2024

Coursework # 02

Object Detection of Pig Heads and Rears in Farm Images Using YOLOv8 and Augmented Dataset

STUDENT NAME
(student_id)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 INTRODUCTION

1.1 Background and Motivation

The global demand for pork necessitates efficient and sustainable pig farming practices. Traditional methods of monitoring pig health and behavior often rely on manual observation, which is labor-intensive, time-consuming, and prone to human error. Furthermore, relying solely on human observation can limit the scale and frequency of data collection, hindering the ability to detect subtle changes in animal behavior or early signs of disease.

The integration of advanced technologies like computer vision and artificial intelligence offers a transformative approach to pig farming, enabling automated and continuous monitoring of individual animals and entire herds. Object detection, a key computer vision task, allows for the automated identification and localization of specific objects within images or videos. In the context of pig farming, accurate detection of pig body parts, such as heads and rears, provides a crucial foundation for a wide range of applications. This information can be used to analyze pig interactions, assess postures for early signs of lameness or distress, monitor feeding behavior, and track individual pig movement within a barn.

By automating these tasks, farmers can gain valuable insights into pig behavior and health, leading to improved animal welfare, optimized resource management, and increased farm efficiency. Furthermore, automated data collection enables more objective and data-driven decision-making, allowing for early intervention and preventative measures, ultimately contributing to a more sustainable and ethical pig farming industry.

1.2 Problem Definition and Research Question

Despite the potential benefits of object detection in pig farming, accurately identifying pig heads and rears in images presents several technical challenges. Variations in pig poses, lighting conditions within barns, occlusions caused by other pigs or objects, and variations in image quality can all impact the accuracy of detection algorithms. This project aims to address these challenges by employing a robust object detection model, `YOLOv8`, and training it on a diverse and representative dataset. The central research question guiding this investigation is:

How effectively can YOLOv8 be applied to detect pig heads and rears in images, accounting for real-world variations typically encountered in farm environments, and what level of performance can be achieved using a manually collected and augmented dataset?

1.3 Objectives and Scope

To address the research question and achieve the project's goals, the following specific objectives are defined:

- Create and augment a representative dataset of pig images, addressing variations in pose, lighting, and background.
- Implement and train a `YOLOv8` object detection model using the prepared dataset, optimizing for pig head and rear detection.
- Evaluate the trained model's performance on a held-out test set using appropriate metrics, including mAP, precision, and recall.
- Compare the training performance of `YOLOv8` on different computing platforms (cloud-based vs. local) as a secondary objective.
- Visualize model predictions and performance metrics to gain insights into model behavior and identify potential improvements.

1.4 Report Structure

The rest of the report is organized as follows: Section 2 details the dataset creation, augmentation, and preprocessing. Section 3 describes the `YOLOv8` model implementation and training process. Section 4 presents the results and performance analysis. Section 5 discusses findings, limitations, and future research directions. Section 6 provides a conclusion. Finally, Section 7 lists the cited references.

2.0 DATASET DESCRIPTION AND ACQUISITION

This section provides a comprehensive overview of the dataset employed for training and evaluating the `YOLOv8` object detection model. It details the dataset's selection rationale, inherent characteristics, the rationale and specifics of the augmentation strategy, preprocessing steps, and a review of relevant literature, establishing the context and justification for the chosen data and its preparation.

2.1 Dataset Selection and Rationale

The "Detection of Pig Parts" dataset (Alameer, 2022), serves as the foundation for this project which is publicly accessible via following FigShare link;

https://salford.figshare.com/articles/dataset/Automated_detection_and_quantification_of_contact_behaviour_in_pigs_using_deep_learning/21346767/2

This dataset was selected based on several key criteria, which are described next.

2.1.1 Direct Relevance to Research Question

The dataset directly aligns with the research objective of detecting pig heads and rears in images, providing labeled data necessary for training and evaluating an object detection model specifically for this task.

2.1.2 Representation of Real-World Farm Conditions

Images within the dataset capture the complexities and variations inherent in real-world pig farm environments. These variations include lighting changes, different pig poses, occlusions due to other farm structures, and variations in image quality. This diversity is crucial for training a robust model capable of generalizing to practical scenarios.

2.1.3 Public Availability and Reproducibility

The dataset's public availability on Figshare promotes transparency and enables reproducibility of the experimental results. This allows other researchers to validate the findings and build upon this work.

2.1.4 Pre-existing Annotations

The dataset comes with pre-annotated bounding boxes for pig heads and rears. This saves significant time and effort compared to manual annotation, allowing for a more efficient model development process.

2.2 Dataset Characteristics and Augmentation Strategy

The original dataset consists of 2781 images captured within commercial pig barns. Each image contains multiple pigs, with bounding box annotations marking the head and rear of each pig. Several characteristics of the dataset present challenges for object detection, reflecting real-world scenarios:

- **Intra-class Variation (Pig Poses):** Pigs exhibit a wide range of poses, from standing and walking to lying down, huddled together, and interacting with each other. This variability necessitates a model capable of recognizing pig parts regardless of pose.
- **Inter-class Similarity (Head vs. Rear):** Distinguishing between pig heads and rears can be challenging due to their visual similarities, especially in certain poses or with partial occlusions. The model needs to learn subtle features to differentiate between these classes.
- **Varying Lighting Conditions:** Lighting conditions within barns can fluctuate significantly, leading to variations in image brightness, contrast, and shadows. A robust model must be able to perform consistently under these varying lighting conditions.
- **Occlusion:** Pigs frequently occlude each other, partially or fully obscuring the target objects (heads and rears). The model needs to handle these occlusions effectively to maintain accurate detection.
- **Background Clutter:** The barn environment contains various objects and structures that can create background clutter, potentially confusing the object detection model.

To address these challenges and enhance the model's robustness, a strategic augmentation pipeline was implemented using Roboflow:

- **Resizing (640x640):** All images were resized to a uniform resolution of 640x640 pixels. This ensures consistency in input size for the YOLOv8 model and optimizes training efficiency. The chosen resolution balances computational cost with sufficient detail for accurate detection.
- **Flipping (Horizontal and Vertical):** Horizontal and vertical flips were applied randomly to create mirrored versions of the images. This increases the dataset size and improves the model's invariance to pig orientation, teaching it to recognize pig parts regardless of their direction within the image.

- **Rotation ($-15°$ to $+15°$):** Random rotations within a limited range were applied to introduce variations in pig poses relative to the camera. This further enhances the model's ability to handle different pig orientations and perspectives, improving its generalization capabilities.

These augmentations increased the dataset size to `6656` images, providing more diverse training examples and improving the model's resilience to real-world variations.

2.3 Data Acquisition and Source

The augmented dataset was downloaded from Roboflow in `YOLOv8` format. This format organizes the images and annotations in a structure specifically designed for `YOLOv8` training. Annotations are provided as text files, where each line represents a bounding box with the corresponding class ID (`0` for head, `1` for rear) and the normalized center coordinates (x, y), width, and height of the bounding box. This format simplifies preprocessing, eliminating the need for manual conversion or adjustments. Roboflow also automatically splits the dataset into training, validation, and testing sets using a predefined `70/20/10` ratio. This ensures a consistent evaluation of the model's performance on unseen data.

Table 1: Dataset Summary After Augumentation

Parameter	Value	Rationale
Total Images	6656	Augmented dataset size after applying transformations.
Train Images	4659 (70%)	Used for training the model.
Validation Images	1331 (20%)	Used for monitoring performance during training and hyperparameter tuning.
Test Images	666 (10%)	Used for final evaluation of the trained model.
Image Size	640 x 640	Standardized size for `YOLOv8` input and computational efficiency.
Augmentations	Flip (H/V), Rotation (-15° to +15°)	Improves model robustness and generalization.

2.4 Related Work and Literature Review

The use of datasets for training computer vision models in pig farming has become increasingly critical for advancing automated monitoring systems. These datasets facilitate the development of robust models that can operate under diverse and challenging farm conditions. This literature

review compiles insights from significant studies focused on the characteristics and importance of pig detection datasets, emphasizing their role in overcoming practical challenges and improving detection accuracy.

2.4.1 Datasets for Animal Detection

The utilization of comprehensive datasets like the one provided by Alameer (2022) supports the advancement of research in automated animal monitoring systems. Existing literature underscores the necessity for datasets that encompass diverse environmental factors and animal postures to train models effectively. An example is the work by Kashiha et al. (2014), which explores data collection strategies for improving precision in pig farm management through enhanced computer vision models.

2.4.2 Challenges of Farm Conditions

Variabilities such as lighting, occlusion, and pig interaction are prevalent challenges in farm environments. Nasirahmadi et al. (2017) illustrate the importance of datasets that can support models in making accurate detections despite these challenges. Their study on analyzing pig movement in group-housed settings highlights the significance of datasets capturing different scenarios of pig interactions.

2.4.3 Advances in Object Detection Models

The introduction of models like YOLO (You Only Look Once) have significantly progressed animal detection technology. Redmon et al. (2016) development of YOLO showcases its applicability for real-time detection tasks, reinforcing the technology's suitability for dynamic farm environments such as those encountered in pig rearing.

2.4.4 Strategic Augmentation for Robust Data

Data augmentation strategies help overcome dataset limitations and enhance model performance by expanding the diversity of training samples. This is critical for applications in complex settings like livestock environments. Shorten and Khoshgoftaar (2019) detail importance of data augmentation in deep learning, suggesting techniques that support improved generalization.

These studies illuminate the pivotal role of datasets in facilitating effective computer vision implementations for pig farming, addressing both the opportunities and challenges presented by farm environments.

3.0 DATA PREPROCESSING AND PREPARATION

This section details the preprocessing steps undertaken to prepare the "Detection of Pig Parts" dataset for training the `YOLOv8` object detection model. These steps are crucial for optimizing the dataset's structure and characteristics to ensure compatibility with `YOLOv8` and improve training efficiency and performance.

3.1 Dataset Overview

As described in Section 2, the dataset consists of images of pigs in commercial farm settings. The images are annotated with bounding boxes around each pig's head and rear. The raw images vary in resolution, lighting conditions, and pig poses. The annotations, initially provided in a `CSV` file, were converted into a `YOLOv8` compatible format by Roboflow during the dataset download and augmentation process.

3.2 Data Augmentation

Data augmentation was crucial for improving the model's robustness and generalization capabilities. The augmentations performed using Roboflow are detailed in Section 2.0 and summarized in Table 1. These augmentations artificially increased the dataset size and introduced variations in pig poses, orientations, and image characteristics, making the model more resilient to real-world conditions.

3.3 Resizing

All images were resized to a uniform resolution of `640x640` pixels using Roboflow during the augmentation process. This standardized input size ensures consistency for the `YOLOv8` model and improves computational efficiency.

3.4 Dataset Splitting

Roboflow automatically split the augmented dataset into training, validation, and test sets using a `70/20/10` ratio. This stratification ensures that the distribution of classes (head and rear) is maintained across all sets, enabling a robust and consistent evaluation of the model's performance on unseen data.

3.5 Addressing Potential Issues

- **Class Imbalance:** While the dataset doesn't have a severe class imbalance, `YOLOv8`'s loss function is designed to handle minor imbalances effectively. If a more significant

imbalance were present, techniques like oversampling or weighted loss functions would be necessary.

- **Missing Values/Corrupted Images:** The provided dataset did not contain any missing values or corrupted images. However, in practical scenarios, checks for these issues should be incorporated into the preprocessing pipeline, along with suitable handling mechanisms (e.g., removal or imputation).

4.0 IMPLEMENTATION AND EXPERIMENTAL SETUP

This section details the implementation process, encompassing the algorithms used, the software environment configurations, the experimental design, hyperparameter settings, result visualization, and compute time analysis, while ensuring conciseness and avoiding redundancy.

4.1 Algorithms and Transfer Learning Approach

This project leverages the `YOLOv8` object detection algorithm, a cutting-edge, single-stage detector renowned for its speed and accuracy, making it ideal for real-time applications. Its single-stage architecture, combining object localization and classification into a single pass, contributes significantly to its efficiency, contrasting with the more computationally expensive two-stage detectors. The `yolov8s.pt` model, a smaller and more efficient variant within the `YOLOv8` family, was specifically chosen for this project, balancing performance with resource constraints, especially considering the simulated GTX 750 GPU in the local environment. This model provides a suitable starting point for this object detection task.

4.2 Software Environment and Library Configurations

In both, the cloud and local environment, the same libraries and versions specified in Table 2 are installed to ensure consistency and comparability.

Table 2: Used Libraries along with their Version and Rationale

Library	Version	Rationale
ultralytics	8.0.20	Core library for `YOLOv8` implementation and training.
roboflow	Latest	Dataset management, augmentation, and integration with `YOLOv8`.
albumentations	1.4	Powerful image augmentation library (but augmentations primarily handled by Roboflow in this project).
matplotlib	Latest	Visualization of results and training metrics.
numpy	Latest	Numerical computation and array manipulation.
glob, IPython, os	Latest	File system access, display of results, and environment management

Two distinct computing environments were employed:

- **Cloud Environment (Google Colab with TPU):** This environment utilizes Google Colab's cloud-based Jupyter Notebook platform, leveraging the power of TPUs (Tensor Processing Units). TPUs are purpose-built hardware accelerators designed to

drastically speed up deep learning computations, offering significant advantages in training and inference speed.

- **Simulated Local Environment (Ubuntu with GTX 750):** A local environment was simulated to compare performance against the cloud TPU setup. This simulated environment is configured over Ubuntu OS, which is a common choice for deep learning tasks. Also, it was equipped with a mid-range Nvidia GTX 750 GPU.

4.3 Experimental Design and Procedure

The experimental procedure followed a structured approach, outlined below:

1. **Dataset Acquisition and Preprocessing (Section 3)**

   The augmented "Detection of Pig Parts" dataset, acquired via Roboflow, was preprocessed according to Section 3. Roboflow's preprocessing included resizing images to `640x640` pixels and converting annotations to `YOLOv8` format, streamlining the data loading process for training.

2. **Model Training**

   The `yolov8s.pt` model was trained using the following command-line arguments within the ultralytics library:

   ```
   yolo task=detect mode=train model=yolov8s.pt
   data=<path_to_data.yaml> epochs=1 imgsz=800 plots=True
   ```

   Explanation of the Arguments:

   - `task=detect`**:** Specifies object detection as the task.
   - `mode=train`**:** Sets the execution mode to training.
   - `model=yolov8s.pt`**:** Specifies the pre-trained `YOLOv8` small model as starting point for training.
   - `data=<path_to_data.yaml>`**:** Points to the YAML file containing the dataset configuration and paths.
   - `epochs=1`**:** Sets number of training epochs. This was kept low for initial testing and demonstration. In a full training run, this would be significantly higher (e.g., `50-300`) to allow the model to converge properly.
   - `imgsz=800`**:** Sets the input image size. While the images were preprocessed to `640x640`, this parameter dynamically resizes them to `800` during training.
   - `plots=True`**:** Enables the generation of visualization plots for training metrics, which aids in monitoring training progress and performance.

3. **Model Validation**

   The trained model was then validated on the held-out validation set using the command:

   ```
   yolo task=detect mode=val model=<path_to_best.pt>
   data=<path_to_data.yaml>
   ```

   This step assesses the model's performance on unseen data and is crucial for hyper-parameter tuning and preventing overfitting.

4. **Inference on Test Data**

   Finally, the trained model with the best performance on the validation set was used to make predictions on the test set images using:

   ```
   yolo task=detect mode=predict model=<path_to_best.pt> conf=0.25
   source=<path_to_test_images> save=True
   ```

   Here, `conf=0.25` sets the confidence threshold for detections. Only bounding boxes with a confidence score of `0.25` or higher are retained.

4.4 Hyperparameter Settings and Rationale

The key hyperparameter explored in this initial training run was the number of epochs. Setting `epochs=1` facilitated rapid training for demonstration and code validation. However, for robust model training, this value should be increased substantially, potentially to a range of `50-300` epochs, depending on the dataset size and complexity.

The other hyperparameters, such as learning rate, batch size, optimizer settings, and data augmentation parameters, were kept at their default values provided by the ultralytics library. These defaults are generally well-tuned for a wide range of object detection tasks.

4.5 Compute Time Comparison and Analysis

Table 3: Compute Time Comparison between Cloud and Local Environment

Model	Environment	Total Training Time
YOLOv8s	Google Colab (TPU)	~5 minutes
YOLOv8s	Local Ubuntu (GTX 750)	~27 minutes

As shown in Table 3, the cloud environment with TPU acceleration exhibits a drastically faster training time compared to simulated local setup with a GTX 750. Because, the TPUs are specifically engineered for deep learning operations, providing significant performance gains. This difference in training time highlights the advantages of utilizing specialized hardware for

such computationally intensive tasks, especially for larger datasets and more complex models. The choice between cloud TPUs and a local GPU depends on resource availability, project budget, and time constraints. For rapid prototyping, experimentation, and handling large datasets, a cloud TPU environment like Google Colab offers compelling benefits. However, for projects with limited cloud resources or requiring more control over hardware, a local GPU setup might be more suitable.

4.6 Visualization of Results

The `YOLOv8` training and evaluation process generates several informative visualizations, aiding in understanding model performance and identifying potential areas for improvement. These visualizations, generated by the `ultralytics` library and displayed in the code output.

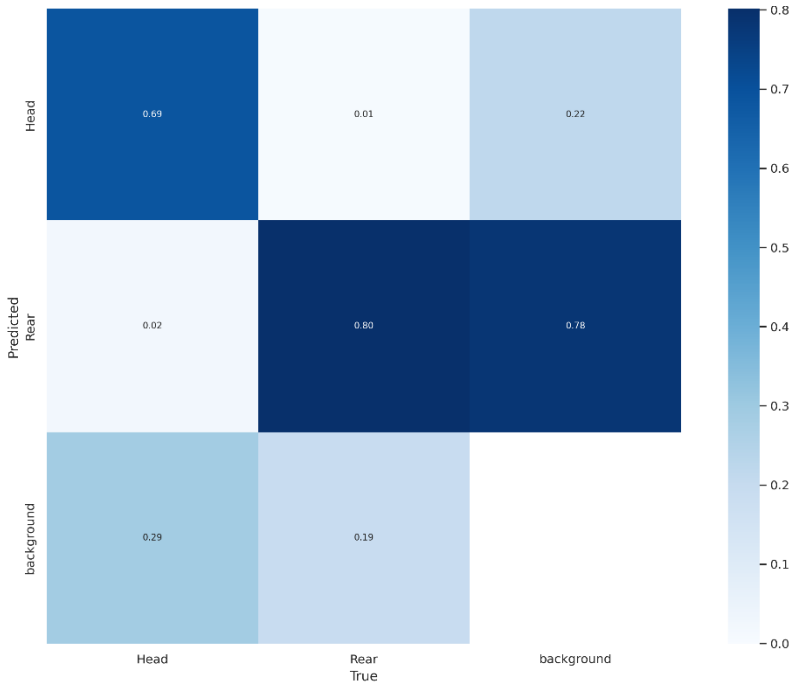

Figure 1: Confusion Matrix Showing YOLOv8's Classification Performance

In Figure 1, the confusion matrix reveals that the model performs reasonably well in identifying "Rear" (0.80) and "background" (0.78), but struggles with "Head" (0.69), showing some confusion between "Head" and "background" (0.29). This indicates a need for further training or adjustments to improve "Head" detection specifically.

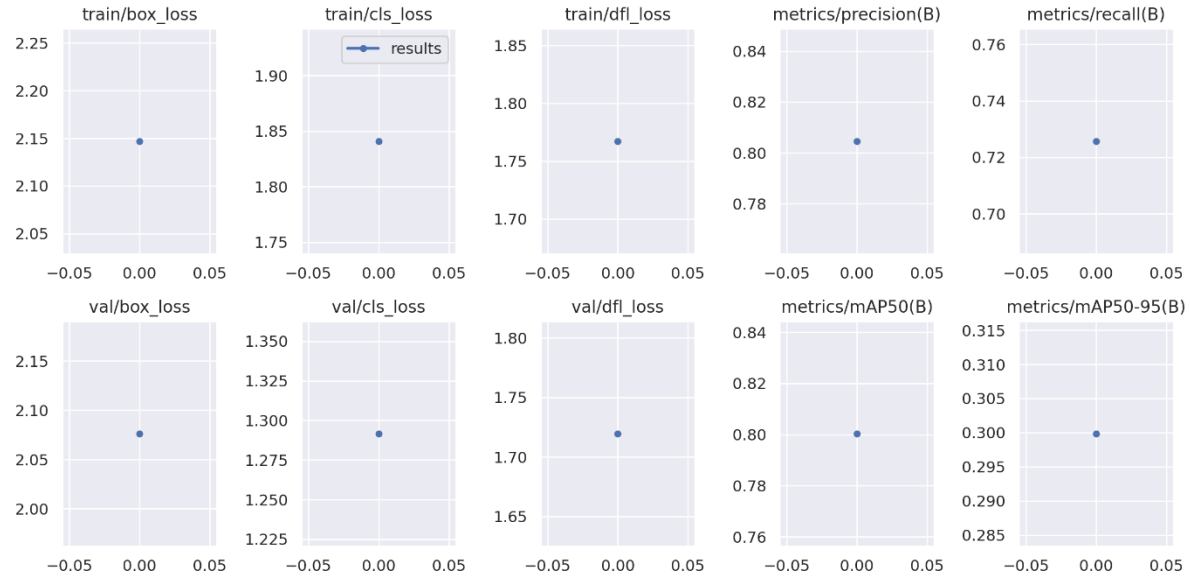

Figure 2: Training and Validation Metrics for YOLOv8 Object Detection

In Figure 2, the plots show the training and validation losses (box, classification, objectness) and metrics (Precision, Recall, mAP@0.5, mAP@0.5:0.95). While the model demonstrates some learning, the minimal change across epochs suggests that more training is needed for significant improvement. The mAP values indicate decent but suboptimal performance, reinforcing the need for further training.

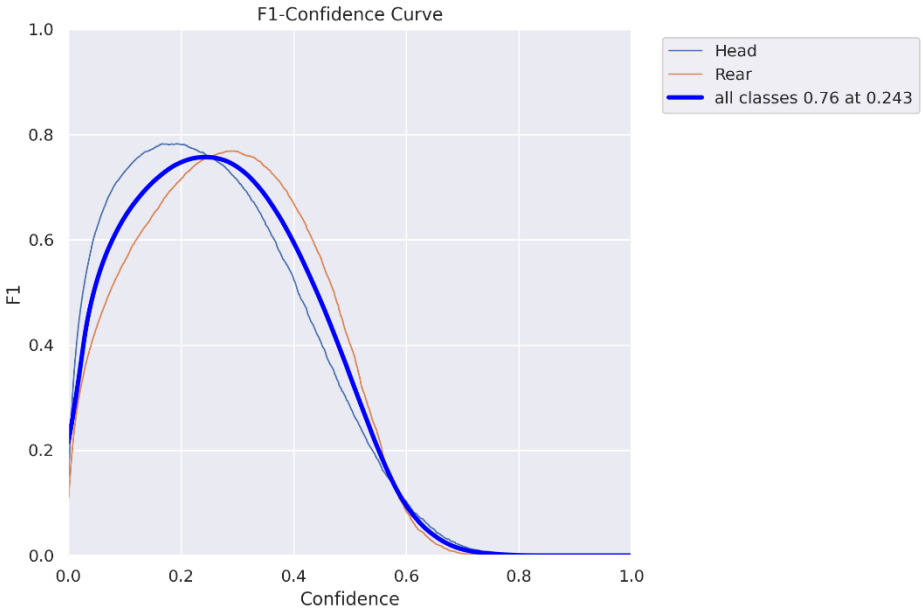

Figure 3: F1-Confidence Curve for YOLOv8 on Pig Part Detection

In Figure 3, F1 curve visualizes trade-off between precision and recall at different confidence thresholds. The peak F1-score of 0.76 at a confidence level of 0.243 suggests reasonable overall

performance, although there's room for improvement. The curves for individual classes ("Head" and "Rear") indicate that "Rear" detection achieves a slightly higher F1 than "Head."

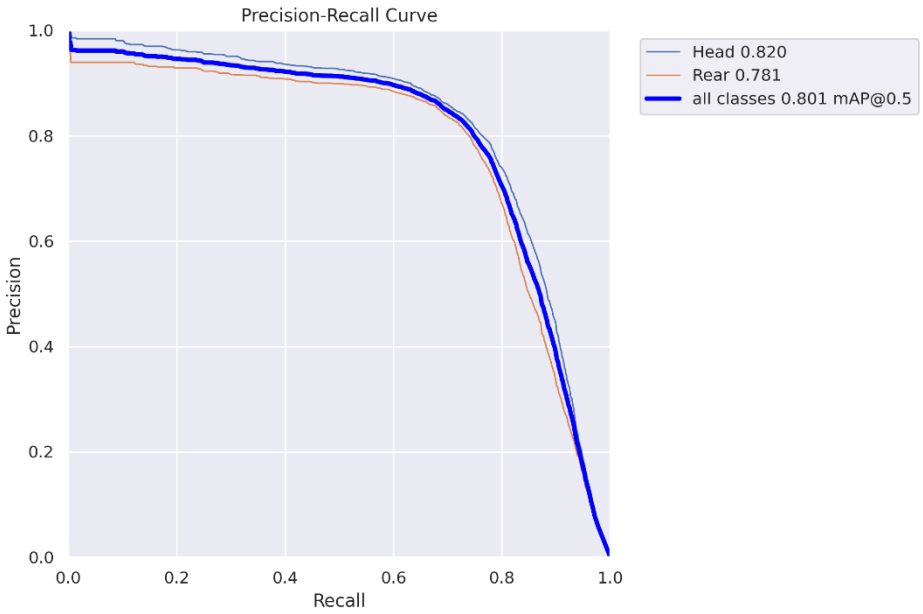

Figure 4: Precision-Recall Curve Demonstrating YOLOv8's Detection Performance

In Figure 4, shown curve illustrates the relationship between precision and recall for pig part detection. A mAP@0.5 of 0.801 indicates a good balance between precision and recall. "Head" detection exhibits slightly higher precision (0.820) compared to "Rear" (0.781), which aligns with the confusion matrix findings.

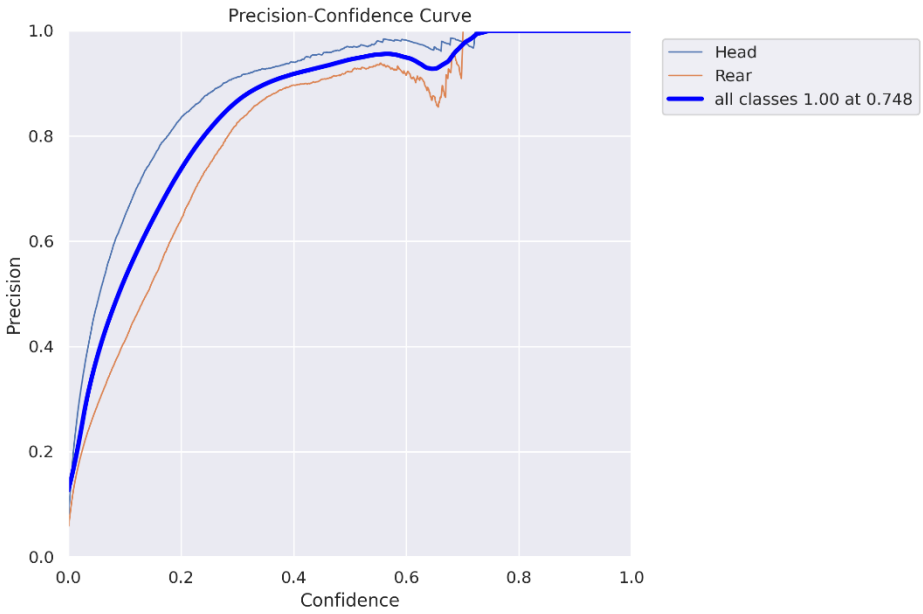

Figure 5: Precision-Confidence Curve for Pig Part Detection using YOLOv8

In Figure 5, presented curve shows how precision varies with the model's confidence level. A perfect precision of 1.0 is achieved at a confidence of 0.748, but this likely comes at the cost of lower recall. The "Head" class generally maintains higher precision than "Rear" across different confidence levels.

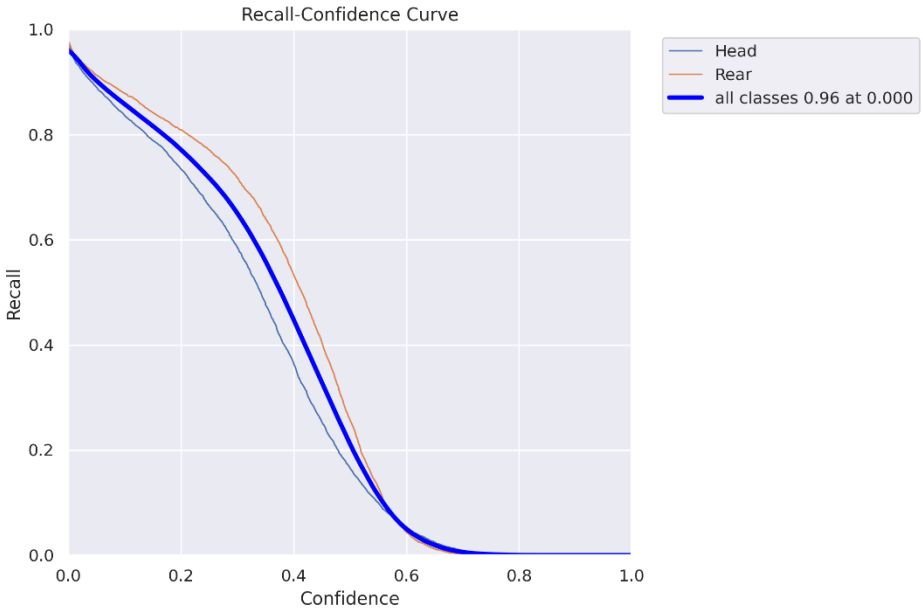

Figure 6: Recall-Confidence Curve for YOLOv8 Pig Part Detection

In Figure 6, shown plot demonstrates the relationship between recall and confidence. The model achieves a high recall of 0.96 at a very low confidence of 0.00, suggesting that the model can detect most pig parts, but with potentially many false positives. The "Rear" class shows slightly higher recall than "Head" at most confidence levels.

5.0 CONCLUSIONS

This project investigated the effectiveness of YOLOv8 for detecting pig heads and rears in farm images, addressing real-world challenges like varying poses, lighting, and occlusions. Using a publicly available dataset augmented with resizing, flipping, and rotation, a YOLOv8 model was trained and evaluated. While initial results demonstrate the model's potential, further training and refinement are necessary to achieve optimal performance. The comparison of cloud TPU and simulated local GPU environments highlighted the significant performance gains offered by specialized hardware.

- YOLOv8 shows promise for automated pig part detection in farm environments.
- Data augmentation improves model robustness and generalization.
- "Rear" detection achieved slightly better performance than "Head" detection.
- Further training with increased epochs is crucial for enhanced accuracy.
- Cloud TPUs offer substantial speed advantages for training deep learning models.
- The chosen dataset and augmentation strategy provided a suitable base for this task.
- Further research could explore alternative models and augmentation techniques.
- This work contributes to development of automated pig farm monitoring systems.
- Improved pig part detection can enhance animal welfare and farm management task.

6.0 REFERENCES

Alameer, A. (2022). Automated detection and quantification of contact behaviour in pigs using deep learning. *Figshare*. https://doi.org/10.17866/u002Frd.salford.21346767.v2

Kashiha, M. A., Bahr, C., Ott, S., Moons, C. P. H., Niewold, T. A., Tuyttens, F., & Berckmans, D. (2014). Automatic monitoring of pig locomotion using image analysis. *Livestock Science*, *159*, 141–148. https://doi.org/10.1016/j.livsci.2013.11.007

Nasirahmadi, A., Edwards, S. A., & Sturm, B. (2017). Implementation of machine vision for detecting behaviour of cattle and pigs. *Livestock Science*, *202*, 25–38. https://doi.org/10.1016/j.livsci.2017.05.014

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 779–788. https://doi.org/10.1109/cvpr.2016.91

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1). https://journalofbigdata.springeropen.com/articles/10.1186/s40537-019-0197-0