	0.608	0.879	0.719	0.758

Table 16: Interaction m-600-8-100 Results [Primary Dataset]

Interaction m-600-16-100 Results [Primary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.626	0.531	0.181	0.270	0.559
	0.30	0.637	0.558	0.238	0.333	0.580
	0.10	0.744	0.670	0.649	0.659	0.727
	0.09	0.760	0.676	0.716	0.695	0.749
	0.08	0.753	0.659	0.734	0.695	0.745
	0.07	0.751	0.644	0.777	0.704	0.749
	0.05	0.714	0.588	0.844	0.693	0.733
500	0.50	0.608	0.456	0.128	0.199	0.526
	0.30	0.626	0.530	0.188	0.277	0.560
	0.10	0.748	0.673	0.663	0.668	0.732
	0.09	0.744	0.653	0.702	0.677	0.733
	0.08	0.740	0.634	0.755	0.689	0.737
	0.07	0.733	0.618	0.787	0.693	0.737
	0.05	0.725	0.596	0.869	0.707	0.747
600	0.50	0.622	0.517	0.163	0.248	0.551
	0.30	0.637	0.565	0.216	0.313	0.578
	0.10	0.732	0.645	0.663	0.654	0.718
	0.09	0.740	0.645	0.709	0.676	0.730
	0.08	0.749	0.641	0.784	0.705	0.749
	0.07	0.743	0.623	0.826	0.710	0.751
	0.05	0.715	0.587	0.858	0.697	0.737

Table 17: Interaction m-600-16-100 Results [Primary Dataset]

4. Interaction Method Results using Secondary Dataset

Interaction m-600-8-100 Results [Secondary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.642	0.650	0.138	0.228	0.582
	0.30	0.656	0.671	0.195	0.302	0.605
	0.10	0.737	0.688	0.571	0.624	0.714
	0.09	0.749	0.682	0.645	0.663	0.732
	0.08	0.747	0.663	0.684	0.674	0.733
	0.07	0.748	0.657	0.713	0.684	0.738
	0.05	0.737	0.617	0.823	0.705	0.746
500	0.50	0.634	0.620	0.110	0.187	0.566
	0.30	0.660	0.718	0.181	0.289	0.613
	0.10	0.751	0.696	0.617	0.654	0.731
	0.09	0.762	0.695	0.670	0.682	0.746
	0.08	0.770	0.690	0.720	0.705	0.758
	0.07	0.774	0.681	0.766	0.721	0.767
	0.05	0.747	0.630	0.816	0.711	0.752
600	0.50	0.619	0.514	0.064	0.114	0.530
	0.30	0.627	0.569	0.103	0.174	0.552
	0.10	0.748	0.689	0.621	0.653	0.729
	0.09	0.748	0.673	0.663	0.668	0.732
	0.08	0.756	0.672	0.706	0.689	0.744
	0.07	0.756	0.658	0.752	0.702	0.75
	0.05	0.757	0.642	0.826	0.722	0.763

Table 18: Interaction m-600-8-100 Results [Secondary Dataset]

Interaction l-400-8-50 Results [Secondary Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.621	0.517	0.106	0.176	0.541
	0.30	0.622	0.522	0.128	0.205	0.546
	0.10	0.728	0.668	0.571	0.616	0.705
	0.09	0.725	0.653	0.599	0.625	0.705
	0.08	0.734	0.654	0.649	0.651	0.718
	0.07	0.738	0.648	0.691	0.669	0.727
	0.05	0.734	0.617	0.801	0.698	0.740
500	0.50	0.627	0.578	0.092	0.159	0.551
	0.30	0.633	0.590	0.128	0.210	0.564
	0.10	0.720	0.664	0.539	0.595	0.694
	0.09	0.741	0.678	0.613	0.644	0.722
	0.08	0.749	0.667	0.688	0.677	0.736
	0.07	0.749	0.654	0.730	0.690	0.741
	0.05	0.733	0.615	0.805	0.697	0.740
600	0.50	0.636	0.618	0.121	0.202	0.569
	0.30	0.646	0.648	0.163	0.261	0.589
	0.10	0.738	0.682	0.592	0.634	0.717
	0.09	0.747	0.679	0.638	0.658	0.729
	0.08	0.743	0.655	0.688	0.671	0.730
	0.07	0.743	0.640	0.745	0.689	0.738
	0.05	0.738	0.621	0.809	0.703	0.745

Table 19: Interaction l-400-8-50 Results [Secondary Dataset]

Interaction m-600-8-100 Results [Secondary (BG) Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.619	0.510	0.092	0.156	0.536
	0.30	0.642	0.622	0.163	0.258	0.582
	0.10	0.744	0.696	0.585	0.636	0.722
	0.09	0.749	0.686	0.635	0.659	0.731
	0.08	0.759	0.673	0.716	0.694	0.748
	0.07	0.763	0.666	0.762	0.711	0.757
	0.05	0.749	0.629	0.840	0.719	0.759
500	0.50	0.634	0.630	0.103	0.177	0.566
	0.30	0.653	0.691	0.167	0.269	0.601
	0.10	0.764	0.713	0.642	0.675	0.746
	0.09	0.763	0.693	0.681	0.687	0.748
	0.08	0.762	0.679	0.713	0.696	0.750
	0.07	0.759	0.660	0.759	0.706	0.753
	0.05	0.753	0.632	0.851	0.725	0.764
600	0.50	0.625	0.558	0.085	0.148	0.545
	0.30	0.637	0.625	0.124	0.207	0.572
	0.10	0.749	0.695	0.613	0.652	0.730
	0.09	0.756	0.686	0.667	0.676	0.740
	0.08	0.762	0.680	0.709	0.694	0.750
	0.07	0.762	0.668	0.748	0.706	0.754
	0.05	0.762	0.645	0.837	0.728	0.768

Table 20: Interaction m-600-8-100 Results [Secondary (BG) Dataset]

Interaction l-400-8-50 Results [Secondary (BG) Dataset]						
Image	**IoU**	**Acc.**	**Prec.**	**Rec.**	**F1**	**nMCC**
400	0.50	0.631	0.632	0.085	0.150	0.560
	0.30	0.646	0.698	0.131	0.221	0.590
	0.10	0.752	0.709	0.596	0.647	0.731
	0.09	0.759	0.697	0.652	0.674	0.742
	0.08	0.767	0.698	0.688	0.693	0.753
	0.07	0.767	0.683	0.727	0.704	0.757
	0.05	0.757	0.643	0.819	0.721	0.762
500	0.50	0.630	0.645	0.071	0.128	0.557
	0.30	0.634	0.643	0.096	0.167	0.566
	0.10	0.760	0.723	0.603	0.658	0.740
	0.09	0.762	0.712	0.631	0.669	0.743
	0.08	0.768	0.703	0.681	0.692	0.753
	0.07	0.778	0.694	0.748	0.720	0.769
	0.05	0.763	0.649	0.826	0.727	0.767
600	0.50	0.629	0.618	0.074	0.133	0.553
	0.30	0.640	0.674	0.110	0.189	0.577
	0.10	0.757	0.729	0.582	0.647	0.736
	0.09	0.762	0.719	0.617	0.664	0.742
	0.08	0.770	0.707	0.677	0.692	0.754
	0.07	0.764	0.682	0.716	0.699	0.753
	0.05	0.757	0.647	0.805	0.717	0.759

Table 21: Interaction l-400-8-50 Results [Secondary (BG) Dataset]

VI. Discussions

A. Discussion of Work

TailSafe, a Pig Head-to-Rear Contact Detection System, is a website application designed to be a decision support tool for farmers in diagnosing issues in pig pens and detecting instances of tail biting. Its primary goal is to evaluate the effectiveness of YOLOv5, a state-of-the-art object detection model, in accurately identifying pig heads and pig rears. This evaluation involves testing various model sizes and adjusting training parameters such as image size, batch size, and epochs. Additionally, TailSafe utilizes the detected pig parts and extracts bounding boxes to calculate IoU scores, enabling the identification of interactions within pig pen images.

A successful method for automated detection of pig heads, pig rears, and interactions within a pig pen has been achieved. The process involved training multiple models to determine the optimal model size and hyperparameters. In total, 24 models were trained for the detection method. Similarly, for the interaction method, extensive testing was conducted to identify the best parameters, including image size for detection and IoU interaction threshold. A total of 9 models were tested, resulting in 189 instances of testing for each parameter combination. Comparisons with previous work revealed notable improvements, particularly in the accurate detection of pig heads and pig rears. Furthermore, the developed models were seamlessly integrated into a website application, allowing users to effortlessly upload single pig pen images and obtain detailed results on contact presence and contact count.

Key Contributions

1. Developed a robust detection model using the YOLOv5 algorithm, capable of accurately detecting pig heads and pig rears in pig pen images.

2. Designed and implemented an interaction method that utilizes the bounding

boxes generated by the detection model. This method computes IoUs and compares it to a calibrated IoU interaction threshold to determine contact presence among pigs within the pen.

3. Conducted comprehensive investigations to assess the performance variations of different YOLOv5 model sizes and training hyperparameters specifically for pig part detection.

4. Explored the impact of image size parameter for detection and IoU interaction thresholds to optimize the identification of contact presence between pigs in pig pen images.

5. Successfully integrated the detection model and interaction method into a user-friendly web-based application. This application serves as a valuable decision support tool for farmers, enabling them to diagnose pig pen-related issues and effectively manage situations such as tail biting outbreaks.

The training and evaluation process resulted in the identification of two distinct models for the contact detection task. The first model, trained on the primary dataset, exhibited higher recall metrics but a notably lower true negative rate. This discrepancy can be attributed to the utilization of the lowest interaction threshold. On the other hand, the second model, trained on the secondary dataset, displayed slightly lower metrics but demonstrated a more balanced performance. This model may prove advantageous in scenarios where balanced performance is desired. Although there were slight variations in model performance across different parameters, the differences were not that substantial. Therefore, alternative approaches for further improvement should be explored such as developing a different interaction method. Regarding the dataset, the data split appeared to enhance the model's performance, while the inclusion of background images did not appear to yield significant improvements.

B. Comparison to Previous Work

Compared to the detection models presented in [1], the developed detection model demonstrates superior performance in detecting pig heads and pig rears. However, when it comes to the interaction method, the results are not as favorable as those reported in previous studies and fall short of expectations. Nevertheless, there are valuable insights for optimizing the detection model, including increasing image sizes and potentially exploring larger model sizes. Similarly, for the interaction method, experimenting with larger image sizes and exploring alternative approaches may lead to improved performance.

C. Issues in Development

During the development of the detection model, the initial setup proved to be relatively straightforward due to the abundance of online resources, such as the notebook tutorials available in [2], which provided guidance for creating and training custom models. Modifying the model backbone was also relatively simple, involving edits to a YAML file, as long as the layers were compatible with YOLOv5. However, one significant challenge encountered was the training process itself, specifically the time and GPU RAM required. Training larger model sizes demanded more time and GPU power, posing a crucial consideration when selecting the appropriate model size, even though larger models were anticipated to enhance detection performance. It is worth noting that this study was conducted using Google Colab, which imposes timeouts for inactivity. Consequently, the researcher had to carefully monitor the training process to ensure it continued uninterrupted, avoiding delays and disconnections.

The interaction computation in the interaction method was relatively lightweight, primarily involving the calculation of IoUs for each pig head and rear combination and comparing them to the specified threshold. However, the detection process using PyTorch proved to be computationally expensive, particularly when using larger model sizes and larger image size parameters. In some cases, the detection

phase would consume around 45GB of GPU RAM, necessitating the utilization of more powerful and costly GPUs available in Google Colab.

VII. Conclusions

The non-augmented dataset obtained from [1], which combined the AUF dataset and AFBI dataset, served as the foundation for this study. To prepare the detection dataset, the images were uploaded to Roboflow for further processing, including splitting and applying various data augmentations. The dataset was successfully augmented through techniques such as rotation, horizontal flip, crop, and brightness adjustments. Finally, the augmented dataset was exported in the YOLOv5 PyTorch format, ready for detection training and evaluation.

To establish a standardized framework for testing and evaluation, a base hyperparameter (400-8-50) was identified for controlling the experiments conducted on different YOLOv5 model sizes: small, medium, and large. Following the training and evaluation process, it was determined that the medium model size exhibited the most favorable performance for the detection task. This choice was based on its close resemblance to the large model size's results, while being more efficient in terms of training time and computational resources. Consequently, the medium model size was designated as the base model.

Subsequently, the hyperparameters were fine-tuned to investigate their impact on the model's performance and identify the optimal combination of image size, batch size, and epochs. The models underwent evaluation using metrics such as mean Average Precision (mAP), Precision-Recall (PR) curve, and F1 curve. Through this evaluation, the best detection model was determined to have the configuration m-600-8-100, signifying the most effective combination of hyperparameters for achieving high performance.

Using the detection models that were developed, the pig heads and pig rears were successfully detected in the interaction dataset, generating bounding boxes around these specific pig parts. The coordinates of these bounding boxes were extracted from the detection results, allowing for the computation of IoU values between various combinations of pig heads and pig rears within the images. Subsequently, the computed IoUs were compared against an IoU interaction threshold,

which was calibrated during the testing phase.

To identify the optimum parameter combination for achieving the most accurate predictions, different image sizes for detection and various IoU interaction thresholds were explored. After thorough evaluation, the best model was found to exhibit superior performance with an image size of 500 and an IoU interaction threshold of 0.05. This specific combination of parameters resulted in the most effective predictions for detecting pig interactions within the dataset.

To enhance the accuracy of interaction predictions, two additional datasets were created based on the recommendations provided in [3]. The first dataset was split in an 80-20 ratio, while the second dataset consisted of an 80-20 split with the inclusion of background images. These datasets underwent the same data processing and training procedures, using the two best models obtained from previous training sessions: m-600-8-100 and m-400-8-50.

By training the models on these new datasets, more balanced metrics were achieved compared to the initial model. Ultimately, the best-performing model among these datasets remained consistent, namely m-600-8-100. This configuration consistently demonstrated superior performance and improved the overall accuracy and balance of the interaction predictions.

The developed methods were successfully integrated into a web-based application called TailSafe. This application provides seamless detection of pig head-to-rear contact and serves as a valuable decision support tool for farmers to diagnose pig pen related issues and tail biting outbreaks. With TailSafe, users can conveniently upload pig pen images, detect contact presence, and access insightful information to aid in managing their farming operations effectively.

VIII. Recommendations

The pig part detection model can be enhanced by improving it precision and recall, consider implementing the follwoing recommendations:

1. Enhance the pig part detection model by training it with larger image sizes. To compensate for the increased training time, consider increasing the batch size to optimize training efficiency.

2. Train for more epochs. Monitor the model's performance during training to prevent overfitting.

3. Augment the existing pig pen dataset or acquire a larger dataset specifically for pig part detection. Aim for a dataset size of over 10,000 images, as recommended for object detection tasks with YOLOv5.

4. Whenever possible, utilize higher resolution images to provide the model with more detailed information during training.

The interaction method has significant potential for improvement, as all metrics can be enhanced across the board. To achieve better performance, consider implementing the following strategies:

1. Increase the image size parameter for the detection part to capture more detailed information.

2. Utilize the balanced detection model trained on the secondary dataset using an 80-20 split for balanced performance.

3. Investigate alternative interaction methods that can leverage the existing detection model for more accurate results.

The website application can be enhanced with various quality-of-life improvements. Consider implementing the following enhancements to provide a better user experience:

1. Implement the functionality to process multiple image uploads simultaneously while optimizing the detection time.

2. Enhance the user experience by enabling the viewing of the original image in the results page, along with the identified contact locations.

3. Provide the option for users to download the processed images directly from the application.

4. Introduce user profiles that allow tracking and analysis of previous detections for enhanced monitoring and insights.

IX. Bibliography

[1] A. Alameer, S. Buijs, N. O'Connell, L. Dalton, M. Larsen, L. Pedersen, and I. Kyriazakis, "Automated detection and quantification of contact behaviour in pigs using deep learning," *Biosystems Engineering*, vol. 224, pp. 118–130, Dec. 2022. Accessed Nov. 18, 2022. doi.org/10.1016/j.biosystemseng.2022.10.002. [ONLINE]. Available: https://www.sciencedirect.com/science/article/pii/S1537511022002240.

[2] G. Jocher, "Yolov5 by ultralytics," 2020. Accessed Nov. 27, 2023. doi.org/10.5281/zenodo.3908559. [ONLINE]. Available: https://github.com/ultralytics/yolov5.

[3] "Tips for best training results." Ultralytics docs. https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/ (accessed on May 27, 2023), 2020.

[4] C. Imane, "Yolo v5 model architecture [explained]." OpenGenus IQ: Computing Expertise Legacy. https://iq.opengenus.org/yolov5/ (accessed on May 27, 2023).

[5] Y. Gomez, A. H. Stygar, I. J. Boumans, E. A. Bokkers, L. J. Pedersen, J. K. Niemi, M. Pastell, X. Manteca, and P. Llonch, "A systematic review on validated precision livestock farming technologies for pig production and its potential to assess animal welfare," *Frontiers in Veterinary Science*, vol. 8, May 2021. Accessed Nov. 18, 2022. doi.org/10.3389/fvets.2021.660565. [ONLINE]. Available: https://www.frontiersin.org/articles/10.3389/fvets.2021.660565/full.

[6] L. Bergamini, S. Pini, A. Simoni, R. Vezzani, S. Calderara, R. D'Eath, and R. Fisher, "Extracting accurate long-term behavior changes from a large pig dataset," *Proceedings of the 16th International Joint Conference on Computer Vision, Imaging and Com-*

puter Graphics Theory and Applications, Feb. 2021. Accessed Nov. 18, 2022. dx.doi.org/10.5220/0010288405240533. [ONLINE]. Available: https://www.researchgate.net/publication/349382558_Extracting_Accurate_Long-term_Behavior_Changes_from_a_Large_Pig_Dataset.

[7] A. Bhujel, E. Arulmozhi, B.-E. Moon, and H.-T. Kim, "Deep-learning-based automatic monitoring of pigs' physico-temporal activities at different greenhouse gas concentrations," *Animals*, vol. 11, p. 3089, Oct. 2021. Accessed Nov. 18, 2022. doi.org/10.3390/ani11113089. [ONLINE]. Available: https://www.mdpi.com/2076-2615/11/11/3089.

[8] "Tail biting." Ontario Ministry of Agriculture, Food and Rural Affairs Organization. http://omafra.gov.on.ca/english/livestock/swine/news/mayjun12a2.htm (accessed Dec. 10, 2022), 2022.

[9] Y. J. Kim, M. H. Song, S. I. Lee, J. H. Lee, H. J. Oh, J. W. An, S. Y. Chang, Y. B. Go, B. J. Park, M. S. Jo, C. G. Lee, H. B. Kim, and J. H. Cho, "Evaluation of pig behavior changes related to temperature, relative humidity, volatile organic compounds, and illuminance," *Journal of Animal Science and Technology*, vol. 63, pp. 790–798, July 2021. Accessed Nov. 18, 2022. doi.org/10.5187/jast.2021.e89. [ONLINE]. Available: https://www.ejast.org/archive/view_article?pid=jast-63-4-790.

[10] L. E. van der Zande, O. Guzhva, and B. T. Rodenburg, "Individual detection and tracking of group housed pigs in their home pen using computer vision," *Frontiers in Animal Science*, vol. 2, Apr. 2021. Accessed Nov. 18, 2022. doi.org/10.3389/fanim.2021.669312. [ONLINE]. Available: https://www.frontiersin.org/articles/10.3389/fanim.2021.669312/full.

[11] S. Wang, H. Jiang, Y. Qiao, S. Jiang, H. Lin, and Q. Sun, "The research progress of vision-based artificial intelligence in smart pig farming," *Sensors*, vol. 22, p. 6541, Aug. 2022. Accessed Nov. 18, 2022.

doi.org/10.3390/s22176541. [ONLINE]. Available: https://www.mdpi.com/1424-8220/22/17/6541.

[12] H. Shao, J. Pu, and J. Mu, "Pig-posture recognition based on computer vision: Dataset and exploration," *Animals*, vol. 11, p. 1295, Apr. 2021. Accessed Nov. 18, 2022. doi.org/10.3390/ani11051295. [ONLINE]. Available: https://www.mdpi.com/2076-2615/11/5/1295.

[13] Y. Luo, Z. Zeng, H. Lu, and E. Lv, "Posture detection of individual pigs based on lightweight convolution neural networks and efficient channel-wise attention," *Sensors*, vol. 21, p. 8369, Dec. 2021. Accessed Nov. 18, 2022. doi.org/10.3390/s21248369. [ONLINE]. Available: https://www.mdpi.com/1424-8220/21/24/8369.

[14] L. Zhang, H. Gray, X. Ye, L. Collins, and N. Allinson, "Automatic individual pig detection and tracking in pig farms," *Sensors*, vol. 19, p. 1188, Mar. 2019. Accessed Nov. 18, 2022. doi.org/10.3390/s19051188. [ONLINE]. Available: https://www.mdpi.com/1424-8220/19/5/1188.

[15] "What are convolutional neural networks?." IBM Website. https://www.ibm.com/topics/convolutional-neural-networks (accessed on Dec. 10, 2022), 2022.

[16] G. Karimi, "Introduction to yolo algorithm for object detection." Section Website. https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection/ (accessed on Jan. 21, 2023), 2021.

[17] E. Zvornicanin, "What is yolo algorithm?." Baeldung Website. https://www.baeldung.com/cs/yolo-algorithm#:~:text=YOLO%20algorithm%20aims%20to%20predict,Width%20of%20the%20box%20(%20) (accessed on Jan. 21, 2023), 2022.

[18] R. Kanjee, "Yolov5 controversy — is yolov5 real?." Medium Website. https://medium.com/augmented-startups/yolov5-controversy-is-yolov5-real-20e048bebb08 (accessed on May 27, 2023), 2020.

[19] M. R. Munawar and M. Z. Hussain, "Yolor or yolov5 (which one is better)?." Medium Website. https://medium.com/augmented-startups/yolor-or-yolov5-which-one-is-better-2f844d35e1a1#:~:text=YOLOv5%20provides%20better%20FPS%20than,more%20popular%20compared%20to%20YOLOR. (accessed on Jan. 21, 2023), 2022.

[20] "ultralytics/yolov5." Zenodo.org. https://zenodo.org/search?page=1&size=20&q=conceptrecid:"3908559"&sort=-version&all_versions=True (accessed on May 27, 2023).

[21] "Roboflow website." Used for splitting the dataset, converting data annotations to YOLOv5 PyTorch format, and conducting data augmentations on the dataset . https://app.roboflow.com (accessed on May 27, 2023).

[22] A. Suresh, "What is a confusion matrix?." Medium Website. https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5#:~:text=A%20Confusion%20matrix%20is%20an,by%20the%20machine%20learning%20model. (accessed on Jan. 21, 2023), 2020.

X. Appendix

A. Google Colab Notebooks

Pig Contact Detection System Using Convolutional Neural Networks

For Detection, it uses YOLOv5 with CSPDarknet as the backbone
For Interaction, it compares the IoU between the head and rear boxes and a calibrated interaction threshold

Notebook by Joackin Santos

Interaction Notebook
https://colab.research.google.com/drive/1CI2BvEgGtwSc3Kg90TBE88AbD4fOZQgm?usp=sharing

Github Repository for this Notebook and the Web Application for Pig Contact Detection.
Link below:
https://github.com/joackinsantos/Pig-Contact-Detection-Web-App

Github Repository for the modified YOLOv5.
Link below:
https://github.com/joackinsantos/YOLOv5-Modification

Github Repository for the pig datasets.
Link below:
https://github.com/joackinsantos/pig-datasets

Definition of Terms

- Detection : of Head and Rear (Tail)
- Contact : between Head and Rear (Tail)

Resource

https://github.com/ultralytics/yolov5
https://roboflow.com/model/yolov5
https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov5-object-detection-on-custom-data.ipynb#scrollTo=t14hhyqdmw6O
https://docs.ultralytics.com/yolov5/tutorials/tips_for_best_training_results/

Data Preprocessing, Annotations and Augmentations

Data Preprocessing was not necessary as the dataset was provided and preprocessed from:
https://doi.org/10.1016/j.biosystemseng.2022.10.002

Dataset Information

- 2781 total pig pen images (1556 AFBI + 1225 AUF)

ROBOFLOW https://app.roboflow.com/pigdataset/pig-parts/6
Annotations were included in the dataset in the COCO format. The images were uploaded in Roboflow to apply annotations and convert to YOLOv5 PyTorch annotations. Additionally, the specified data augmentation techniques were applied to the training set pre-export:

Augmentation	Specification
Flip	Horizontal
Rotate	90°
Rotate	-35°, +35°
Crop(Scaling)	0% min, 50% max
Brightness	-20%, +20%

Dataset is exported after from Roboflow and uploaded to Google Drive. The dataset is split to (train,validation, and test) 70%:15%:15%.

DATASETS *small datasets are fast training and testing of models*

- Augmented
  - small-pig-dataset
  - mid-pig-dataset
  - big-pig-dataset (set from previous work, not in correct format)
- Non-augmented
  - o-small-pig-dataset
  - o-mid-pig-dataset

new datasets, same augmentations but a few new tweaks to improve detection

- midN-pig-dataset (uses 70%:30%, training & validation spit)
- midNBG-pig-dataset (same split but with background images)

Resource for Background Images:
removed pigs from images https://cleanup.pictures/

Dataset Setup

```
# from google.colab import drive
# drive.mount('/content/drive')

# # data path for annotated datasets using coco
# aug_big_ds_path = '/content/drive/MyDrive/Special-Problem-2023/pig-datasets/augmented/big-p
```

```
# # data paths for annotated datasets from roboflow using PyTorch
# aug_small_ds_path = '/content/drive/MyDrive/Special-Problem-2023/pig-datasets/augmented/sma
# naug_small_ds_path = '/content/drive/MyDrive/Special-Problem-2023/pig-datasets/non-augmente
# aug_mid_ds_path = '/content/drive/MyDrive/Special-Problem-2023/pig-datasets/augmented/mid-p
# naug_mid_ds_path = '/content/drive/MyDrive/Special-Problem-2023/pig-datasets/non-augmented/
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
# clone repository containing yolov5 and datasets
# rename pig-datasets repo to datasets for yolov5
%cd /content
!git clone https://github.com/joackinsantos/YOLOv5-Modification
!git clone https://github.com/joackinsantos/pig-datasets datasets
```

```
/content
Cloning into 'YOLOv5-Modification'...
remote: Enumerating objects: 9888, done.
remote: Counting objects: 100% (80/80), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 9888 (delta 39), reused 69 (delta 34), pack-reused 9808
Receiving objects: 100% (9888/9888), 242.67 MiB | 32.49 MiB/s, done.
Resolving deltas: 100% (123/123), done.
Cloning into 'datasets'...
remote: Enumerating objects: 54055, done.
remote: Counting objects: 100% (9623/9623), done.
remote: Compressing objects: 100% (9616/9616), done.
remote: Total 54055 (delta 7), reused 9623 (delta 7), pack-reused 44432
Receiving objects: 100% (54055/54055), 1.21 GiB | 43.30 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Updating files: 100% (59989/59989), done.
```

```
# data path for annotated dataset using coco
original_data_path = '/content/datasets/big-pig-dataset'

# data path for annotated datasets from roboflow using PyTorch
aug_small_ds_path = '/content/datasets/small-pig-dataset'
naug_small_ds_path = '/content/datasets/o-small-pig-dataset'
aug_mid_ds_path = '/content/datasets/mid-pig-dataset'
naug_mid_ds_path = '/content/datasets/o-mid-pig-dataset'
aug_midN_ds_path = '/content/datasets/midN-pig-dataset'
aug_midNBG_ds_path = '/content/datasets/midNBG-pig-dataset'
```

```
%ls
```

```
datasets/  sample_data/  YOLOv5-Modification/
```

```
# for folder deletions
import shutil
```

```
shutil.rmtree('/content/datasets')
```

Model Setup

```
# clone YOLOv5 repository
%cd YOLOv5-Modification
!pip install -r requirements.txt  # install dependencies (ignore errors)
```

```
/content/YOLOv5-Modification
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/publ
Collecting gitpython>=3.1.30 (from -r requirements.txt (line 5))
  Downloading GitPython-3.1.31-py3-none-any.whl (184 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 184.3/184.3 kB 4.9 MB/s eta 0:00:00
Requirement already satisfied: matplotlib>=3.3 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: opencv-python>=4.1.1 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from -
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (
Collecting thop>=0.1.1 (from -r requirements.txt (line 14))
  Downloading thop-0.1.1.post2209072238-py3-none-any.whl (15 kB)
Requirement already satisfied: torch>=1.7.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: torchvision>=0.8.1 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: setuptools>=65.5.1 in /usr/local/lib/python3.10/dist-pack
Collecting gitdb<5,>=4.0.1 (from gitpython>=3.1.30->-r requirements.txt (line 5))
  Downloading gitdb-4.0.10-py3-none-any.whl (62 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.7/62.7 kB 7.7 MB/s eta 0:00:00
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packag
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-pa
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-p
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-pack
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/di
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from to
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from t
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from tr
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from trit
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (
```

```
Collecting smmap<6,>=3.0.1 (from gitdb<5,>=4.0.1->gitpython>=3.1.30->-r requirements.txt
  Downloading smmap-5.0.0-py3-none-any.whl (24 kB)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-package
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (
Installing collected packages: smmap, gitdb, gitpython, thop
Successfully installed gitdb-4.0.10 gitpython-3.1.31 smmap-5.0.0 thop-0.1.1.post2209072
```

```
# pytorch setup
import torch

from IPython.display import Image, clear_output  # to display images
from utils.downloads import attempt_download  # to download models/datasets

# clear_output()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_propert
```

```
Setup complete. Using torch 2.0.1+cu118 _CudaDeviceProperties(name='Tesla T4', major=7,
```

Displaying Dataset Formats

```
# augmented/{small,mid,big}-pig-dataset
%cat {aug_small_ds_path}/data.yaml
```

```
path: ../datasets/small-pig-dataset
train: ../train/images
val: ../valid/images
test: ../test/images

nc: 2
names: ['Head', 'Rear']

roboflow:
  workspace: pigdataset
  project: pig-parts
  version: 1
  license: CC BY 4.0
  url: https://universe.roboflow.com/pigdataset/pig-parts/dataset/1
```

```
# non-augmented/{small,mid}-pig-dataset
%cat {naug_small_ds_path}/data.yaml
```

```
path: ../datasets/o-small-pig-dataset
train: ../train/images
val: ../valid/images
test: ../test/images
```

```
nc: 2
names: ['Head', 'Rear']

roboflow:
  workspace: pigdataset
  project: small-pigset-noaugment
  version: 1
  license: CC BY 4.0
  url: https://universe.roboflow.com/pigdataset/small-pigset-noaugment/dataset/1
```

roboflow urls are no longer accessible

Defining Model Configuration and Architecture

```
# default yolov5m architecture
%cat /content/YOLOv5-Modification/models/custom_yolov5m.yaml
```

```
# YOLOv5 🚀 by Ultralytics, AGPL-3.0 license

# Parameters
nc: 2  # number of classes
depth_multiple: 0.67  # model depth multiple
width_multiple: 0.75  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]],  # 9
  ]

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
```

```
           [-1, 1, nn.Upsample, [None, 2, 'nearest']],
           [[-1, 4], 1, Concat, [1]],  # cat backbone P3
           [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

           [-1, 1, Conv, [256, 3, 2]],
           [[-1, 14], 1, Concat, [1]],  # cat head P4
           [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

           [-1, 1, Conv, [512, 3, 2]],
           [[-1, 10], 1, Concat, [1]],  # cat head P5
           [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

           [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
          ]
```

This section is for configuring your own yaml files

```
# get classes {num_classes} from yaml
import yaml
with open(aug_small_ds_path + "/data.yaml", 'r') as stream:
    num_classes = str(yaml.safe_load(stream)['nc'])


#customize iPython writefile so we can write variables
from IPython.core.magic import register_line_cell_magic

@register_line_cell_magic
def writetemplate(line, cell):
    with open(line, 'w') as f:
        f.write(cell.format(**globals()))
```

here we can configure our own architecture

```
%%writetemplate /content/YOLOv5-Modification/models/custom_yolov5l.yaml

# Parameters
nc: {num_classes}  # number of classes
depth_multiple: 1.0  # model depth multiple
width_multiple: 1.0  # layer channel multiple
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 v6.0 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Conv, [64, 6, 2, 2]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
```

```
   [-1, 3, C3, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 6, C3, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, C3, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 3, C3, [1024]],
   [-1, 1, SPPF, [1024, 5]],  # 9
  ]

# YOLOv5 v6.0 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, C3, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, C3, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, C3, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, C3, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

Directory for Created Architectures

always 2 classes

backbone	filename
YOLOv5s-CSPDarkNet	custom_yolov5s.yaml
YOLOv5m-CSPDarkNet	custom_yolov5m.yaml
YOLOv5l-CSPDarkNet	ccustom_yolov5l.yaml

Custom Training using YOLOv5

Hyperparameter Tuning

These are the expected parameters and its arguments to be tuned:

parameter	argument
epoch	50, 100
pixel size	400, 500, 600
batch size	8, 16, 32

train.py parameters:

- **img:** define input image size
- **batch:** determine batch size
- **epochs:** define the number of training epochs. (Note: often, 3000+ are common here!)
- **data:** set the path to our yaml file
- **cfg:** specify our model configuration
- **weights:** specify a custom path to weights. (Note: you can download weights from the Ultralytics Google Drive folder)
- **name:** result names
- **nosave:** only save the final checkpoint
- **cache:** cache images for faster training

```
img_size = 600
batch_size = 8
epochs = 100


# train yolov5s on custom data for 100 epochs
# time its performance
# result file name {aug}_{data size}_{yolo size}_{img size}_{batch size}_{epochs}

%%time
%cd /content/YOLOv5-Modification/
!python train.py --img {img_size} --batch {batch_size} --epochs {epochs} --data {aug_midNBG_d
```

```
/content/YOLOv5-Modification
train: weights=, cfg=./models/custom_yolov5m.yaml, data=/content/datasets/midNBG-pig
warning: no common commits
remote: Enumerating objects: 15705, done.
remote: Counting objects: 100% (33/33), done.
remote: Compressing objects: 100% (27/27), done.
remote: Total 15705 (delta 9), reused 23 (delta 6), pack-reused 15672
Receiving objects: 100% (15705/15705), 14.50 MiB | 25.38 MiB/s, done.
Resolving deltas: 100% (10754/10754), done.
From https://github.com/ultralytics/yolov5
 * [new branch]      add/weights_dir   -> ultralytics/add/weights_dir
 * [new branch]      benchmarks        -> ultralytics/benchmarks
 * [new branch]      exp/scaleFill     -> ultralytics/exp/scaleFill
 * [new branch]      exp12             -> ultralytics/exp12
 * [new branch]      exp13             -> ultralytics/exp13
 * [new branch]      exp13-nosoft      -> ultralytics/exp13-nosoft
 * [new branch]      exp13-soft        -> ultralytics/exp13-soft
```

```
 * [new branch]      fix/rgb_albumentations -> ultralytics/fix/rgb_albumentations
 * [new branch]      ghost              -> ultralytics/ghost
 * [new branch]      master             -> ultralytics/master
 * [new branch]      snyk-fix-09d8acaff69195c641ab1e77119e4c6b -> ultralytics/snyk-f
 * [new branch]      snyk-fix-6c9eeb3b0a7596bd3087093a737a3c49 -> ultralytics/snyk-f
 * [new branch]      study_activations -> ultralytics/study_activations
 * [new branch]      ultralytics/HUB   -> ultralytics/ultralytics/HUB
 * [new branch]      update/cls-album  -> ultralytics/update/cls-album
 * [new branch]      update/textlogger -> ultralytics/update/textlogger
 * [new branch]      update/threaded   -> ultralytics/update/threaded
 * [new tag]           v1.0              -> v1.0
 * [new tag]           v2.0              -> v2.0
 * [new tag]           v3.0              -> v3.0
 * [new tag]           v3.1              -> v3.1
 * [new tag]           v4.0              -> v4.0
 * [new tag]           v5.0              -> v5.0
 * [new tag]           v6.0              -> v6.0
 * [new tag]           v6.1              -> v6.1
 * [new tag]           v6.2              -> v6.2
 * [new tag]           v7.0              -> v7.0
github: ⚠️ YOLOv5 is out of date by 2656 commits. Use 'git pull ultralytics master'
requirements: /content/requirements.txt not found, check failed.
YOLOv5 🚀 12625aa8 Python-3.10.11 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

hyperparameters: lr0=0.01, lrf=0.01, momentum=0.937, weight_decay=0.0005, warmup_epo
ClearML: run 'pip install clearml' to automatically track, visualize and remotely tr
Comet: run 'pip install comet_ml' to automatically track and visualize YOLOv5 🚀 rur
TensorBoard: Start with 'tensorboard --logdir runs/train', view at http://localhost:
Downloading https://ultralytics.com/assets/Arial.ttf to /root/.config/Ultralytics/Ar
100% 755k/755k [00:00<00:00, 18.4MB/s]

                from  n    params  module                                argument
  0               -1  1      5280  models.common.Conv                    [3, 48,
  1               -1  1     41664  models.common.Conv                    [48, 96,
  2               -1  2     65280  models.common.C3                      [96, 96,
  3               -1  1    166272  models.common.Conv                    [96, 192
  4               -1  4    444672  models.common.C3                      [192, 19
  5               -1  1    664320  models.common.Conv                    [192, 38
  6               -1  6   2512896  models.common.C3                      [384, 38
  7               -1  1   2655744  models.common.Conv                    [384, 76
```

Detection Model Evaluation

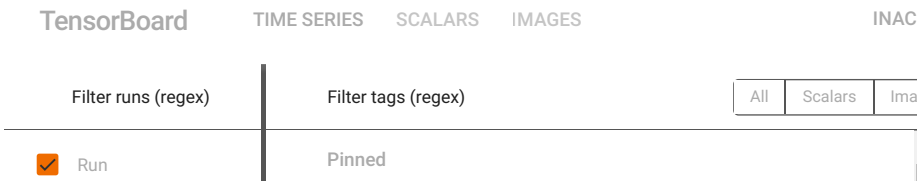
```
! python val.py  --img {img_size} --batch-size {batch_size} --weights /content/YOLOv5-Modific

val: data=/content/datasets/midNBG-pig-dataset/data.yaml, weights=['/content/YOLOv5-Modi
requirements: /content/requirements.txt not found, check failed.
YOLOv5 🚀 12625aa8 Python-3.10.11 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
custom_YOLOv5m summary: 212 layers, 20856975 parameters, 0 gradients, 47.9 GFLOPs
WARNING ⚠️ --img-size 600 must be multiple of max stride 32, updating to 608
```

```
val: Scanning /content/datasets/midNBG-pig-dataset/valid/labels.cache... 610 images, 55
                 Class     Images  Instances          P          R      mAP50   mAP50-95
                   all        610      10676      0.938      0.909      0.947      0.458
                  Head        610       5281      0.936      0.904      0.945      0.462
                  Rear        610       5395       0.94      0.914      0.949      0.454
Speed: 0.1ms pre-process, 9.5ms inference, 4.0ms NMS per image at shape (8, 3, 608, 608)
Results saved to runs/val/exp
```

```
# Start tensorboard
# Launch after you have started training
# logs save in the folder "runs"
%load_ext tensorboard
# %reload_ext tensorboard
%tensorboard --logdir runs
```



Running Inference with Trained Weights


```
%ls
```

Testing

```
# this is for testing
!python detect.py --weights ./runs/train/aug_midNBG_m_{img_size}_{batch_size}_{epochs}/weight
```

```
detect: weights=['./runs/train/aug_midNBG_m_600_8_100/weights/best.pt'], source=../d
requirements: /content/requirements.txt not found, check failed.
YOLOv5 🚀 12625aa8 Python-3.10.11 torch-2.0.1+cu118 CUDA:0 (Tesla T4, 15102MiB)

Fusing layers...
custom_YOLOv5m summary: 212 layers, 20856975 parameters, 0 gradients, 47.9 GFLOPs
WARNING ⚠️ --img-size [600, 600] must be multiple of max stride 32, updating to [608
image 1/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 2/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 3/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 4/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 5/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 6/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 7/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 8/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 9/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_-
image 10/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 11/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 12/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 13/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 14/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 15/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 16/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 17/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 18/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 19/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 20/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 21/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 22/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 23/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 24/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 25/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 26/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
```

```
image 27/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 28/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 29/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 30/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 31/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 32/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 33/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 34/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 35/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 36/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 37/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 38/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 39/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 40/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 41/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 42/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 43/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 44/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 45/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 46/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 47/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 48/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 49/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
image 50/610 /content/datasets/midNBG-pig-dataset/valid/images/2016_02_07__07_15_47_
```

Display Inferences

```
import glob
from IPython.display import Image, display

for imageName in glob.glob('/content/YOLOv5-Modification/runs/detect/exp/*.jpg'): #assuming J
    display(Image(filename=imageName))
    print("\n")
```

Collecting training results, testing results

- download the folder for training results
- download the folder for detect results

```
!zip -r /content/train.zip /content/YOLOv5-Modification/runs/train/aug_midNBG_m_{img_size}_{b
```

```
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/ (stored 0%)
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/val_batch1_pred
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/opt.yaml (deflat
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/weights/ (storec
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/weights/best.pt
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/weights/last.pt
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/val_batch2_label
  adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/F1_curve.png (de
```

```
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/train_batch1.jpg
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/results.csv (def
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/val_batch0_pred
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/train_batch0.jpg
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/R_curve.png (def
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/hyp.yaml (deflat
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/PR_curve.png (de
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/confusion_matrix
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/labels_correlogr
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/train_batch2.jpg
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/val_batch1_label
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/val_batch0_label
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/labels.jpg (defl
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/P_curve.png (def
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/events.out.tfeve
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/val_batch2_pred
adding: content/YOLOv5-Modification/runs/train/aug_midNBG_m_600_8_100/results.png (def
```

```
!zip -r /content/exp.zip /content/YOLOv5-Modification/runs/detect/exp
```

```
adding: content/YOLOv5-Modification/runs/detect/exp/ (stored 0%)
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_20__00_01_26_-826-0003
adding: content/YOLOv5-Modification/runs/detect/exp/Pen2_201911081153GP041615-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911081208GP051730-0000
adding: content/YOLOv5-Modification/runs/detect/exp/bg_colored-60-_jpg.rf.f96c7613
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-831-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen2_201911081229GP061615-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_18__00_07_12_-856-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_19__00_07_12_-856-0002
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201912121218GP061858-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-836-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_21__00_01_26_-856-0005
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_19__00_07_12_-856-0002
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911261541GP171855-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_20__00_01_26_-856-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_20__00_02_53_-848-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_18__00_07_12_-856-0001
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_18__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911261448GP141855-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_19__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_21__00_01_26_-856-0005
adding: content/YOLOv5-Modification/runs/detect/exp/bg_colored-95-_jpg.rf.5147e2b3
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-822-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_21__00_08_39_-848-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_18__00_07_12_-856-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_20__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-823-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_21__00_08_39_-848-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-822-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_20__00_01_26_-856-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_21__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_21__00_01_26_-826-0005
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-822-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_21__00_01_26_-826-0005
```

```
adding: content/YOLOv5-Modification/runs/detect/exp/Pen2_201911081340GP101615-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_18__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201912121218GP061858-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen2_201911081136GP031615-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911081057GP011730-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_18__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_18__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_18__00_07_12_-856-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_20__00_02_53_-848-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_20__00_01_26_-826-0000
adding: content/YOLOv5-Modification/runs/detect/exp/bg_mono-42-_jpg.rf.033985260bb
adding: content/YOLOv5-Modification/runs/detect/exp/2016_02_07__07_15_47_-822-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201912121143GP041858-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_20__00_01_26_-856-0004
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911261430GP131855-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201912121218GP061858-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_10_21__00_01_26_-837-0000
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201912121201GP051858-0000
adding: content/YOLOv5-Modification/runs/detect/exp/bg_colored-110-_jpg.rf.581c7d1
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911081208GP051730-0000
adding: content/YOLOv5-Modification/runs/detect/exp/2016_09_21__00_01_26_-856-0005
adding: content/YOLOv5-Modification/runs/detect/exp/bg_colored-120-_jpg.rf.09cf94d
adding: content/YOLOv5-Modification/runs/detect/exp/Pen1_201911081057GP011730-0000
```

```
from google.colab import files

files.download("/content/train.zip")
```

```
files.download("/content/exp.zip")
```

```
# delete zips to follow naming
```

Pig Contact Detection System Using Convolutional Neural Networks

For Detection, it uses YOLOv5 with CSPDarknet as the backbone
For Interaction, it compares the IoU between the head and rear boxes and a calibrated interaction threshold

Notebook by Joackin Santos

Main Notebook: https://colab.research.google.com/drive/1orWlSQr0xjVmVI-2SvIuBQiu3wvIgBS3?usp=sharing

This is the accompanying notebook for the Pig Contact Detection System. This notebook uses the weights (*.pt) from the trained models to output the coordinates of the bounding boxes from detect.py of YOLOv5. The output is used to get the IoU (Intersection over Union) of the heads and rears in an image. This will be used to compute for interaction, based on a calibrated IoU threshold.

This will use the AFBI and AUF datasets to get predictions and truth values (annotated) to evaluate the interaction method.

Upon evaluation of the best threshold, the method will be extracted for single image use, to be integrated to the website application.

Setup

```
# for folder deletions
import shutil
shutil.rmtree('/content/YOLOv5-Modification')
```

Cloning Repositories

```
# clone repository containing yolov5 and datasets
# rename pig-datasets repo to datasets for yolov5
%cd /content
!git clone -b interaction-method https://github.com/joackinsantos/YOLOv5-Modification
!git clone https://github.com/joackinsantos/pig-datasets datasets
```

```
/content
Cloning into 'YOLOv5-Modification'...
remote: Enumerating objects: 9895, done.
remote: Counting objects: 100% (87/87), done.
remote: Compressing objects: 100% (50/50), done.
```

```
remote: Total 9895 (delta 41), reused 79 (delta 35), pack-reused 9808
Receiving objects: 100% (9895/9895), 475.52 MiB | 23.79 MiB/s, done.
Resolving deltas: 100% (125/125), done.
Updating files: 100% (162/162), done.
Cloning into 'datasets'...
remote: Enumerating objects: 54055, done.
remote: Counting objects: 100% (9623/9623), done.
remote: Compressing objects: 100% (9616/9616), done.
remote: Total 54055 (delta 7), reused 9623 (delta 7), pack-reused 44432
Receiving objects: 100% (54055/54055), 1.21 GiB | 20.08 MiB/s, done.
Resolving deltas: 100% (60/60), done.
Updating files: 100% (59989/59989), done.
```

Data Paths

```
# resource paths
AFBI_path = '/content/datasets/interaction-set/AFBI'
AUF_path = '/content/datasets/interaction-set/AUF'
Combined_path = '/content/datasets/interaction-set/Combined'
Combined_csv_path = '/content/datasets/interaction-set/Combined.csv'
img_AFBI_path = '/content/datasets/interaction-set/AFBI/AFBI001.jpg'
img_AUF_path = '/content/datasets/interaction-set/AUF/AUPF190.jpg'
weight_path = '/content/YOLOv5-Modification/test-weights/'
```

```
%ls
```

```
datasets/  sample_data/  YOLOv5-Modification/
```

Detection *detect.py was modified to extract bounding box properties such as:*

- image names
- confidence scores
- classfication and class index
- minX, minY, maxX, maxY

This is exported as the "bounding-boxes.csv" file

This uses the detect.py from YOLOv5 ultralytics repository

```
%cd /content/YOLOv5-Modification
!python detect.py --img 600 --weights {weight_path}/test.pt --source {Combined_path}
```

This is the detect implementation in the website (img size can be adjusted)

```
# pytorch setup
# %cd /content
import torch

from IPython.display import Image, clear_output  # to display images
from utils.downloads import attempt_download  # to download models/datasets

# clear_output()
torch.cuda.empty_cache()
print('Setup complete. Using torch %s %s' % (torch.__version__, torch.cuda.get_device_propert


path_hubconfig = '/content/YOLOv5-Modification'
path_weightfile = '/content/YOLOv5-Modification/best-weights/aug_midNBG_m_600_8_100.pt'


import torch
import os

img_size = 500

img_AFBI_path = '/content/datasets/interaction-set/AFBI/AFBI001.jpg'
img_AUF_path = '/content/datasets/interaction-set/AUF/AUPF190.jpg'

model = torch.hub.load(path_hubconfig, 'custom',
                    path=path_weightfile, source='local',)

jpg_files = []

for filename in os.listdir(Combined_path):
    if filename.endswith(".jpg"):
        # print(filename)
        file_path = os.path.join(Combined_path, filename)
        jpg_files.append(file_path)

# to sort in order of the true value file "Combined csv"
jpg_files.sort()
print(jpg_files)
# print(len(jpg_files))

results_torch = model(jpg_files, size=img_size)
results_torch.print()
```

```
YOLOv5 🚀 1cd2751 Python-3.10.11 torch-2.0.1+cu118 CUDA:0 (Tesla V100-SXM2-16GB, 161

requirements: /content/requirements.txt not found, check failed.
Fusing layers...
custom_YOLOv5m summary: 212 layers, 20856975 parameters, 0 gradients
Adding AutoShape...
['/content/datasets/interaction-set/Combined/AFBI001.jpg', '/content/datasets/intera
image 1/738: 411x811 9 Heads, 8 Rears
image 2/738: 411x811 8 Heads, 8 Rears
image 3/738: 411x811 8 Heads, 8 Rears
```

```
image 4/738: 411x811 9 Heads, 9 Rears
image 5/738: 411x811 9 Heads, 8 Rears
image 6/738: 411x811 8 Heads, 8 Rears
image 7/738: 411x811 8 Heads, 9 Rears
image 8/738: 411x811 9 Heads, 7 Rears
image 9/738: 411x811 10 Heads, 9 Rears
image 10/738: 411x811 8 Heads, 9 Rears
image 11/738: 411x811 8 Heads, 9 Rears
image 12/738: 411x811 9 Heads, 9 Rears
image 13/738: 411x811 8 Heads, 9 Rears
image 14/738: 411x811 8 Heads, 8 Rears
image 15/738: 411x811 7 Heads, 9 Rears
image 16/738: 411x811 9 Heads, 9 Rears
image 17/738: 411x811 9 Heads, 9 Rears
image 18/738: 411x811 9 Heads, 8 Rears
image 19/738: 411x811 9 Heads, 9 Rears
image 20/738: 411x811 10 Heads, 12 Rears
image 21/738: 411x811 10 Heads, 8 Rears
image 22/738: 411x811 9 Heads, 9 Rears
image 23/738: 411x811 9 Heads, 9 Rears
image 24/738: 411x811 9 Heads, 9 Rears
image 25/738: 411x811 10 Heads, 9 Rears
image 26/738: 411x811 7 Heads, 9 Rears
image 27/738: 411x811 8 Heads, 9 Rears
image 28/738: 411x811 10 Heads, 8 Rears
image 29/738: 411x811 9 Heads, 9 Rears
image 30/738: 411x811 8 Heads, 8 Rears
image 31/738: 411x811 9 Heads, 10 Rears
image 32/738: 411x811 9 Heads, 9 Rears
image 33/738: 411x811 10 Heads, 11 Rears
image 34/738: 411x811 10 Heads, 9 Rears
image 35/738: 411x811 10 Heads, 10 Rears
image 36/738: 411x811 9 Heads, 11 Rears
image 37/738: 411x811 10 Heads, 11 Rears
image 38/738: 411x811 9 Heads, 10 Rears
image 39/738: 411x811 10 Heads, 10 Rears
image 40/738: 411x811 9 Heads, 9 Rears
image 41/738: 411x811 11 Heads, 9 Rears
image 42/738: 411x811 10 Heads, 10 Rears
image 43/738: 411x811 11 Heads, 10 Rears
image 44/738: 411x811 11 Heads, 10 Rears
image 45/738: 411x811 12 Heads, 10 Rears
image 46/738: 411x811 11 Heads, 10 Rears
image 47/738: 411x811 12 Heads, 11 Rears
image 48/738: 411x811 10 Heads, 10 Rears
image 49/738: 411x811 10 Heads, 10 Rears
image 50/738: 411x811 10 Heads, 10 Rears
```

```
# sample df of bounding-box csv
%cd /content/YOLOv5-Modification
import pandas as pd

df = pd.read_csv('test.csv')
df
```

This is the Interaction Method (detect.py)

```
%cd /content/YOLOv5-Modification
import pandas as pd
import os

df = pd.read_csv('bounding-boxes.csv')

# variables
iou_threshold = 0.05

# df results
results_df = pd.DataFrame(columns=['Image_Name', 'Classification', 'Interaction_Count'])

def compute_interactions(data, threshold, results):
    # unique image names
    image_names = data['Image_Name'].unique()

    # loop for each unique name
    for name in image_names:
      # interaction variables
      interaction_flag = False
      interaction_count = 0

      # subset dataframe with this image name
      name_df = data[data['Image_Name'] == name]

      # subset dataframes of heads and rears
      head_df = name_df[name_df['Object_Name'] == 'Head'].reset_index(drop=True)
      rear_df = name_df[name_df['Object_Name'] == 'Rear'].reset_index(drop=True)

      # loop over heads and rears in the image
      for i, head_row in head_df.iterrows():
        for j, rear_row in rear_df.iterrows():
            # bounding box indexing
            # df[4] = minX, df[5] = maxX, df[6] = minY, df[7] = maxY

            head_minX = head_df.iloc[i, 4]
            head_maxX = head_df.iloc[i, 5]
            head_minY = head_df.iloc[i, 6]
            head_maxY = head_df.iloc[i, 7]

            rear_minX = rear_df.iloc[j, 4]
            rear_maxX = rear_df.iloc[j, 5]
            rear_minY = rear_df.iloc[j, 6]
            rear_maxY = rear_df.iloc[j, 7]

            # this is for the headBox and rearBox
            curr_iou = compute_iou(head_minX, head_maxX, head_minY, head_maxY,
```

```
                            rear_minX, rear_maxX, rear_minY, rear_maxY)

                # print("computed the IoU:", curr_iou)

                if (curr_iou >= threshold):
                  interaction_flag = True
                  interaction_count += 1

                # # tester
                # print(i, j)
                # print(head_minX, head_minY, head_maxX, head_maxY)
                # print(rear_minX, rear_minY, rear_maxX, rear_maxY)

          temp_list = [name,
                       1 if interaction_flag else 0,
                       interaction_count]
          results.loc[len(results)] = temp_list
        return results

    def compute_iou(head_minX, head_maxX, head_minY, head_maxY, rear_minX, rear_maxX, rear_minY,
        # determine (x,y) coordinates of the intersection rectangle
        x_left = max(head_minX, rear_minX)
        y_top = max(head_minY, rear_minY)
        x_right = min(head_maxX, rear_maxX)
        y_bottom = min(head_maxY, rear_maxY)

        # compute the area of intersection rectangle
        intersection_area = max(0, x_right - x_left + 1) * max(0, y_bottom - y_top + 1)

        # compute area of both prediction and ground-truth
        # rectangles
        headArea = (head_maxX - head_minX + 1) * (head_maxY - head_minY + 1)
        rearArea = (rear_maxX - rear_minX + 1) * (rear_maxY - rear_minY + 1)
        union_area = float(headArea + rearArea - intersection_area)

        # compute IoU
        iou = intersection_area / union_area
        return iou

    res = compute_interactions(df, iou_threshold, results_df).copy()
    if os.path.exists('results.csv'):
        os.remove('results.csv')
    res.to_csv('results.csv', index=False)

    # print(res)


         /content/YOLOv5-Modification
```

This is Interaction Method (PyTorch)

```
%cd /content/YOLOv5-Modification
import pandas as pd
import os

# df format [xmin ymin xmax ymax confidence class name], xyxy index is the name
bb = results_torch.pandas().xyxy

# variables
iou_threshold = 0.05

# df results
results_df = pd.DataFrame(columns=['Classification', 'Interaction_Count'])

def compute_interactions(data, threshold, results):
    # loop for each unique name/ image
    for image in bb:
      # interaction variables
      interaction_flag = False
      interaction_count = 0

      # subset dataframe with this image name
      name_df = image

      # subset dataframes of heads and rears
      head_df = name_df[name_df['name'] == 'Head'].reset_index(drop=True)
      rear_df = name_df[name_df['name'] == 'Rear'].reset_index(drop=True)

      # loop over heads and rears in the image
      for i, head_row in head_df.iterrows():
        for j, rear_row in rear_df.iterrows():
            # bounding box indexing
            # df[0] = minX, df[1] = minY, df[2] = maxX, df[3] = maxY

            head_minX = head_df.iloc[i, 0]
            head_minY = head_df.iloc[i, 1]
            head_maxX = head_df.iloc[i, 2]
            head_maxY = head_df.iloc[i, 3]

            rear_minX = rear_df.iloc[j, 0]
            rear_minY = rear_df.iloc[j, 1]
            rear_maxX = rear_df.iloc[j, 2]
            rear_maxY = rear_df.iloc[j, 3]

            # this is for the headBox and rearBox
            curr_iou = compute_iou(head_minX, head_maxX, head_minY, head_maxY,
                        rear_minX, rear_maxX, rear_minY, rear_maxY)

            # print("computed the IoU:", curr_iou)

            if (curr_iou >= threshold):
```

```
                interaction_flag = True
                interaction_count += 1

            # # tester
            # print(i, j)
            # print(head_minX, head_minY, head_maxX, head_maxY)
            # print(rear_minX, rear_minY, rear_maxX, rear_maxY)

      temp_list = [1 if interaction_flag else 0,
                   interaction_count]
      results.loc[len(results)] = temp_list
    return results

def compute_iou(head_minX, head_maxX, head_minY, head_maxY, rear_minX, rear_maxX, rear_minY,
    # determine (x,y) coordinates of the intersection rectangle
    x_left = max(head_minX, rear_minX)
    y_top = max(head_minY, rear_minY)
    x_right = min(head_maxX, rear_maxX)
    y_bottom = min(head_maxY, rear_maxY)

    # compute the area of intersection rectangle
    intersection_area = max(0, x_right - x_left + 1) * max(0, y_bottom - y_top + 1)

    # compute area of both prediction and ground-truth
    # rectangles
    headArea = (head_maxX - head_minX + 1) * (head_maxY - head_minY + 1)
    rearArea = (rear_maxX - rear_minX + 1) * (rear_maxY - rear_minY + 1)
    union_area = float(headArea + rearArea - intersection_area)

    # compute IoU
    iou = intersection_area / union_area
    return iou

res = compute_interactions(bb, iou_threshold, results_df).copy()
if os.path.exists('results.csv'):
    os.remove('results.csv')
res.to_csv('results.csv', index=False)

# print(res)
```

```
/content/YOLOv5-Modification
```

Evaluations

Getting and Processing Necessary Data

```
# if os.path.exists(Combined_csv_path):
#     os.remove(Combined_csv_path)
```

```
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)


# True Values
truth_df = pd.read_csv(Combined_csv_path)
# Changing column names, removing extra \', binarizing the classification
binarize = lambda x: 0 if x == 0 else 1

truth_df['Image_Name'] = truth_df['Image_Name'].str.replace('\'', '')
truth_df['Classification'] = truth_df['Classification'].apply(binarize)

# truth_df


# Getting truth_df counts
truth_df['is_AFBI'] = truth_df['Image_Name'].str.contains('AFBI').astype(int)
truth_df['is_AUF'] = truth_df['Image_Name'].str.contains('AUPF').astype(int)
AFBI_grouped = truth_df.groupby('is_AFBI')
# AUF_grouped = truth_df.groupby('is_AUF')

# This creates separated dfs for the two data sets
for name, group in AFBI_grouped:
    df_name = 'AFBI_group_df' if name else 'AUF_group_df'
    globals()[df_name] = group.copy()

truth = truth_df['Classification'].value_counts()
afbi = AFBI_group_df['Classification'].value_counts()
auf = AUF_group_df['Classification'].value_counts()
print("truth:\n", truth,'\n',"afbi:\n",afbi,'\n'," auf:\n",auf,'\n')
```

```
truth:
 0    456
1    282
Name: Classification, dtype: int64
 afbi:
 0    322
1    111
Name: Classification, dtype: int64
  auf:
 1    171
0    134
Name: Classification, dtype: int64
```

```
# Predicted Values
pred_df = pd.read_csv('results.csv')
# pred_df


# Getting pred_df counts
# pred_df['is_AFBI'] = pred_df['Image_Name'].str.contains('AFBI').astype(int)
# pred_df['is_AUF'] = pred_df['Image_Name'].str.contains('AUPF').astype(int)
```

```
# AFBI_grouped = pred_df.groupby('is_AFBI')

# AUF_grouped = truth_df.groupby('is_AUF')

# This creates separated dfs for the two data sets
# for name, group in AFBI_grouped:
#     df_name = 'AFBI_group_df' if name else 'AUF_group_df'
#     globals()[df_name] = group.copy()

pred = pred_df['Classification'].value_counts()
# afbi = AFBI_group_df['Classification'].value_counts()
# auf = AUF_group_df['Classification'].value_counts()
# print("pred:\n", pred,'\n',"afbi:\n",afbi,'\n'," auf:\n",auf,'\n')
print("pred:\n", pred)
```

```
pred:
 1    380
0    358
Name: Classification, dtype: int64
```

Evaluation Proper

```
print(truth_df.shape[0],
      pred_df.shape[0])
```

```
738 738
```

```
y_true = truth_df["Classification"]
y_pred = pred_df["Classification"]

from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusio

accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)
f1 = f1_score(y_true, y_pred)
confusion_matrix = confusion_matrix(y_true, y_pred)
mcc = matthews_corrcoef(y_true, y_pred)
n_mcc = (mcc + 1)/2   # normalize to set in range [0,1]
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
print("nMCC:", n_mcc)
print("Confusion Matrix:\n", confusion_matrix)
```

```
Accuracy: 0.7533875338753387
Precision: 0.631578947368421
Recall: 0.851063829787234
F1 Score: 0.7250755287009063
```

```
nMCC: 0.7644718895000668
Confusion Matrix:
 [[316 140]
 [ 42 240]]
```

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
plt.clf()

# assume y_true and y_pred are the true and predicted labels
cm = confusion_matrix(y_true, y_pred)

# create a heatmap using seaborn
sns.heatmap(cm, annot=True, cmap="Blues", fmt="d")

# add axis labels and title
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")

# show the plot
plt.show()
plt.savefig(f"conf_mat_{img_size}_{iou_threshold}.png")
```

```
from google.colab import files
files.download(f'/content/YOLOv5-Modification/conf_mat_{img_size}_{iou_threshold}.png')
```

B. Source Code

1. Django Web Framework
views.py (contains methods)

```
import io
import os
import cv2
import torch
import shutil
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from PIL import Image as im
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix

from django.core.exceptions import *
from django.shortcuts import render, redirect
from django.contrib import messages
from django.views.generic.edit import CreateView
from .models import *


def home(request):

    if request.method == "POST":
        image = UploadImage()
        if len(request.FILES) != 0:
            image.image = request.FILES['image']
            filename = image.image.name

            if image.image.size > 5*1024*1024:
                # raise ValidationError('File size too large.')
                print("File size too large.")
                return render(request, 'home.html')
            elif not filename.lower().endswith(('.jpg', '.jpeg', '.png')):
                # raise ValidationError('File is not an image.')
                print("File is not an image.")
                return render(request, 'home.html')
            else:
                image.save()
        image.save()
        messages.success(request, "Image Uploaded!")
        return redirect('upload/')
    return render(request, 'home.html')

def upload(request):
    image = UploadImage.objects.all().last()
    data = {
        'image':image,
    }
    print(image)
    return render(request, 'upload.html', data)

# Use this to modify Contact Detection Performance
# model (img_size, threshold_value)
# aug_mid_m_600_8_100 (500, 0.05)
model_weight = 'aug_mid_m_600_8_100.pt'
threshold_value = 0.05
img_size = 500

# IMAGE DETECTION AND INTERACTION
def results(request):
    # GET UPLOADED IMAGE
    image = UploadImage.objects.all().last()
    img_bytes = image.image.read()
    img = im.open(io.BytesIO(img_bytes))

    ## DETECTION METHOD
    # LOADING THE MODEL (can be loaded remotely)
    # path_hubconfig = "joackinsantos/YOLOv5-Modification:website-integration"
    path_hubconfig = "YOLOv5-Modification"
    path_weightfile = f"best-weights/{model_weight}"
    model = torch.hub.load(path_hubconfig, 'custom',
                        path=path_weightfile, source='local',
                        force_reload=True)
    results = model(img, size=img_size)

    # PROCESSING DETECTED IMAGE
    # remove previous detected image directory to save space
    detect_image_path = 'runs/detect'
    exp_path = f'{detect_image_path}/exp'
    if os.path.exists(exp_path):
        shutil.rmtree(exp_path)
    # creates new detected image
    results.print()
    results.save()
    # move directory and change image name
    old_path = f'{detect_image_path}/exp/image0.jpg'
    new_path = f'{detect_image_path}/image0.jpg'
    output_path = f'{detect_image_path}/detect-{str(image)}'
    if not os.path.exists(new_path) and not os.path.exists(output_path):
        shutil.move(old_path, new_path) # move to non refreshed dir
        os.rename(new_path, output_path) # rename detected image
```

```
    ## INTERACTION METHOD
    # bb = bounding box
    bb = results.pandas().xyxy[0]
    iou_threshold = threshold_value
    results_df = pd.DataFrame(columns=['Classification', 'Interaction_Count'])
    interaction_flag = False
    interaction_count = 0

    # separating by class names
    head_df = bb[bb['name'] == 'Head'].reset_index(drop=True)
    rear_df = bb[bb['name'] == 'Rear'].reset_index(drop=True)

    # loop over heads and rears in the image
    for i, head_row in head_df.iterrows():
        for j, rear_row in rear_df.iterrows():
            # bounding box indexing
            # df[0] = minX, df[1] = minY, df[2] = maxX, df[3] = maxY

            head_minX = head_df.iloc[i, 0]
            head_minY = head_df.iloc[i, 1]
            head_maxX = head_df.iloc[i, 2]
            head_maxY = head_df.iloc[i, 3]

            rear_minX = rear_df.iloc[j, 0]
            rear_minY = rear_df.iloc[j, 1]
            rear_maxX = rear_df.iloc[j, 2]
            rear_maxY = rear_df.iloc[j, 3]

            curr_iou = compute_iou(head_minX, head_maxX, head_minY, head_maxY,
                       rear_minX, rear_maxX, rear_minY, rear_maxY)

            # print(curr_iou)
            if(curr_iou >= iou_threshold):
                interaction_flag = True
                interaction_count += 1

    temp_list = [1 if interaction_flag else 0,
                 interaction_count]
    results_df.loc[len(results_df)] = temp_list

    row = results_df.iloc[0]
    classification = 'With Contact' if row['Classification'] == 1 else "Without Contact"
    count = row['Interaction_Count']

    data = {
        'image': f'../{output_path}',
        'class': classification,
        'interaction_count': count
    }

    return render(request, 'results.html', data)

def test(request):
    return render(request, 'test.html')



# FUNCTIONS
def compute_iou(head_minX, head_maxX, head_minY, head_maxY, rear_minX, rear_maxX, rear_minY, rear_maxY):
    # determine (x,y) coordinates of the intersection rectangle
    x_left = max(head_minX, rear_minX)
    y_top = max(head_minY, rear_minY)
    x_right = min(head_maxX, rear_maxX)
    y_bottom = min(head_maxY, rear_maxY)

    # compute the area of intersection rectangle
    intersection_area = max(0, x_right - x_left + 1) * max(0, y_bottom - y_top + 1)

    # compute area of both prediction and ground-truth
    # rectangles
    headArea = (head_maxX - head_minX + 1) * (head_maxY - head_minY + 1)
    rearArea = (rear_maxX - rear_minX + 1) * (rear_maxY - rear_minY + 1)
    union_area = float(headArea + rearArea - intersection_area)

    # compute IoU
    iou = intersection_area / union_area
    return iou

# USEFUL TESTERS
# Tester for Head and Rear dfs
# use to test order
# print(head_df.head())
# print(rear_df.head())
# head_minX = head_df.iloc[0, 0]
# head_minY = head_df.iloc[0, 1]
# head_maxX = head_df.iloc[0, 2]
# head_maxY = head_df.iloc[0, 3]
# print(head_minX, head_minY, head_maxX, head_maxY)
```

models.py

```
import os

from django.db import models
from django.utils.translation import gettext_lazy as _

# Create your models here.
```

```
class UploadImage(models.Model):
    image = models.ImageField(_("image"),upload_to='pig-images/', null=True, blank=True)

    def __str__(self):
        return str(os.path.split(self.image.path)[-1])
```

urls.py

```
"""PCD_project URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/4.1/topics/http/urls/
Examples:
Function views
    1. Add an import: from my_app import views
    2. Add a URL to urlpatterns: path('', views.home, name='home')
Class-based views
    1. Add an import: from other_app.views import Home
    2. Add a URL to urlpatterns: path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns: path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from PCD_app.views import *

from django.conf import settings
from django.conf.urls.static import static


urlpatterns = [
    path("admin/", admin.site.urls),
    path("", home, name="home"),
    path("upload/", upload, name="upload"),
    path("results/", results, name="results"),
    path("test/", test, name="test"),
]

if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

main.css

```
body {
    overflow-x: hidden;
    position: relative;
    z-index: -1;
}

.main-div{
    background-color: white;
    font-family: 'Reem Kufi Fun', sans-serif !important;
    font-weight:normal;
    padding:0;
}

.upload-div{
    margin-top: 2%;
    margin-bottom: 3%;
}

.how-div{
    margin-top: 4%;
    margin-bottom: 3%;
}

.upload-button{
    font-size:medium;
    display:inline-block;
}

.carousel-inner .carousel-item {
    width: 700px;
    height: auto;
}

.upload-image-container{
    position: relative;
    display: inline-block;
}

.upload-image{
    width: 1000px;
    height: auto;
    border: 7px solid #1C8572;
    box-shadow:0px 3px 2px rgb(85, 85, 85);
    transition: transform 0.3s, filter 0.3s;
    z-index: 9999;
    cursor: zoom-in;
}

img.zoomed {
    position: relative;
    transform: scale(1.5);
    z-index: 9999;
    cursor: zoom-out;
```

```
}

.overlay {
    opacity:0;
    background-color: black;
    position:fixed;
    width:100%;
    height:100%;
    top:0px;
    left:0px;
    z-index:-1;
}


.upload-text{
    background-color: #047675;
    font-size: 20px;
    color:white;
    padding-top: 5px;
    padding-bottom: 5px;
    padding-left: 75px;
    padding-right: 75px;
    border-radius: 5px;
}

.results-box-1{
    background-color: #1C8572;
    color: white;
    border-radius: 3px;
    align-items: center;
    text-align: center;
    box-shadow:0px 1px 2px rgb(85, 85, 85);
}

.results-box-2{
    background-color: #1C8572;
    opacity: 85%;
    color:white;
    border-radius: 3px;
    align-items: center;
    text-align: center;
    box-shadow:0px 1px 2px rgb(85, 85, 85);
}

.btn{
    color: white;
    border: none;
    border-radius: 15px;
    padding-top: 4%;
    padding-bottom: 4%;
    font-weight: bold;
    background-color: #047675;
    box-shadow:0px 3px 2px rgb(85, 85, 85);
}
.btn:hover{
    background-color: #036260;
    color: white;
}
.btn:active, .open > .dropdown-toggle.btn{
    background-color: #036260 !important;
    border-color:#036260!important;
}

.btn-back{
    color: white !important;
    border: none;
    border-radius: 15px;
    font-weight: bold;
    background-color: #047675;
    box-shadow:0px 3px 2px rgb(85, 85, 85);
}
.btn-back:hover{
    background-color: #036260;
    color: white;
}
.btn-back:active, .open > .dropdown-toggle.btn{
    background-color: #036260 !important;
    border-color:#036260!important;
}

.upload-helper{
    font-size: small;
}

.circle {
    width: 33px;
    height: 33px;
    line-height: 33px;
    border-radius: 50%;
    font-size: 18px;
    color: #047675;
    text-align: center;
    background: white;
    box-shadow:0px 2px 2px rgb(85, 85, 85);
}
.circle-colored {
    width: 22px;
    height: 22px;
    line-height: 33px;
```

```
    border-radius: 50%;
    font-size: 18px;
    color: #047675;
    text-align: center;
    background:#047675;
    box-shadow:0px 2px 2px rgb(85, 85, 85);
}

.how-title{
    font-size: 20px;
    font-weight: bolder;
}

.seg-text{
    font-size: 17px;
}

.description-div{
    background-color: #ecf5f1;
}

.description-margin{
    margin-left: 12%;
    margin-right: 12%;
    padding-top: 5%;
    padding-bottom: 5%;
}

.pig-home{
    width: 375px;
    height: auto;
    border-radius: 50%;
    box-shadow:0px 5px 10px rgb(85, 85, 85);
    /* -webkit-filter: grayscale(100%); Safari 6.0 - 9.0
    filter: grayscale(100%); */
}

.description-logo{
    padding-left: 25px;
    padding-right: 25px;
}

.description-title{
    background-color: #047675;
    color: white;
    font-size: x-large;
    font-weight: bold;
    border-radius: 10px;
}

.title-accompany{
    background-color: #20937F;
    color: white;
    font-size: large;
    -webkit-mask-image: -webkit-gradient(linear, center top, center bottom,
                       color-stop(0.75, rgba(0,0,0,1)),
                       color-stop(1.00, rgba(0,0,0,0)));
    mask-image: -webkit-gradient(linear, center top, center bottom,
                       color-stop(0.75, rgba(0,0,0,1)),
                       color-stop(1.00, rgba(0,0,0,0)));
}

.pig-title-icon{
    width: 60px;
    height: auto;
}

.description-logotext{
    font-size: larger;
}

.description-text{
    font-size: 18px;
    margin-right: 10%;
    margin-left: 10%;
}

.description-foottext{
    font-size: smaller;
    margin-right: 10%;
    margin-left: 10%;
}

.redirect-button{
    font-size: 20px;
    border-radius: 15px;
    padding-top: 2%;
    padding-bottom: 2%;
    font-weight: bold;
    background-color: #047675;
    box-shadow:0px 3px 2px rgb(85, 85, 85);
}
```

navbar.css

```
.navbar{
    background-color: #047675;
    font-family: 'Reem Kufi Fun', sans-serif !important;
```

```
    box-shadow:0px 2px 3px rgb(85, 85, 85);
}


.navbar-brand{
    font-size: 175%;
    text-shadow:40px 50px 50px rgb(85, 85, 85);
}

.pig-icon{
    width: 70px;
    height: auto;
}
```

footer.css

```
.footer{
    background-color: #047675;
    color: white;
    font-family: 'Lato', sans-serif !important;
    font-weight: bold;
}
```

2. Website HTML files

main.html

```
<!doctype html>
<html lang="en">
    <head>
        {% load static %}
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
        <!-- Fonts -->
        <link rel="preconnect" href="https://fonts.googleapis.com">
        <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
        <link href="https://fonts.googleapis.com/css2?family=Lato:ital,wght@0
            ↪ ,100;0,300;0,400;0,700;0,900;1,100;1,300;1,400;1,700;1,900&display=swap" rel="stylesheet">
        <link href="https://fonts.googleapis.com/css2?family=Reem+Kufi+Fun:wght@400;500;600;700&display=swap" rel="stylesheet">
        <!-- Bootstrap CSS -->
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/css/bootstrap.min.css" integrity="sha384-
            ↪ ggOyROiXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.1.1/css/all.min.css" integrity="sha512-
            ↪ KfkfwYDsLkIlwQp6LFnl8zNdLGxu9YAA1QvwINks4PhcElQSvqcyVLLD9aMhXd13uQjoXtEKNosOWaZqXgel0g==" crossorigin="anonymous"
            ↪  referrerpolicy="no-referrer" />
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/mdb-ui-kit/6.3.0/mdb.min.css"/>
        <link rel="icon" type="image/png" href="{% static 'image/pig_icon_2.png' %}">
        <title>TailSafe</title>
    </head>

    <body>
        {% load static %}
        <!-- <link rel='stylesheet' href={% static 'css/main.css' %}> -->
        <!-- Navbar -->
        {% include 'navbar.html' %}

        <!-- Content -->
        {% block content %}

        {% endblock content %}

        {% include 'footer.html' %}

        <!-- Optional JavaScript -->
        <!-- jQuery first, then Popper.js, then Bootstrap JS -->
        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965
            ↪ DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/popper.js@1.14.7/dist/umd/popper.min.js" integrity="sha384-
            ↪ UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
        <script src="https://cdn.jsdelivr.net/npm/bootstrap@4.3.1/dist/js/bootstrap.min.js" integrity="sha384-
            ↪ JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
    </body>
</html>
```

navbar.html

```
{% load static %}
<link rel="stylesheet" href={% static 'css/navbar.css' %}>

<!-- Navbar Bootstrap -->
<div class="navbar-container">
    <nav class="navbar navbar-expand-lg navbar-light px-4 py-2 d-flex row">
        <div class="navbar-icon d-flex flex-row justify-content-center">
            <a href={% url 'home' %} >
                <img class="pig-icon"src={% static 'image/pig_icon_2.png' %} alt="pig icon">
            </a>
            <a class="navbar-brand font-weight-bold text-white ml-2" href={% url 'home' %}>
                TailSafe
            </a>
        </div>
    </nav>
</div>
```

footer.html

```
{% load static %}
<link rel="stylesheet" href={% static 'css/footer.css' %}>


<footer class="footer text-center text-lg-start">
    <div class="text-center p-3" style="background-color: rgba(0, 0, 0, 0.2);">
      TailSafe by
      <a class="text-white" href="https://github.com/joackinsantos"> <u>Joackin Santos</u></a>
    </div>
</footer>
```

home.html

```
{% extends 'main.html' %}
{% load static %}
{% block content %}
<link rel="stylesheet" href={% static 'css/main.css' %}>

<div class="container-fluid main-div d-flex flex-column">
    <div class="description-div container-fluid text-center">
        <div class="description-margin d-flex flex-row justify-content-center">
            <div class="description-logo">
                <div class="d-flex flex-column align-self-center">
                    <div class="description-image mt-3 mr-2">
                        <img class="pig-home" src={% static 'image/pig-closeup-cropped.jpg' %} alt="pig icon">
                    </div>
                </div>
            </div>
            <div class="description-group d-flex flex-column justify-content-center align-self-center">
                <div class="description-title d-flex flex-column">
                    <div class="justify-content-center mt-2 py-1 d-flex flex-row">
                        <div class="d-flex mr-2">
                            <img class="pig-title-icon"src={% static 'image/pig_icon_2.png' %} alt="pig icon">
                        </div>
                        <div class="d-flex align-self-center mt-2 mb-1">
                            TailSafe
                        </div>
                    </div>
                    <div class="title-accompany mt-1 pt-2 d-flex flex-row justify-content-center">
                        <p>
                            Empowering Farm Decision Making with Pig Tail Biting Detection
                        </p>
                    </div>
                </div>
                <div class="description-text text-justify text-wrap mt-4">
                    <p class="">
                        Tail biting is a significant welfare problem in pig farming where one pig bites the tail of another,
                        causing harm, pain, and stress in the pen. This can be a sign of various issues such as poor living
                            ↪ conditions,
                        health problems, and boredom, which can lead to abnormal behavior. <br><br>

                        TailSafe is a website application developed to detect head-to-rear contact between pigs in a pen by analyzing
                        uploaded pig pen images. This can help diagnose if tail biting is possibly present in a pen to act as a
                            ↪ decision
                        support tool for medical interventation and maintain welfare in the pig farm.
                    </p>
                </div>
                <div class="description-foottext text-uppercase text-justify text-wrap mt-4 text-center">
                    <a href="#upload-section">
                        <button class="redirect-button btn container-fluid text-uppercase mt-2">
                            Start detecting now
                        </button>
                    </a>
                    <p class="mt-3">
                        Pig Head-to-Rear Contact Detection Website Application
                    </p>
                </div>
            </div>
        </div>
    </div>
    <div class="how-div">
        <div class="how-title container-fluid text-center mb-5 font-weight-bold">
            How to Use TailSafe for Pig Tail Biting Detection
        </div>
        <div class="container segment-container">
            <div class="row">
              <div class="col-sm">
                <div class="row text-justify pr-1">
                    <p class="col-0 mt-2 circle">1</p>
                    <p class="seg-text col p-1 ml-2 mr-4">
                        Select a pig pen image by clicking on "Choose File" then click the "Upload Image" button to upload it.
                    </p>
                </div>
              </div>
              <div class="col-sm">
                <div class="row text-justify pr-1">
                    <p class="col-0 mt-2 circle">2</p>
                    <p class="seg-text col p-1 ml-2 mr-4">
                        Confirm the selected image by clicking the "Submit and Process" button to initiate image processing.
                    </p>
                </div>
              </div>
              <div class="col-sm">
                <div class="row text-justify pr-1">
                    <p class="col-0 mt-2 circle">3</p>
                    <p class="seg-text col p-1 ml-2 mr-4">
```

```
                    View the results to determine if the image contains any contacts and the total counts of contacts detected.
                </p>
              </div>
            </div>
          </div>
        </div>
    </div>
    <hr class="hr hr-blurry" />
    <div class="upload-div d-flex flex-column mx-auto">
        <div id="upload-section" class="upload-text-container d-flex justify-content-center mx-3 mt-2 mb-2 ">
            <p class="upload-text head-text font-weight-bold">
                Detect head-to-rear contact between pigs
            </p>
        </div>
        <form action="" method="POST" enctype="multipart/form-data">
            <div class="upload-button-container d-flex flex-column justify-content-center mt-5">
                <div class="custom-file custom-file-button mb-2">
                    <input id="fileInput" class="form-control form-control-lg" type="file" Required name="image" />
                </div>
                <button type="submit" class="upload-button btn container-fluid text-uppercase mt-2">
                    <i class="fa-solid fa-plus mr-2"></i>UPLOAD IMAGE
                </button>
                {% csrf_token %}
            </div>
        </form>
        <div class="upload-helper-container d-flex justify-content-center">
            <p class="upload-helper mt-1">
                please upload images only
            </p>
        </div>
    </div>
</div>
{% endblock content %}
```

upload.html

```
{% extends 'main.html' %}
{% load static %}
{% block content %}
<link rel="stylesheet" href={% static 'css/main.css' %}>

<div class="container-fluid main-div d-flex flex-column">
    <div class="how-div mt-5 mb-2">
        <div class="how-title container-fluid text-center mb-4 font-weight-bold">
            How to Use TailSafe for Pig Tail Biting Detection
        </div>
        <div class="container segment-container">
            <div class="row">
              <div class="col-sm">
                <div class="row text-justify pr-1">
                    <p class="col-0 mt-2 circle">1</p>
                    <p class="seg-text col p-1 ml-2 mr-4">
                        Select a pig pen image by clicking on "Choose File" then click the "Upload Image" button to upload it.
                    </p>
                </div>
              </div>
              <div class="col-sm">
                <div class="row text-justify pr-1">
                    <p class="col-0 mt-2 circle">2</p>
                    <p class="seg-text col p-1 ml-2 mr-4">
                        Confirm the selected image by clicking the "Submit and Process" button to initiate image processing.
                    </p>
                </div>
              </div>
              <div class="col-sm">
                <div class="row text-justify pr-1">
                    <p class="col-0 mt-2 circle">3</p>
                    <p class="seg-text col p-1 ml-2 mr-4">
                        View the results to determine if the image contains any contacts and the total counts of contacts detected.
                    </p>
                </div>
              </div>
            </div>
          </div>
    </div>
    <hr class="hr hr-blurry" />
    <div class="upload-div d-flex flex-column mx-auto">
        <div class="upload-text-container d-flex justify-content-center mx-3 mt-1 mb-1">
            <p class="upload-text head-text">
                Detect head-to-rear contact between pigs
            </p>
        </div>
        <div class="upload-image-container d-flex justify-content-center">
            <img class="upload-image" src="{{ image.image.url }}" onclick="zoomImage(this)" alt="default">
            <div class="overlay"></div>
        </div>
        <div class="upload-button-container d-flex justify-content-center mt-5">
            <div class="">
                <a href={% url 'results' %} type="button" class="upload-button btn container-fluid text-uppercase px-5 py-3">
                    SUBMIT AND PROCESS
                </a>
            </div>

        </div>
    </div>
    <hr class="hr hr-blurry" />
    <div class="back-button-container d-flex justify-content-center mt-2 mb-4">
      <a onclick="redirectToMainPage()" type="button" class="upload-button btn-back text-uppercase p-2 px-4">
```

```
      <i class="fa-solid fa-arrow-left mr-2"></i>BACK
    </a>
  </div>
</div>

<script>
  function redirectToMainPage() {
    window.location.href = '/#upload-section';
  }
  function zoomImage(element) {
      element.classList.toggle('zoomed');
      document.body.classList.toggle('darken-page');
      var overlay = document.querySelector('.overlay');
      overlay.classList.toggle('show');

      if (overlay.classList.contains('show')) {
          overlay.style.opacity = '0.5';
          overlay.style.zIndex = '1000';
      } else {
          overlay.style.opacity = '';
          overlay.style.zIndex = '';
      }

      if (element.classList.contains('zoomed')) {
          element.style.cursor = 'zoom-out';
      } else {
          element.style.cursor = '';
      }
  }
</script>
{% endblock content %}
```

results.html

```
{% extends 'main.html' %}
{% load static %}
{% block content %}
<link rel="stylesheet" href={% static 'css/main.css' %}>

<div class="container-fluid main-div d-flex flex-column">
    <div class="upload-div d-flex flex-column mx-auto mb-4">
        <div class="upload-text-container d-flex justify-content-center mx-3 mt-4 mb-2">
            <p class="upload-text head-text">Detect head-to-rear contact between pigs</p>
        </div>
        <div class="upload-image-container d-flex justify-content-center">
            <img class="upload-image" src="{{ image }}" onclick="zoomImage(this)" alt="default">
            <div class="overlay"></div>
        </div>
        <!-- Results segment-->
        <div class="results-container container mt-3 px-5 font-weight-bold">
            <div class="row text-center">
              <div class="col results-box-1 mr-1 text-uppercase py-2 align-self-center">
                Classification
              </div>
              <div class="col results-box-2 py-2 align-self-center">
                {{ class }}
              </div>
            </div>
            <div class="row mt-1">
                <div class="col results-box-1 mr-1 text-uppercase py-2 align-self-center">
                    Contact Count
                  </div>
                <div class="col results-box-2 py-2 align-self-center">
                    {{ interaction_count }}
                </div>
            </div>
          </div>
        <!-- Results segment -->
    </div>
    <hr class="hr hr-blurry" />
    <div class="how-div mt-3 mb-1">
      <div class="how-title container-fluid text-center mb-4 font-weight-bold">
          Making Sense of the Results Above
      </div>
      <div class="container segment-container">
          <div class="row">
            <div class="col-sm">
              <div class="row text-justify pr-1">
                  <p class="col-0 mt-2 circle-colored">1</p>
                  <p class="seg-text col p-1 ml-2 mr-4">
                    The <b>Classification</b> result provides two possible values: <b>With Contact</b> and <b>Without Contact.</b>
                    If the result is <b>With Contact</b>, it indicates that the system has detected instances of head-to-rear
                    contact or tail biting in the provided image. Otherwise, if the result is <b>Without Contact</b>, it means
                    that no such contact was detected.
                  </p>
              </div>
            </div>
            <div class="col-sm">
              <div class="row text-justify pr-1">
                  <p class="col-0 mt-2 circle-colored">1</p>
                  <p class="seg-text col p-1 ml-2 mr-4">
                    Assuming the classification result indicates <b>With Contact</b>, the <b>Contact Count</b> result represents the
                    ↪ total
                    number of occurrences of head-to-rear contact or tail biting in the provided image. If there is no contact
                    detected, the <b>Contact Count</b> will be zero.
                  </p>
              </div>
            </div>
```

```
        </div>
      </div>
  </div>
  <hr class="hr hr-blurry" />
  <div class="back-button-container d-flex justify-content-center mt-2 mb-4">
    <a href={% url 'home' %} type="button" class="upload-button btn-back text-uppercase p-2 px-4 mr-2">
        <i class="fa-solid fa-house-chimney mr-2"></i>HOME
    </a>
    <a onclick="redirectToMainPage()" type="button" class="upload-button btn-back text-uppercase p-2 px-4">
      <i class="fa-solid fa-arrow-left mr-2"></i>DETECT AGAIN
    </a>
  </div>
</div>

<script>
  function redirectToMainPage() {
    window.location.href = '/#upload-section';
  }
  function zoomImage(element) {
        element.classList.toggle('zoomed');
        document.body.classList.toggle('darken-page');
        var overlay = document.querySelector('.overlay');
        overlay.classList.toggle('show');

        if (overlay.classList.contains('show')) {
            overlay.style.opacity = '0.5';
            overlay.style.zIndex = '1000';
        } else {
            overlay.style.opacity = '';
            overlay.style.zIndex = '';
        }

        if (element.classList.contains('zoomed')) {
            element.style.cursor = 'zoom-out';
        } else {
            element.style.cursor = '';
        }
    }
</script>
{% endblock content %}
```

XI. Acknowledgment

I would like to express my heartfelt gratitude to all those who have supported me throughout my journey at the University of the Philippines Manila.

First and foremost, I want to thank God for granting me strength, guidance, and inspiration throughout this endeavor.

To my loving family, I am deeply grateful for providing me with the necessary resources, such as my laptop and funding for essential software, enabling me to pursue and complete my SP. Your unwavering support and belief in me have been instrumental in my success.

A special appreciation goes to my partner, whose unwavering support and invaluable insights have played a significant role in shaping the topic and direction of my SP. Your encouragement and contributions have been truly remarkable.

I extend my sincere thanks to my SP adviser, Dr. Vincent Magboo, for his guidance, expertise, and mentorship. Your valuable advice, from refining the research paper to making crucial decisions, has been instrumental in keeping me on the right path. I am truly grateful for your unwavering support.

To my dear friends, thank you for allowing me to share my challenges and for being there for me. Your support, encouragement, and understanding have been invaluable during this journey.

I would also like to express my gratitude to my blockmates for their knowledge sharing and allowing me to learn from their experiences. Your camaraderie and collaborative spirit have greatly enriched my understanding.

In conclusion, I want to express my heartfelt appreciation to everyone who has been part of this incredible journey. Your unwavering support, encouragement, and belief in me have been invaluable. Thank you all for everything.