5-2-2023

ORGAN LOCALIZATION AND DETECTION IN SOW’S USING MACHINE LEARNING AND DEEP LEARNING IN COMPUTER VISION

Iyad Almadani

ORGAN LOCALIZATION AND DETECTION IN SOW'S USING MACHINE LEARNING AND DEEP LEARNING IN COMPUTER VISION

by

Iyad Mohd Eid H Almadani

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

Major: Electrical And Computer Engineering

The University of Memphis

May 2023

Acknowledgements

First and foremost, All praises and gratitude to Allah who gives me the strength to accomplish this degree. This thesis is dedicated to my sacrificer mother, my supportive brother Murad as well as my educator and virtuous father, MOHAMMAD EID, who has passed away(may Allah elevate his rank in paradise).

Additionally, I want to express my most profound appreciation to my friend Mohammed Abu Hussein, and I am also indebted to my supervisor Dr. Aaron L Robinson for always having my back.

Abbreviations

Abbreviations	
Abbreviation	Meaning
SVM	support vector machine
HOG	Histogram of Oriented Gradients
PCA	Principle Component Analysis
CNN	Convolutional Neural Network
YOLO	You Only Look Once
HSV	Hue, Saturation, and value
IOU	Intersection Over Union
LWIR	Long-Wave Infrared
ROI	Regions Of Interest

Thesis Statement and Contribution Summary

Farming and animal husbandry has benefited from recent developments in automated harvesting, scientific studies in plant genomes, and recent reproductive cycle research. However, even with these recent advances, the industry is still heavily dependent upon human specialists with many years of experience to limit the costs associated with insemination and to maximize the probability of pregnancy associated with insemination. To address this issue, this thesis poses the question of how can the dependence on human experts be decreased through the application of machine learning techniques? To begin to answer this question, this treatment focuses on object detection and segmentation using thermal images. Specifically, this research focuses on the problem of detecting the sow's vulva. This thesis also poses the question of whether current research results can contribute to an automated process of determining the estrus cycle and ovulation in sows? This thesis assembles and presents research and results that begin to address these questions.

The contributions of this thesis may be summarized as follows:

1. Development of machine learning techniques to reduce the farming industry dependence on human experts.
2. Development of segmentation results for vulva detection and localization.
3. Derivation of a new feature extractor based on concatenating two feature extractors.
4. Evaluation of traditional semi-supervised ML techniques on infrared images

Table of Contents

List of Figures

List of Tables

Abstract

The objective of computer vision research is to endow computers with human-like perception to enable the capability to detect their surroundings, interpret the data they sense, take appropriate actions, and learn from their experiences to improve future performance. The area has progressed from using traditional pattern recognition and image processing technologies to advanced techniques in image understanding such as model-based and knowledge-based vision. In the past few years, there has been a surge of interest in machine learning algorithms for computer vision-based applications. Machine learning technology has the potential to significantly contribute to the development of flexible and robust vision algorithms that will improve the performance of practical vision systems with a higher level of competence and greater generality. Additionally, the development of machine learning-based architectures has the potential to reduce system development time while simultaneously achieving the above-stated performance improvements. This work proposes utilizing a computer vision-based approach that leverages machine and deep learning systems to aid the detection and identification of sow reproduction cycles by segmentation and object detection techniques. A lightweight machine learning system is proposed for object detection to address dataset collection issues in one of the most crucial and potentially lucrative farming applications. This technique was designed to detect the vulvae region in pre-estrous sows using a single thermal image. In the first experiment, the support vector machine (SVM) classifier was used after extracting features determined by 12 Gabor filters. The features are then concatenated with the features obtained from the Histogram of oriented gradients (HOG) to produce the results of the first experiment. In the second experiment, the number of distinct Gabor filters used was increased from 12 to 96. The system is trained on cropped image windows and uses the Gaussian pyramid technique to look for the vulva in the input image. The resulting process is shown to be lightweight, simple, and robust when applied to and evaluated on a large number of images. The results from extensive qualitative and quantitative testing experiments are included. The

experimental results include false detection, missing detection, and favorable detection rates. The results indicate state-of-the-art accuracy. Additionally, the project was expanded by utilizing the ”You Only Look Once” (YOLO) deep learning Object Detection models for fast object detection. The results from object detection have been used to label images for segmentation. The bounding box from the detected area was systematically colored to achieve the segmented and labeled images. Then these segmented images are used as custom data to train U-Net segmentation.

The first step involves building a machine-learning model using Gabor filters and HOG for feature extraction and SVM for classification. The results discovered the deficiency of the model, therefore a second stage was suggested in which the dataset was trained using YOLOv3-dependent deep learning object detection. The resulting segmentation model is found to be the best choice to aid the process of vulva localization. Since the model depends on the original gray-scale image and the mask of the region of interest (ROI), a custom dataset containing these features was obtained, augmented, and used to train a U-Net segmentation model. The results of the final approach show that the proposed system can segment the sow's vulva region even in low-rank images and has excellent performance efficiency. Furthermore, the resulting algorithm can be used to improve the automation of estrous detection by providing reliable ROI identification and segmentation and enabling beneficial temporal change detection and tracking in future efforts.

Chapter 1

Introduction

Although significant progress has been made toward more precise object localization in recent years, state-of-the-art object detectors have become increasingly expensive. For example, one of the latest Amoeba Net-based NASFPN detector [29] requires 167M parameters and 3045B FLOPs to achieve state-of-the-art accuracy (30x more than Retina Net [21]). Large model sizes and high processing costs prevent them from being used in many real-world applications where model size and latency are critical, such as robotics and self-driving cars. As a result of these real-world resource constraints, model efficiency becomes increasingly important for object detection. Several previous studies attempted to develop more efficient detector architectures, such as one-stage [21, 22, 28, 29] and anchor-free detectors [18, 39, 44], as well as compress current models [14, 23]. In the previously mentioned examples, the additional procedural efficiency is frequently accompanied by a requisite sacrifice in accuracy. Furthermore, most previous research has focused on a single or limited set of resource requirements. This holds true despite the fact that a large array of real-world applications, ranging from mobile devices to data centers, frequently necessitate a wide variety of resource limits. Object detection in animal farming is one of these applications [2]. Humans have devised several algorithms and approaches in this area to support farming industry production goals.

Farmers breed animals for various reasons, including increased productivity, disease resistance, effective reproduction, and climate adaptation, with heat and drought being the most common. Highly productive animals use a more significant portion of their nutritional intake to produce desired items such as milk or meat rather than simply maintaining their bodies. Increased longevity and reproductive success mean that feed calories are concentrated among the most productive animals, lowering the number of replacement animals needed in a herd. In recent years, technology has made this much more manageable. Therefore, models were built to make use of the recent advances in machine learning and their associated libraries and methodologies. However, despite the recent inventions of advanced machine learning techniques (e.g. [1, 3, 16, 17, 32, 38, 45]), the majority of these models require a large amount of training data, are computationally expensive to train, or are accompanied by model dependent inference processes that are prohibitively long.

Additionally, applying these advanced techniques to problems where datasets are scarce, such as sow's vulva detection, is largely unachievable even after significant data augmentation efforts. Therefore, building a lightweight model with enhanced accuracy as an intuitive solution to this problem is complex and was established as one of the goals of this research. Specifically, the model requires a lightweight machine and deep learning classifier that requires annotated or segmented dataset prior to training. In an effort to address these issues, initial efforts attempted to make use of transfer learning or fine-tune the pre-trained model. The problem with these approaches is that most of the existing models were trained on and applicable to RGB-based color images. The proposed estrous detection model in this treatment is based on thermal images and thus the number of channels available for feature detection and extraction is effectively reduced by a factor of 3. The direct effect of the channel reduction is decreased performance for thermal imagery even after fine-tuning the model.

The summarized difficulties in object localization, application of machine learning to farming production, small datasets, and infrared transfer learning all support the goals of the research presented in this thesis. The study aims to develop a lightweight object localization model that can be used to classify pigs from other entities in a scene, such as humans. The model can also be used to detect/classify regions within a particular animal such as the vulva area of the sows. Specifically, the model will detect Regions Of Interest (ROI) where temperature changes indicate the current phase of the sow's estrous cycle.

The first step involves building a machine-learning model using Gabor filters and HOG for feature extraction and SVM for classification. The results discovered the deficiency of the model, therefore a second stage was suggested in which the dataset was trained using YOLOv3-dependent deep learning object detection. The resulting segmentation model is found to be the best choice to aid the process of vulva localization. Since the model depends on the original gray-scale image and the mask of the region of interest (ROI), a custom dataset containing these features was obtained, augmented, and used to train a U-Net segmentation model. The results of the final approach show that the proposed system can segment the sow's vulva region even in low-rank images and has excellent performance efficiency.

Chapter 2

Object Detection Using Machine Learning

This chapter will provide an in-depth discussion of a machine learning-based object detection approach applied to images captured in the infrared spectrum. Details will include how the model incorporates a support vector machine (SVM) [37] classifier after extracting features using Gabor [20] filters. An SVM is chosen in this system because it outperforms other simple supervised machine learning classifiers across a wide range of applications [41]. The system is trained on cropped image windows and uses the Gaussian pyramid technique to locate the vulva in the input image, making it lightweight, simple, and robust to test on a hugely different number of images.

2.1 Background

This section summarizes some of the current and relevant approaches to segmentation and classification in the infrared with a specific focus on applications in animal husbandry. Additionally, given the dependence of the results presented in this chapter on Histogram of Oriented Gradients (HOG), Gabor filters, and Support Vector Machines (SVMs), these topics and some of their applications in the infrared will also be covered in depth.

2.1.1 Histogram of Oriented Gradients

Machine learning is a widely suitable approach with numerous benefits to many applications. That widespread applicability results in current machine learning algorithms that evolve quickly over time. Once-optimal algorithms are frequently updated to make way for more efficient implementations. For example, a Histogram of Oriented Gradients (HOG) is an algorithm that can expand its range of applicability and evolve its originally intended purpose through the inclusion of some common machine learning algorithms. The combination of the two enables the approach, which was initially focused on human detection, to be extended to provide benefits to many other arenas. Further, this additional support is easily obtained and is one of the multiple reasons for the longevity of HOG and its associated excellent results.

The histogram of oriented gradients (HOG) [8] is a feature descriptor used in object recognition and image classification procedures in machine vision and image processing. Feature descriptors simply refer to a representation of an image that only pulls valuable information from the image and ignores the rest.

The HOG descriptor characterizes an object's structure or form by extracting the orientation and gradient of the edges because the magnitude of gradients is large around edges and corners (regions of abrupt intensity changes) and that edges and corners pack in a lot more information about object shape than flat regions. The gradients and orientations are computed in 'localized' segments. The entire image is split into smaller sub-regions, with each region's gradients and orientation determined separately. Finally, the HOG process creates distinct histograms for each of these sections.

2.1.2 Histogram of Oriented Gradients (HOG) Calculation

This section details the steps required to calculate HOG feature descriptors. To begin the illustration of the calculations required for each step, refer to the thermal image shown in figure 2.1. The image shows multiple sows separated in individual pens. It depicts an environment with characteristics typical of the initial step in the estrous detection process. The separation into individual stalls permits observation of sow activity and physical changes to physical changes to the sow's ears, head, and vulva. Thus, the need for effective segmentation and the extraction of relevant features is underscored.

Preprocessing in HOG

The pre-processing steps for the histogram of oriented gradients consist of 2 key components. The first step is to resize the image to 64 × 128 pixels. This is due to the features being extracted by splitting the image

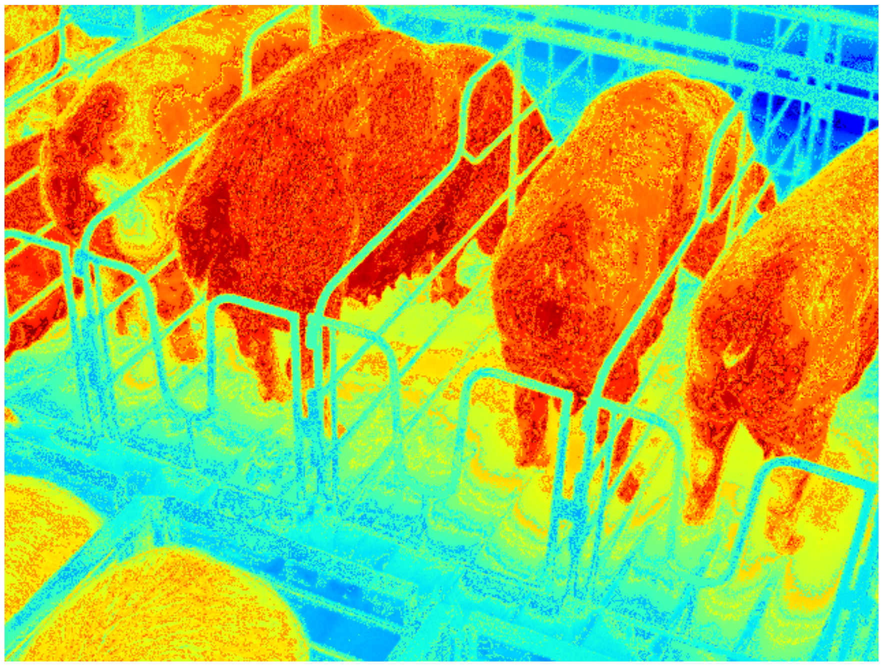


Figure 2.1: Thermal image of pigs in barn.

into 8*8 and 16*16 patches. As a result, having this size made all of our computations a lot easier.

Calculate Image Gradients

This section summarized the methods used to calculate the pixel intensity gradients in the image. Gradients quantify image intensity changes in the x and y coordinate directions. For this research, the gradients were estimated using subsections from the image as shown in figure 2.2.

For this patch, the estimated gradient values were achieved using the Sobel operator in OpenCV with kernel size 1, The same results were obtained as well in [5].

The gradient estimation process at the pixel denoted $P(x, y)$ may be summarized as follows. The pixel intensity value x_1 to the left of $P(x, y)$ should be subtracted from the pixel intensity value x_2 to the right to of $p(x, y)$ to calculate the gradient (or change) in the x-direction. In the same way, the pixel intensity value y_1) below the selected pixel should be subtracted from the pixel intensity value y_2 above it to compute the gradient in the y-direction. Hence the resultant gradients are:

Gradient in the X direction: $G(x) = x_2 - x_1$

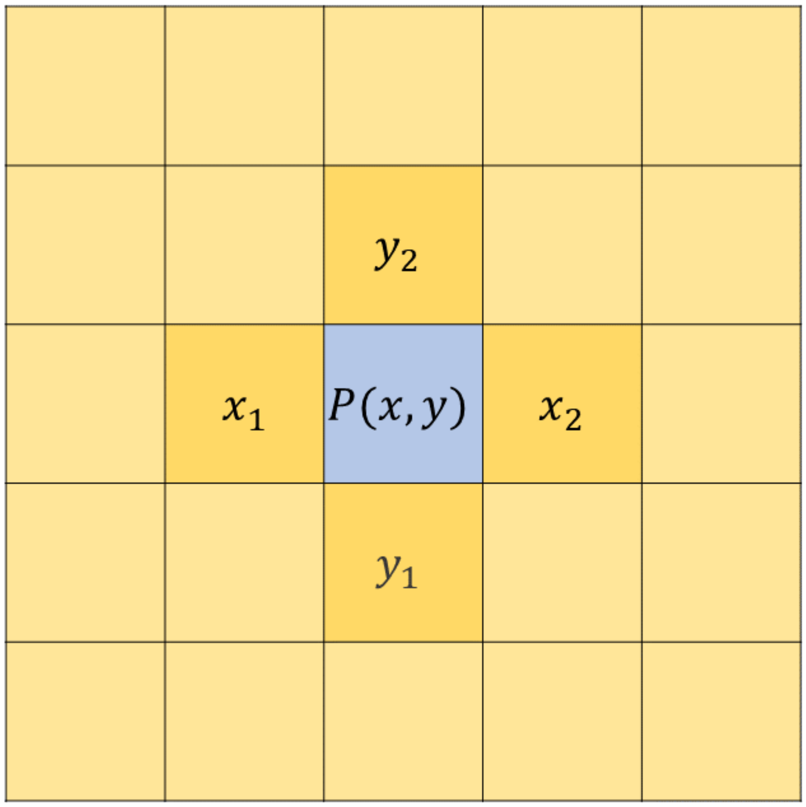


Figure 2.2: Pixel values for the patch

Gradient in the Y direction: $G(y) = y_2 - y_1$

Now two new matrices were obtained, one in the x-direction and the other in the y-direction, as a result of this operation. The same procedure was followed for each of the image's pixels. After that, the Pythagoras theorem was used to compute each pixel's gradient magnitude and direction based on the previous phase's gradients.

(Total Gradient Magnitude) $= [\sqrt{(Gx)^2} + \sqrt{(Gy)^2}]$

Next, for the same pixel, the direction θ was calculated.

$$\tan(\theta) = [Gy/Gx] \qquad \theta = atan(Gy/Gx)$$

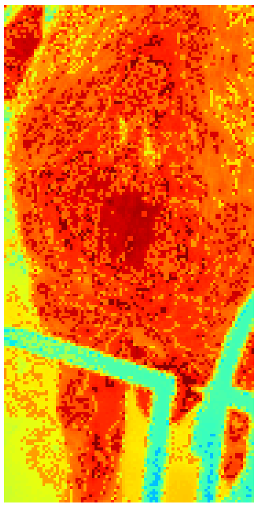


Figure 2.3: Resized image to 64*128 after cropping the RIO

Histogram Creation Using Gradients and Orientation

A histogram is a graph that can be used to depict the frequency distribution of continuous data. For the purposes of this study, the histogram is constructed with the angle or orientation on the x-axis and the frequency on the y-axis. The pixel's θ value was calculated as described above and used to indicate the orientation of the pixel and to update the frequency table. The frequency table representing angles and their presence in the image is constructed by repeating the gradient orientation calculation for all pixels under consideration and summing the number of orientations satisfying the specified number of ranges. With the specified angle range values on the x-axis and the number of orientations satisfying the criteria on the y-axis, this frequency table may be used to create a histogram.

Histogram of Gradients Calculation in 8×8 cells

Refer to the image in Figure 2.3 The image was split into 8×8 cells in this stage. A histogram of gradient orientations was created for each of the 8×8 cells according to the method summarized in the previous section. As a result, a collection of orientation histograms for the smaller patches are obtained and can then be used as features to represent the entire image.

Normalize gradients in 16×16 cell

The Histogram of Oriented Gradients (HOG) for the image's 8×8 patches was created. However, because the image's gradients are sensitive to overall lighting, the histograms had to be ”normalized” to reduce the effect of lighting differences. A 16×16 block was made by joining four 8×8 cells. Additionally, each of the 8×8 cells has an associated 9×1 histogram matrix. As a result, four 9×1 matrices or a single 36×1 was collected.

Each of these values was divided by the square root of the sum of squares of the values to normalize this matrix. Mathematically, for a given vector V:
$V = [a1, a2, a3, \dots .a36]$
The root of the sum of squares were calculated:
$L2norm = [\sqrt{(}a1)^2 + (a2)^2 + (a3)^2 + \dots .(a36)^2]$
normalize vector = (vector / L2norm)

The resultant vectors

As shown in Figure 2.4, in one 16*16 patch, there are 4 histograms, and the vectors in 8*8 patch represent the 9 bins histogram. Each bin represents a specific orientation, and the length of the vector represents the vector's magnitude.

To find the resultant vector for one 16*16 patch, we need to break down each vector into its main components. The resultant vector formula consists of two parts: magnitude and direction. As shown in Figure 2.5 where $\sum x$ is the x-component of the resultant, which is the sum of the x-components of each single vector and $\sum y$ is the y-component of the resultant, which is the sum of the y-components of each single vector. The angle of the resultant vector with the positive x-axis, can be found by taking the inverse tangent of the slope of the resultant.

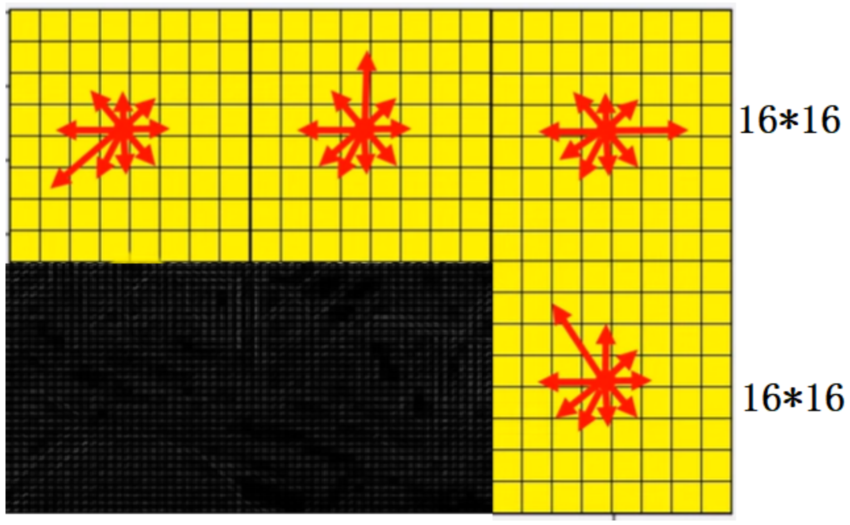


Figure 2.4: Final vectors

The Complete Image

As shown in Figure 2.6, the result ended up with a final image of 105(15 * 7) blocks, Each of these 105 blocks has an associated 36X1 vector that can be used as a descriptive feature. As a result, the total number of features in the image would be 105 x 36×1 = 3780. These features will be used later to support a machine-learning approach for classification.

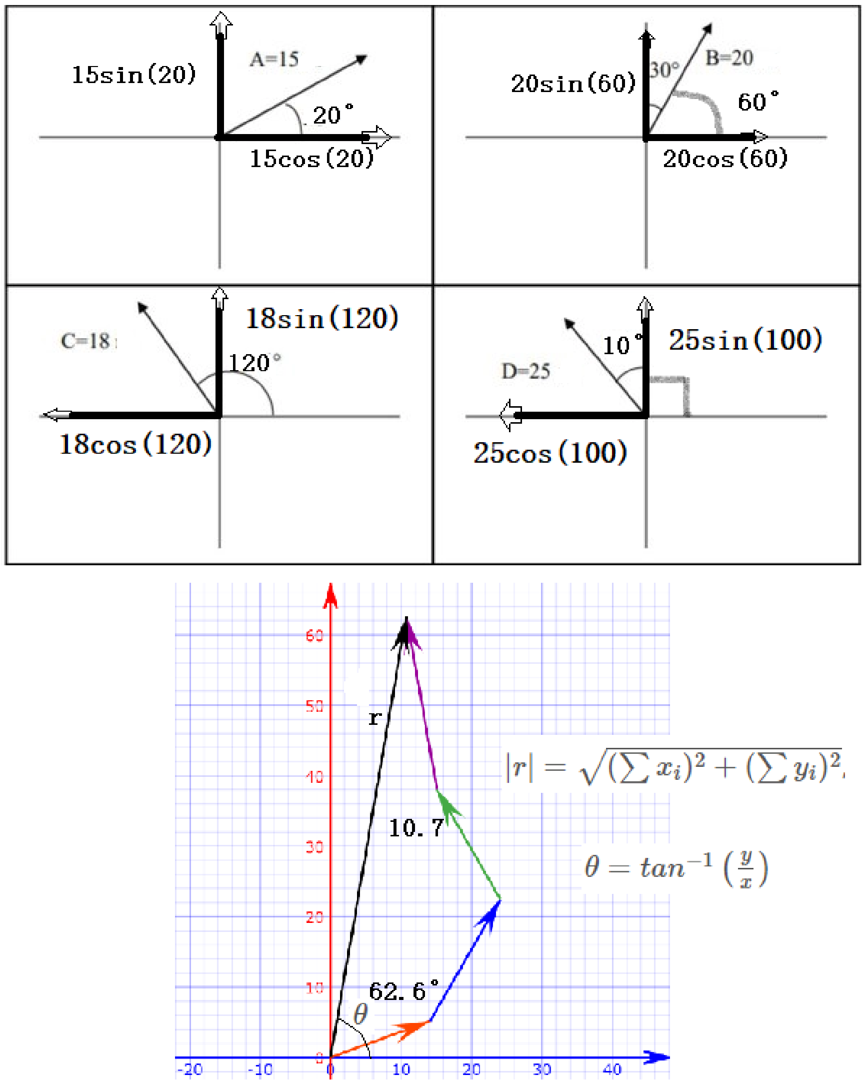


Figure 2.5: Resultant vector calculation

2.1.3 Gabor Filter

A Gabor Kernel is a convolution filter representing a combination of Gaussian and a sinusoidal term. The Gaussian component provides weights to limit the spatial extent of the kernel and the sin component provides the direction and periodicity component. Mathematically, the 2-dimensional Gabor kernel is given by:

$$g(x, y; \lambda, \theta, \phi, \gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\phi\right)$$

Where

$$x' = x\cos\theta + y\sin\theta, y' = -x\sin\theta + y\cos\theta.$$

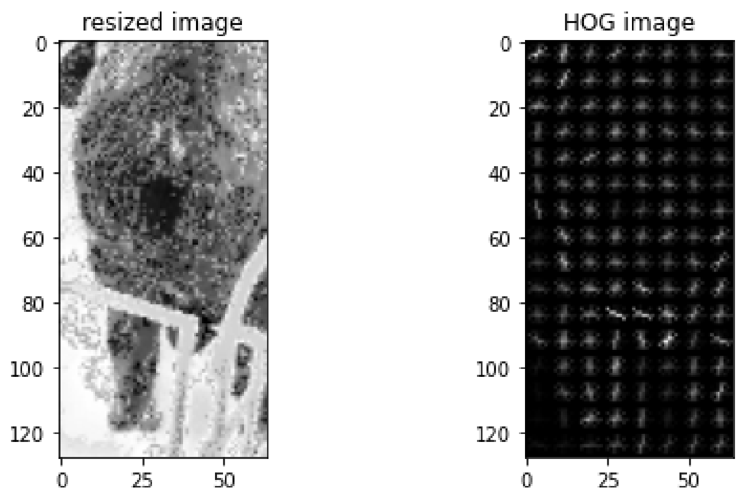


Figure 2.6: The calculated HOG descriptor for an image.

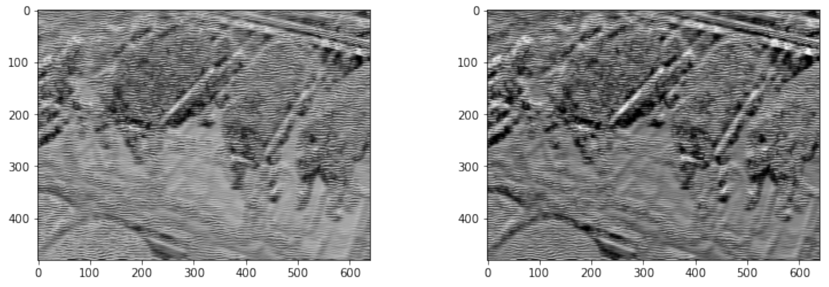


Figure 2.7: Different Gabor responses for two different Gabor kernels

The following parameters are present in the filter:

Wavelength: The number of cycles/pixel is specified by λ.

Orientation: The angle of the normal to the sinusoid is θ.

Phase: The offset of the sinusoid is ϕ.

Aspect Ratio: Ellipticity is produced with $\gamma \neq 1$.

The spatial envelope of the Gaussian, σ, is controlled by the bandwidth, which at unity gives $\sigma = 0.56\lambda$.

Figure 2.7 demonstrates two different Gabor responses for a sample image to two kernels with different σ, and γ. The concatenation of the responses acquired by the different Gabor filters was utilized to generate better texture feature maps, which will be used to enhance the classification of different regions of the sow.

2.1.4 Support Vector Machine

SVM, or SUPPORT VECTOR MACHINE, is a linear model for classification and regression issues. It can handle linear and nonlinear problems and is useful for a wide range of applications. The algorithm creates a line or hyper-plane that divides the information into classes. Figure 2.8 shows one example to classify two different classes via linear SVM.

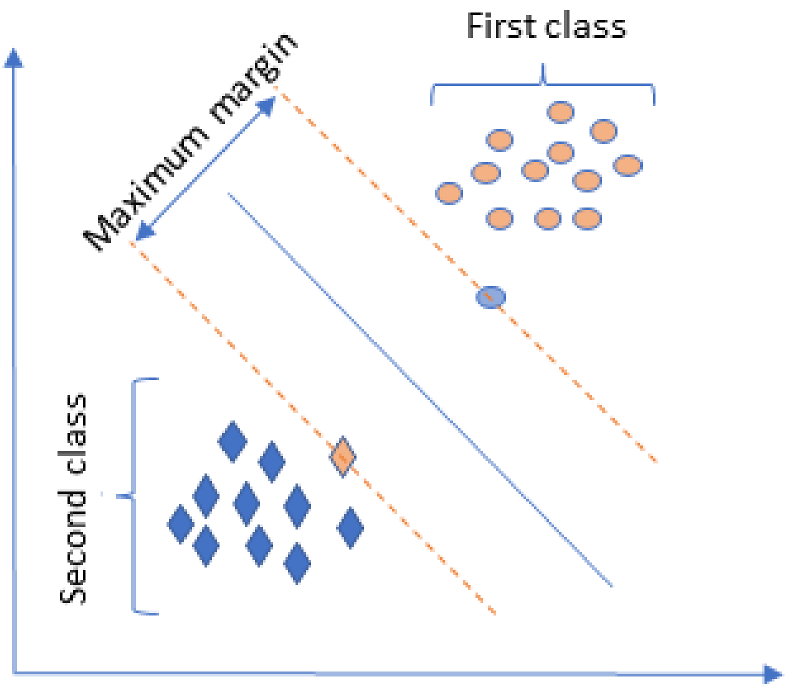


Figure 2.8: A demonstration of linear SVM.

How does it work?

Assume having two tags, red and blue as shown in figure 2.8, and two features, x and y, in our data. We want a classifier that can determine if a pair of (x,y) coordinates is red or blue. A support vector machine uses these data points to calculate the optimum hyperplane for separating the tags. This line represents the decision boundary: everything on one side will be classified as blue, and anything on the other will be classified as red. The most efficient hyperplane is the one that maximizes the margins from both red and blue tags for SVM. In other words, the hyperplane with the greatest distance to the nearest element of each tag. Figure 2.9 shows how the greatest distance is achieved mathematically.

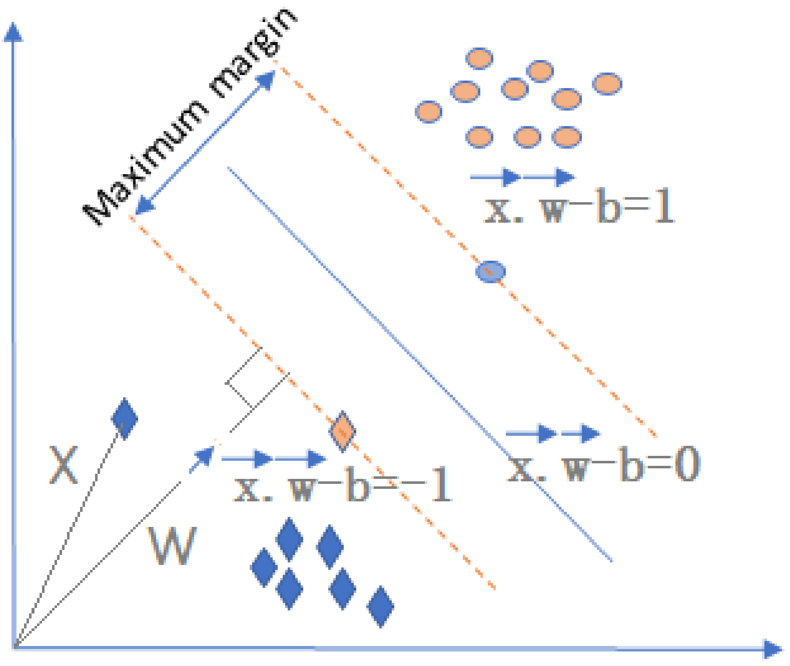


Figure 2.9: Math behind SVM

SVM for nonlinear data

What if we have data like shown in figure figure 2.10. Note that in this instance a linear hyperplane separation of the two classes is not possible. So here we are going to create a third dimension. up until this point, we have only made use of the two dimensions x and y. In this new development, we establish a new z dimension and specify that it be computed in the following manner: $z = \sqrt{x^2 + y^2}$. Our decision boundary is a radius z perimeter that divides both tags using nonlinear SVM. At this point, a procedure similar to the one used to calculate the distance from the linear hyperplane is employed to determine the radius and center of the derived nonlinear hyperplane.

Selecting a kernel function

Now that we have the feature vectors, the only thing left is to select a kernel function for our model. Every problem is unique, and the kernel function is decided by the data.

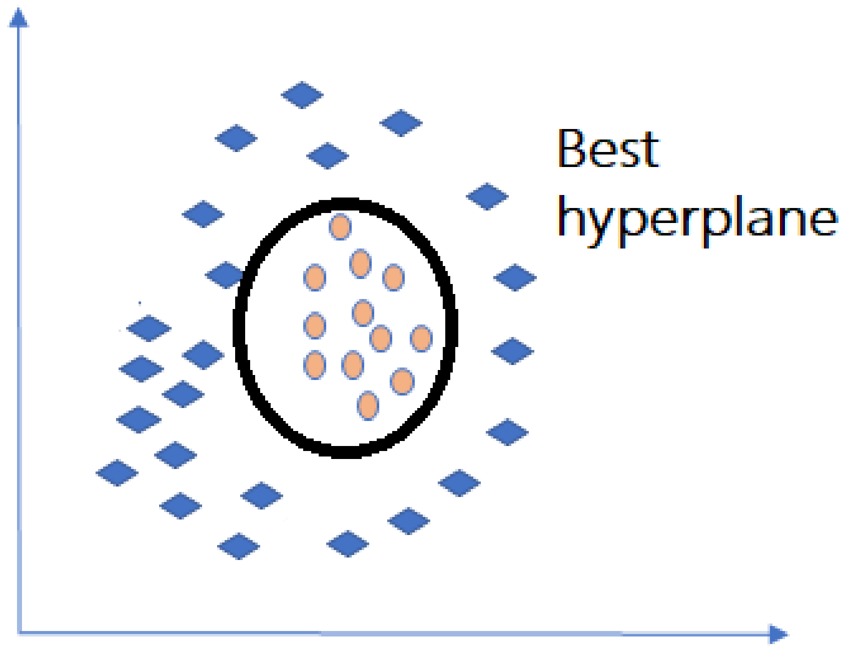


Figure 2.10: A demonstration of nonlinear SVM.

Using SVM in our experiment

To detect/classify pigs in the input scene, this research focused on the use of the Supporting Vector Machine (SVM) method. The system swiftly and accurately classifies pigs from other animals/targets in the scene. Also, for a better result, the dataset containing the images of the identified pigs is augmented and presented so that it may be utilized for other reasons. The strategy that takes us to the final model will be detailed in the next section.

2.1.5 Infrared Images and applications

The infrared spectrum encompasses light with wavelengths ranging from 0.9 to 12μm. Infrared (IR) imaging technology is used in many applications ranging from thermal detection of objects when very photons from the visible spectrum are available to determine the temperature of an item without contact. All things generate electromagnetic radiation, particularly in the infrared wavelength range, which the human eye cannot see. Figure 2.11 shows the light spectrum for both visible and infrared light. All of these benefits and a host of others lead to their inclusion in several studies on object detection using machine and deep learning. For example, this study [36] makes use of one of the oldest using machine-learning approaches to detect pedestrians using infrared images. The study first extracts image features using histograms of oriented gradients (HOG). Then used (SVM) classifier model.

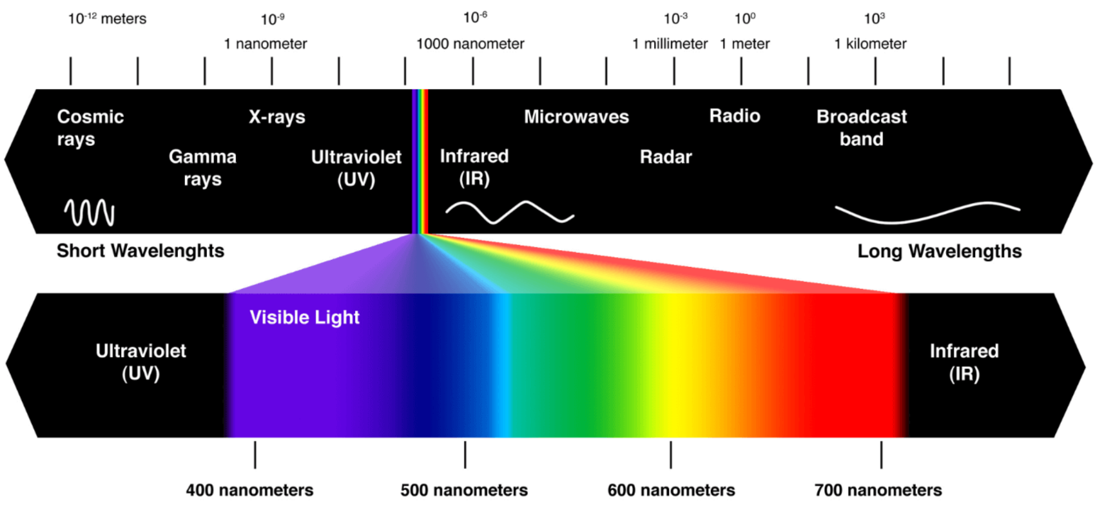


Figure 2.11: The light spectrum shows both visible light and infrared light

2.2 Method Overview

2.2.1 Vulva Localization Using HOG with 12 Gabor filters

Many machine-learning approaches focus on the use of one feature to detect/segment objects. The model developed in this study is based on features derived from the Gabor filter concatenated with HOG features. By using this model we can easily and effectively get the desired results. The procedure followed has been split into two main steps which focus separately on training and testing data-set. In order to support the previously stated core goals of agricultural reproduction, the dataset contains thermal images of pigs and their surrounding stalls.

In order to effectively train the SVM classifier, a negative/labeled sample set is required. A sample of both positive and negative samples are shown side by side in Figure 2.12.

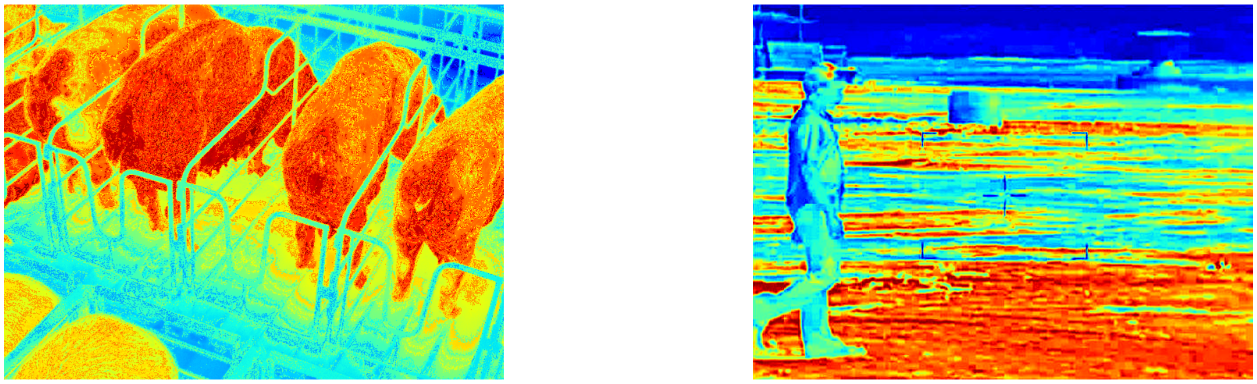

(a) Positive sample **(b)** Negative sample.

Figure 2.12: A sample is taken from the dataset.

Training dataset

The datasets used contains a combination of thermal images of pigs and humans in different poses. As a positive dataset, we used images from pigs, while we sorted humans' images as negative.

After resizing the input images to the size of $N \times N$, the HOG feature maps were calculated from the input images. Then the Gabor responses are calculated for the input image. A huge number of Gabor responses can be obtained by using different combinations of filter parameters. Lastly, we concatenated them into a single 3D feature map. The same procedure was used for the negative samples.

Testing dataset

The testing stage of the method starts by defining the size of the sliding window to be the same as the size of the corresponding image in the training data ($N \times N$). Gaussian pyramids were implemented to increase the probability of accurate detection regardless of the size of the pig in the image. The sliding window scanned each layer of the Gaussian pyramid for responses similar to the ones on which the SVM model was trained. This technique in image processing breaks down an image into successively smaller groups of pixels which will assures better detection regardless of the pig's location, size, or direction in the 2D image.

2.2.2 Vulva Localization Using 96 Gabor filters

Figure 2.13 presents the full pipeline components to be covered in detail in the following subsections.

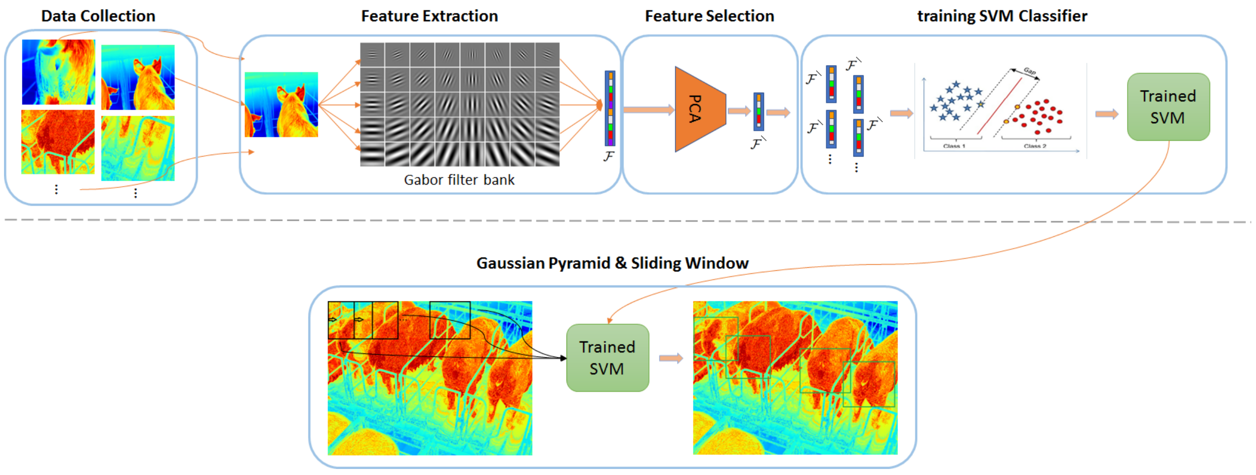


Figure 2.13: An overview of our classification system construction. The First stage contains collecting thermal RGB images dataset. Then, we **extract features** using Gabor filter bank followed by **feature selection** via PCA. Consequently, our SVM classifier is trained using the resulting features. Finally, we employ a series of **Gaussian pyramids with sliding windows** to establish the vulva locations through the trained SVM.

Dataset Collection

To train our machine learning classifier, a dataset with images containing only the vulva (positive dataset) and images representing anything other than the vulva (negative dataset) is needed. To do so, we used a Long-Wave Infrared (LWIR) camera to capture 175 images for sows in a climate-controlled barn. Then we cropped the parts that contained only the pig's vulva. In contrast, we created images with other pig parts, such as ears and heads, as well as other unspecified scenes, as shown in Figure 2.14. LWIR cameras are constructed to capture and image photons emitted due to thermal energy in one of the three major infrared ranges. The other two major infrared ranges are Medium Wavelength Infrared (MWIR) and Very Long Wavelength Infrared (VLIR). LWIR infrared covers the wavelengths that range from 8,000nm to 14,000nm (8 μm to 14μm) [6]. An LWIR camera can detect the thermal emissions of animals, vehicles, and people as they stand out when the temperature of the environment is different. Since it measures heat rather than visible light, LWIR imaging is frequently used as a thermal imaging solution. The contrast in the captured images is actively enhanced according to the documentation provided by the manufacturer in [11]. Active Contrast Enhancement (ACE) eliminates the effects of temperature shifts during daytime compared to nighttime and the temperature gradient during different seasons.

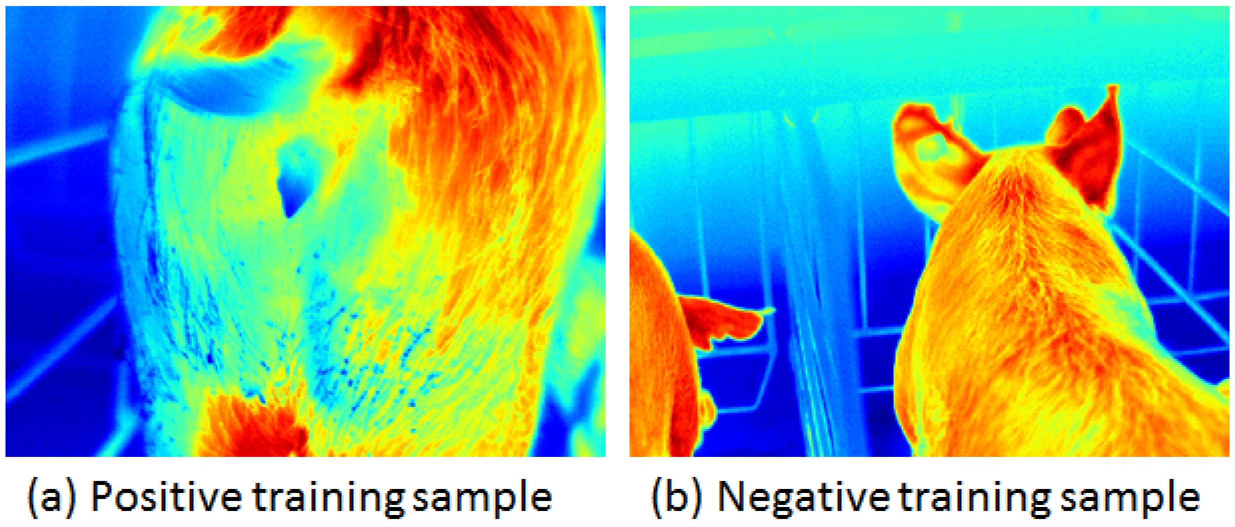


Figure 2.14: Image in (a) presents a sample training set image that contains only the pig's vulva. Images such as these construct our positive training dataset. Images such as the one shown in (b) are representative of the images defining our negative training dataset

Dataset Augmentation. Machine learning algorithms require significant amounts of training and testing data to effectively extract the information necessary for accurate function on unseen data. In our case, although we had significant amounts of data, the number of images collected was not sufficient for an effective algorithmic function. Therefore, a number of data augmentation methods had to be implemented to increase the effective size of the available data. Specifically, in order to obtain a large enough dataset, the images are flipped, mirrored, and rotated 90 degrees. An example of this augmentation is shown in Figure 2.15 With the additional images, the dataset was shown to be sufficient to make effective inferences on unknown data.

Feature Extraction

In order to train our system, we extracted 96 image features using the Gabor filter bank. We chose Gabor filters to incorporate textures in the classification process in addition to the HOG set of features which can be insufficient, as demonstrated in previous work [2]. The previous study demonstrated how HOG features in thermal images could cause the model to confuse any surface that falls within the same temperature range to be classified as a pig. Another advantage of Gabor filters over Gaussian derivatives is related to greater

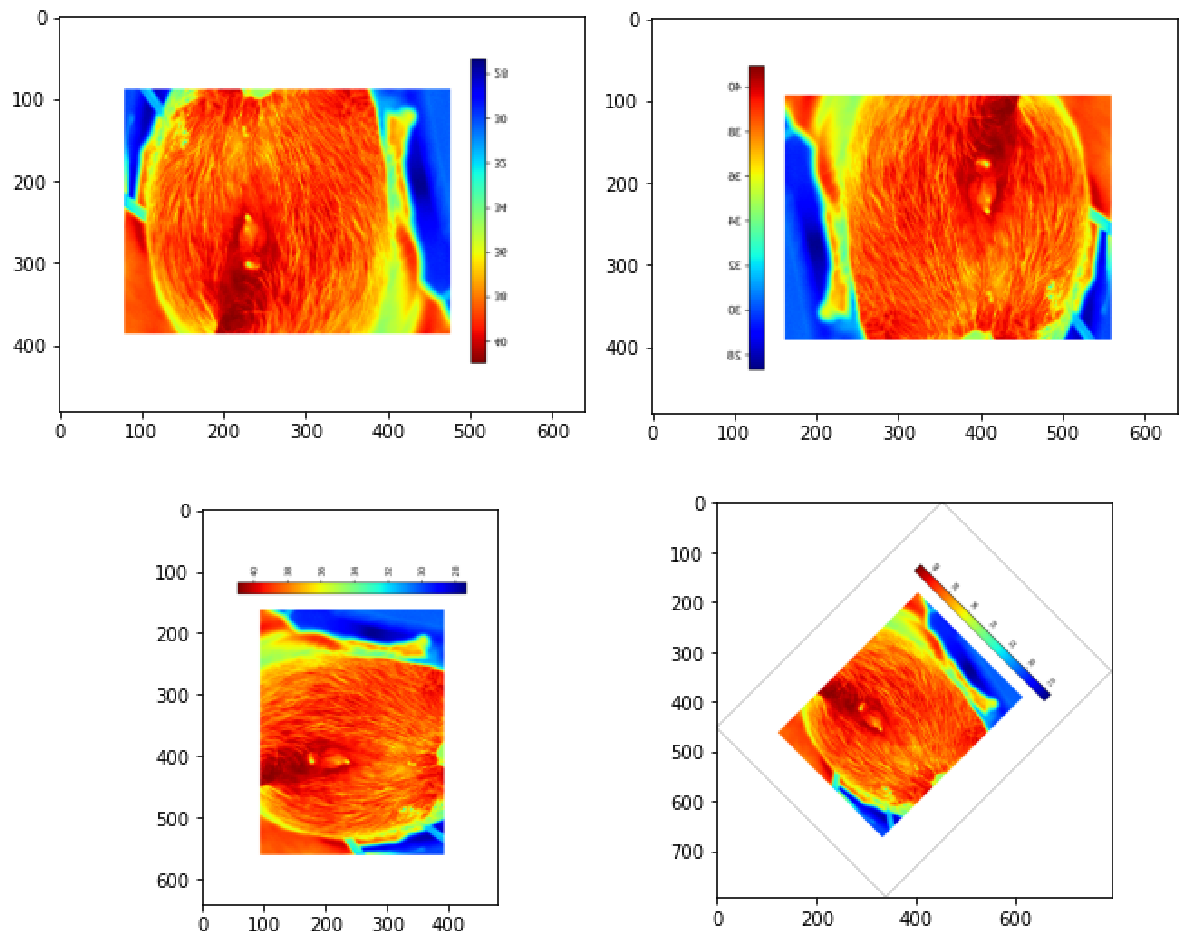


Figure 2.15: Data Augmentation

kernel/function shape flexibility due to the inherent inclusion of a larger general set of parameters (degrees of freedom) [40]. Consequently, we generated 96 Gabor filters via manipulating its parameters, as shown in the following equation:

$$g(x, y; \lambda, \theta, \phi, \gamma) = \exp\left(-\frac{x'^2+\gamma^2 y'^2}{2\sigma^2}\right)\cos\left(2\pi\frac{x'}{\lambda}+\phi\right)$$

while λ, θ, γ, and ϕ represent the wavelength, orientation, spatial aspect ratio, and phase offset, respectively. We choose a set of λ values to be equal to $(0, \pi/2, \pi/3, \pi/4)$, while θ and γ equal to $(0, 1/4\pi, 2/4\pi, 3/4\pi)$ and $(0, \pi/2, \pi/3, \pi/4)$, respectively. We also set both the size of the Gabor kernel and sinusoidal function phase offset ϕ to zero.

To construct our features, we applied the 96 Gabor kernels and collected their responses for each image such that we got 96 Gabor responses for each. Then we calculated the power which is the sum of the squared for each filtered image produced to exclude the redundancy in features. As a result, we ended up with 96 features for each training image dataset.

Normalization: As a final step before creating a classifier, we normalized the final feature to ensure that all data falls between 0 and 1.

Feature Selection

Generating Gabor kernels using the full range for parameters mentioned in the previous section can lead to a high number of filters. This can increase the dimension of the problem and cause the model to become prohibitively complex and expensive to train [7]. Often when dealing with a high number of filters, there will exist a sub-set of the feature filters where the information extracted is relevant to the model. The rest tend to be redundant and carry no useful information. By eliminating the irrelevant features, we cut down on the resources needed to train and perform predictions and enhance the accuracy of the model since these features can act as noise to the model. Therefore, some feature selection is needed after we build our kernels. In the paper [26], the authors apply the generated filters on the input images, then eliminate redundant responses to be used in their segmentation by calculating the error in the reconstructed image using a subset of the filtered images. The authors then iterate through the filters to get the highest reconstruction variance. To complete a similar procedure, we leveraged Principle component Analysis (PCA) [33] to exclude redundant features which extracted in the previous subsection.

Principal Component Analysis (PCA): Varying Gabor kernel parameters $(\theta, \lambda, \sigma, \gamma, \mu)$ resulted in 96 kernels. Convolving our dataset with those filters will result in $96 \times D$ feature maps. Having this many feature maps means having highly correlated samples to train our ML model. The high correlation will cause overfitting and subsequent poor algorithmic generalization performance. We applied PCA to eliminate correlation in the training data by excluding highly correlated responses from the Gabor filters. We find the major principal component and eliminate the least significant 5% of the principal components. We found the 5% ratio empirically. When we tried to reduce the dimensionality further, it resulted in reduced performance. PCA performs better in cases of fewer samples per class. In comparison, other dimensionality reduction techniques such as LDA perform better with large datasets having multiple classes [4].

Training SVM Classifier

SVM, or Support Vector Machine, is a linear model for classification and regression issues. It can handle linear and nonlinear problems and is useful for a wide range of applications. The algorithm creates a line or hyper-plane that divides the information into classes. Figure 2.8 shows one example to classify two different classes via linear SVM.

For sorting the final features into two categories, we design an SVM classifier by partitioning the data into training and testing splits. To do so, we used 80 percent of the data for training and the remaining 20 percent for testing.

Gaussian Pyramid & Sliding Window

After training our model, we needed to keep track of a contiguous sequence of elements, so a sliding window was defined. Performing the search over a sliding window is an essential task in object detection since it allows localizing where the object precisely resides in the image. Three arguments are provided to use the sliding window function: image, step size, and window size. A step size of 30 pixels was used, where the window size has to be the same as the size of the images in the training data. Along with the sliding window, as shown in Figure 2.16 and 2.17 the Gaussian pyramid was used to break down an image into successively smaller groups of pixels for *efficient multi-scale object detection*.

Utilizing both a sliding window and a Gaussian pyramid, we are able to detect objects in images at various scales and locations. Finally, we eliminated detections of the same object as shown in Figure 2.18 which overlapped by 95% or more of the target area, which resulted in a single bounding box.

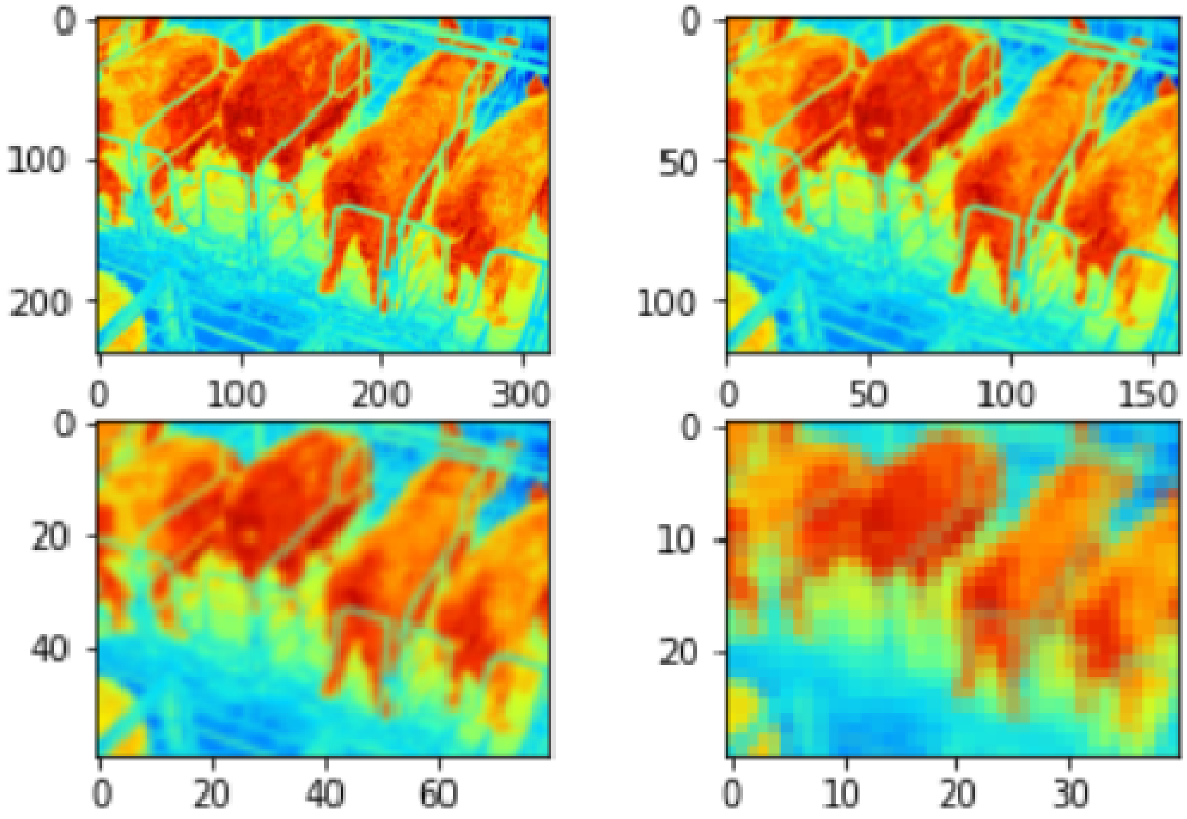


Figure 2.16: Various sizes of image (layers) achieved by using scale down pyramid

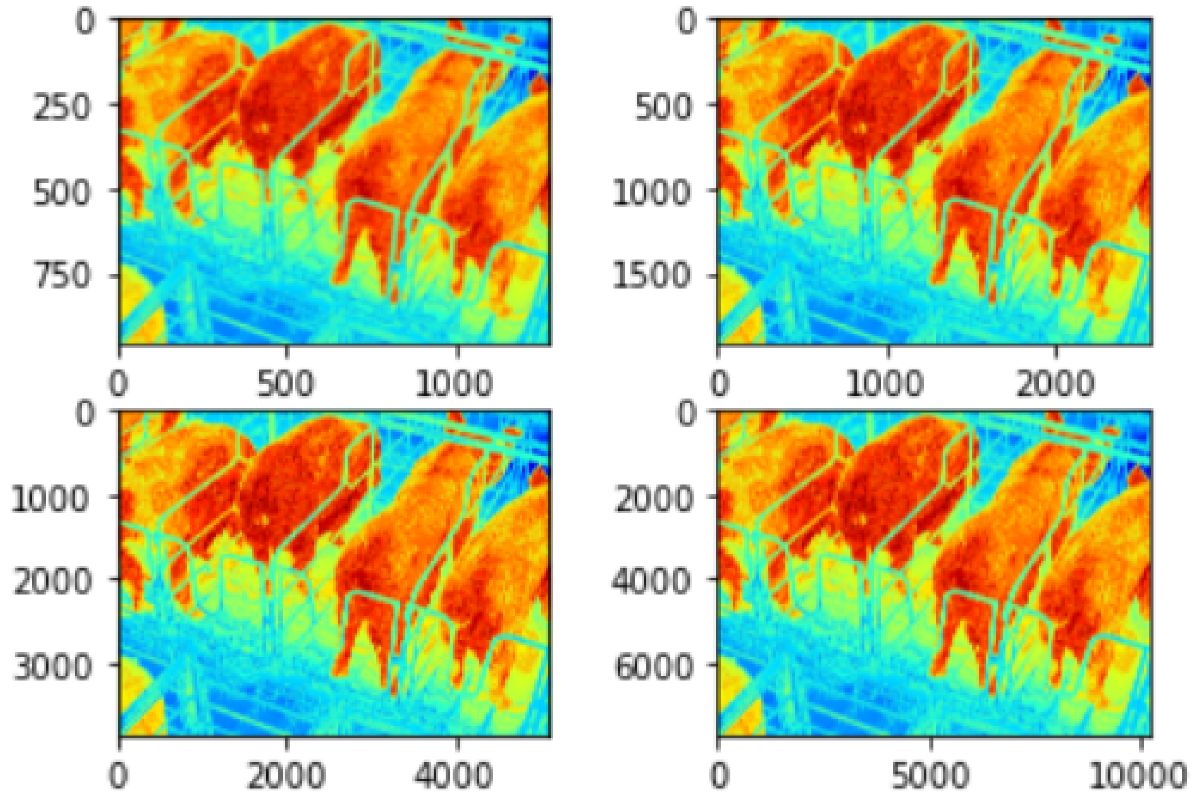


Figure 2.17: Various sizes of image (layers) achieved by using scale up pyramid

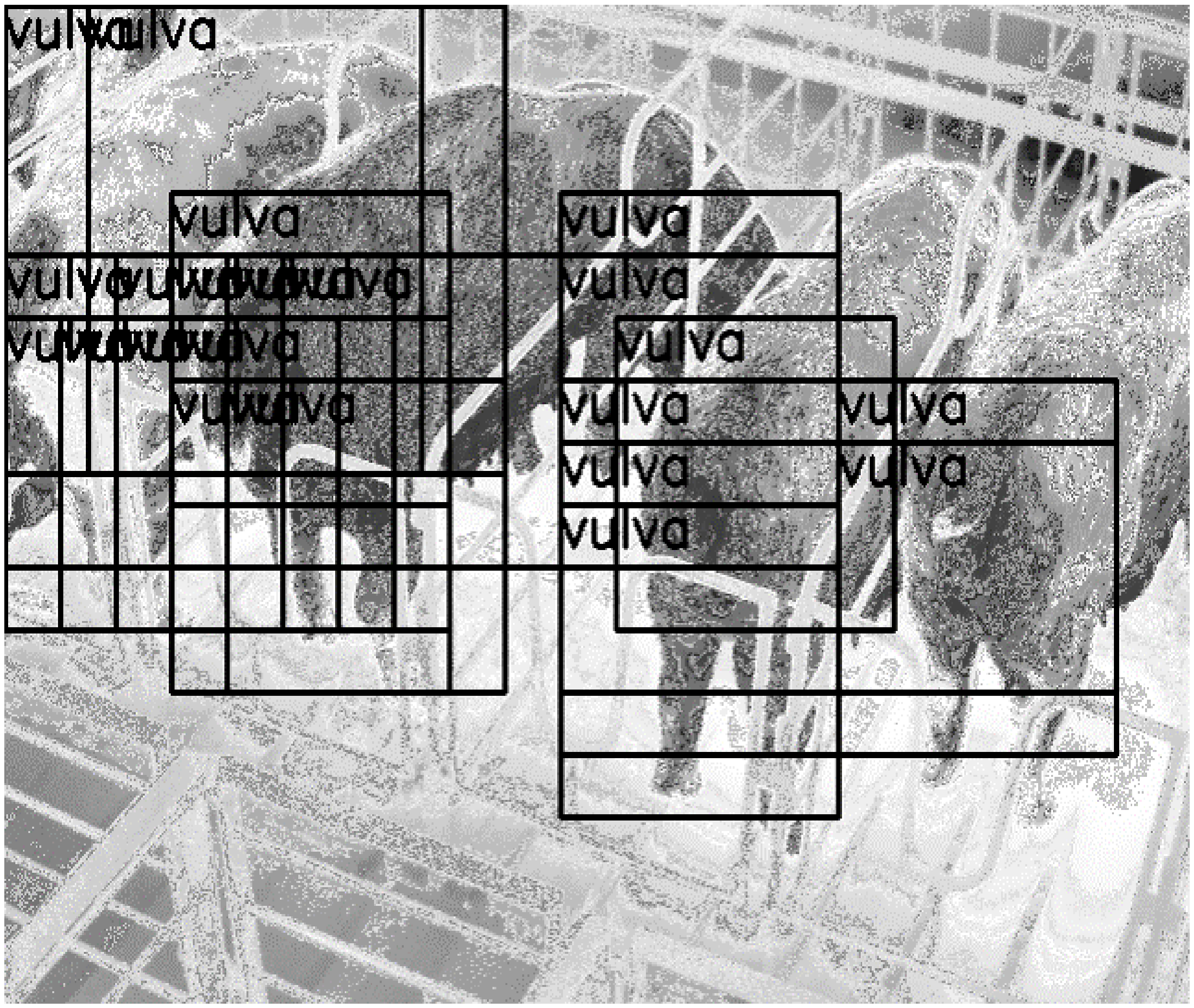


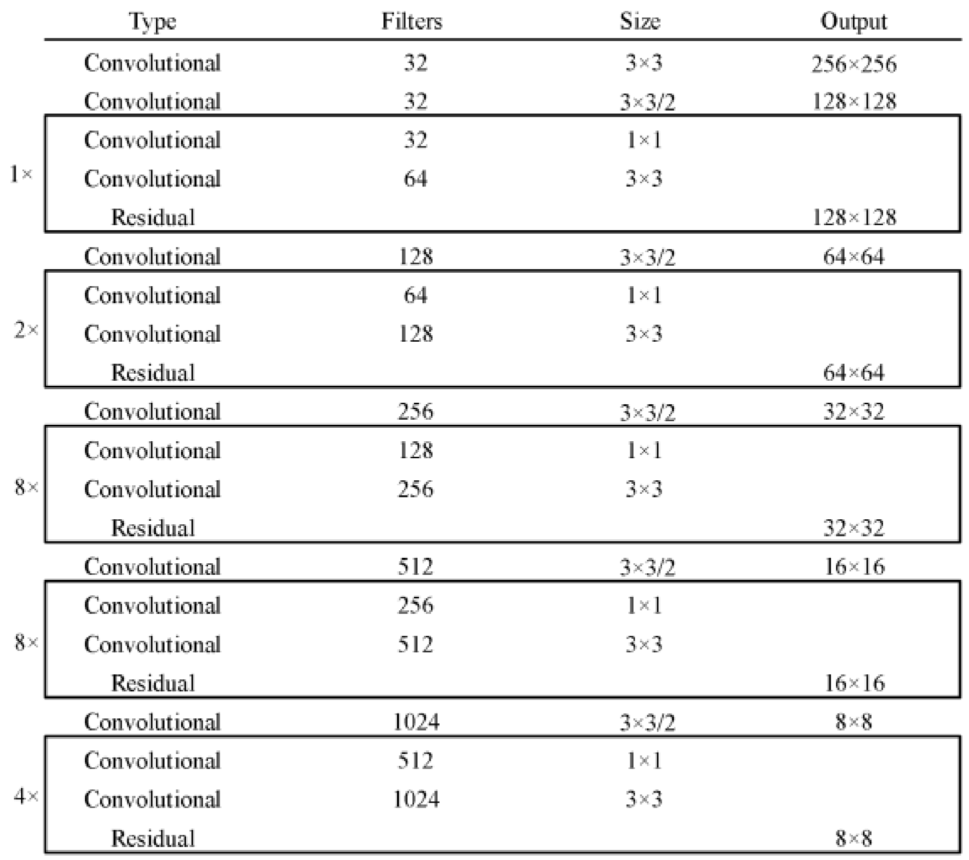
Figure 2.18: Overlapping of bounding box

Chapter 3

Object Detection Implementation Using Deep Learning

3.1 Background

This section summarizes some object detection approaches utilizing Deep Neural Networks. In order to ensure complete coverage of this section, we will include a detailed discussion of several deep-learning approaches to object detection and segmentation. For example, YOLOv3 [29] is an object detector proposed by Joseph et al. It determines the score for each detection using logistic regression. This approach improves detecting speed and takes input images in various sizes. Darknet-53 [29] is used by YOLOv3 to extract features. Darknet-53 is far more powerful than Darknet-19 [27] while remaining more efficient than ResNet-101 [13] or ResNet-152 [13].A 53-layer feature extractor Darknet53 was built in the YOLOv3 model using continuous 1×1 and 3×3 convolutions and residual modules, as seen in the figure 3.1. YOLOv3 uses multiscale prediction, which implies it is identified on several scale feature maps. For this reason, the accuracy of target detection is improved. figure 3.2 shows its structure detail.



	Type	Filters	Size	Output
	Convolutional	32	3×3	256×256
	Convolutional	32	3×3/2	128×128
1×	Convolutional	32	1×1	
	Convolutional	64	3×3	
	Residual			128×128
	Convolutional	128	3×3/2	64×64
2×	Convolutional	64	1×1	
	Convolutional	128	3×3	
	Residual			64×64
	Convolutional	256	3×3/2	32×32
8×	Convolutional	128	1×1	
	Convolutional	256	3×3	
	Residual			32×32
	Convolutional	512	3×3/2	16×16
8×	Convolutional	256	1×1	
	Convolutional	512	3×3	
	Residual			16×16
	Convolutional	1024	3×3/2	8×8
4×	Convolutional	512	1×1	
	Convolutional	1024	3×3	
	Residual			8×8

Figure 3.1: The structure of Darknet53.

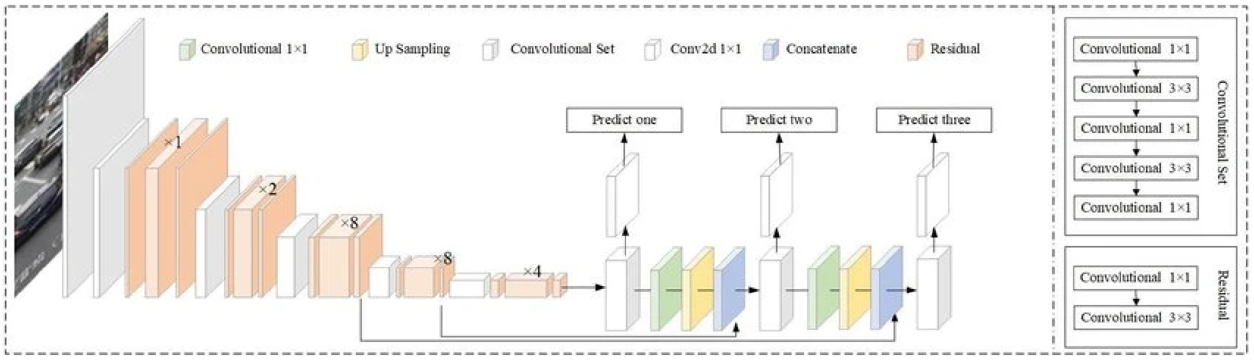


Figure 3.2: Structure detail of YOLOv3. It uses Darknet-53 as the backbone network and uses three-scale predictions.

The YOLO algorithm is a multiple-step approach to object detection. First, it divides the image into N grids with each having an equal dimensional region of $S \times S$. Each of these N grids is responsible for detecting and locating the object under search. Similarly, these grids estimate B bounding box coordinates relative to cell coordinates, the object label, and its probability of being present in the cell. YOLO utilizes of Non-Maximal Suppression [25] to get rid of many duplicate predictions due to multiple cells predicting the exact same object with different bounding box predictions figure 3.3. It fulfills this by first looking at the probability scores associated with each decision and taking the largest one. Following this, it suppresses the bounding boxes having the largest Intersection over Union with the current high probability bounding box.

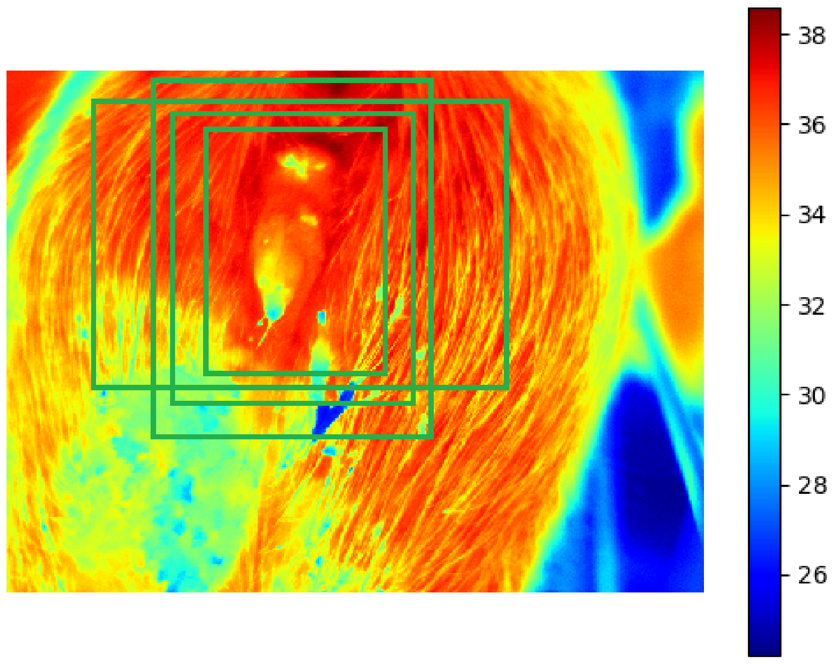


Figure 3.3: Duplicate predictions

3.2 Method overview

3.2.1 Image annotation

Image annotation is defined as the task of labeling digital images. It is generally used to prepare images for use in machine learning algorithms and is an integral component of creating algorithms that perform computer vision tasks. Put more simply, annotated images are used to train computers to annotate images on their own. This is highly beneficial for a wide array of processing needs since hand-labeling images for algorithmic use is tedious and time-consuming. For the purposes of the research contained in this treatment, the graphical image annotation tool LabelImg was used to complete the labeling.

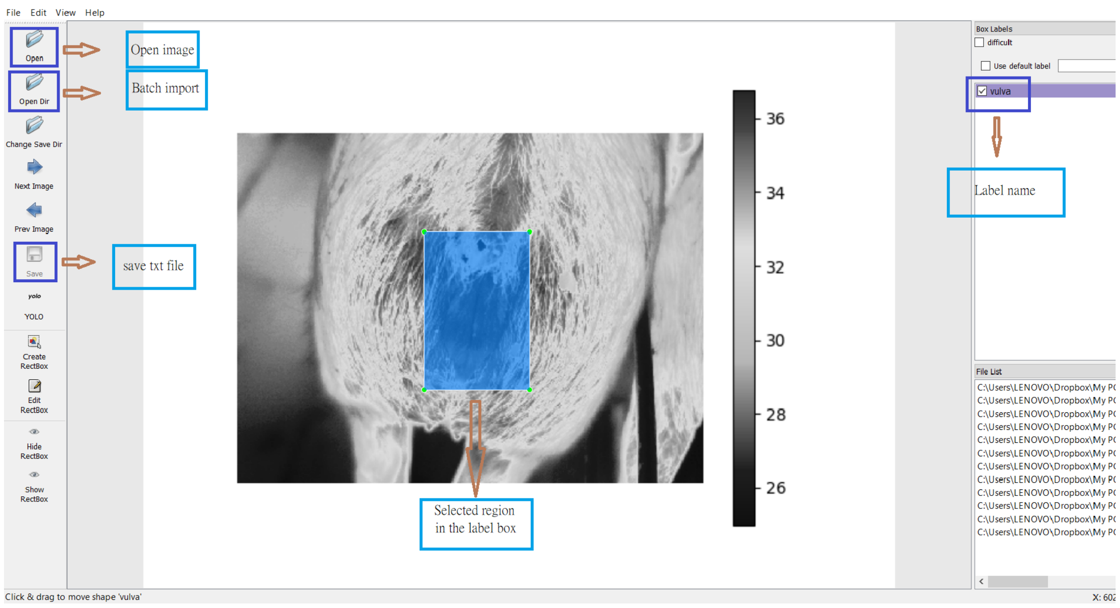


Figure 3.4: The labeling interface for vulva dataset

When labeling images using LabelImg, there are a few steps to consider as shown in Fig 3.4:

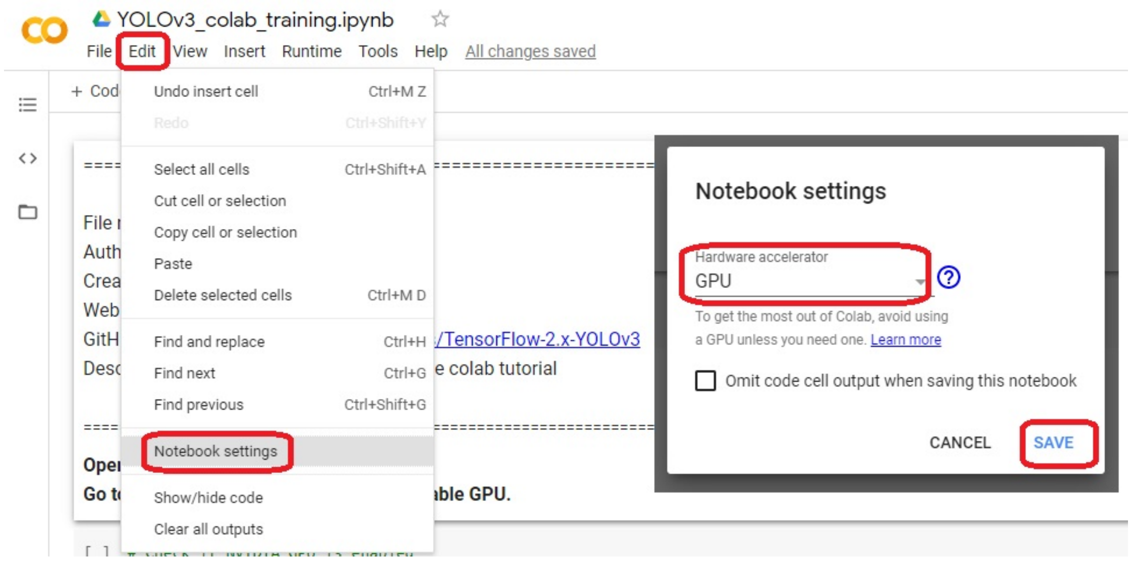
1- Label the complete circumference of an object; including a small amount of non-object buffer is preferable rather than omitting a piece of the object with a rectangle label. Look for boxes that closely resemble the items wanted to name but do not chop off any of them. This method will significantly improve our model's understanding of the edges.

2- For labeling obscured items completely; if an object is out of view because another object is in front of it, name it as if it could be seen entirety. This will help our model grasp the absolute boundaries of items.

3.2.2 Set up Google Colab

Colab allows anybody to develop and run Python code in the browser, making it ideal for machine learning and data analysis. Colab, in technical terms, is a hosted Jupyter notebook service that requires no setup and provides free access to computer resources such as GPUs. The main difference between Google Colab and Jupyter Notebook is that Google Colab is cloud-based, whereas Jupyter is not. This implies that if you work in Google Colab, you won't have to download or install anything on your computer. It also means that you may rest assured that your work will automatically save and backup to the cloud without your intervention. And because it syncs effortlessly between platforms, Google Colab is ideal for users who need to work across many devices, such as one computer at home, one at work, a laptop, and a tablet. On the other hand, Jupyter Notebook runs on your local system and saves files to your hard disk. Jupyter does have an autosaving interval (which you can modify), but it does not back up to the cloud, so if something goes wrong with your system, you're out of luck. Jupyter cannot sync or share files between devices without using a third-party file-sharing service such as DropBox or GitHub.
Jupyter Notebook vs. Google Colab couldn't be discussed without bringing up cooperation. Google Colab, as the name implies, is designed to make it simple to share your notes with everyone - even if they aren't data scientists. Other people can access your notebook without downloading any software, which is a huge benefit if you frequently deal with non-techies who need to view the data. Another major feature that distinguishes Colab from the competition is the ability to import large datasets from Google Drive into Colab.

To setup Google colab GPU need to be enabled as shown in Fig 3.5



Notebook settings

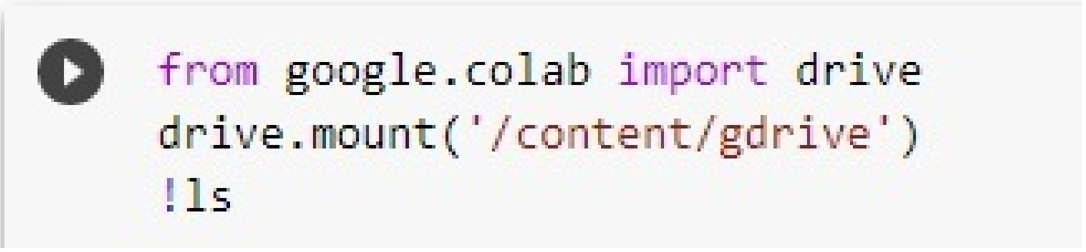
Hardware accelerator: GPU

To get the most out of Colab, avoid using a GPU unless you need one. Learn more

☐ Omit code cell output when saving this notebook

Figure 3.5: Set up Google Colab

To import data from Google drive two lines code need to be written as shown in Fig 3.6



```
from google.colab import drive
drive.mount('/content/gdrive')
!ls
```

Figure 3.6: connect Colab with our Google Drive

3.2.3 Training

In order to start training, Darknet need to be installed. Darknet is an open-source neural network framework that supports CPU and GPU computation. Darknet-19 was used as the backbone feature extractor in YOLOv2, whereas Darknet-53 is currently used in YOLOv3. Darknet-53 is another backbone created by YOLO inventors Joseph Redmon and Ali Farhadi. Darknet-53 includes 53 convolutional layers instead of 19 in Darknet-19, making it more powerful and efficient than rival backbones (ResNet-101 or ResNet-152). Using the chart from Redmon and Farhadi's YOLOv3 work Table 3.1, which shows that Darknet-52 is 1.5 times faster than ResNet101. The precision represented does not imply any trade-off between accuracy and speed amongst Darknet backbones since it is still as accurate as ResNet-152 while being two times faster.

Comparison of backbones					
Backbone	Top-1	Top-5	Ops	BFLOP/s	FPS
Darknet-19	74.1	91.8	7.29	1246	171
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

Table 3.1: Comparison of backbones. Accuracy, billions of operations (Ops), billion floating-point operations per second (BFLOP/s), and frames per second (FPS) for various networks

In order to ensure algorithmic accuracy YOLO needs certain specific files to know how and what to train. The YOLOv3 tiny model (yolov3-tiny.cfg) will need to be modified by training it on the custom detector. This modification includes the following steps:

1- Uncomment the lines 5,6, and 7 and change the training batch to 64 and subdivisions to 2.
2- Change the number of filters for convolutional layer "[convolution]" just before every yolo output "[yolo]" such that the number of filters = number of anchors x (5 + number of classes) = 3*(5+1) = 18. The number 5 is the count of parameters center-x, center-y, width, height, and objectness Score. So, change lines 127 and 171 to "filters=18".
3- For every yolo layer [yolo] change the number of classes to 1 as in lines 135 and 177.

In addition to weights from the darknet53 model, weights were used from a pre-trained model on Imagenet [10]. Where a pre-trained model refers to a model or a saved network created by someone else and trained on a large dataset to solve a similar problem, these large datasets were obtained from Imagenet, which is an image database organized according to the WordNet hierarchy, where each node of the hierarchy is represented by hundreds of thousands of images.

Chapter 4

Image Segmentation Using U-Net

4.1 Background

4.1.1 Image segmentation

Image segmentation is a sub-field of digital image processing and computer vision that tries to group the same segments or regions of an image under their corresponding class labels. Image segmentation is a subset of image classification that does localization in addition to classification. Image segmentation is essentially a subset of image classification, with the model identifying where a related item is present by outlining the entity's boundaries. Most image segmentation models in computer vision are composed of an encoder-decoder network, as opposed to a single encoder network in classifiers. The main task of the encoder is to encode a latent space representation of the input, which the decoder decodes to generate segment maps that outline the position of each item in the image.

Image segmentation tasks are divided into three categories based on the amount and kind of information conveyed; semantic segmentation [12], instance segmentation [31], and panoptic segmentation [9]. Semantic segmentation is the categorization of pixels in an image into semantic classes. Pixels belonging to a specific class are simply categorized into that class, with no further information or context taken into account. In instance segmentation, pixels are divided into groups based on "instances" rather than classes. While panoptic segmentation is a combination of semantic and instance segmentation in one image.

Semantic segmentation

Semantic segmentation is the process of categorizing each pixel as belonging to a specific label. It does not differ across instances of the same item. For example, in Figure 4.1 the image contains two pigs, and semantic segmentation assigns the same label to all of the pigs' pixels.

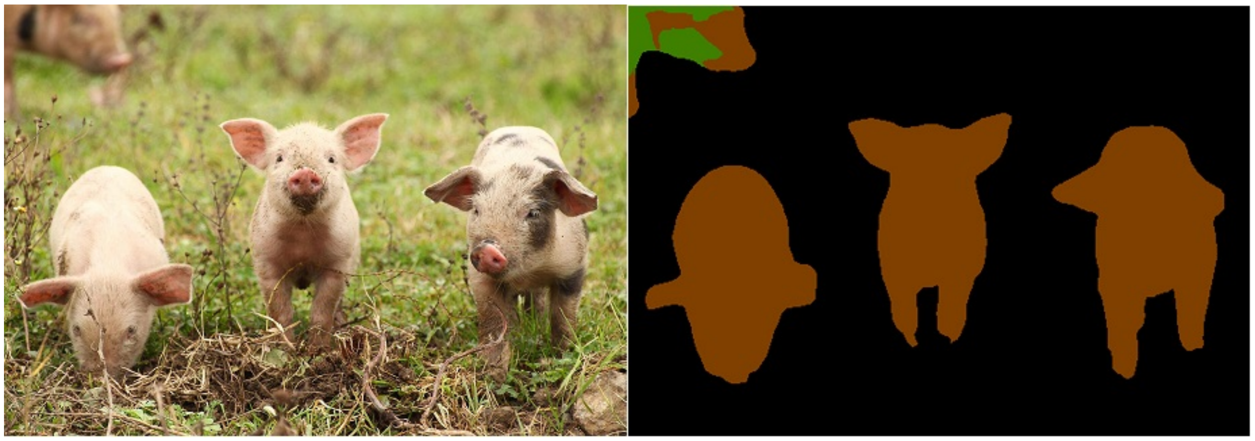


Figure 4.1: Semantic segmentation

Instance segmentation

Instance segmentation models sort pixels based on "instances" rather than classes. An instance segmentation algorithm has no previous knowledge of what class a recognized region belongs to, but it may separate overlapping or remarkably similar object regions based on their borders. Figure 4.2 shows how instance segmentation work.

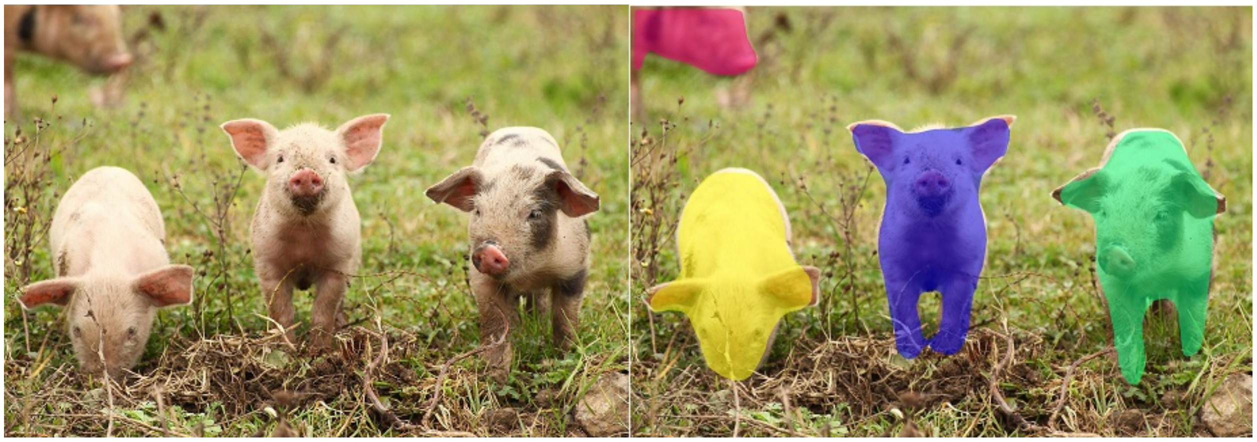


Figure 4.2: Instance segmentation

Panoptic segmentation

The most recently developed segmentation task, panoptic segmentation, may be defined as a mix of semantic segmentation and instance segmentation in which each instance of an item in an image is separated, and the object's identification is predicted.

4.1.2 U-Net semantic segmentation

U-Net is a particular type of architecture for image segmentation purposes. When come to architecture, that means an arrangement of deep learning tools like convolutional layers and max pooling. The U-Net is an excellent design that solves the majority of the problems that arise. This method employs the notion of fully convolutional networks. The U-Net goal is to collect both context features and localization. The sort of architecture constructed effectively to complete this procedure. The implementation's basic idea is to use consecutive contracting layers followed by up-sampling operators to get higher-resolution outputs from the input images. By looking at the architecture depicted in figure 4.3, it's clearly obvious why it is most likely referred to as U-Net architecture. The shape of the newly formed architecture is in the shape of a 'U', thus the name. The architecture can tell that the network constructed is a fully convolutional neural network just by looking at the structure and the multiple pieces involved in building this architecture. They did not employ any further layers, such as dense, flatten, or other similar layers. The representation shows a decreasing path called the encoder, followed by an increasing path called the decoder. According to the design, the model processes an input image, which is then followed by a pair of convolutional layers using the ReLU activation function. The picture size decreases from 572X572 to 570X570 and then to 568X568. This decrease is due to unpadded convolutions, which reduce total dimensionality. The encoder block constantly reduces image size with the aid of strides 2's max-pooling layers. In the encoder design, convolutional layers also have been repeated with a growing number of filters. When we approach the decoder aspect, the number of filters in the convolutional layers was beginning to decrease, followed by a steady up-sampling in the subsequent layers all the way to the top. The usage of skip connections to connect the previous outputs to the layers in the decoder blocks can be noticed. There is a handful of convolutional layers followed by the last convolution layer in the final convolution block. This layer has 2 filters with the proper function to show the output. This last layer can be adjusted depending on the scope of the project.

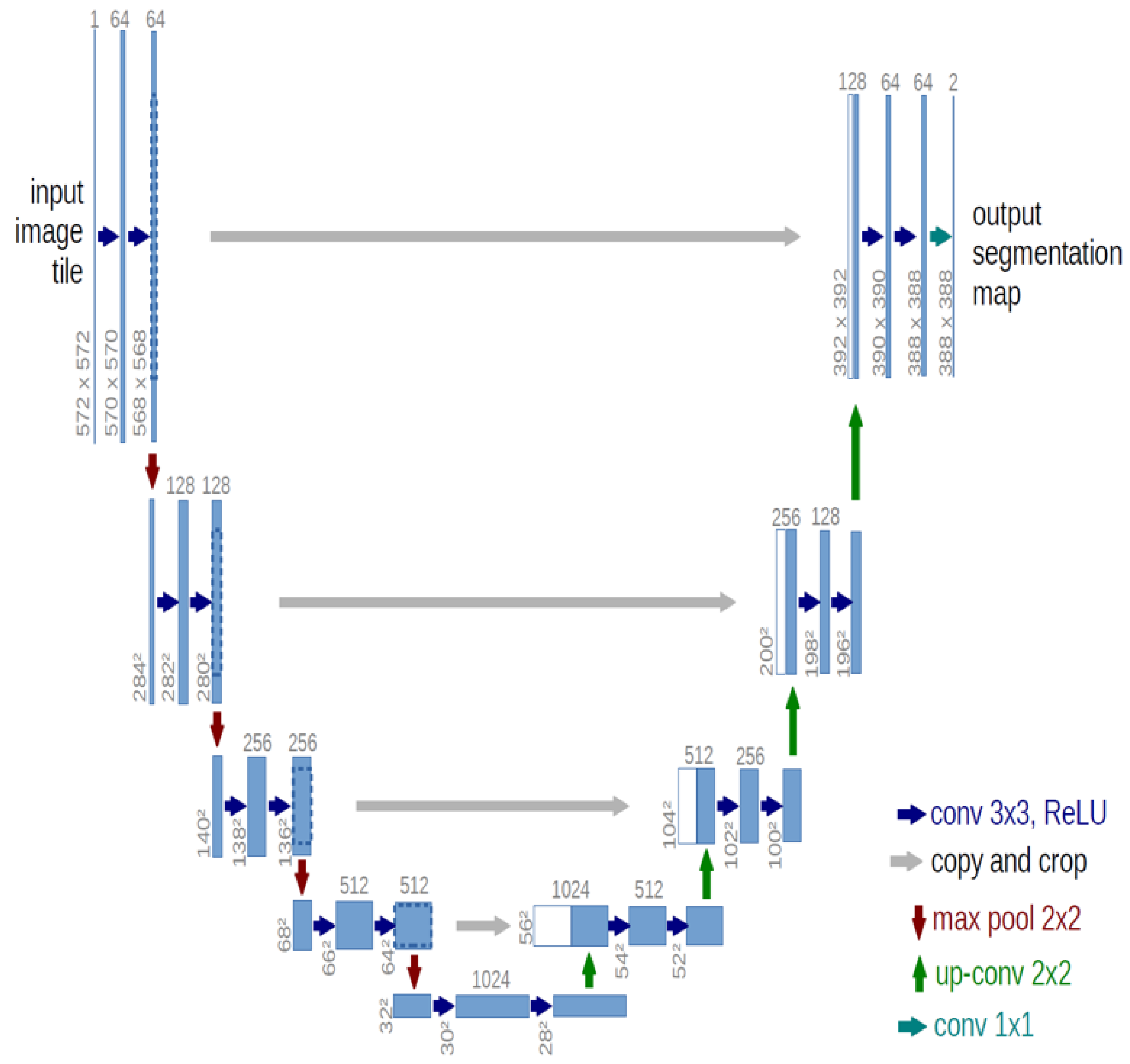


Figure 4.3: U-Net architecture

4.2 Introduction

Nowadays, Farmers breed animals for various reasons, including increased productivity, disease resistance, effective reproduction, and climate adaptation, with heat and drought being the most common. Highly productive animals use a more significant portion of their nutritional intake to produce desired items such as milk or meat rather than simply maintaining their bodies. Increased longevity and reproductive success mean that feed calories are concentrated among the most productive animals, lowering the number of replacement animals needed in a herd. With the recent growth of the intelligent pig breeding industry, precise pig management has become critical, and individual pig recognition is the key to precision breeding. However, how important is pig breeding, There has been little research into this. Previous research sought to construct a model that can detect pig faces [19] using YOLO [24], posture [35], touching pigs [15] and sow using machine learning [2]. The goal of this work is to detect sow's vulvae using the U-Net segmentation model [32], which is the most effective technique to manage breeding on the farm. A segmentation model is chosen to assist in detecting the ovulation cycle as it relates to temperature fluctuations in the vulva. According to the study [34], the temperature and size changes in the vulva indicate which part of the estrous cycle the sow is in. This will be covered in further detail in the future research section. In this work, The U-Net semantic segmentation was used after extracting masks via HSV color space. The system is trained on gray-scale images along with masks obtained by defining the HSV ranges for red on the segmented images. the U-Net technique to look for the vulva in the input image makes it lightweight, simple, and robust to be evaluated on a hugely different number of images.

4.3 Method overview

Figure 4.4 presents the full pipeline components that were covered in detail in the following subsections.

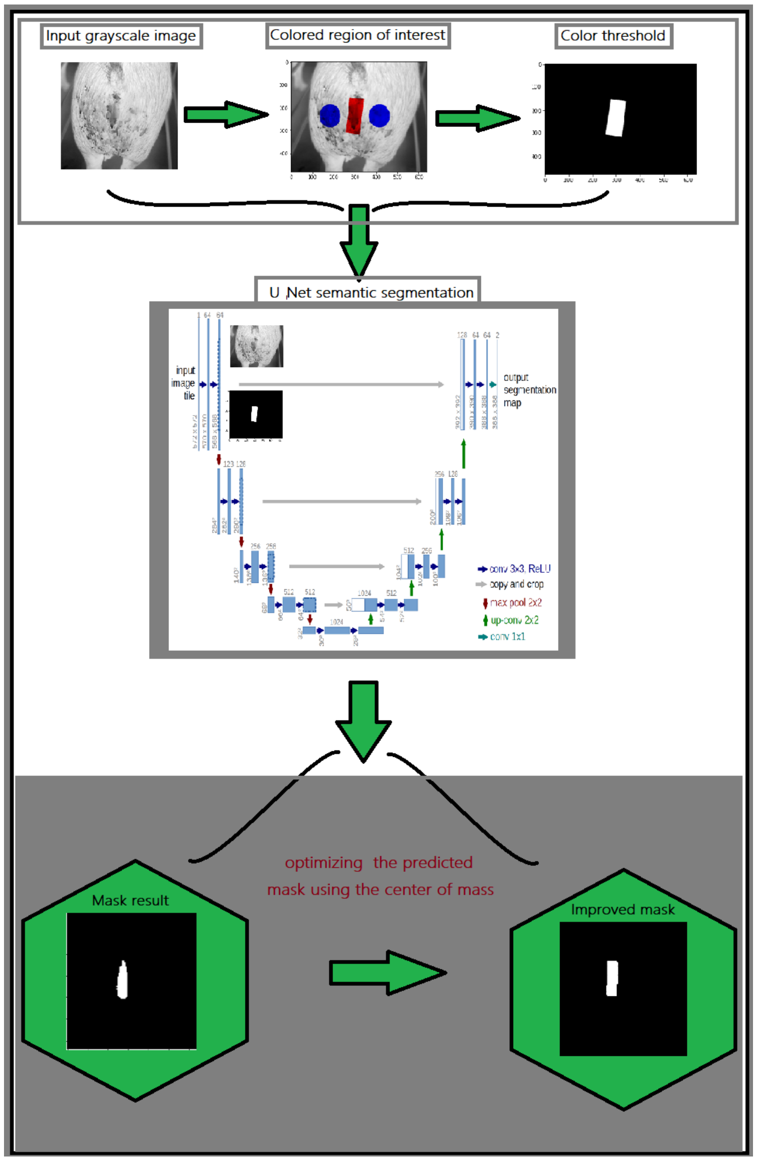


Figure 4.4: An overview of the segmentation model used in this chapter. The first stage contains extract masks of a region of interest. Then train grayscale images along with extracted masks in U-Net semantic segmentation model. Finally, using the center of mass for optimizing the predicted mask to get better Intersection over union (IOU) values.

4.3.1 Color threshold

In order to extract the masks of the region of interest, segment this region in red. Then threshold the image to get the binary image using the Hue, Saturation, and Value of each pixel. Unlike RGB, HSV allows not only filtering based on the colors of the pixels but also on the intensity of color and the brightness since Hue is Measuring the color of the pixel, Saturation measures the intensity of the pixel's color, while the pixel's brightness is measured by value. The figure 4.5 shows a sample of the segmented image and its mask.

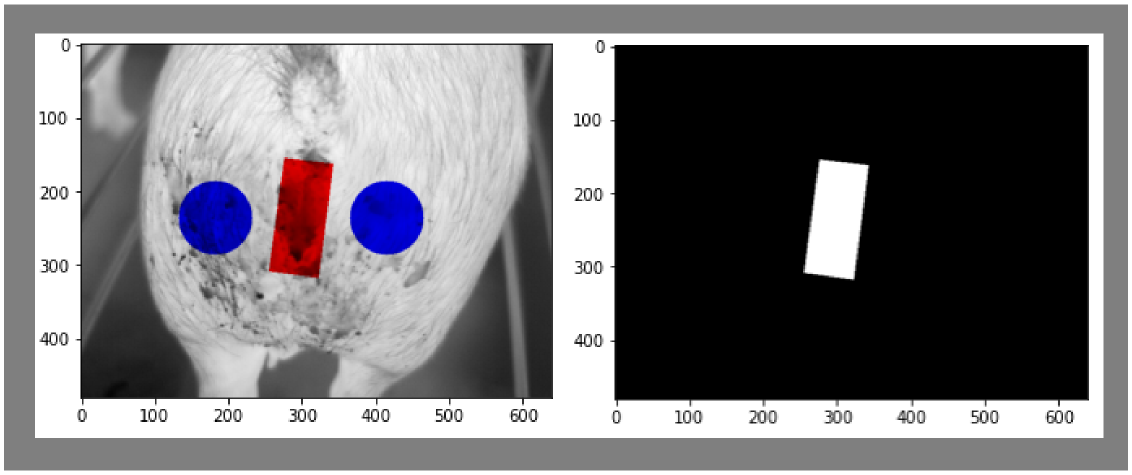


Figure 4.5: Segmented image and its mask. The lower and upper limits for HSV are (161, 155, 84) (179, 255, 255) respectively

4.3.2 Training masks

The dataset used to build this model is a combination of grayscale images and their masks. The table 4.1 explains the U-Net semantic segmentation architecture in detail.

Contraction path (Encoder)			
Block	layer	# filters	Activation Function
Block 1	Conv2D (3,3)	16	relu
	Conv2D (3,3)	16	relu
Block 2	Conv2D (3,3)	32	relu
	Conv2D (3,3)	32	relu
Block 3	Conv2D (3,3)	64	relu
	Conv2D (3,3)	64	relu
Block 4	Conv2D (3,3)	128	relu
	Conv2D (3,3)	128	relu
Block 5	Conv2D (3,3)	256	relu
	Conv2D (3,3)	256	relu
Expansive path (Decoder)			
Block	layer	# filters	Activation Function
Block 6	Conv2DTranspose (2,2)	128	....
	Concatenate(c4,Transpose(2,2)	128	......
	Conv2D (3,3)	128	relu
Block 7	Conv2DTranspose (2,2)	64	......
	Concatenate(c3,Transpose(2,2)	64	......
	Conv2D (3,3)	64	relu
Block 8	Conv2DTranspose (2,2)	32	......
	Concatenate(c2,Transpose(2,2)	32	......
	Conv2D (3,3)	32	relu
Block 9	Conv2DTranspose (2,2)	16	....
	Concatenate(c1,Transpose(2,2)	16	......
	Conv2D (3,3)	16	relu

Table 4.1: U-Net architecture.

4.3.3 Intersection Over Union

IOU (Intersection over Union) is a phrase used to define the amount to which two boxes overlap. The bigger the overlap zone, the greater the IOU [30]. Which is a suitable indicator for determining the model's robustness. For the segmentation model, An Intersection over Union score greater than 0.5 is normally considered a "good" prediction. this model shows 0.50 IOU and 0.58 after improving it with a minimum area rectangle.

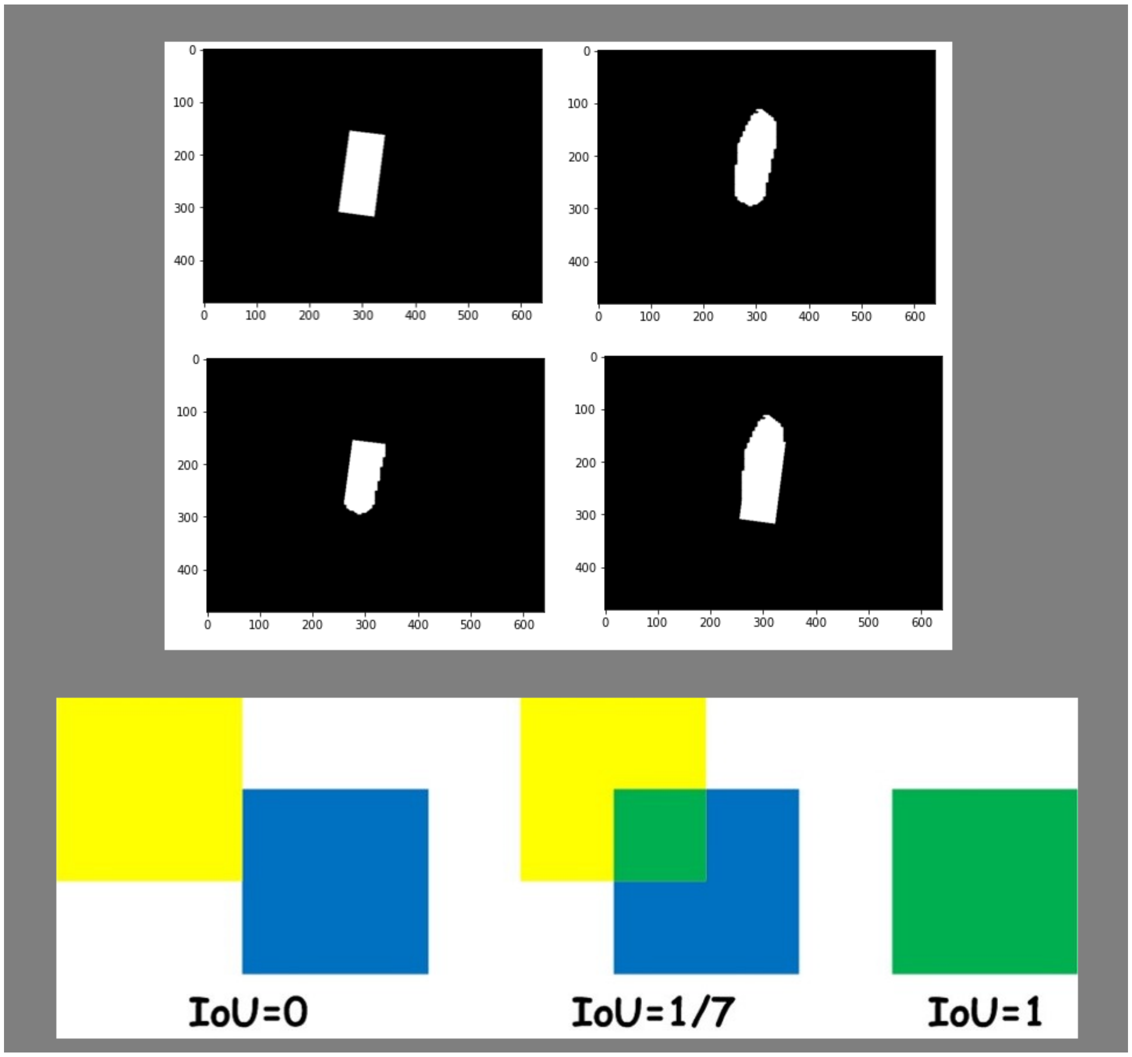


Figure 4.6: Intersection Over Union. The first left image is the ground truth mask. The first right image is the predicted. The second left shows the intersection between the ground truth and the predicted mask, while the second right represents the union

4.3.4 Minimum area rectangle

In order to improve the results that come from the U-Net model, the masks returned to the original rectangle shape by drawing a minimum rectangle around the predicted masks with considering the rotation. The figure 4.7 shows two rectangles; the green one is the regular bounding rectangle, while the blue one is the minimum area rectangle.

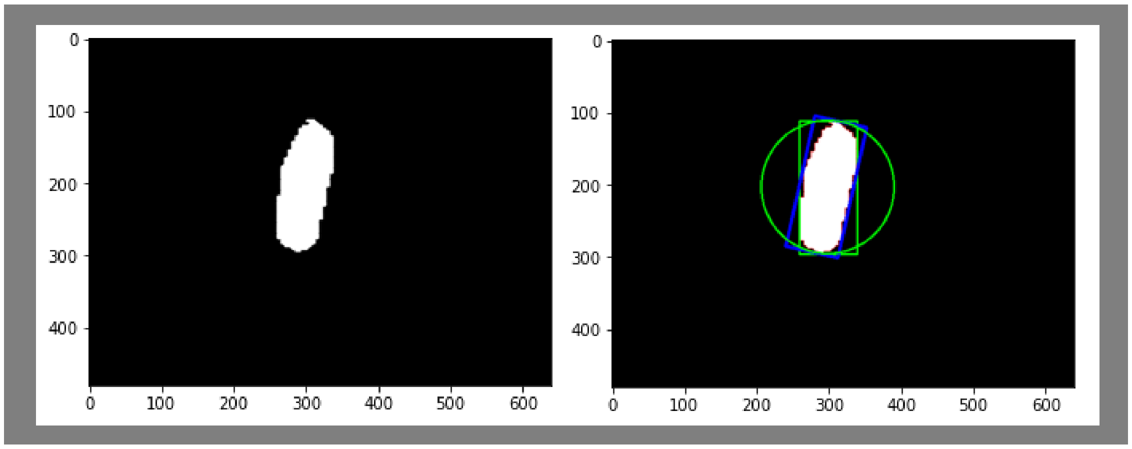


Figure 4.7: Bounding rectangle with the minimum area

Chapter 5

Thesis Results

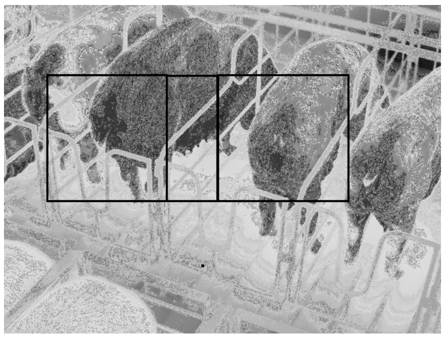
5.1 Results of machine learning model

This section presents and summarizes the results obtained from the application of the machine and deep learning-based object detection models detailed in the previous sections. The results address the effects of the number of Gabor filters and the chosen target on algorithmic performance. A brief comparison to a Convolutional Neural Networks-based approach provides a baseline for qualitative and quantitative efficacy measures.

5.1.1 Using HOG with 12 Gabor filters

The results shown in figure 5.1a summarize the outputs of the object detection algorithm for the HOG feature alone without the inclusion of the Gabor filter enhancement. After concatenating 12 Gabor filters with the model, Figure 5.2d shows the results achieved which is more accurate and reliable.

The graph in Figure 5.3 presents the accuracy of the object detection algorithm in relation to the number of Gabor kernels used in generating the feature maps. The results illustrate that of any number of kernels over 12 (Gabor feature maps) begins to significantly decrease the model's accuracy. This can be attributed to the similarity in the concatenated responses where the overall feature map starts to become similar in all regions or textures. Therefore the model will find it hard to distinguish between regions. A significant drop in the model's accuracy was noticed after using 12 filters.


(a) Result 1

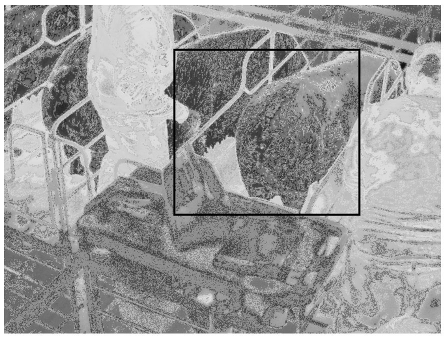
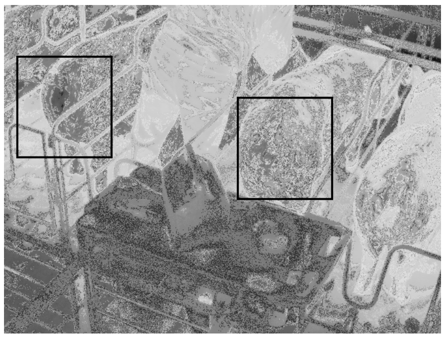
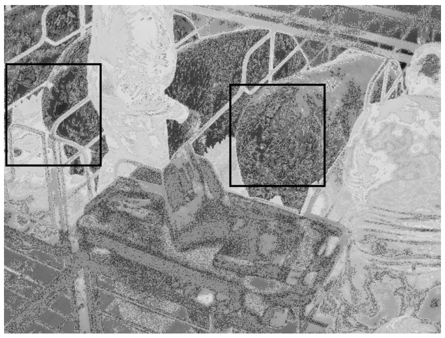
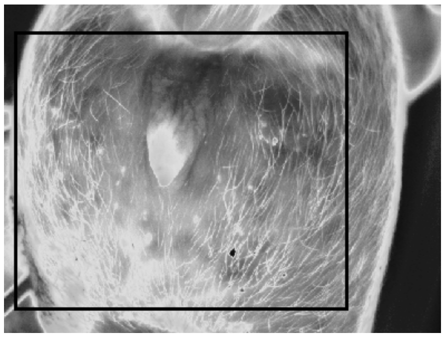

(b) Result 2

Figure 5.1: Results using HOG without Gabor filters


(a) Result 1


(b) Result 2


(c) Result 3

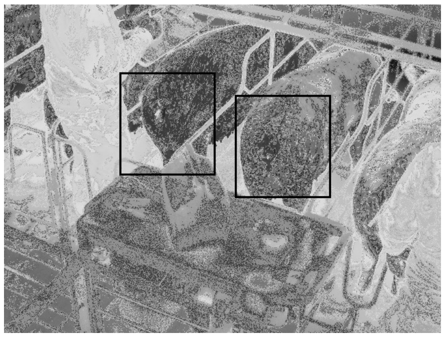

(d) Result 4

Figure 5.2: Final results after incorporating Gabor responses in the feature maps.

5.1.2 Results using 96 Gabor filters

The system was evaluated using qualitative and quantitative assessments. Figure 5.4 shows the quantitative results which depict the confusion matrix of our classification model. It demonstrates that our system can correctly classify the pig's vulva by 93%. It also shows that the percentages of false detection and missing

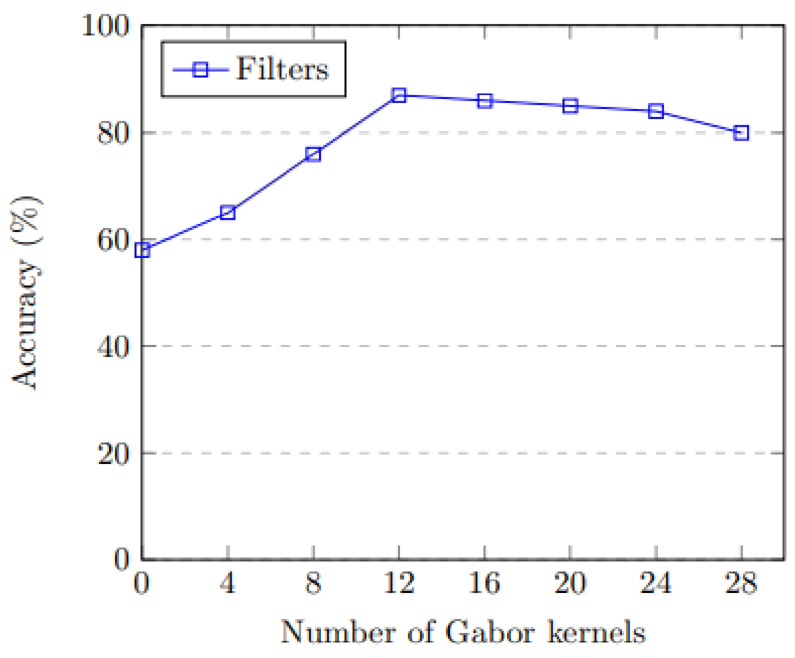
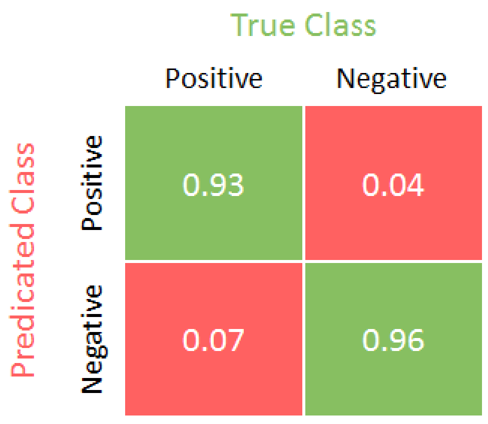


Figure 5.3: Gabor filter accuracy.

detection are 4% and 7%, respectively. These values emphasize the high performance the system achieves with considering the limited dataset.



		True Class	
		Positive	Negative
Predicated Class	Positive	0.93	0.04
	Negative	0.07	0.96

Figure 5.4: The resultant confusion matrix of our trained system.

On the other hand, Figure 5.5 shows the qualitative results that our system achieves on a high-level range of image scenarios. For instance, Figure 5.5(f) and Figure 5.5(g) present the high detection response of our system. When a farmer covers a sow's hindquarter with their hand in Figure 5.5(f), the system is unable to detect vulva in that instance. In contrast, the system becomes able to detect the sow's vulva right after the person removed their hand in Figure 5.5(g). On the left side of the 5.5, the figures represent the robustness of our system on close-up images. These outcomes can be seen in Figures 5.5(a) and 5.5(e) on one side, and all other images captured from a distance on the other side. These results demonstrate the importance of using the Gaussian pyramid technique, which derives the robustness of our system in a variety of scenarios and scales.

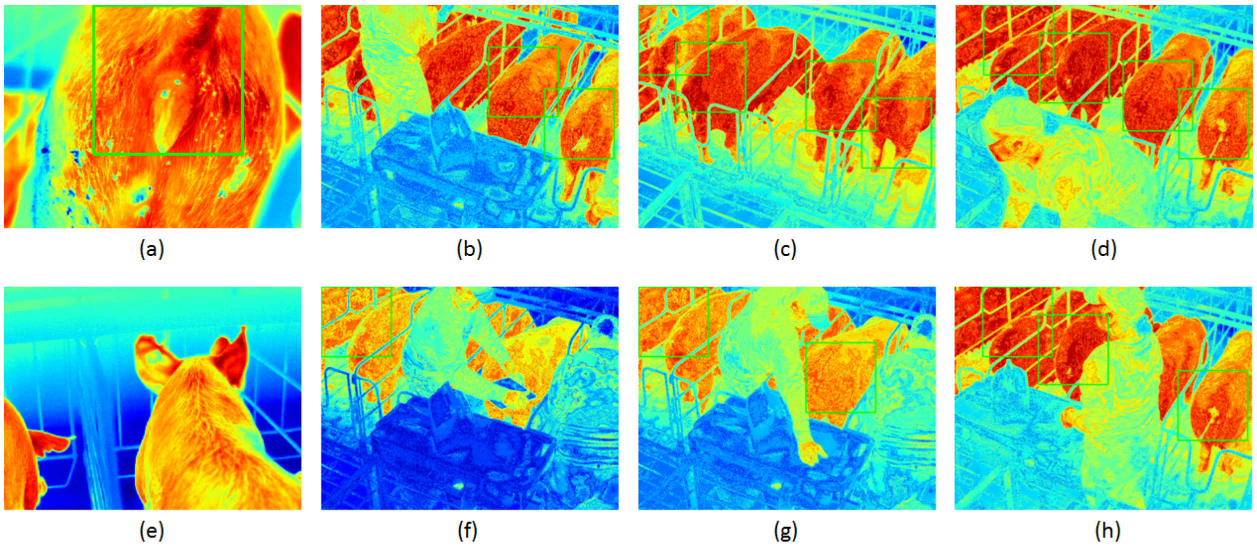


Figure 5.5: A qualitative results of the Vulva detection system. The results cover a wide range of scenarios indicating the robustness of our system. Results show in infrared as they appear in the camera.

To further assess, the performance of the proposed model has been compared to the convolutional neural network (CNN). Due to the limited dataset available and less variation between images in our dataset, the model reached overfitting. The results of the CNN pipeline on our dataset are shown in Figures 5.6. The figure depicts how a CNN pipeline is unable to solve this type of problem. This outcome leads us to the conclusion that, despite its lightweight, our model outperforms CNNs on our dataset, producing reliable and high-accuracy results.

Runtime: Figure 5.7 depicts the runtime evaluation measures of our system using Intel core i7-7500U CPU with respect to the number of Gabor kernels used. These times represent the average running times of the testing algorithm considering the entire test dataset. In the experiments, the model was run on 96 Gabor kernels and obtained results in 0.4 ms. It emphasizes the system's lightness and ability to operate in real

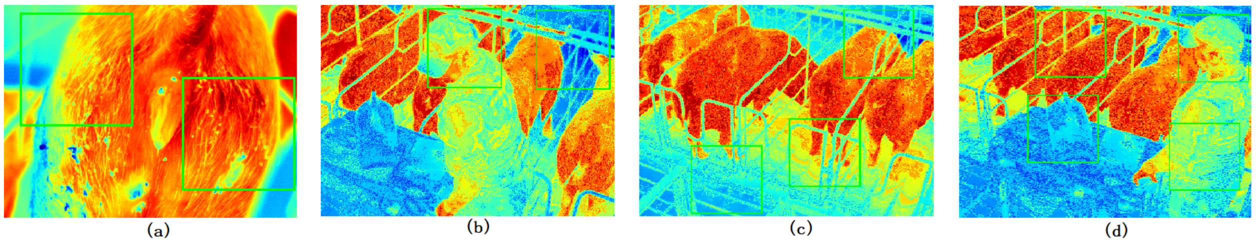


Figure 5.6: The results of the CNN model [43] on our dataset. The results show that CNN is unable to address the dataset limitation of these types of problems in a wide range of scenarios.

time. Also, It shows that the calculated time slightly increases when kernels are less than 400, but it changes dramatically when more than 400 Gabor kernels are used.

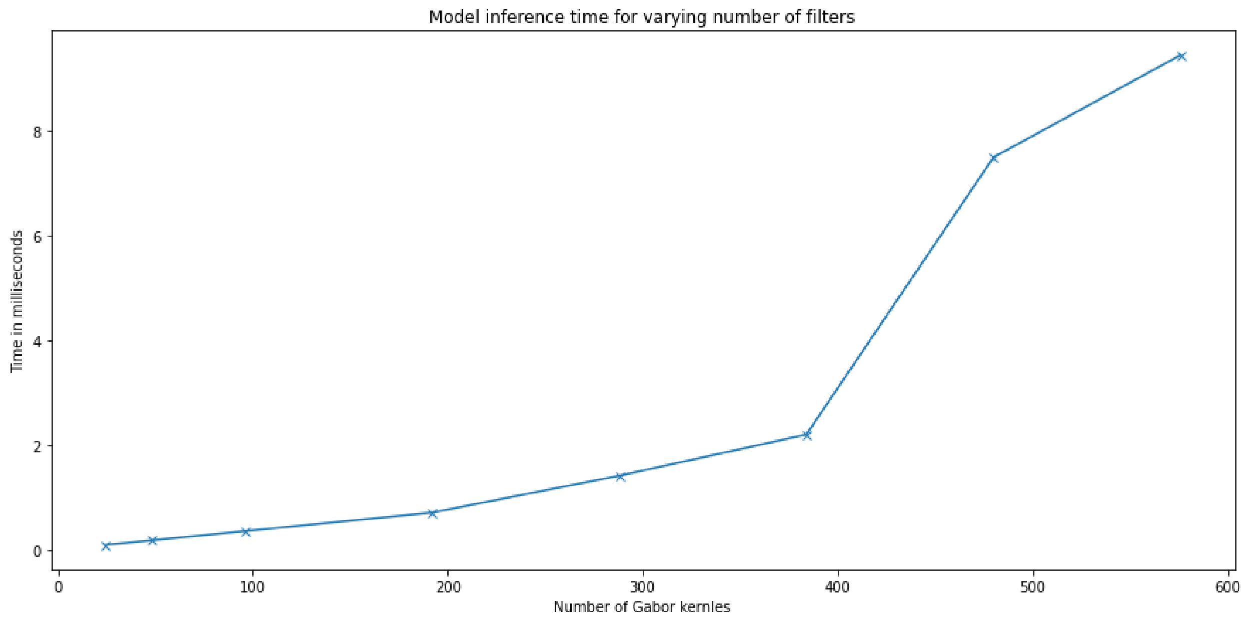


Figure 5.7: Model inference time for the varying number of filters

5.2 Results of YOLOv3 model

The interpretation of a YOLO model prediction results is just as nuanced as the model's implementation. A successful interpretation and accuracy rating is determined by several elements, including the box confidence score and class confidence score utilized when developing a YOLOv3 computer vision model. The results were evaluated using the mean average precision (mAP), The mAP computes a score by comparing the ground-truth bounding box to the detected box. The higher the score, the more precise the model's detections. Fig 5.8 shows the results that were obtained utilizing our custom gray-scale dataset. The images show the detection of the vulva and ear in the same model as well as prediction accuracy.

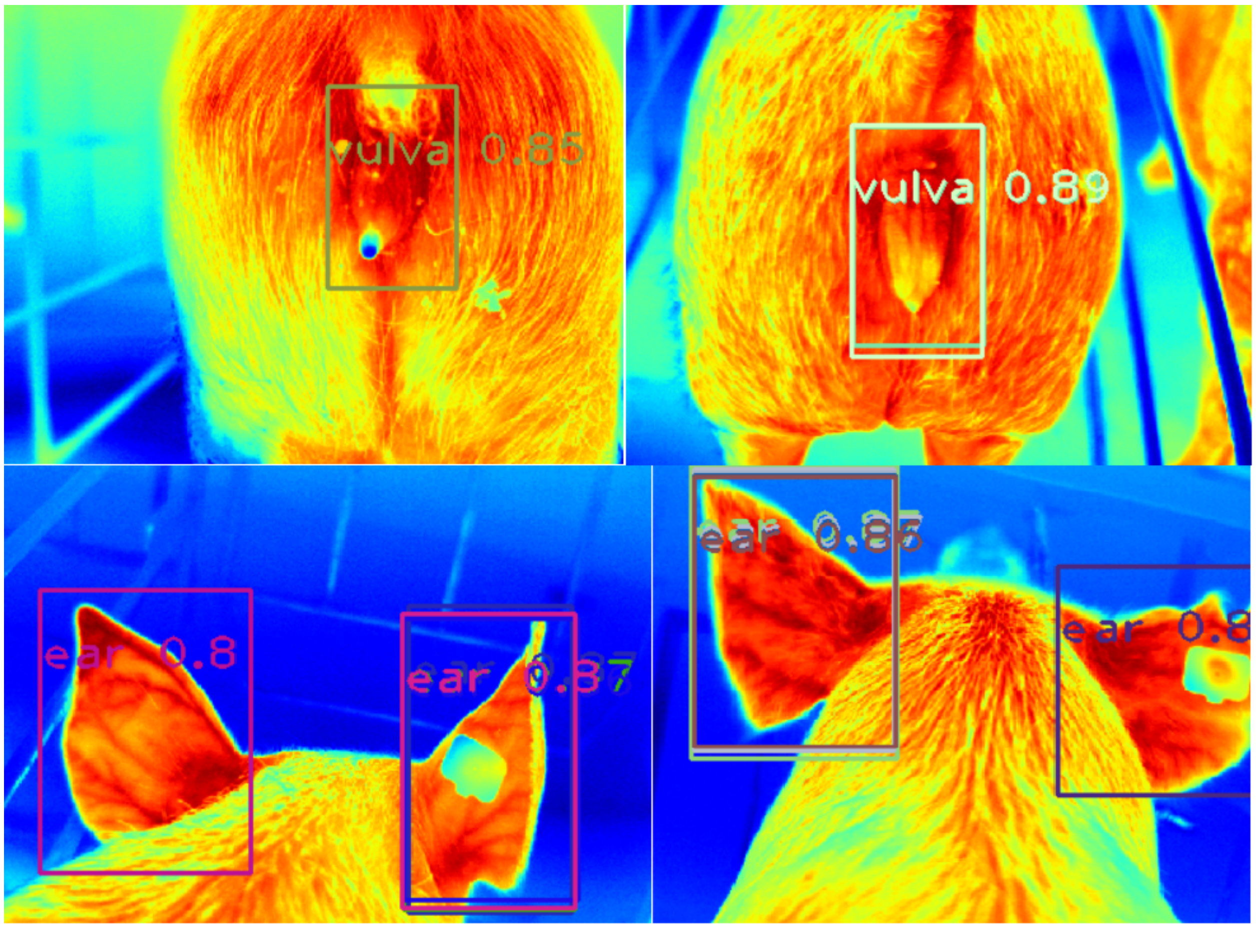


Figure 5.8: YOLOv3 results

5.3 Conclusion and future work

In the first chapter, the problem of detecting vulva in sows using a machine learning-based algorithm has been studied. The base purpose behind this topic was to localize vulva in thermal images. this began by using three main steps, which are: collecting dataset, feature selection, and extraction, then normalization. Collecting and annotating a sufficient number of images was the most challenging part. The challenge of data collection prompted the development of this lightweight machine-learning algorithm for detecting sows. Nonetheless, when the sliding window was examined alongside the Gaussian pyramid techniques for scalable feature extraction, the system shows robustness and works efficiently with a high positive detection rate and low false and missing detection rates.

YOLO was explored in the second chapter. The ”You.Only.Look.Once” (YOLO) model family is a collection of end-to-end deep learning models [42] built for quick object detection. The method employs a single deep convolutional neural network that divides the input into a grid of cells, each of which immediately predicts a bounding box and object classification. As a consequence, several candidate bounding boxes are generated for each object, and then a single entity is selected in a final prediction by using a non-max suppression technique.

In the third chapter, U-Net semantic segmentation model has been utilized. Semantic image segmentation aims to label pixels of an image with a compatible class of what is being represented. This task is known as ”dense prediction” because the predicting was for every pixel in the image. The U-Net architecture has two paths. The first path is the contraction path (also known as the encoder), which is used to store the image's context. The encoder is just a stack of convolutional and max pooling layers. The symmetric expanding path (also known as the decoder) is utilized to achieve exact localization via transposed convolutions. Using U-Net allows us to obtain the most precise results. Despite the fact that accuracy and model speed are trade-offs, the model demonstrates the capacity to recognize the vulva with the greatest IOU and the highest speed. The table 5.1 summaries the comparison between the different models used in this project.

This work is part of a more extensive framework to automate the process of breeding sows. The first step is to accurately localize the Regions Of Interest (vulva, ears, and back of the thighs where ovaries can be visible to alter the temperature of the outer skin). Time series analysis will be performed to identify patterns in the ROIs for accurate estrous predictions.

Comparison between models			
Approach	Mean IOU	Improved IOU	FPS
SVM	0.400	0.450	35
SVM after Gabor	0.490	0.515	25
YOLOv3	0.520	0.520	40
U-Net	0.500	0.586	30

Table 5.1: Comparison between models

Bibliography

[1] Murad Al-Madani, Umair bin Waheed, and Mudassir Masood. Fast and accurate dictionary learning for seismic data denoising using convolutional sparse coding. In *SEG International Exposition and Annual Meeting*. OnePetro, 2019.

[2] Iyad Almadani, Mohammed Abuhussein, and Aaron L Robinson. Sow localization in thermal images using gabor filters. In *FICC2022*, accepted in 2021.

[3] Murad Almadani, Umair bin Waheed, Mudassir Masood, and Yangkang Chen. Dictionary learning with convolutional structure for seismic data denoising and interpolation. *Geophysics*, 86(5):1–102, 2021.

[4] Abhishek Bansal, Kapil Mehta, and Sahil Arora. Face recognition using pca and lda algorithm. In *2012 second international conference on Advanced Computing & Communication Technologies*, pages 251–254. IEEE, 2012.

[5] Gary Bradski and Adrian Kaehler. Opencv. *Dr. Dobb's journal of software tools*, 3, 2000.

[6] Dario Cabib, Moshe Lavi, Amir Gil, and Uri Milman. Long wave infrared (8 to 14 microns) hyperspectral imager based on an uncooled thermal camera and the traditional ci block interferometer (si-lwir-uc). In *Infrared Technology and Applications*, volume 8012, page 80123H. International Society for Optics and Photonics, 2011.

[7] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers and Electrical Engineering*, 40(1):16–28, 2014. 40th-year commemorative issue.

[8] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[9] Daan De Geus, Panagiotis Meletis, and Gijs Dubbelman. Panoptic segmentation with a joint semantic and instance segmentation network. *arXiv preprint arXiv:1809.02110*, 2018.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] FLIR. *FLIR Camera Adjustments LWIR Video Camera.* Number 102-PS242-100–01. Jun 2014.

[12] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, and Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. *Applied Soft Computing*, 70:41–65, 2018.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pages 770–778. IEEE, June 2016.

[14] Rachel Huang, Jonathan Pedoeem, and Cuixian Chen. Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2503–2510. IEEE, 2018.

[15] Miso Ju, Younchang Choi, Jihyun Seo, Jaewon Sa, Sungju Lee, Yongwha Chung, and Daihee Park. A kinect-based segmentation of touching-pigs for real-time monitoring. *Sensors*, 18(6):1746, 2018.

[16] Phil Kim. Convolutional neural network. In *MATLAB deep learning*, pages 121–147. Springer, 2017.

[17] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[18] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*, pages 734–750, 2018.

[19] Guangbo Li, Jun Jiao, Guolong Shi, Huimin Ma, Lichuan Gu, and Liang Tao. Fast recognition of pig faces based on improved yolov3. In *Journal of Physics: Conference Series*, volume 2171, page 012005. IOP Publishing, 2022.

[20] Weitao Li, Kezhi Mao, Hong Zhang, and Tianyou Chai. Selection of gabor filters for improved texture feature extraction. In *2010 IEEE International Conference on Image Processing*, pages 361–364. IEEE, 2010.

[21] Tsung-Yi Lin. Piotr dollã ar, ross b. girshick, kaiming he, bharath hariharan, and serge j. belongie. feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[23] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[24] Qi-Chao Mao, Hong-Mei Sun, Yan-Bo Liu, and Rui-Sheng Jia. Mini-yolov3: real-time object detector for embedded applications. *Ieee Access*, 7:133529–133538, 2019.

[25] Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *Proceedings of the 18th International Conference on Pattern Recognition - Volume 03*, ICPR '06, pages 850–855, Washington, DC, USA, 2006. IEEE Computer Society.

[26] Agyztia Premana, Akhmad Pandhu Wijaya, and Moch Arief Soeleman. Image segmentation using gabor filter and k-means clustering method. In *2017 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pages 95–99. IEEE, 2017.

[27] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016.

[28] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[29] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[30] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 658–666, 2019.

[31] Bernardino Romera-Paredes and Philip Hilaire Sean Torr. Recurrent instance segmentation. In *European conference on computer vision*, pages 312–329. Springer, 2016.

[32] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[33] PK Sathish and S Balaji. A complete person re-identification model using kernel-pca-based gabor-filtered hybrid descriptors. *International Journal of Multimedia Information Retrieval*, 7(4):221–229, 2018.

[34] Vasco G Simões, Faouzi Lyazrhi, Nicole Picard-Hagen, Véronique Gayrard, Guy-Pierre Martineau, and Agnès Waret-Szkuta. Variations in the vulvar temperature of sows during proestrus and estrus as determined by infrared thermography and its relation to ovulation. *Theriogenology*, 82(8):1080–1085, 2014.

[35] Saraswathi Sivamani, Seong Ho Choi, Dong Hoon Lee, Jihwan Park, and Sunil Chon. Automatic posture detection of pigs on real-time using yolo framework. *Int. J. Res. Trends Innov*, 5:81–88, 2020.

[36] Frédéric Suard, Alain Rakotomamonjy, Abdelaziz Bensrhair, and Alberto Broggi. Pedestrian detection using infrared images and histograms of oriented gradients. In *2006 IEEE Intelligent Vehicles Symposium*, pages 206–212. IEEE, 2006.

[37] Shan Suthaharan. Support vector machine. In *Machine learning models and algorithms for big data classification*, pages 207–235. Springer, 2016.

[38] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

[39] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[40] Eduardo Zalama, Jaime Gómez-García-Bermejo, Roberto Medina, and José Llamas. Road crack detection using visual features extracted by gabor filters. *Computer-Aided Civil and Infrastructure Engineering*, 29(5):342–358, 2014.

[41] EA Zanaty. Support vector machines (svms) versus multilayer perception (mlp) in data classification. *Egyptian Informatics Journal*, 13(3):177–183, 2012.

[42] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[43] Liquan Zhao and Shuaiyang Li. Object detection algorithm based on improved yolov3. *Electronics*, 9(3):537, 2020.

[44] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[45] Ev Zisselman, Jeremias Sulam, and Michael Elad. A local block coordinate descent algorithm for the csc model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8208–8217, 2019.