

AN AUTOMATED 3D POSE ESTIMATION SYSTEM FOR SOW HEALTH MONITORING

By

Steven Yik

A THESIS

Submitted to  
Michigan State University  
in partial fulfillment of the requirements  
for the degree of

Electrical Engineering – Master of Science

2020

# ABSTRACT

## AN AUTOMATED 3D POSE ESTIMATION SYSTEM FOR SOW HEALTH MONITORING

By

Steven Yik

Pork ranks as one of the most consumed meats globally, presenting both a challenge and an opportunity to both improve the care of swineherds and to increase efficiency of production. Conditions such as lameness and poor body composition impair productivity and animal welfare, while current assessment methods are subjective and labor-intensive, resulting in slower production and ambiguous quality classifications. Precision Livestock Farming (PLF) proposes using technology to monitor animals, assess health, and apply data-driven interventions to increase welfare and produce higher quality products. For sows, body shape and motion characteristics provide important health indicators that could be assessed automatically through appropriate PLF sensors and techniques.

This thesis developed a sow PLF health assessment device using artificial intelligence, including both a hardware system and a novel training algorithm. First, a data collection device, the *SIMKit*, was built using modern dense depth sensors, which can reveal detailed shape characteristics of a sow. Second, a new annotation method called, *Transfer Labeling* was developed, enabling the semi-automated annotation of a large dataset of sow depth images. This dataset was used to train a convolutional neural network (CNN) to detect and track pig poses. Results show that *Transfer Labeling* produces annotations with sub-centimeter accuracy with much-reduced human effort. It is anticipated that this will lead to much-improved sow health monitoring.

## ACKNOWLEDGMENTS

I want to take this time to offer my sincere thanks to Dr. Madonna Benjamin for her support and encouragement throughout this project and research career. Her passion for veterinary practice and welfare for animals has opened me up to a completely different domain, to which I would not have had any exposure. The inclusion of many administrative activities has exposed me to many technical challenges when dealing with research and broad avenues to incorporate when making decisions on project direction. She has helped me to become more mindful and become the engineer I am today.

I would also like to thank Dr. Daniel Morris for his expertise in computer vision and guiding me throughout this research project. Many lessons learned and useful practice techniques have helped shape how I tackle new tasks for future projects. I would also gratefully acknowledge Dr. Michael Lavagnino for his great support in analysis and assistance-ship for acquiring more students to fulfill areas of need on the project. Finally, acknowledgment of Dr. Vaibhav Srivastava for serving on my defense committee and providing feedback to my work.

I want to show my appreciation to the farms that allowed for us to record data at their facilities:

- Valley View
- MSU Swine Farms
- Barton Farm

Here is a list of investors that made this project possible, thank you for believing in our work:

- Michigan Pork Producers - Pigs Across State Lines Without Change of Ownership
- Michigan Alliance for Animal Agriculture 2018 Grant
- MTRAC Program - by the State of Michigan 21st Century Jobs Fund received through the Michigan Strategic Fund and administered by the Michigan Economic Development Corporation

## TABLE OF CONTENTS

LIST OF TABLES . . . . .	vi
LIST OF FIGURES . . . . .	vii
CHAPTER 1 INTRODUCTION AND OBJECTIVES . . . . .	1
1.1 Motivation . . . . .	1
1.2 Approach . . . . .	2
1.3 Research Contributions . . . . .	4
1.4 Thesis Organization . . . . .	4
CHAPTER 2 LITERATURE REVIEW . . . . .	6
2.1 Sow Welfare Challenges . . . . .	6
2.1.1 Lameness . . . . .	6
2.1.2 Body Condition . . . . .	7
2.1.3 Sow Group Behavior . . . . .	9
2.2 Precision Livestock Farming . . . . .	9
2.3 Computer Vision For Sow Health Evaluation . . . . .	10
2.3.1 Lameness Detection . . . . .	10
2.3.2 Body Condition Analysis . . . . .	10
2.3.3 Observing Sow Group Behavior . . . . .	11
2.4 Artificial Neural Networks . . . . .	11
2.4.1 Neuron Structure and Network Architecture . . . . .	12
2.4.2 2D Convolution Layers . . . . .	14
2.4.3 Pooling Layers and Receptive Field . . . . .	15
2.4.4 Skip Connections and Residual Blocks . . . . .	17
2.5 Summary . . . . .	17
CHAPTER 3 SOW DATA COLLECTION . . . . .	19
3.1 Introduction . . . . .	19
3.2 Farm Environmental Hazards . . . . .	19
3.2.1 Biosecurity and Connectivity . . . . .	20
3.3 SIMkit Collection Device . . . . .	21
3.3.1 Design Requirements . . . . .	21
3.3.2 Concept Design . . . . .	22
3.3.3 Prototype Designs . . . . .	23
3.3.4 Sensor Selection and Placement . . . . .	25
3.3.5 Computing Platform and Data Acquisition . . . . .	26
3.3.6 Data Transportation Pipeline . . . . .	28
3.4 Data Preprocessing . . . . .	29
3.4.1 Motion Detection Filtering . . . . .	29
3.4.2 Depth Image Preprocessing . . . . .	30



3.4.3	Health Record Processing . . . . .	31
3.5	Summary . . . . .	31
CHAPTER 4 SEMI-AUTOMATED LABELING . . . . .		33
4.1	Overview . . . . .	33
4.2	Introduction . . . . .	33
4.3	Annotation Framework . . . . .	34
4.4	Transfer Labeling . . . . .	34
4.5	Preliminary Preparation . . . . .	35
4.6	Mark Detection Neural Network . . . . .	36
4.6.1	Mark Labeling . . . . .	36
4.6.2	Network Architecture and Implementation . . . . .	37
4.6.2.1	Input and Ground Truth Layers . . . . .	37
4.6.2.2	Architecture . . . . .	38
4.6.2.3	Implementation Details . . . . .	39
4.6.3	Predictive Performance . . . . .	40
4.7	Optical Flow Joint Association . . . . .	42
4.7.1	Outline . . . . .	43
4.7.2	Development . . . . .	44
4.7.3	Implementation and Labeling Procedure . . . . .	46
4.8	Evaluation and Performance . . . . .	47
4.9	Summary on Semi-Automated Labeling . . . . .	47
CHAPTER 5 POSE ESTIMATION NETWORK . . . . .		49
5.1	General . . . . .	49
5.2	Pose Network Architecture . . . . .	49
5.2.1	Hourglass Module . . . . .	50
5.2.2	Complete Architecture Design . . . . .	50
5.3	Dataset Preprocessing . . . . .	52
5.4	Network Implementation . . . . .	53
5.4.1	Input and Output Layers . . . . .	53
5.4.2	Loss Function . . . . .	53
5.5	Predictive Performance . . . . .	54
5.5.1	Heatmap Joint Extraction . . . . .	57
5.5.2	Accuracy Evaluation . . . . .	57
5.6	Summary . . . . .	60
5.7	Alternative Approaches and Future Work . . . . .	61
CHAPTER 6 CONCLUSIONS . . . . .		63
APPENDICES . . . . .		65
APPENDIX A	SIMKIT - HARDWARE . . . . .	66
APPENDIX B	SIMKIT - SOFTWARE . . . . .	82
BIBLIOGRAPHY . . . . .		84

## LIST OF TABLES

Table 1.1: Reasons for sow removal by total percentage [36]. . . . .	2
Table 3.1: SIMKit preliminary hazard analysis . . . . .	22
Table 4.1: Association results for landmarks in IR images. After human assignment of IDs, only 1.3% of images need human attention. . . . .	47
Table 5.1: Average error and miss detection rate between depth-based pose-estimation network and annotations . . . . .	57
Table 5.2: Predicted joint deviation along the axial and lateral axis of the sow for a 99.7% confidence interval. . . . .	60

## LIST OF FIGURES

Figure 2.1: Visual body condition scoring with the following rankings: Thin=1; Good=3; Fat=5 . . . . .	8
Figure 2.2: Caliper measurement of angularity along the last rib for body condition assessment. Colored regions on the caliper indicate the body condition score from Figure 2.1, [19] . . . . .	8
Figure 2.3: Neuron architecture . . . . .	12
Figure 2.4: A simple neural network architecture. . . . .	14
Figure 2.5: Top-down view of a convolution operation. The kernel is represented as the blue square and the convolution operation performs a region-wise dot product across the image with the neural network able to produce a set of filters to match an output. . . . .	15
Figure 2.6: Max-pooling operation with a kernel size of 2. Max-pool denotes that the maximum value in in the region is the output. Each color denotes the region from the input matrix used to compute the output. . . . .	16
Figure 2.7: Residual block architecture . . . . .	17
Figure 3.1: A simplified view of how the <i>SIMkit</i> device operated in the farm setting. . . . .	23
Figure 3.2: <i>SIMKit</i> prototype using a Raspberry Pi as the base processor along with an Intel Realsense camera. . . . .	23
Figure 3.3: CAD rendering of the final Simkit design. Features include: tactile buttons for interfacing with the monitor, 2-axis rotation on camera mounts for re-positioning, and easy mounting with compliant conduit PVC piping. . . . .	24
Figure 3.4: Detailed view of the camera holders. There are 2 pivot axes designed for roll and pitch movement. There is a location for a fan and the design allows for enclosed wiring to the processor. . . . .	25
Figure 3.5: Simkit installed on the ceiling within a farm hallway. Camera field of view is indicated by the blue overlaid region in the image . . . . .	27
Figure 3.6: A block diagram outlining how data is obtained, processed, and transferred to a remote database. . . . .	29

Figure 3.7: Processed depth image with a color map and contour plot overlay. . . . .	31
Figure 4.1: These marks are paint crayon markings on the sows back, used to obtain image labels. . . . .	36
Figure 4.2: A high level diagram of the mark detection neural network architecture. Each orange box denotes a sequence of 3: convolution>batch_norm>relu layers followed by either a max_pool (shrinking in size) or upsampling layer (growing in size) with all layers having 64 channels each. . . . .	40
Figure 4.3: A graph showing the network loss during training for 75 epochs (Blue: training, Red: validation). . . . .	41
Figure 4.4: A histogram of the total error between the detected and human labeled joint center locations in pixels. The average error is 2 pixels or 0.5 cm accuracy. . . .	42
Figure 4.5: The output of the mark detection network as a heatmap overlaid over the input infrared image. Marks in red denote high detection probability while blue indicates low detections. . . . .	42
Figure 4.6: Visual representation of the joint flow vectors. The middle image is the initialized set of joint labels from the labeler. The top image indicates predicted joint locations on the $i + 1$ frame for forward motion overlaid on the infrared image. Likewise, the bottom image is the same but for the $i - 1$ frame which is backwards motion. . . . .	45
Figure 4.7: The full joint detection and association process for generating annotations. Each image corresponds to separate step in the process and are described in section 4.7.3. . . . .	46
Figure 5.1: The architecture of a single hourglass module within the Stacked Hourglass Network. . . . .	50
Figure 5.2: The residual module specific to the Stacked Hourglass Network design. Numbers below the convolution blocks are the number of channels in that layer. . . .	51
Figure 5.3: An high level overview of the Stacked Hourglass Network architecture. Blue blocks denote convolution layers used to match channel sizes between each module. Red blocks are an hourglass module described in section 5.2.1. The circle shows the locations in the network where supervised loss is evaluated. . . .	51
Figure 5.4: Network structure in between the hourglass modules within the Stacked Hourglass Network design. The red symbol denotes an hourglass module. Numbers below the convolution blocks are the number of channels in that layer. . . .	52

Figure 5.5: Sample transformed depth image, joint image at the last rib, and weight map. . .	52
Figure 5.6: Hourglass module loss graphs (loss/epoch). Red line denotes the validation loss and blue denotes the training loss. . . . .	54
Figure 5.7: The loss graph of the training and validation from the complete stacked hourglass network per epoch. Red denotes the validation and blue being training. . .	54
Figure 5.8: Error graph between the infrared mark extracted joint locations and the depth pose estimation network. The average error is 8.7 pixels (2.1cm) with a 2.4% miss rate. . . . .	55
Figure 5.9: Error graph between the human annotated joint locations and the depth pose estimation network. The average error is 11.2 pixels (2.6cm) with a 2.6% miss rate. . . . .	55
Figure 5.10: A sample output of the pose detection network. A heatmap of joint existence is generated for all 8 joints with red denoting the highest probability and blue being the lowest. The size of the sow is roughly 1.2m in length. . . . .	56
Figure 5.11: Evaluation of the 99.7% confidence region for predicted joint locations indicated by the blue ellipses. . . . .	58
Figure 5.12: Network sample outputs for multiple sow poses. Left is the ground truth infrared image with the labeled joints indicated in red. Right is the compiled network output on the depth image with detected joint locations represented by the heatmap. . . . .	59
Figure 5.13: Compiled image of all joints detected overlaid over the input infrared image with a drawn skeleton to represent the full body pose. . . . .	61

# CHAPTER 1

## INTRODUCTION AND OBJECTIVES

### 1.1 Motivation

Pork, produced from the muscles of pigs, is the most consumed meat in the world, and the demand will continually grow with the population. To meet this need from both export and domestic market standpoints, the U.S. has expanded pork production capabilities from 63 to 77 million pigs since 2013. Consequently, the number of breeding female pigs giving birth (farrowing) has increased from 5.67 million to 6.38 million [41]. The massive influx of pork demand has caused challenges for farmers trying to find and maintain labor and ways to increase pork production.

Raising breeding stock is human labor-intensive, requiring a trained workforce standard of one stock-person for 300 sows. Yet, farmers are facing challenges in securing a labor force due to the physical demands when working with livestock and high employee turnover, limited access to labor in rural areas, and diminishing numbers of foreign-born workers. Also, to maintain assurance standards for safe, quality pork products, farmers must show verification of employee training in all areas of pig production that they participate in [8].

Another difficulty in increasing sow production is how to increase the amount of pork produced. The breeding female pig (sows and gilts) is the limiting factor in production. The number of times a sow will give birth to piglets is defined as sow longevity. On average, sows are replaced after bearing 3–4 litters [13], but this varies greatly, depending on reproductive history, sow health, and the availability of replacement animals. Studies have shown that retaining sows in the herd longer (longevity) has economic benefits such as the amount of pork produced and increased farm revenue [14]. Longevity is determined by the sows ability to maintain productivity over the potential productivity of her replacement. A replacement breeding animal (gilt) fills the production space vacated by lost sows either due to mortality (death) or from culling (removal from the herd).

Thereby, an indicator of herd sow longevity, turnover rate, is the sum of mortality and culling. Two common reasons cited for sow mortality and premature culling of sows are lameness and poor body condition [13], with a combined contribution of 30% total loss, Table 1.1 shows removal percentages by category. Research in cattle suggests evidence to target the management of body condition to minimize the risk of lameness [32]. Unfortunately, stockperson assessment of these health and welfare indicators is currently subjective [30, 2, 3] and often too late to remedy the outcomes [1].

**Table 1.1:** Reasons for sow removal by total percentage [36].

Reason for removal	Proportion of total (%)
Reproduction	26.95
Locomotion	15.49
Low production	12.77
Disease	12.99
Misc	7.70

Subsequently, the swine industry requires a tool to easily train stockpersons and a device to quantitative assess sow health and welfare conditions. Precision livestock farming (PLF) is the use of technology to monitor attributes of individual livestock for animal welfare. PLF methods use technology to take quantitative measurements from sensors to monitor animals. This type of device provides a tool for stockpersons and can promote more efficient farming practices through early detection of factors related to sow longevity.

## 1.2 Approach

The PLF system developed in this thesis performs individualized health monitoring of sows using: 1) visual data recording system and 2) a neural network to specifically observe sow functional morphology. Functional morphology is the study of an animal's structure to determine its function. Observing the functional morphology reveals the development or loss of fat and muscle throughout her production cycle [9] and locomotion patterns [9, 34]. By training a neural network system to

perform these evaluations, it provides an approach to assess animal health quantitatively.

Neural networks are a type of artificial intelligence that learns the inverse relationship between the desired output and input. For input data depth, images are collected from sensors to observe animal motion and shape. The network observes functional morphological features, and it extracts information and outputs the pose of a sow, used to analyze the body structure. For a biological organism, the skeletal structure is the structural foundation. The software equivalent is estimating pose, a process used to track and locate an object within an image or video. A neural network can learn a function to relate sensor information that encodes the animal's structure to produce pose estimates used for health analysis. For this network to learn this task, a large dataset of known inputs and outputs is required. Therefore, the acquisition or creation of a large dataset is required to train such a network.

Currently, there are very few datasets publicly available to develop complex models that incorporate various modes of sow health. Since one suited for sow pose estimation is not publicly accessible, the creation of a sow dataset is necessary. Therefore, we have created our own dataset with infrared and depth images of sows using a custom designed data collection system. Furthermore, the dataset includes various key indicators of sow health, such as any medical conditions or breeding patterns, which is useful for development of prediction models.

The network's goal is to generate pose estimates based on depth image data. Pose provides a structure to observe kinematic motion and establishes reference points for body condition assessments. Utilizing a pose estimation network establishes a basis for health monitoring, and will be used to predict future reproductive performance and risk of culling.



### **1.3 Research Contributions**

The first objective of this research is to develop the hardware necessary to collect data of sows in a noninvasive manner. The second is to develop a model to assess the pose from sow functional morphology. These two objectives are divided into the following tasks:

1. Developed a robust data collection hardware system, utilizing dense depth and infrared sensors for image creation.
2. Created a neural network semi-automatic labeling method for efficient annotation for large sets of images.
3. Performed evaluation of the new labeling method vs. human labeling.
4. Developed of a neural network-based pose estimation on depth images for sow health monitoring.
5. Performed evaluations of the depth-based pose-estimation neural network for detection accuracy.

### **1.4 Thesis Organization**

Chapter 2 offers a review of the literature addressing sow health and welfare challenges, and methods and technologies researchers have developed within sow health evaluation. Chapter 3 provides an in-depth look at the development of a robust data collection process. It includes hardware and software details of the collection device as well as various factors regarding revisions of the design. Chapter 4 presents a new method to create a large dataset utilizing various computer vision techniques to create large amounts of annotations efficiently. An in-depth discussion of each step within the labeling pipeline, key-point extraction model, optical flow label transfer, and fusion algorithm is provided. Finally, Chapter 5 highlights the development processes of the pose estimation network with full details on network architecture, I/O formats, loss function, and

performance metrics. A discussion of suggested modification and future extensions are provided to improve sow pose estimation.

## **CHAPTER 2**

### **LITERATURE REVIEW**

This chapter provides a review of the literature related to this work as well as an introduction to the methods used to address sow health. First, current sow welfare challenges are discussed, with an emphasis on body condition, lameness, and sow movement behavior. Next included is summary of Precision Livestock Farming (PLF) sensor devices, and methodology used to improve sow welfare and farm efficiency. Finally, to underline this research, an overview of computer vision techniques and fundamentals of artificial neural network development.

### **2.1 Sow Welfare Challenges**

Observing sow welfare can help narrow down factors that affect sow reproductive performance. Often, harmful mental or physiological effects prevent a sow from breeding to their full potential and slowing down injury or disease recovery, leading to early culling of the sow. When detected early in development, taking precautionary measures can prevent loss of sows within the herd. This section will focus on a general description of lameness, body condition, and sow group interaction to provide insight towards poor reproductive health.

#### **2.1.1 Lameness**

Lameness is defined by *Merriam-Webster*<sup>7</sup> as "having a body part and especially a limb so disabled as to impair freedom of movement" or as 'impaired movement or deviation from normal gait' [23]. Lameness is a painful condition that negatively affects sow herds by contributing to lower reproductive efficiency and returns to investments. It is a multifactorial problem due to genetic, mechanical, chemical, and biological processes. Lameness is also at a severe disadvantage when it comes to accessing food and water, particularly if they have to compete with pen mates, suffering from both pain as well as hunger and thirst. Not only is lameness a welfare concern, but the cur-

rent incidence in breeding animals results in significant economic losses due to low reproductive capability and early culling or euthanasia of sows. Gruhot [14] notes that sows identified as lame have 1.4 fewer litters per animal and results in 15% [28] of herd culling. As there is a paucity of licensed analgesics for food animals, producers in the United States have few alternatives and often choose to cull affected sows, especially as severity and duration increase [8].

Pain-related behavior can be used as a tool for on-farm grading of lameness, typically by humans, observing selected attributes of lame animals. In dairy cattle, an on-farm five-point lameness or locomotion scoring system emphasizes attributes of lame cows through posture and gait [37] as the cows transition from the field or barn into the milking parlor. This scoring system in cattle has reasonable reliability in terms of intra- and inter-observer agreement [24].

A lameness scoring system for sows would ideally categorize the degree of lameness demonstrated from locomotion. A four-point scale [sound (0), mildly lame (1), moderately lame (2), and severely lame (3)] developed by Zinpro [12] has been used in sows to quantify and evaluate herd lameness prevalence. However, the accuracy of these qualitative methods are highly variable among observers [25]. Therefore there is a need for more objective and quantitative methods to assess lameness in sows.

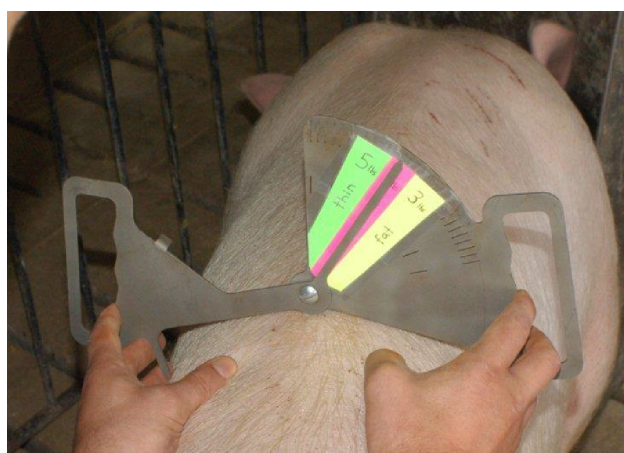
### **2.1.2 Body Condition**

Body condition scoring (BCS) is an on-farm scoring measure [thin (1), good (2), and overweight (3)] of the physical assessment of the body composition. BCS are commonly used to evaluate quality and quantity requirements for a sow's diet [10]. Figure 2.1 shows a visual comparison of three sows with different body compositions. Studies to evaluate BCS to reasons for culling sows with BCS = 1, associated with lameness, reduced farrowing rate and are less likely to exhibit estrus. Whereas overweight sows (BCS=3) were culled due farrowing problems, reduced colostrum production and increased mortality of piglets [20, 21, 18].



**Figure 2.1:** Visual body condition scoring with the following rankings: Thin=1; Good=3; Fat=5

The caliper device, seen in Figure 2.2, was developed in 2012 and is currently being used to evaluate body composition by measuring around the last rib of the sow. The technology quantifies the angularity from the spinous process to the transverse process of a sow's back. The sow caliper is based on the premise that as a sow loses weight, fat and muscle her back becomes more angular [19]. A lower number represents a “thinner” sow and a greater number represents an “overweight” sow. Obtaining accurate caliper evaluations requires that the sow is standing and for many sows, that they are restrained in a stall. In addition, utilizing this information requires that the stockperson can both reach the last rib and transcribe the reading.



**Figure 2.2:** Caliper measurement of angularity along the last rib for body condition assessment. Colored regions on the caliper indicate the body condition score from Figure 2.1, [19]

### **2.1.3 Sow Group Behavior**

Sows are herd animals and isolation from others, especially while entering an unfamiliar space, is stressful. In these situation sows will maintain both visual and body contact by moving side by side or retaining loose bunches. Pigs have excellent hearing and an acute sense of smell. Thereby, sows will use their hearing to track people or novelties they can't see, and are likely, to stop and investigate sights and smells [7]. To conduct locomotion patterns of illness such as lameness, we should also recognize that to reduce the risk of predation, sows might adapt movement in the presence of human observers. [43].

## **2.2 Precision Livestock Farming**

Precision livestock farming (PLF) runs on the ideology that "[A]n animal enjoying good health and welfare might provide the best guarantee of product quality in the long term" [6]. PLF promotes the use of increasing implementation of technological advances originally developed for video gaming (PlayStation, Xbox) to progress livestock production so that it is both more efficient and more focused on the welfare of the animals. Such advances are necessary to ensure that innovations can emerge from applications using cameras, microphones and sensors to enhance the farmers' eyes, ears and nose in everyday farming. This technology for remote monitoring of livestock, termed precision livestock farming, is the ability to automatically track individual livestock in real time [5]. PLF technology employing computer vision, implies automated remote detection and monitoring of identified individuals for animal health and welfare using real-time image analysis for livestock tracking, weight and body condition estimation, and functional metrics such as locomotion [6, 26, 11]. The data derived from these analyses is useful in developing a model that can offer a range of real-time management tools to improve health, welfare, production yields, and environmental impact.

## **2.3 Computer Vision For Sow Health Evaluation**

Computer vision is the interdisciplinary field of how computers gain a high level of understanding using images or videos. This focus is primarily utilized for tasks like tracking, motion and pose estimation, and object detection. How computer vision is useful in sow health evaluation is by providing a method of analysis for different aspects of the discussed welfare challenges covered in section 2.1. This section covers various techniques, either being in research or being used in the swine industry to improve sow welfare.

### **2.3.1 Lameness Detection**

To identify lameness and observe various characteristics that cause it, the assessment of kinematic motion, such as walking and running patterns, are studied. Research has validated the use of depth imaging for motion analysis by comparing the motion capabilities to traditional high-resolution infrared marker motion capture systems [38]. Also, analysis has been conducted on pairwise series of depth images to evaluate motion variation using principal component analysis to classify or score animals according to patterns that would indicate lameness [22]. These studies have advanced technology towards more concrete classifications of lameness, but it still requires more development to construct a robust system for tracking and motion variations.

### **2.3.2 Body Condition Analysis**

Methods for analyzing body condition revolve around 3D imaging as it best encapsulates shape information of a physical object. Weight assessment on sows have been primarily done with scales. However, research work using the Microsoft Kinect™ cameras, a 3D reconstruction of the sow's body is used to volumetrically estimate weight using a non-linear regression model [29]. This operation can perform as a non-contact alternative as sows can be non-cooperative when trying to be placed on a scale. Other methods of analysis consists of target key-points along the sow's body using a pair of depth cameras to observe posture [42]. This method constructs a pointcloud

representation and extracts limb positions through planar fitting and clustering in order to obtain body measurements. The current research focuses on specific body measuring techniques, but a holistic assessment has yet to be conducted for classification or scoring.

### **2.3.3 Observing Sow Group Behavior**

Other tasks such as sow segmentation is useful to isolate an individual sow. Research using YOLO [33], a specific type of fully-convolutional neural network (CNN), generates bounding boxes to find regions where there are pigs within an image. In cases where pigs are touching each other or overlapping, the network can have some difficulties. By using some correction algorithms, researchers have been able to distinguish individual animals in a cluster [17]. Furthermore, research on individual posture detection has been successful using CNNs [31]. These networks are useful for identifying behavioral states along with tracking for monitoring activities sow activity: length of time standing or lying down. The use of more advanced computer vision techniques, like artificial neural networks, have become more popular as they have shown to be capable of performing high-level monitoring tasks.

## **2.4 Artificial Neural Networks**

Artificial neural networks are a highly interconnected network of many simple functions called neurons. These neurons are parallelized and grouped, called a layer, to perform a specific operation. Within the network, many layers are interconnected together in a specific way to form an overall architecture.

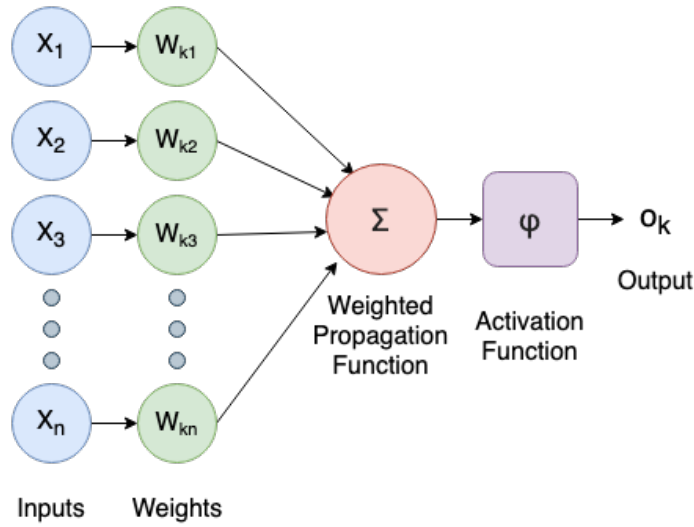
These networks try to model a high-level task by learning a relationship between the input-output mapping through an iterative process of training and validation. Training is performed by computing gradients across the predicted and the ground-truth, or known output, and adjusting the neuron weights that help converge the network using a loss function as an optimization method.



The next sections will discuss neural network fundamentals and the necessary layers utilized in the project. There is a large variety of network architectures, but this paper will primarily focus on convolutional neural networks (CNN), as these network architectures are designed for image and video input.

### 2.4.1 Neuron Structure and Network Architecture

The neuron performs the primary computational operation based on the input of one layer and outputs the value to the next layer. Figure 2.3 shows a basic outline of the neuron architecture:



**Figure 2.3:** Neuron architecture

The inputs,  $x_1-x_n$ , are the outputs of a previous layer with  $n$  neurons. The weights,  $w_{k1}-w_{kn}$ , is a tensor that multiplies the input with its respective weight before performing a predetermined propagation function [15]. The value of this weight determines the amount of effect that the previous neuron has on the current. Generally, positive values are excitatory connections, and negative values are inhibitory connections when referred to the biological neuron [35]. The propagation function, denoted by  $\Sigma$ , is a fixed function used to perform a specific operation; specific functions used in this research will be discussed. The activation function,  $\phi$ , generally a non-linear function that determines the overall output of the neuron. The activation is important as all subsequent parts of

the neuron are linear operations; this function enables the network to learn non-linear properties between the source and desired output. The output of the activation function is passed along to all neurons in subsequent layers. Equations 2.1 and 2.2 show the mathematical operations:

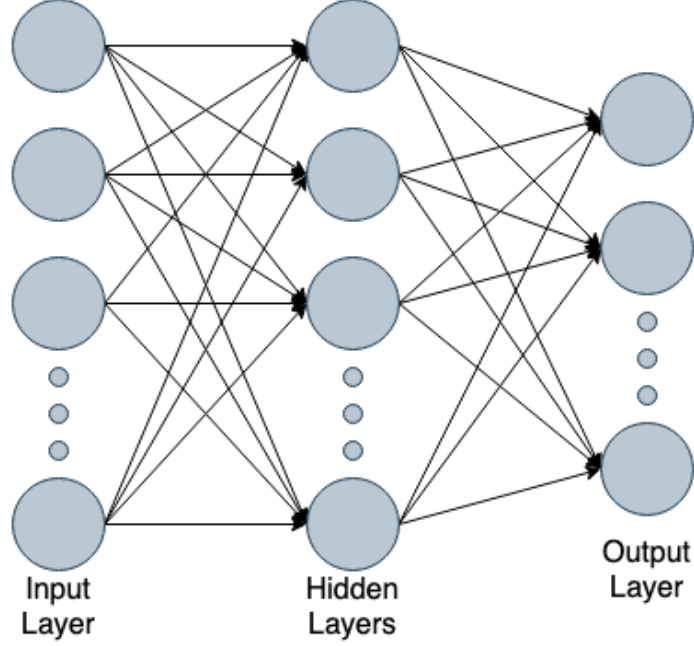
$$prop_{out} = \sum_{i=1}^n (W_{ki}x_i) \quad (2.1)$$

$$o_k = \phi(prop_{out}) \quad (2.2)$$

where the propagation function,  $\Sigma$ , is the simple summation and  $\phi$  is an arbitrary activation function.

The combination of multiple neurons in parallel is called a layer. The combination of sequential layers form an overall neural network architecture. Figure 2.4 shows a detailed view of how layers are interconnected and the propagation flow between input and output. Each circle represents a single neuron with the outputs of multiple neurons, becoming the inputs of subsequent neurons in the next layer.

Along with the network architecture, a loss function and optimizer are requirements to dictate how the network learns to perform a particular operation. The loss function helps determine what the network should observe to produce the same result as the given output effectively. This function is unique to the type of application of the designed network. The optimizer is a pipeline of functions that alter network weights in order to converge the network output towards the ground truth. Generally, optimizers are a variant of gradient descent, which computes the gradient value of the output and propagates backward towards the input to target which weights contribute to the desired output. Enhancements such as momentum and modified learning rates further decrease the learning time and increase the accuracy of the output.



**Figure 2.4:** A simple neural network architecture.

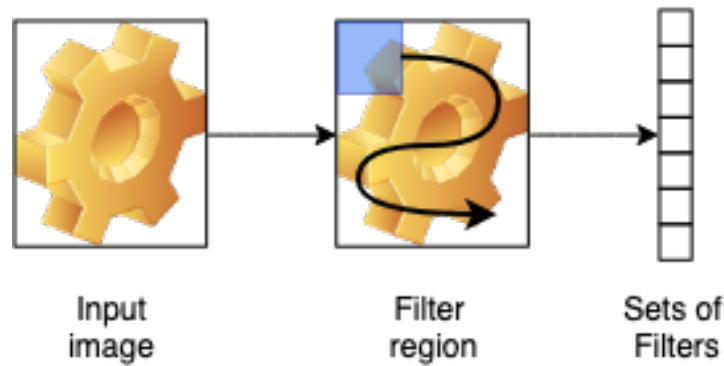
### 2.4.2 2D Convolution Layers

Convolutions are the essential layers when it comes to image-based neural networks. A subclass of neural networks called convolutional neural networks (CNN's) are based on the convolution layers to find patterns and characteristics of an image. The convolution operation, from a top-down perspective, is a filter (kernel) performed on an image. Equation 2.3 is the mathematical operation performed in a linear algebra representation:

$$conv_{out} = (I \otimes K)_{x,y} = \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} (I_{i,j} K_{x-i,y-j}) \quad (2.3)$$

where  $I$  is the image represented as a matrix and  $K$  is the kernel of operation. Effectively the convolution operation outputs how the shape of the kernel matrix modifies the image. For better understanding, Figure 2.5 shows a top-down overview of how the convolution operates over an image. The kernel, represented in blue, is slid across the image performing element-wise multiplications to produce a filtered image. The kernel values, represented as the neuron weights, are shared across the image. The number of channels that compose the image determines the number

of kernels. Thereby creating a set of filters used to extract specific patterns and features within the image. During network training, these kernel values are altered to produce specially designed set filters to generate the given output.

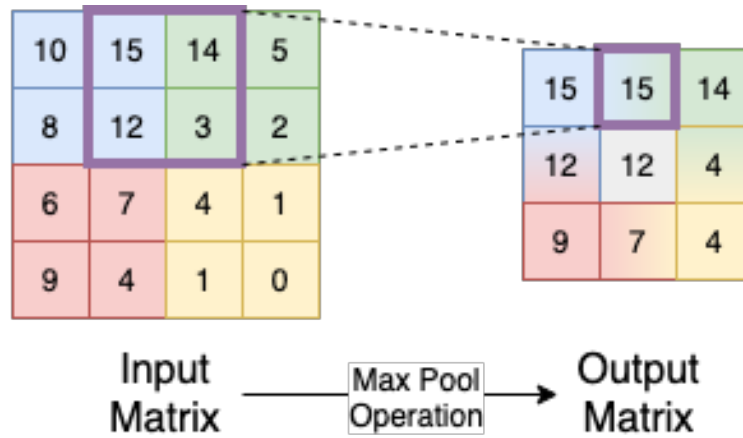


**Figure 2.5:** Top-down view of a convolution operation. The kernel is represented as the blue square and the convolution operation performs a region-wise dot product across the image with the neural network able to produce a set of filters to match an output.

### 2.4.3 Pooling Layers and Receptive Field

The pooling operation acts like the convolution operator, which has its kernel. This kernel notes how the output is calculated based on the type of pooling: min, max, average, etc. For max-pooling, the maximum value is chosen within the kernel, and the out is assigned that value. Figure 2.6 shows how the max-pool operation is performed on a 4x4 matrix. The purpose of this layer is to reduce the number of dimensions of a given input. In the case of images, the number of dimensions is given by the resolution size and channels, thereby having an exponential growth in dimensions as resolution size increases. Pooling used to reduce the computational complexity, effectively 'down-sampling' the image relative to the kernel size.

The combination of convolutions and pooling layers help construct a structure where a filter can observe larger regions of an image without the need to increase the kernel size. The main idea is



**Figure 2.6:** Max-pooling operation with a kernel size of 2. Max-pool denotes that the maximum value in the region is the output. Each color denotes the region from the input matrix used to compute the output.

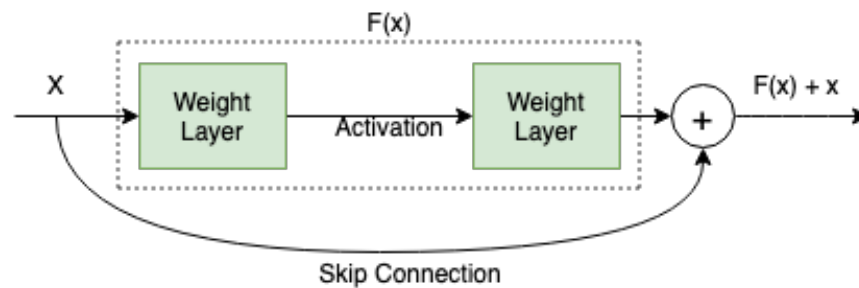
that increasing the kernel size will result in more memory usage to perform larger dot-products. Thereby exponentially increasing the number of element-wise multiplications. So instead of making the kernel larger, making the image smaller leads to a decreased number of multiplication operations and maintains the memory usage. So if the image is changing, how large of an area that the kernel operating on? To answer this question, let us observe the number of connections between each neuron. For example, if we observe a single neuron that as previously had a 2x2 max-pool operation performed, it has four connections; since there were four neurons in a 2x2 kernel. This concept can extend to multiple layers and various operations.

The receptive field is a hyperparameter in which denotes the number of connections. It is useful because it also is equivalent to the size of the kernel operating region. Therefore, a kernel with a receptive field similar to the input resolution is effectively operating on the whole image. This relation is critical when it comes to images as different features are useful in different receptive fields. For example, at lower fields, the kernel is observing detailed texture while at large receptive fields, it observes object shape and positional relationships.

#### 2.4.4 Skip Connections and Residual Blocks

When dealing with images, the use of convolutions as a filter method is excellent for extracting features. However, when concatenating multiple filters together, fine details can be lost due to the previous layers of filters. The utilization of skip connections between layers can mend this issue. The primary purpose of the skip connection is to retain the original information before filtering and reincorporate it back after filtering. In a mathematical sense, this is a simple addition after a function operation. Applications of skip connections are critical in maintaining previously known information and allowing for shortcuts between layers to enhance network learning.

Residual blocks are a set of network layers designed around skip connections—Figure 2.7 shows the architecture for a residual block. From the figure,  $F(x)$  is a sequence of arbitrarily defined layers. Residual blocks operate by having the network find the difference between the input and the output of the block. The result is the content in between, referred to as the residual. It helps the network to skip learning on intermediate layers in which do not contribute to the desired output. Residual blocks are a fundamental building block for larger CNN architectures such as ResNet [39].



**Figure 2.7:** Residual block architecture

## 2.5 Summary

This chapter provided a review of sow welfare challenges along with precision livestock farming technologies aiding to solve these issues, namely: lameness, body condition assessment, and group

interactions between sows. An overview of artificial neural networks was presented and discussed. In-depth structures that make up neural networks were explained with mathematical operations and top-down perspectives to demonstrate the overall impact these structures have. Specifics for convolutional neural networks were explained as these are critical concepts used in the research project and will be utilized in upcoming sections discussing network architecture construction.

## CHAPTER 3

### SOW DATA COLLECTION

#### 3.1 Introduction

A dataset is required to train a network to perform individualized health monitoring. This dataset is preferred to contain information sources about characteristics of the sow's health along with a health record of each sow with indications of lameness or other measures of reproductive health. These metrics can be used to develop sow health detection and prediction devices.

Currently, we are not aware of any open-source datasets that can be used to estimate sow pose. Finding an automated way to collect sow data allows for the creation of a dataset used to train a network to perform health monitoring. Therefore, I have created a custom hardware collection device to obtain data, known as the *SIMKit*. The device was constructed such that it can withstand the harsh farm environment. With many factors that need to be accounted for, the farm environment poses significant challenges when creating robust technological hardware. The next section discusses several types of environmental hazards.

#### 3.2 Farm Environmental Hazards

Many environmental hazards around the farm can inhibit the proper operation of hardware. Farms are prone to the buildup of particulate (dust, dirt, skin cells and other animal by-products) in the air and on surfaces. Without regular cleaning maintenance, a build up of these particulates on non-porous material surfaces can lead to device failure; sources include obstruction of camera lens or clogged cooling fans leading to device overheating. Surfaces can have a build up of grime on non-porous materials, causing sensor camera lenses to be obstructed and cooling fans to be clogged. Additionally, depending on the location and exposure of the device, it can be susceptible to physical damage or obscured by the animal's position. Devices close to the ground would require substantial



protection from moving forces of animals, people and equipment as well as a higher cleaning and maintenance routine. Aerially mounted devices would have less physical disruption, but need to be able to operate at extensive ranges as well as incorporate a method to remove pests such as spiders, flies, and rodents from landing, obstructing or destroying sensors.

Chemical hazards are also a significant concern for cathodic corrosion. Production of swine results in corrosive biproducts such as feces and urine. A corrosive gas compound produced in pig urine is ammonia ( $NH_3$ ). Copper used in wiring within circuit boards, is highly reactive in the presence of ammonia, causing deterioration of connections. For example, electrical shorts occur when the buildup of hexamine-copper(II)  $[Cu(NH_3)_6]^{2+}$  ions form, along with impurities, on the surface of circuit board leads. Also, sows are provided ad libitum access to water and routine cleaning in the barns makes the environment have a high moisture content. Moisture accelerates a corrosion process and will shorten the lifespan of materials if not properly protected.

Thermal fluctuations contribute to device operations since barns are not insulated from outdoor temperature variations. This leads to extreme ranges in temperatures, possibly causing hardware malfunction if it exceeds rated operating conditions for sensors and processors. Mitigation techniques involve the usage of cooling fans, heat sinks, or sealed enclosures. These help regulate the device to minimize the large spikes in temperature change.

### **3.2.1 Biosecurity and Connectivity**

A considerable focus to maintain food animal health includes biosecurity standards which minimize the introduction of disease via people and fomites (inanimate objects) from outside the farm. As a result, all hardware that enters a farm environment must be disinfected and then remain on-site to meet compliance of the farm's biosecurity plan and protocol. These biosecurity protocols, while good for animal health, present problems as the prototypes cannot be brought back to the lab for adjustments. The prototypes must be near completion, from a hardware perspective, before on-farm testing can occur. Biosecurity measures also imply that any transfer of data using hardware such

as external drives, must be kept at a minimum as the transportation of storage devices for collected can also be a carrier for disease spread.

Due to these restrictions placed on prototype and material entry into farms, some consideration of utilizing the internet for data transfer were discussed, but the availability of strong connections may be limited. The primary reason is that many farms are sited in remote locations with minimal access to common networks such as cellular and satellite connectivity, as well as internet access. Locations with these amenities, the type of data collected is a concern as devices, like cameras, can generate upwards of several gigabytes of information per second. Therefore, large storage devices like hard drives and flash drives are a reasonable medium for data transfer. Another key factor that should be accounted for is cyber-security. Having proper encryption methods and hardware protection measures are both vital to protect private information and in maintaining good business practices between the device developer and farm operators.

### **3.3 SIMkit Collection Device**

The *SIMkit* (Sows In Motion kit) collection device is the custom hardware platform used to operate, process, and store data from the selected sensors. The first section outlines each component's requirement specifications, design choices and technical construction for various parts. Second, is an overview of the logistics of data transfer between the farm site and research lab. And finally, the details of the data preprocessing procedure used to format the data for the dataset.

#### **3.3.1 Design Requirements**

The design requirements for the *SIMKit* were determined by the hazards initially identified in Section 3.2. The creation of a preliminary hazard analysis (PHA), seen on Table 3.1, establishes a baseline for what the device needs to incorporate. The hardware components were selected using the PHA as a guideline. Sensors and processing units were determined based on the form factor

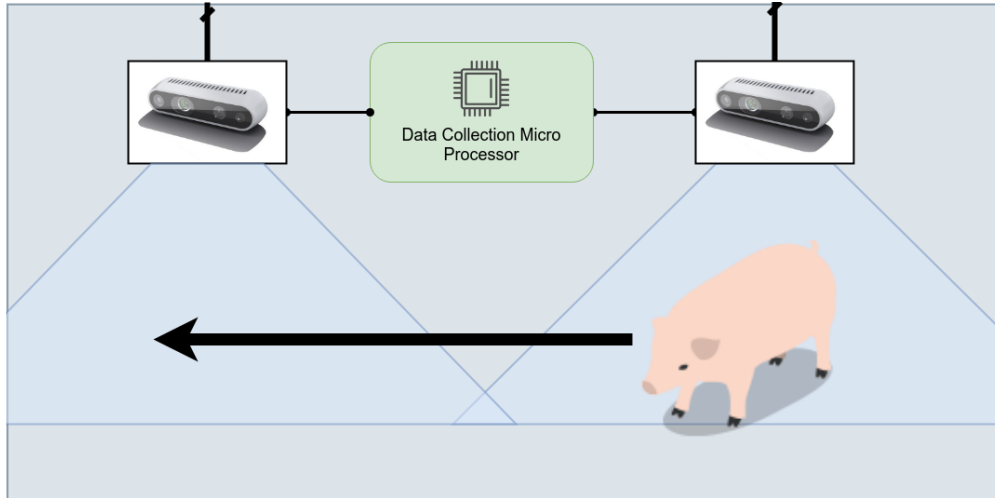
and capabilities used to record the data.

**Table 3.1:** SIMKit preliminary hazard analysis

Device	Hazard	Detection	Corrective Action / Mitigation
Processor and Interfaces	particulate	N/A	PLA/PETG enclosure
	corrosion	N/A	PLA/PETG enclosure
	moisture	N/A	PLA/PETG enclosure
	overheating	onboard temp-sensor	heatsink + cooling fans
Sensors	particulate	observation / sensor blockage	routine maintenance
	corrosion	N/A	onboard conformal coating
	moisture	N/A	cooling fans
	overheating	onboard temp-sensor	heatsink case + cooling fans
	pests	N/A	cooling fans
Wiring	particulate	N/A	PVC encasement
	corrosion	N/A	PVC encasement
	moisture	N/A	PVC encasement
	pests	observation	PVC encasement

### 3.3.2 Concept Design

The conceptual design of the *SIMkit* was envisioned to non-invasively measure sows in motion without disturbing normal farm operations. Specifically, to record data as sows moved between facilities (farrowing barn to breeding barn) during normal production transition. The objective is to identify functional morphological features associated with lameness, and observe body condition from a dorsal (top-down) view. Therefore, depth imaging cameras were selected since it provides a 3D map of the sow within the environment and operates at high enough frequencies to record motion data. Figure 3.1 provides a visual concept. Depth imaging cameras were placed such that sows would travel through the hallway, without inhibiting normal sow movement, and the camera's field of view during barn loading/unloading sessions.

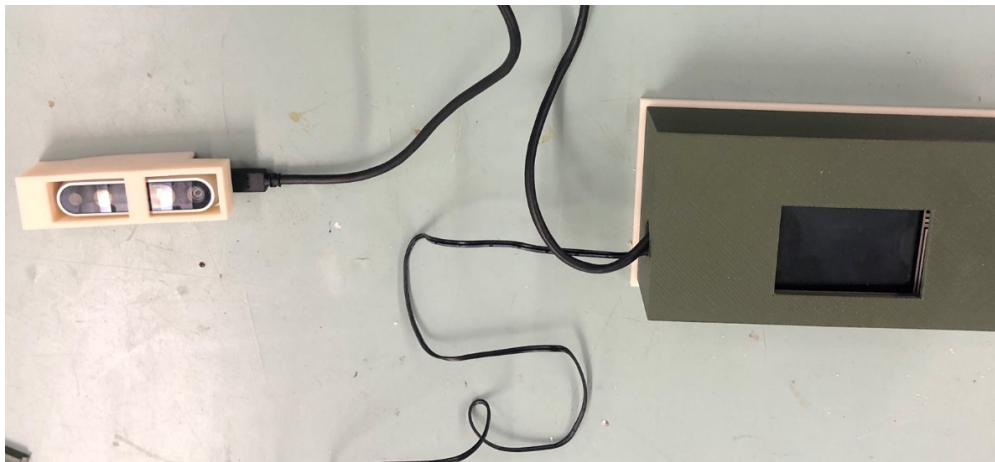


**Figure 3.1:** A simplified view of how the *SIMkit* device operated in the farm setting.

### 3.3.3 Prototype Designs

With the design specifications and concept planned, several prototype design iterations were developed using easily available materials. After many designs, the *SIMkit* hardware design became a balance between operational feasibility and manufacturing complexity.

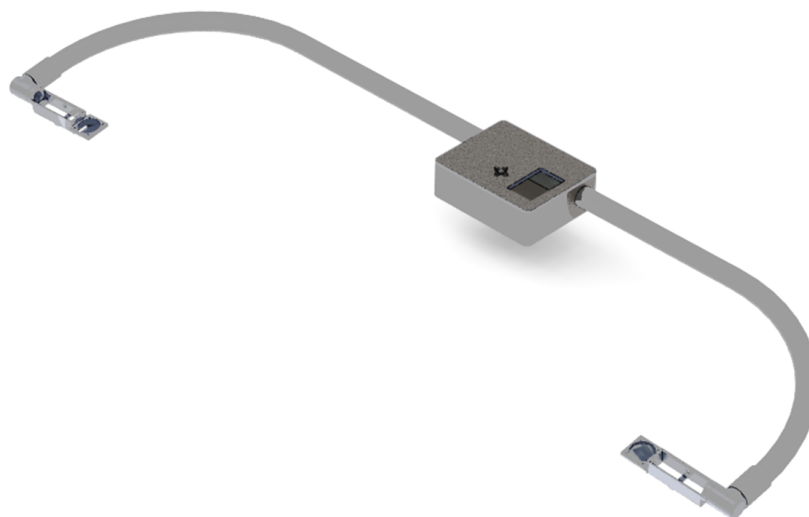
For the first prototype, the Raspberry Pi 3b+ was the only available lightweight computer with easy development capabilities. However, many of the Raspberry Pi based prototype devices, as seen in Figure 3.2, ended up being re-designed due to bulk and lack of convenient functionality once tested within the commercial farms.



**Figure 3.2:** *SIMkit* prototype using a Raspberry Pi as the base processor along with an Intel Realsense camera.

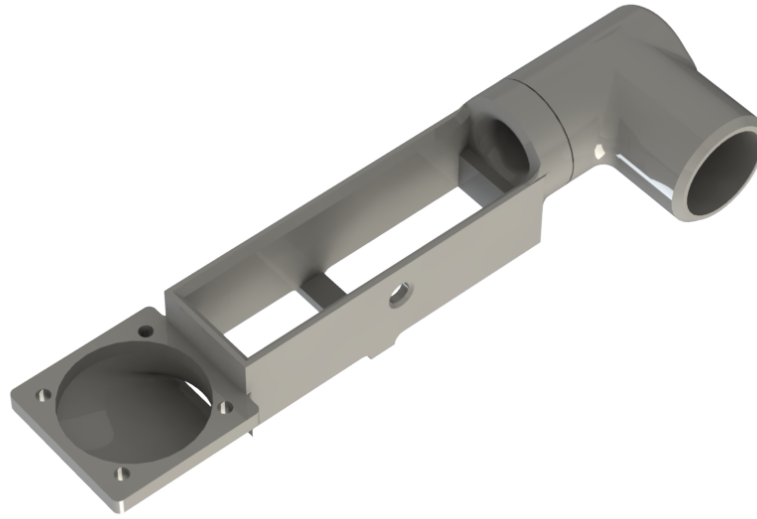
Another challenge to these devices was setup times, often taking upwards of an hour and causing delays to normal farm operations.

The next iterations, using the Nvidia Jetson board, were modeled in CAD (computer-aided design), and simulations were ran to test durability and actuation. Figure 3.3 shows the final design implementation. This design features: adjustable 2-axis rotation camera mounts for calibration (Figure 3.4), 3.5 inch LCD with tactile buttons for an interactive user interface, and fully enclosed system with considerations of the hazards listed in Table 3.1. Individual CAD drawings are included in Appendix A.1 and full build instructions in Appendix A.2.



**Figure 3.3:** CAD rendering of the final Simkit design. Features include: tactile buttons for interfacing with the monitor, 2-axis rotation on camera mounts for re-positioning, and easy mounting with compliant conduit PVC piping.

The modeled parts were prepared and 3D printed using PLA and PETG plastics. The advantage to these materials include a lightweight and durable structure to the enclosure while being resistant to ammonia and moisture leakage. Extra materials, like PVC piping and fasteners, were used to join the 3D parts together. Construction does involve some work with wiring up the interface buttons



**Figure 3.4:** Detailed view of the camera holders. There are 2 pivot axes designed for roll and pitch movement. There is a location for a fan and the design allows for enclosed wiring to the processor.

and fans, but the overall build is simple, with the need of just screwing down components to the box and wire management to keep everything enclosed.

### 3.3.4 Sensor Selection and Placement

Multiple imaging sensors were tested and evaluated based on the following key criteria: ease of development, scalability, and accuracy. In the initial and current project development, both the Microsoft Kinect™ v1 and Intel® Realsense™ D series cameras were the popular depth imaging cameras for research due to their ease of development integration and customer support.

Disadvantages of the Kinect™ include the need for a main power connection and thereby subject to normal ventilation and equipment drawn power surges. In addition, the generated depth map was not as easily obtained through complications with Matlab® and storing raw image formats with near real-time performance. Alternatively Intel® Realsense™ D435 series camera runs off of USB2.0 or USB3.0 interfaces with a dedicated USB, bus at up to 90fps at distinct resolutions and

is compatible with Python3+ and C++11+ API's for software development

With a depth camera, sensor position placement relative to the sow is important as to decide how to best capture motion and body composition. From previous works, observing the back and sides provide the best views for obtaining context clues for muscle and fat deposits and limb movement either through observing the leg motion directly or indirectly through back muscle displacement [29, 38]. When looking for sources of motion abnormality, a good location is a top-down view of the sow. This view would contain information about limb position via muscle movement along the back. It will show the relative trajectory, Figure 3.7 shows a depth image that contains lots of information on the sow's back. Tracking the top-down motion will reveal lameness by observing a predicted trajectory with a normal sow compared to a lame one. Also, lame sows demonstrate a distinctive periodic lateral movement pattern as to avoid applying pressure to injuries. Therefore, the depth cameras will be mounted on the ceiling of the barns. The cameras are placed within 2–3 meters from the ground, to minimize measurement error, as seen in Figure 3.5.

### **3.3.5 Computing Platform and Data Acquisition**

For a computing platform, a full-sized computer is not viable due to biosecurity and complications with setup and operation. As a data collection device, the processing needed should contain video encoding and image processing hardware with relatively high clock speeds. Here are the minimum requirements that were decided when choosing a computing unit:

- 1Ghz quad core processor
- 2GB of RAM
- Digital Signal Processing (DSP) chip for video or image processing
- USB 2.0 or 3.0 interfaces for camera compatibility



**Figure 3.5:** Simkit installed on the ceiling within a farm hallway. Camera field of view is indicated by the blue overlaid region in the image

- Small form factor so it can easily be mounted
- I/O pins for buttons / LED indicators for user interfaces

The *Simkit* prototypes has two different versions with different computers: the Raspberry Pi 3 Model B+ and later, the NVIDIA Jetson™ Nano Developer Kit. The Raspberry Pi 3b+ is a 1.4Ghz quad-core single-board computer that was used along with the Intel® Movidus™ Neural Compute Stick as a co-processor for neural network acceleration. The *Simkit* firmware was created on the Linux kernel with integration with the Debian operating system with the user interface application developed in Python3.6+.

The Raspberry Pi based prototype setup consisted of a 3.5-inch touch LCD for a user interface



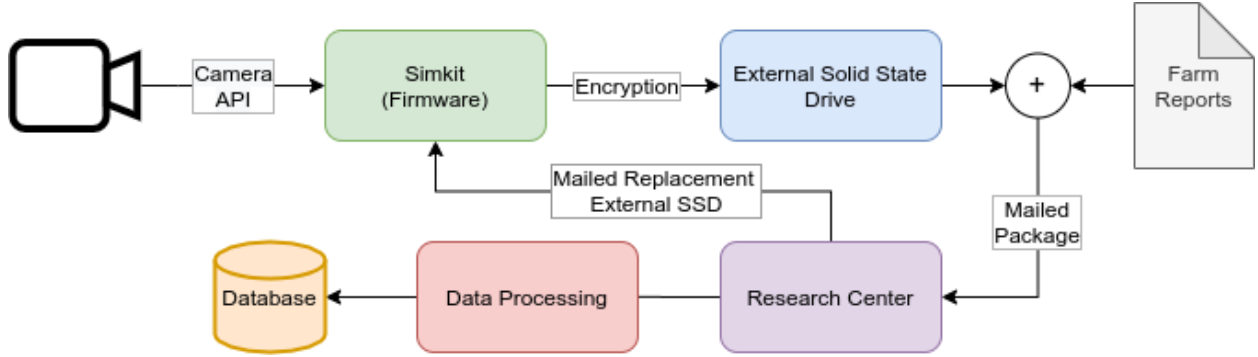
and a single depth camera powered by an external 5v/2.5A power transformer. Both infrared and depth images at roughly 8fps with dimensions of roughly 15x20x5cm. The overall performance was lackluster, and further analysis of the system reveals that the need for USB3.0 is a necessity as the camera was being underpowered from the 5v USB power lines on the circuit board. There was not enough bandwidth to support a high frame rate with multiple streams being saved.

The NVIDIA Jetson™ Nano System on Module (SOM) setup consisted of a dual Realsense™ camera arrangement spaced at 1 meter apart and powered through the USB3.0 interfaces and an external SSD for raw image storage. The same 3.5 inch LCD was used as an interface, but using a separate circuit to have tactile buttons as farmers preferred this method over a touch screen. An uninterruptible power supply was used with a 5v/3.5A power transformer, as this board required more power for more processing units. The overall performance is at 15-20fps for both infrared and depth images for both cameras. There are some bandwidth issues with the USB3.0 bus being multiplexed between 2 cameras, which may cause one not to operate occasionally, but a single camera has not shown any issues. A single camera variation is the current setup being utilized for the *Simkit*, and all data that is shown is from recorded data using this device.

### **3.3.6 Data Transportation Pipeline**

As previously mentioned, farm environments will most likely not have a high-speed internet connection sufficient for large file transfer. Keeping track of the data transfer from the sensor to storing it in a remote database location is essential. Figure 3.6 outlines the information transfer pipeline.

The *Simkit* performs all of the hardware to software transduction processes, data encryption, and stores the final files to the connected external solid-state drive. The farmers compile the sow production records, in paper format, such as sow identification, number of pigs weaned and caliper scores for the days recorded and ships the hard drive and paper reports to the research lab for validation and analysis. A replacement hard drive is sent back with specific hardware IDs



**Figure 3.6:** A block diagram outlining how data is obtained, processed, and transferred to a remote database.

and file structures for the *Simkit* to accept as functional. Drive replacement is a simple USB-C connection performed when the device is turned off. The next section outlines the specific details that encompass the data processing needed before storing the files in the database.

### 3.4 Data Preprocessing

Data preprocessing is a series of procedures to prepare the data for later usage. It is useful to process and filter out unnecessary information during data collection, such images that do not contain sows ("no-sow"). This section discusses the various algorithms used to filter and refine the raw data recorded at farms in preparation for neural network training. The raw data format consists of a folder for each camera used in that specific recording with sub-folders indicating which sensor each image is coming from. The images are stored in a .png format using the OpenCV library in Python3.6. Infrared images are read using the "cv2.imread()" function utilizing the "IMREAD\_GRAYSCALE" flag. For the depth images, the same function is called, but using the "IMREAD\_ANYDEPTH" flag. A generated text file is created to indicate the scale factor applied to the depth image values to obtain a real-world distance measurement in meters.

#### 3.4.1 Motion Detection Filtering

A motion detection algorithm is used to filter the raw data of the image frame where there is no sow within the frame. Motion is detected with the following algorithm (Alg 1):

---

**Algorithm 1** Coarse Motion Detection

---

**Input:**  $\{I_0, \dots, I_i, \dots, I_n\}$  as image sequence  
threshold = minimum number of pixels to constitute an object  
image\_list = a sequence of image numbers  
**for** every image  $I_i$  in the sequence **do**  
    background  $\leftarrow$  find background pixels on  $I_i$  based on  $I_{i-1}$  to  $I_{i-200}$   
    foreground  $\leftarrow I_i$  - background  
    **if** sum(foreground) > threshold **then**  
        append image number to image\_list  
    **end if**  
**end for**  
**Return:** image\_list

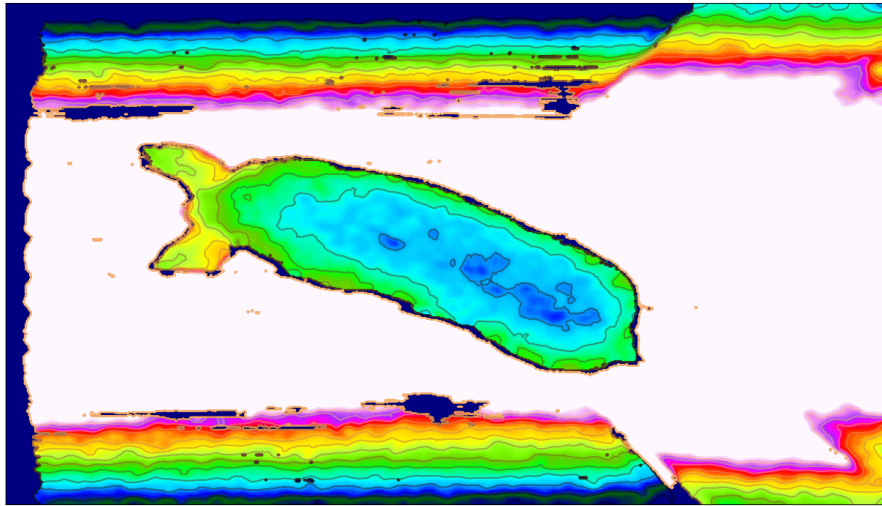
---

The background detection algorithm being used is the Mixture of Gaussians (MOG2). It operates by creating a variable  $k$  number of gaussian distribution for every pixel to model the color value. A history of  $n$  images (defaulted to be 200) is used with a weight distribution being proportional to the amount of time a color has remained in that pixel. Meaning that the produced output image will have high weights for anything classified as background and low weights for foreground [45, 46]. The threshold value for an object is tuned such that at least 4% of the total pixels are foreground. It is equivalent to having enough visibility for a sow's head to be within the frame. The final output of the course motion detection algorithm is a list of all images that have contained any motion. This list is saved off into a CSV file for a quick filter to find instances of pigs.

### 3.4.2 Depth Image Preprocessing

Depth images from the Intel® Realsense™ cameras are encoded as a 16bit value of depth and scaled to minimize rounding errors. These images are packaged within the .png format, which allows for easy extraction of 16bit values, but will not be visible using a standard photo viewer as they use 8bit encoding. The images are scaled to obtain real-world depth values, using the correction factor extracted from the camera used to get a distance in meters away. Then, each pixel is clipped to a max distance of 2.5 meters is the distance from the camera to the barn floor. This value mapped to a normalized floating-point value from 0.0 to 1.0 for all images. This preparation is essential for

neural networks as inputs should be consistent with a strict range of values. Figure 3.7 is a visualization of the normalized depth image and displayed using a colormap to help differentiate distance.



**Figure 3.7:** Processed depth image with a color map and contour plot overlay.

### 3.4.3 Health Record Processing

The sow records of individual sows are collected in paper format and transferred to the farm office for entry into an electronic sow performance database. The records track the sows history and include health information such as vaccination records, number of births (parity) farrowing date; number of piglets born alive, still, and mummified; piglets weaned; lactation length; wean to estrus interval; mortality; and reasons for cull. This information is transcribed into a database format and can be exported and interpreted as numbers for a neural network ground truth value.

## 3.5 Summary

The *SIMKit* data collection device is a specially designed piece of farm technology that is robust to harsh farm environments. This technology is used to obtain a top-down infrared and depth image view of sows walking to extract motion and topology metrics within a time-dependent structure.

It operates as a stand-alone unit with data being safely and efficiently transferred to the research center for analysis. The video data is then filtered and prepared for subsequent neural network training and stored as a big data.

## CHAPTER 4

### SEMI-AUTOMATED LABELING

#### 4.1 Overview

This chapter proposes *transfer labeling*, a semi-automated procedure for annotating key points in video data. This procedure detects features in one modality and transferring them, along with their labels, to another modality. Our application uses marked locations of sow joints only detectable in the infrared image to create labels for depth images. This design is divided into two parts: mark detection and joint association. Sow markers are detected on an image with a simple CNN detector. A human labeler then assigns a joint label to the detected marks on that image. Finally, optical flow is then used to propagate the joint labels to the remaining images in the video sequence. Analysis of this procedure is quantitatively evaluated and compared to the conventional human labeling method.

#### 4.2 Introduction

Labeled datasets are an integral part of supervised learning, but creating one is a tedious and laborious task. Often these datasets are large and contain annotations for every input. The conventional method for annotations requires people to hand label each set of inputs with a corresponding output. This labor-intensive process is very time consuming, which grows the larger dataset is. Alternatives, such as motion capture devices, used to mark locations, affects the data collected due to physical indicators that are attached to the subject. Therefore, the images collected have visual indicators, which is not representative of the actual input.

*Transfer learning* overcomes this problem by capturing motion with marks that are visible in one modality while being invisible in a second modality. Since marks are detectable in the first modality, we use infrared, this setup enables automated labeling. These labels can be transferred to the

second modality, in our case of depth, while maintaining the integrity of the data. We seek to use the unmodified depth images with annotations to train a network.

### 4.3 Annotation Framework

Annotations are represented in many formats; for pose estimation, a list of coordinates encodes a sow's joint locations. A coordinate is assigned to be the central location of a joint with an accompanying label to determine the classification: *head*, *neck*, *left-shoulder*, *right-shoulder*, *last-rib*, *left-hip*, *right-hip*, and *tail*. These values are stored within a dictionary entry that contains: *the reference input image number*, *date of recording* and *list of joint locations*. These values are the essential information required for the neural network to begin training. *Transfer labeling* is used to obtain joint locations efficiently by automating portions of the labeling process.

### 4.4 Transfer Labeling

Precise manual labeling of keypoints in-depth images is challenging due to a lack of local texture cues. Instead of labeling depth images directly, labels can be generated in a different modality and then applied. Depth images are calculated based on two infrared cameras with known intrinsics and extrinsics to calculate depth. By placing label markings only detectable in infrared images, these markings are extracted as labels for depth images. This method provides a way to determine accurate label locations without modifying depth image data so they can be used as training data.

*Transfer labeling* can apply for depth and infrared image pairs since they represent two distinct types of information: distance, and infrared reflectivity. This configuration provides an added convenience of pixel-wise alignment. This benefit is due to how depth images are generated from image disparity from stereo infrared imaging cameras. This relationship makes transferring labels directly from one image type to another simple and accurate.

The distinction of different marks on sows to their joint labels is a challenging task for computers. Humans can perform this task with ease with very little training. Therefore, a human labeler assigns these label IDs to the sow markings for a single image within a sequence. The label IDs for the remaining images is accomplished through mark tracking. Given an initial set of labels, they can be tracked forward and backward in time. The tracking of pixels between sequential frames is called optical flow, and this technique is used to associate a set of joint labels throughout a whole sequence of frames. The time required to annotate a sequence goes from a whole sequence (roughly 200 images) to one image, thus drastically decreasing the amount of time and effort required to generate annotations.

## **4.5 Preliminary Preparation**

Before data collection recording, each sow was palpated and marked on 8 locations: top of the skull (head), articulation between thoracic and cervical spine (neck), top of the scapula (left-shoulder/right-shoulder), spinal position of the last-rib (last rib), hook bones (left-hip/right-hip), and tail head (tail). These locations were selected because they provide consistent body landmarks for the pose. The head mark is placed between the ears. By palpating the sow's spine, the neck mark is located on the first thoracic vertebrae. The shoulder marks indicate the top of each shoulder blade. The last rib location is determined by palpation of the sides of the sow and traced up to the back spinal location. Hip mark locations are placed at the top of the hook bone. Lastly, the tail mark is placed at the tail head.

A circle using a dark-colored solid paint crayon stick is used to place the marks on sows. These crayons, commonly referred to as livestock markers, are used in the farm environment and are food safe. The application of paint makes a distinctly identifiable mark on the sow's skin as it has different infrared material properties. Figure 4.1 shows the placement of the joint markers on an



infrared image.



**Figure 4.1:** These marks are paint crayon markings on the sows back, used to obtain image labels.

## 4.6 Mark Detection Neural Network

The joint markers on pigs need to be extracted from the infrared images to obtain annotation label locations. The marks need to be discriminated from the background and separated from extraneous markings on the sow from farm operations. A simple convolutional neural network (CNN) can easily extract well-defined features with very high accuracy. Therefore, a CNN was constructed to detect these joint markers. The CNN extracts the mark features and constructs a confidence map denoting the central location of a drawn mark on the sow. This section discusses the CNN construction and evaluation using the neural network concepts described in section 2.4.

### 4.6.1 Mark Labeling

A hand-labeled dataset is used to train the mark detection network. Hand labeling is performed through a custom-built interactive program I created. It displays images in a window, and the human labeler can click on the joint pixels that should be detected. These pixel locations constitute

the ground truth values for training and evaluating the neural network. The human labeler can also indicate ignore regions, not to include markings on sows originally placed by stockpersons during farm operations. The ignore regions are used during training to neither learn to detect nor ignore extraneous markings not related to the joints. The hand-labeled dataset consists of roughly 157 images with sows in various locations and some with multiple sows within the frame. These images were selected with the following criteria: one image per video sequence of pigs, all joint markings in the image are given an annotation, and locations of pigs are chosen to be varied to avoid learning relative occurrence within the image.

#### **4.6.2 Network Architecture and Implementation**

The neural network is constructed within an environment container to maintain environment run-time consistency and avoid version differencing issues. The program of choice is Nvidia Docker<sup>TM</sup> which is an extension of Docker<sup>TM</sup>. Docker is a container management system that allows for the construction of a virtualized software package that includes all dependencies for applications to run.

I have created this docker image that runs on Ubuntu 18.04LTS as the base operating system with CUDA acceleration. The neural network framework is constructed using Python 3.6 within Tensorflow 2.1 with OpenCV 4.2, Matplotlib, and Scikit-image as support packages. The training server contains an Nvidia GTX 1080Ti graphics card, 64Gbs of RAM, and an Intel core I5 processor with an image of the docker container for training and development.

##### **4.6.2.1 Input and Ground Truth Layers**

With the annotations and images, the data needs preprocessing during the training session since loading all of the images can cause RAM overflow issues. To avoid this, I have constructed a data generator that formats the input and ground truth data from the dataset for the network.

The input layer of the CNN is constructed by taking the input image filename from the annotations and decoding the image and normalizing the values from [0.0-1.0] and storing it into RAM. This image is then rescaled to 256x512px resolution to match the network input. A uniformly randomized set of image operations: scaling by 0.8–1.1 times larger, rotation of [-20,20] degrees, and a vertical mirroring across the vertical centerline of the image are performed with background pixels having a value of 0.0. These specific values are stored and again utilized for the ground truth. This type of augmentation allows for more variation in images during training so that the network is discouraged and we hope generalizes better to sows it has not previously observed.

The ground truth is generated by taking the set of pixel locations from the annotations and constructing a confidence map of size 256x512px resolution. It is constructed using a Gaussian distribution with a mean of 0px and a variance of 36px for each joint. A weight map is also generated, which is used for ignoring regions in the image and equally distributing learning between joints and background pixels. It is created by copying the joint confidence map assigning a value 1.0 where there are joints and a value of 0.0 for regions to ignore. For the rest of the pixels, a value of the number of remaining pixels / the total image resolution is assigned, denoting it as the background. This construction provides an equal amount of weighting between the background pixels and the joints to effectively train the network with sparse ground truth such as this one. Both the joint confidence and weight map are then augmented using the same values as the accompanying input image to get a matching ground truth, then storing it into RAM. The neural network then utilizes these images for training/validation and then deleted to free up resources on the computer for the next set of images.

#### **4.6.2.2 Architecture**

The mark detection CNN must extract joint marking features to determine their locations. The marks on the sows within the infrared image are distinct ellipsoids with high contrast on the sow's skin. The network has to identify characteristics of each mark to learn their locations. Therefore,

the receptive field of the network must be large enough to 'observe' the whole sow in a filter to learn the relative placements on the body. The first half of the CNN, as seen in Figure 4.2, has an hourglass-like structure. This construction of convolutions followed by pooling increases the receptive field of the network, thereby filtering at different image resolutions.

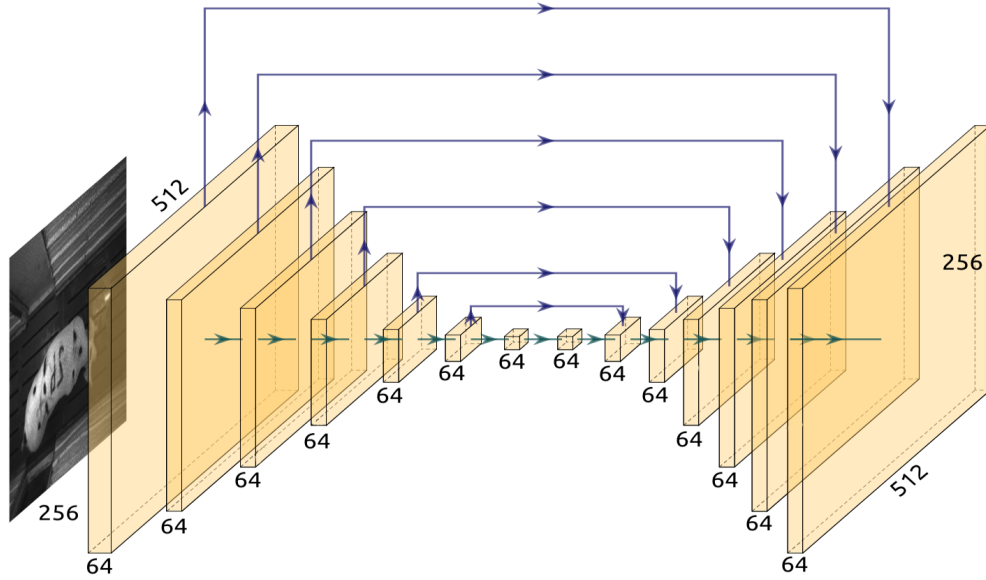
Now that the network can observe fine and coarse details from the image, the output must be shaped such that it produces a confidence map of the detected joints. The simplest option would be to up-sample the second to last layer to match the size of the final output, but that would cause inaccurate results as the detection regions would be large or ambiguous. Using the inverted pyramid structure to incrementally build-up to the correct output shape creates a more refined output. All of the sequential convolution layers use of skip connections to mitigate the loss of details, such as sharp edges. These connections re-introduce back the original inputs and recover details lost in previous filters. Each 'tier' of each pyramid is linked together with a skip connection since they already have the same layer shape; thus, easily added together. The overall network shape is referred to as an 'hourglass' structure.

#### 4.6.2.3 Implementation Details

The network utilizes a custom loss function for training. It is a variation on the standard cross-entropy loss but includes a weight map to alter the learning rates for specific regions of the image. The weighted cross-entropy loss is specified as:

$$\mathcal{L}_{CE}(z_i, y_i) = \sum_{i=1}^N w_i [-y_i * z_i + \log(1 + \exp(z_i))] \quad (4.1)$$

where  $w_i$  is the pixel weight,  $z_i$  is the output of the network, and  $y_i$  is the probability that the given pixel is the mark center, the ground truth. This loss is minimized when the sigmoid of  $z_i$  is equal to  $y_i$ . Here the pixel weight is chosen to balance the contribution to the loss of the small num-



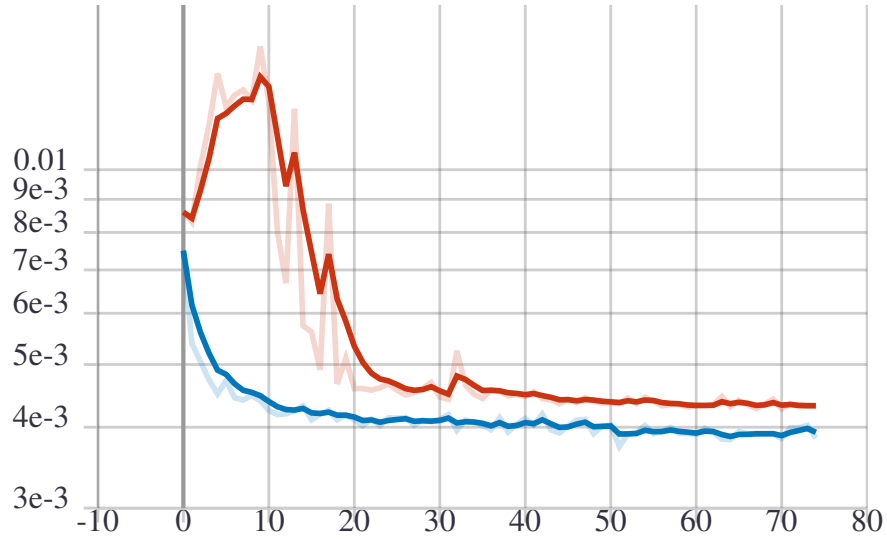
**Figure 4.2:** A high level diagram of the mark detection neural network architecture. Each orange box denotes a sequence of 3: convolution>batch\_norm>relu layers followed by either a max\_pool (shrinking in size) or upsampling layer (growing in size) with all layers having 64 channels each.

ber of pixels on marks with the far greater number on the background as specified in section 4.6.2.1.

The training session consisted of 75 epochs with a batch size of 8 images per batch using RMSprop optimizer with a learning rate of  $1e - 4$ . The training session took roughly an hour on the specified hardware in section 4.6.2. Figure 4.3 shows the value of the loss function output for the training and validation at the end of each epoch. The graph shows that the training and validation loss converged well, and the network learned something from the data.

### 4.6.3 Predictive Performance

A testing dataset was used to evaluate the network performance, and the error is defined to be the euclidian distance between the human-labeled joint center locations, and the network predicted centers. First, the output of the network is fed into a sigmoid function to generate a confidence map. To determine the centers of the joint in the confidence map, a peak local maximum filter is



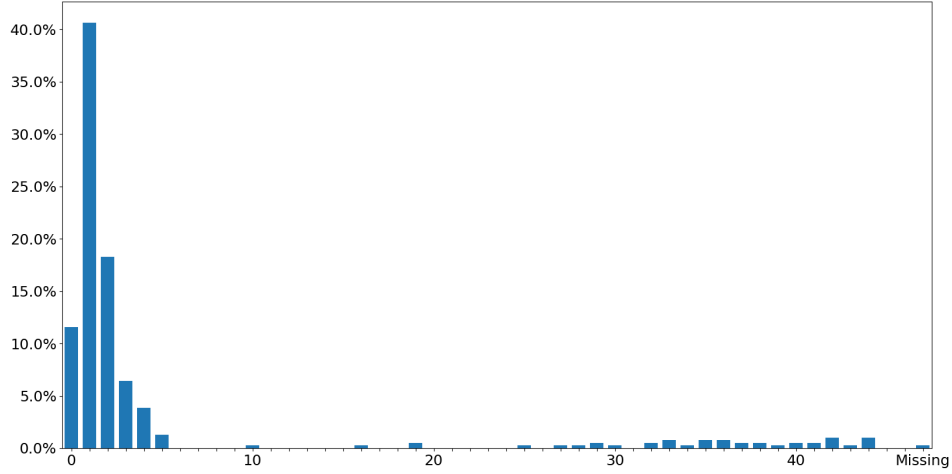
**Figure 4.3:** A graph showing the network loss during training for 75 epochs (Blue: training, Red: validation).

applied. This algorithm operates by dilating the input and using it to mask over the original image and comparing the locations of the maximums, returning a list of pixels. Comparisons between the annotations list and network output list is made by taking the L2 norm of the difference to determine the error. Figure 4.4 shows a histogram representing the error distance vs. the rate of occurrence.

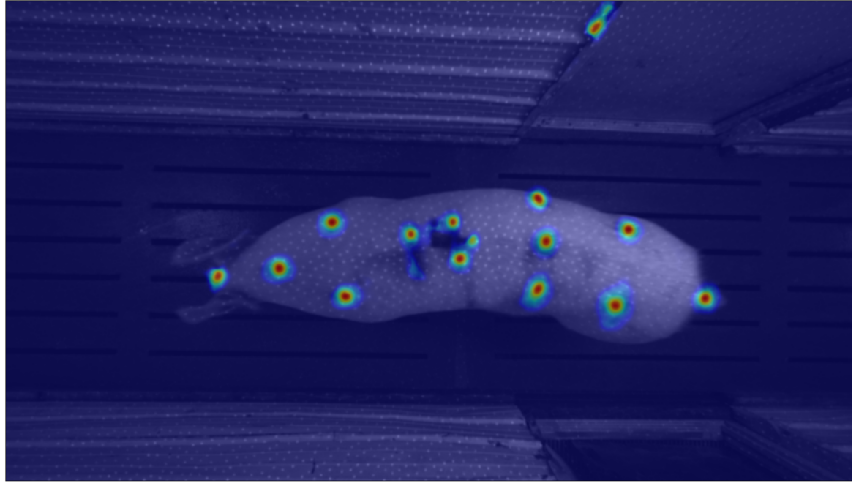
The average error throughout the testing set is evaluated to be 2 pixels. A conversion ratio to centimeters is calculated by dividing by the focal length of the camera given by the intrinsic, determined to be 0.235cm/pixels with an average accuracy of 0.5cm.

To evaluate the model holistically, plotting the confidence map over the infrared image provides intuition on how the network performs. Figure 4.5 is representation with the confidence map having an alpha channel of 0.7 to view the input infrared image.

The network performance does very well at detecting the marked locations as well as extraneous marks on the pig. But extraneous marks are not a significant concern as these marks are filtered



**Figure 4.4:** A histogram of the total error between the detected and human labeled joint center locations in pixels. The average error is 2 pixels or 0.5 cm accuracy.



**Figure 4.5:** The output of the mark detection network as a heatmap overlaid over the input infrared image. Marks in red denote high detection probability while blue indicates low detections.

during the joint association step of the *transfer labeling* method.

## 4.7 Optical Flow Joint Association

With the challenge of the detection of joints completed, the task of joint identification, the assignment of joint names to marked locations, needs to be completed. It is challenging to design a filter

to identify these joints automatically. It is due to variations in how sows orient their bodies and how to account for visual occlusion due to perspective changes. For example, when a sow moves their head, this causes the markers to be occluded and possibly not be detected. What is the best way to account for missed and false detections and how to correct for these possible occurrences?

Given that automated identification of joints involves a complex process of filtering and error correction, a human can perform this task with minimal effort if given a set of joint locations. So, if we can obtain an initial set of identified joints, ID propagation can be done throughout a sequence by tracking differences in movements. Optical flow algorithms can detect pixel-wise movements between images and estimate motion represented by a vector. These motion vectors can provide a heuristic to where a specific joint has traveled.

The following subsections describe the outline, development, and implementation of joint identification and association. Each topic discusses high-level reasoning's for each design choice and how to mitigate the flaws that accompany them.

#### **4.7.1 Outline**

To establish a clear path in how joint identification and association occur, the following procedures outline how this algorithm operates:

This method minimizes the amount of human labeling necessary for obtaining annotations by exploiting key similarities between frames of a sequence of images. An initial set of joint IDs are human-labeled for an image. These IDs and coordinates are sequentially generated for adjacent frames using optical flow as a tracker. The mark detection network evaluates the true location by searching around the estimated coordinate from the tracker. This operation is performed on all frames in the sequence while correcting for poor motion estimates and missed detections. This process accounts for human labeling error, since the mark detection network corrects for poorly marked initial labels.



---

**Algorithm 2** Joint ID Labeling and Association

---

**Input:**  $\{I_0, \dots, I_i, \dots, I_n\}$  as image sequence  
**for** every image  $I_i$  in the sequence **do**  
    **if**  $I_i == I_0$  **then**  
        human assigns IDs to marks on  $I_i$   
    **else**  
         $M \leftarrow$  do mark detection on  $I_i$   
         $F \leftarrow$  do optical flow between  $I_{i-1}$  and  $I_i$   
        **for** number of joints **do**  
             $joint_{flow} \leftarrow F(joint)$   
            search for local max on  $M$  around  $joint_{flow}$   
            assign max as new joint  
        **end for**  
    **end if**  
**end for**  
**Return:** joints

---

#### 4.7.2 Development

The optical flow algorithm is a crucial component of association performance. It tracks the movement of detected joints, providing estimate locations for sequential frames. The movements are calculated by taking the gradient of the image light intensity through the  $x$  and  $y$  axes. As a result, motion vectors can be calculated based on the gradient values [4, 16].

There are two different classes of optical flow algorithms: sparse and dense flow. Sparse flow produces vectors for 'interesting' features such as corners or edges of objects. Dense flow produces a vector for every pixel in the image with higher accuracy but at the cost of computational time. This type of flow is chosen because computation time is not a concern, and the higher accuracy helps obtain better joint location estimates.

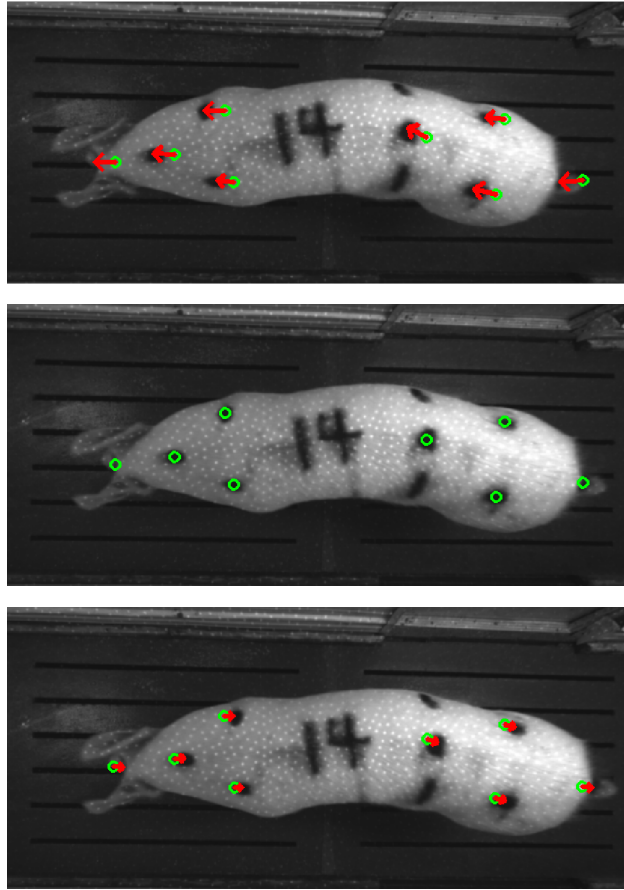
Modern dense optical flow algorithms are neural network-based. Generally, this approach takes two video images as inputs and outputs a flow image, which can be expressed as:

$$(u, v) = \mathcal{F}(I_{t-1}, I_t) \quad (4.2)$$

where  $u$  is the motion in the  $x$  direction,  $v$  is the motion in the  $y$  direction and  $\mathcal{F}$  is the network

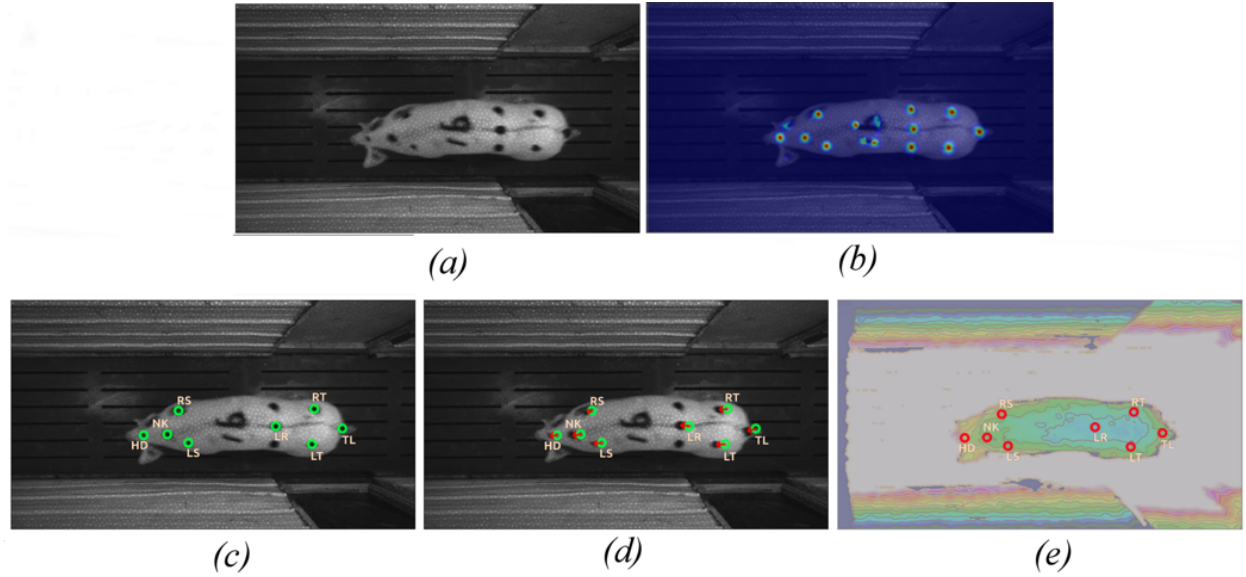
with inputs  $I_{t-1}$  and  $I_t$  being consecutive images.

The implementation chosen was the OpenCV version of the DeepFlow [44] algorithm. This network blends a deep matching algorithm with an energy optimization framework to match repetitive textures between the two images. Since the flow is calculated based on a strictly a pair-wise image set, there is no delineation between forward and backward motion. It allows for parallel computation of forward and backward movement starting from the initially labeled image. Figure 4.6 shows the motion vectors just for the joint location as the remaining pixels are not necessary.



**Figure 4.6:** Visual representation of the joint flow vectors. The middle image is the initialized set of joint labels from the labeler. The top image indicates predicted joint locations on the  $i + 1$  frame for forward motion overlaid on the infrared image. Likewise, the bottom image is the same but for the  $i - 1$  frame which is backwards motion.

### 4.7.3 Implementation and Labeling Procedure



**Figure 4.7:** The full joint detection and association process for generating annotations. Each image corresponds to separate step in the process and are described in section 4.7.3.

The labeling procedure is performed in a series of five steps. First, an infrared image is selected where all marks are visible on the sow, Fig 4.7(a). Next, the mark detection network extracts the mark coordinates from the image, Fig 4.7(b). A human labeler determines joint IDs using the custom labeling interface, Fig 4.7(c). After the initialization of the joint IDs, DeepFlow generates flow vectors for the identified joints. These vectors are added to the current joint coordinates to obtain an estimate for the next frame, Fig 4.7(d). Afterward, a new mark detection image is generated using the next frame. At each joint estimate, a search space of  $\pm 6\text{px}$  is used to find the true joint coordinates on the mark detection image. Joint IDs are given to the new coordinates and stored as annotations.

This approach is used to filter out false detections in the confidence map and is a secondary check if the detector does not find anything in that location. In cases where no joints are detected, the predicted optical flow location is trusted, and the human labeler is notified to check the validity of the joints. The labeler can either modify the predicted joint locations or proceed with the predicted

locations. This loop of joint propagation occurs throughout the whole image sequence. Finally, the generated annotations are used for the depth images, Fig 4.7(e), as the infrared and depth are pixel-wise aligned.

## 4.8 Evaluation and Performance

Evaluation of *transfer labeling* is done by comparing the joint locations between in hand and semi-automated images. The mark detection network determines the final joint coordinates, as described in the labeling procedure. Therefore, the mark network determines *transfer labeling* accuracy, which has a 2px error, as seen in Figure 4.4.

Time saved by using *transfer labeling* is represented in Table 4.1. It measures the number of human interactions required to produce the final labeled dataset. Overall, there is a 98.7% success rate, meaning that the human labeler would, on average, need to hand annotate  $0.013 * dataset\_length + num\_image\_sequences$ , number of images.

**Table 4.1:** Association results for landmarks in IR images. After human assignment of IDs, only 1.3% of images need human attention.

Number of Pig Traversal Sequences	158
Average number of Images per Sequence	126
Average number of Interventions per Sequence (excluding initial labeling)	1.5
Success Rate for Automated Association	98.7%

## 4.9 Summary on Semi-Automated Labeling

The proposed method of *transfer labeling* is a way to more efficiently create annotations for depth image sequences designed for pose-estimation. It uses a multi-modal system to determine key points for a single modality by observing markers only detectable in another. This method is applied to obtain sow pose annotations for a large dataset.

The implementation of *transfer labeling* is divided into two sections: detection and association. The detector is used to find markings on the sow's back to denote joints in which a pose-estimation network is supposed to predict. A custom hourglass CNN architecture network was designed to perform this task, which has an accuracy of 2 pixels or 0.5 cm. A human labeler then identifies the joint label for 8 detected marks in an image. Association is used to propagate these initial labels throughout the rest of the images in the sequence using the DeepFlow optical flow algorithm to obtain new estimates for joints. Estimates serve as initial search locations for joints on the next image. Newly detected marks are found from the network, and joint IDs are passed to the true joint coordinates. These IDs and associated coordinates are stored as annotations. The cycle of mark detection, joint prediction, and ID propagation is repeated for all images in the video sequence. The stored annotations are transferred to depth images since the infrared images are pixel-wise aligned.

The evaluation of *transfer labeling* is determined to have a 2px error with a 0.25% miss rate between generated and human annotation locations. It also reduces the number of human assignments required to generate the dataset by 98.7%.

## **CHAPTER 5**

### **POSE ESTIMATION NETWORK**

#### **5.1 General**

Pose is the detection of objects in images and videos. A collection of key points, or joints, are found and used to represent detections. Sow pose estimation uses a convolutional neural network that determines joint coordinates from sows depth images. These joints represent a skeleton structure, useful in observing sow functional morphology. This network is an intermediate step towards a quantitative method to evaluate sow body condition and lameness factors based on motion and body structure.

This chapter is divided into five different sections: network architecture, dataset processing, implementation, predictive performance, and a summary. The network architecture section describes the architecture of choice and how each module in the design is beneficial for evaluating pose. The dataset processing section covers preparatory procedures within the data loader. The network implementation describes the training hyperparameters and loss function used. The predictive performance evaluates how the network compares to the generated training and human labels—also a brief discussion on possible failure cases and possible improvements to the network. Lastly, the summary outlines a high-level overview of the network.

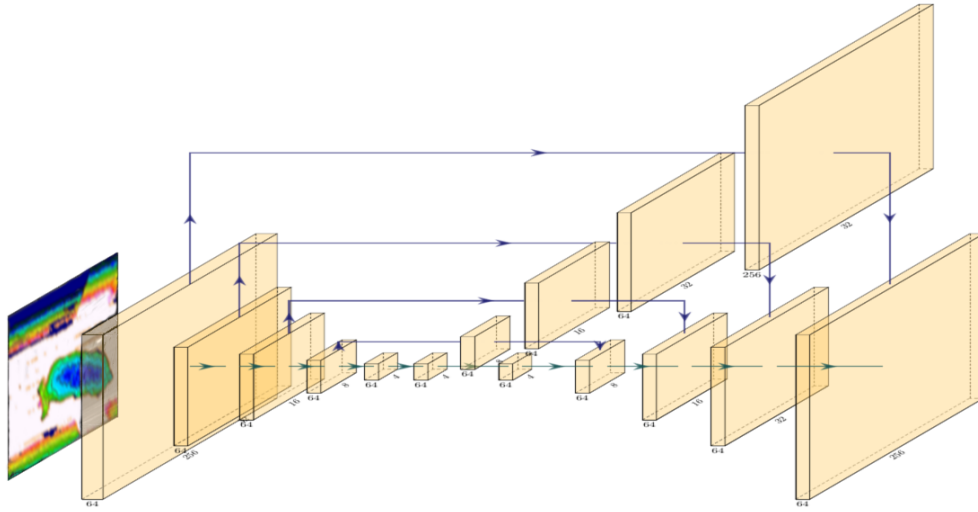
#### **5.2 Pose Network Architecture**

The network architecture is an implementation of Stacked Hourglass Network [27] modified for sows. This network structure is similar to the mark detection network, section 4.6.2.2, with its successive pooling and up-sampling layers to find different features at various resolutions. Differences include the use of two hourglass blocks and have more network parameters. It also

features intermediate supervised learning for better accuracy and reduced training time.

### 5.2.1 Hourglass Module

As previously mentioned in section 4.6.2.2, the hourglass module [27] contains successive convolution layers paired with either a pooling or up-sampling layer dependent on the layer of operation. Figure 5.1 shows the whole hourglass architecture design for the pose estimation task.

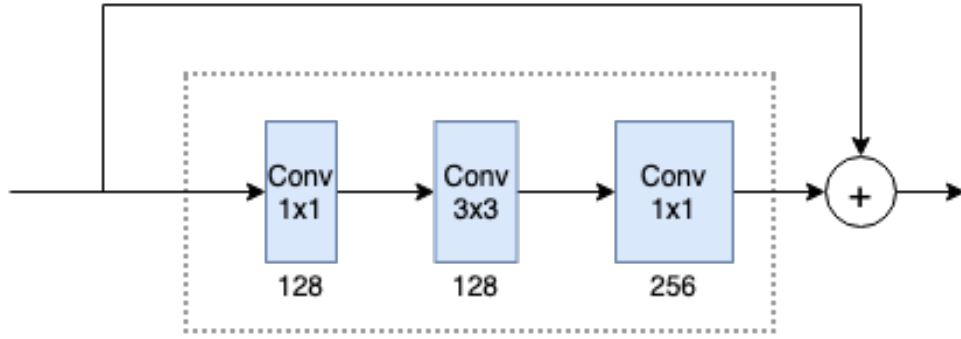


**Figure 5.1:** The architecture of a single hourglass module within the Stacked Hourglass Network.

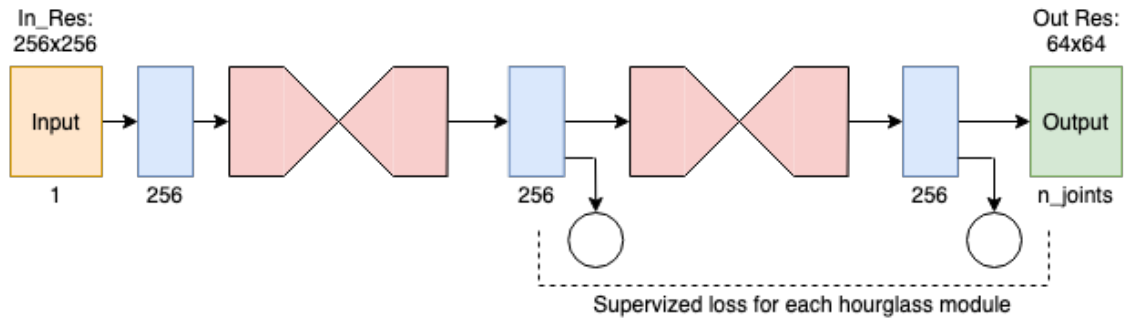
The difference between this design and the mark detection network design is that the skip connections now include a set of convolutional operations, and there is a range of channels for each convolution. The skip connection setup is referred to as a residual module, covered in section 2.4.4, with a specific set of convolution pairings, seen in Figure 5.2.

### 5.2.2 Complete Architecture Design

With the hourglass module established, the high-level design of the architecture is a combination of the previously mentioned modules. Figure 5.3 shows the overall architecture for a 2-stacked hourglass configuration used for this network.



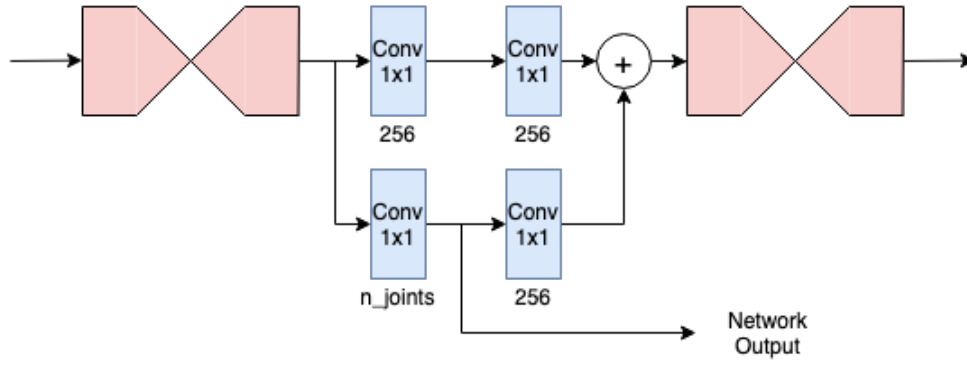
**Figure 5.2:** The residual module specific to the Stacked Hourglass Network design. Numbers below the convolution blocks are the number of channels in that layer.



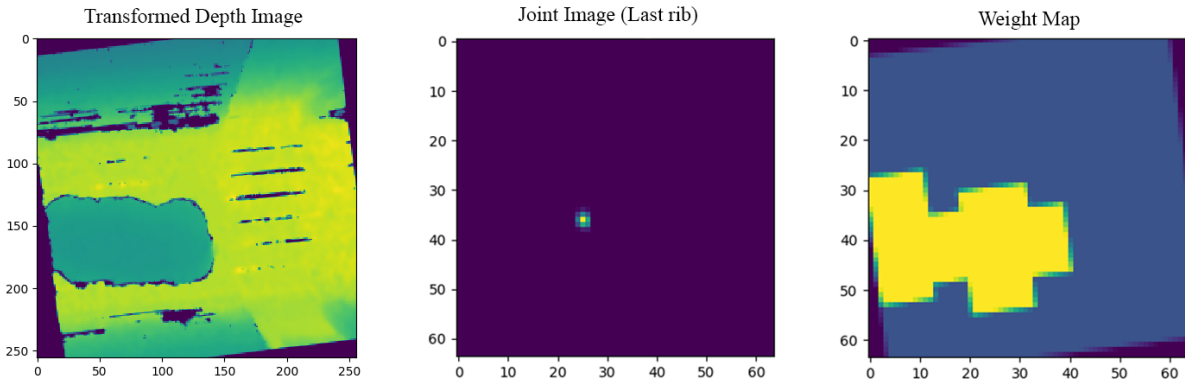
**Figure 5.3:** An high level overview of the Stacked Hourglass Network architecture. Blue blocks denote convolution layers used to match channel sizes between each module. Red blocks are an hourglass module described in section 5.2.1. The circle shows the locations in the network where supervised loss is evaluated.

The input is defined to be a 256x256px resolution image with an output resolution of 64x64px. An initial convolution layer is used to shape the input to have 256 channels to match the channel size of the hourglass module. After the hourglass portion, there is an intermediate set of layers used to extract the joint information used for loss evaluation. A detailed view can be seen in Figure 5.4. The output of the hourglass is forked into two different paths: one for passing information to the next hourglass and the other for producing a set of confidence maps, used for training evaluation. This intermediate evaluation is to make the training of the large network more efficient and to increase overall network accuracy.





**Figure 5.4:** Network structure in between the hourglass modules within the Stacked Hourglass Network design. The red symbol denotes an hourglass module. Numbers below the convolution blocks are the number of channels in that layer.



**Figure 5.5:** Sample transformed depth image, joint image at the last rib, and weight map.

### 5.3 Dataset Preprocessing

This network uses the joint annotations generated from *transfer labeling* as well as the associated depth images. This dataset contains information from 3 recording sessions of 158 sows with over 20k images. A data loader was created to convert the dataset annotations into joint location prediction images with an associated weight map for network training.

Joint location images are made by generating a Gaussian distribution with zero mean and six-pixel variance at each joint. These images represent the probability of the existence of a joint at each pixel, or a joint confidence map. Weight maps are used to balance the training between joint and background pixels equally. These maps are made by assigning foreground pixels a value of 1.0 to

a circular region that is 0.15% the image resolution about all joint locations. The size of the region is determined to encompass the whole sow. Background pixels have a value of  $\frac{n_f}{n_b}$ , where  $n_f$  is the number of foreground pixels and  $n_b$  is the number of background pixels. This value provides an evenly distributed weight between background and foreground pixels. All images and maps are then rescaled to meet the pose network input resolution. Finally, image augmentation is done by: scaling, rotation, and vertical mirroring to increase input variability. Figure 5.5 shows the input depth image along with the joint image and weight map at the last rib location.

## 5.4 Network Implementation

### 5.4.1 Input and Output Layers

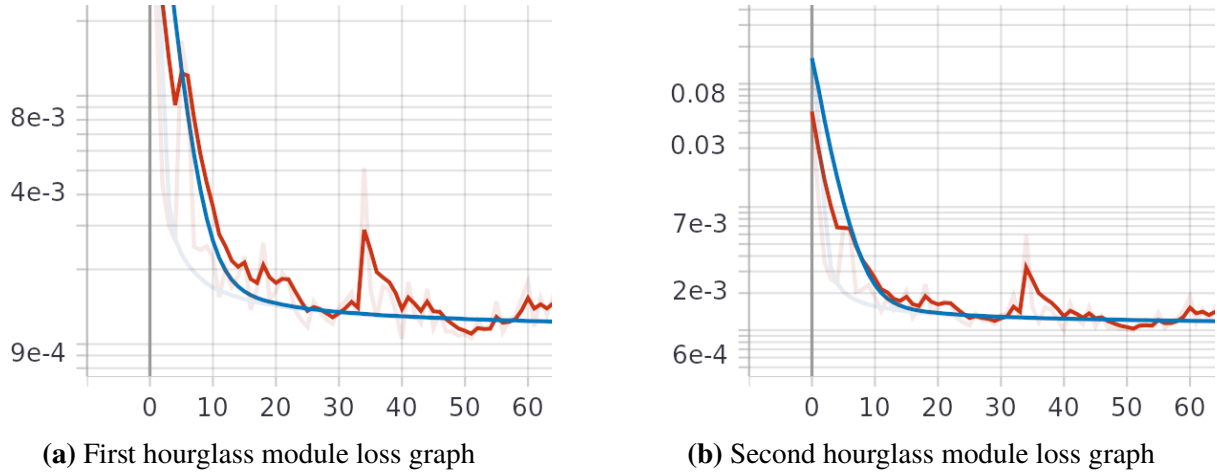
The input layer uses a normalized depth image at 256x256 pixel resolution. The dataset images normalized from [0,1] and are rescaled to meet this requirement. The output layer takes the form of a list of stacked confidence and weight maps, since the network architecture uses intermediate supervision between hourglass modules. Therefore, a set of two confidence and weight map pairs are stacked to guide the training of the individual hourglass modules. Each pair has a 64x64 resolution image with 16 channels, with the first eight channels being a set of confidence maps at every joint, and the last eight channels are the accompanying weight maps.

### 5.4.2 Loss Function

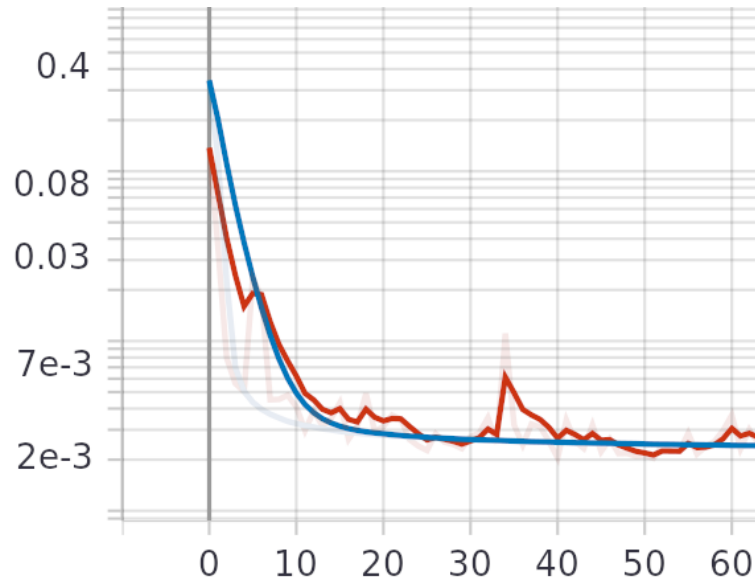
A weighted cross-entropy loss is used for optimization, defined in Algorithm 4.1. Cross-entropy loss minimizes the distance between the predicted and ground truth distribution, making the network learn the joint confidence maps. A weight factor is applied to increase the rate of learning of joint locations by reducing the importance of background pixels. These pixels, like walls and the floor, have a minimal change compared to locations where there is a sow. The network is encouraged to learn parameters related to joint locations, resulting in faster convergence and lowering training time.

## 5.5 Predictive Performance

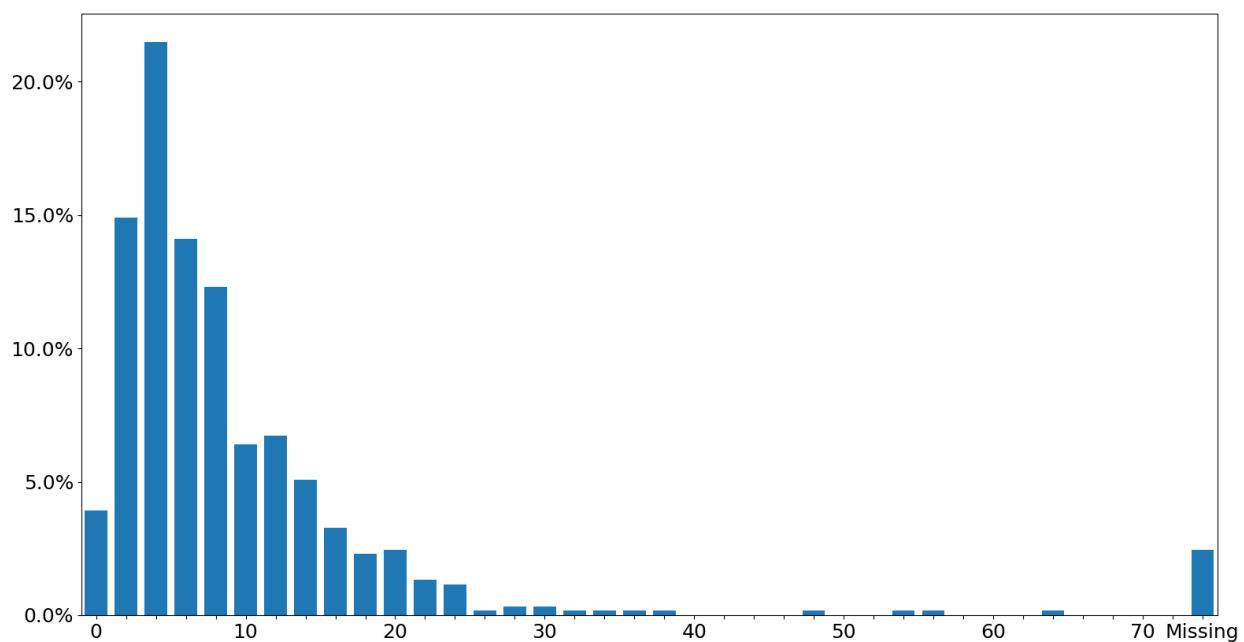
The network was trained with all network weights were saved after each epoch. The 51st epoch weights was selected for evaluation because it has the lowest validation lost. RMSprop optimizer [40] with a learning rate of  $8e - 5$  was used. Figures 5.6 are the total loss of the network.



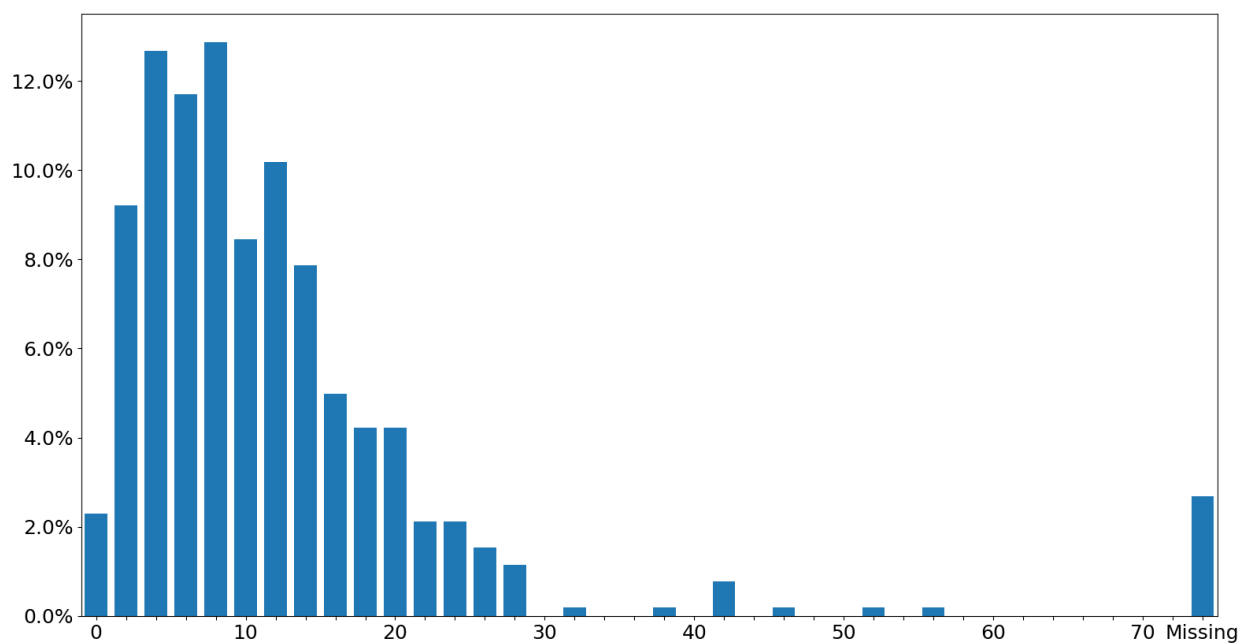
**Figure 5.6:** Hourglass module loss graphs (loss/epoch). Red line denotes the validation loss and blue denotes the training loss.



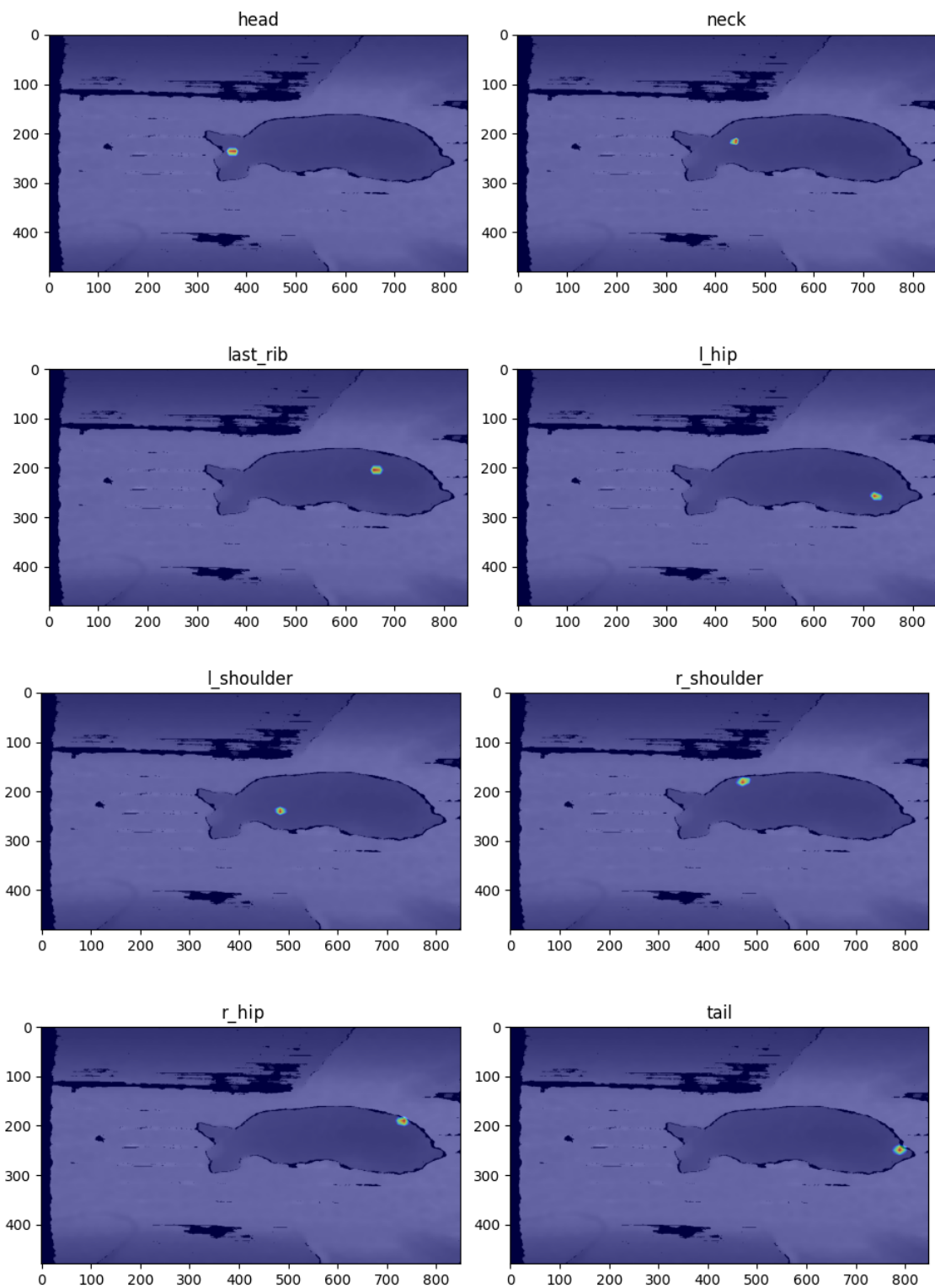
**Figure 5.7:** The loss graph of the training and validation from the complete stacked hourglass network per epoch. Red denotes the validation and blue being training.



**Figure 5.8:** Error graph between the infrared mark extracted joint locations and the depth pose estimation network. The average error is 8.7 pixels (2.1cm) with a 2.4% miss rate.



**Figure 5.9:** Error graph between the human annotated joint locations and the depth pose estimation network. The average error is 11.2 pixels (2.6cm) with a 2.6% miss rate.



**Figure 5.10:** A sample output of the pose detection network. A heatmap of joint existence is generated for all 8 joints with red denoting the highest probability and blue being the lowest. The size of the sow is roughly 1.2m in length.

### 5.5.1 Heatmap Joint Extraction

Images of predicted joint locations are overlaid on the input image was used to observe the network output. Figure 5.10 is this representation for a sample input. Local maximums of the network produced confidence maps were determined at each predicted joint location. This process uses the peak local max algorithm from *scikit-image* to extract the joint locations for accuracy evaluation.

### 5.5.2 Accuracy Evaluation

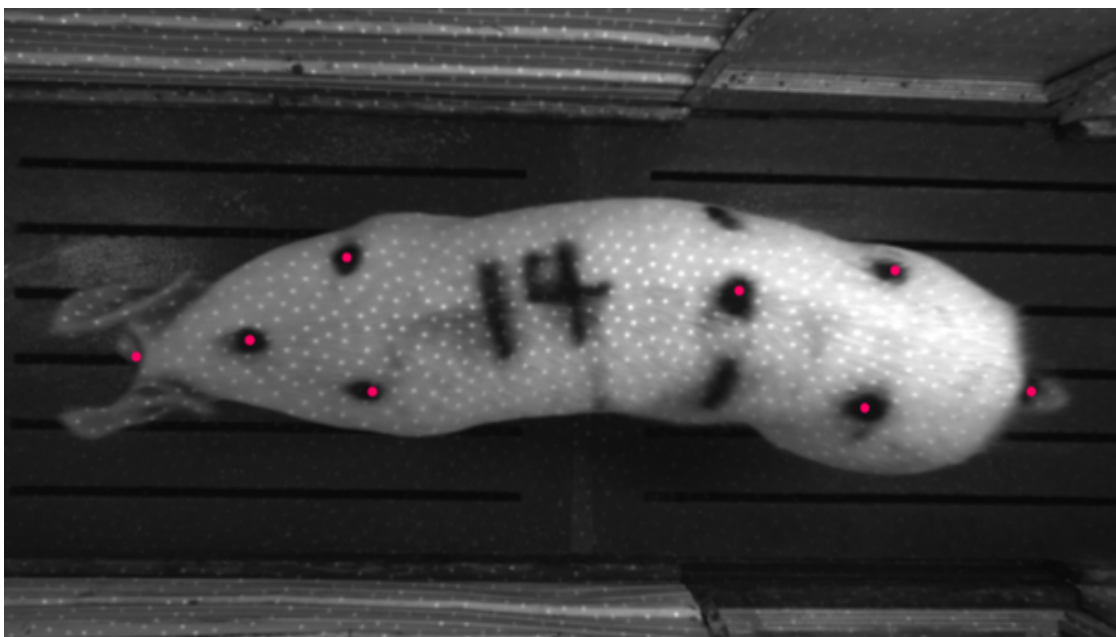
Network error is defined to be the norm between the network and annotation joint locations. Two evaluations were performed, one between the generated joint marking locations (Figure 5.8) and the other from the human-annotated set of joints (Figure 5.9). The generated joint to network location comparison shows how well the network can determine joint locations from depth images. The human to network evaluation shows the total error using *transfer labeling* and depth images.

**Table 5.1:** Average error and miss detection rate between depth-based pose-estimation network and annotations

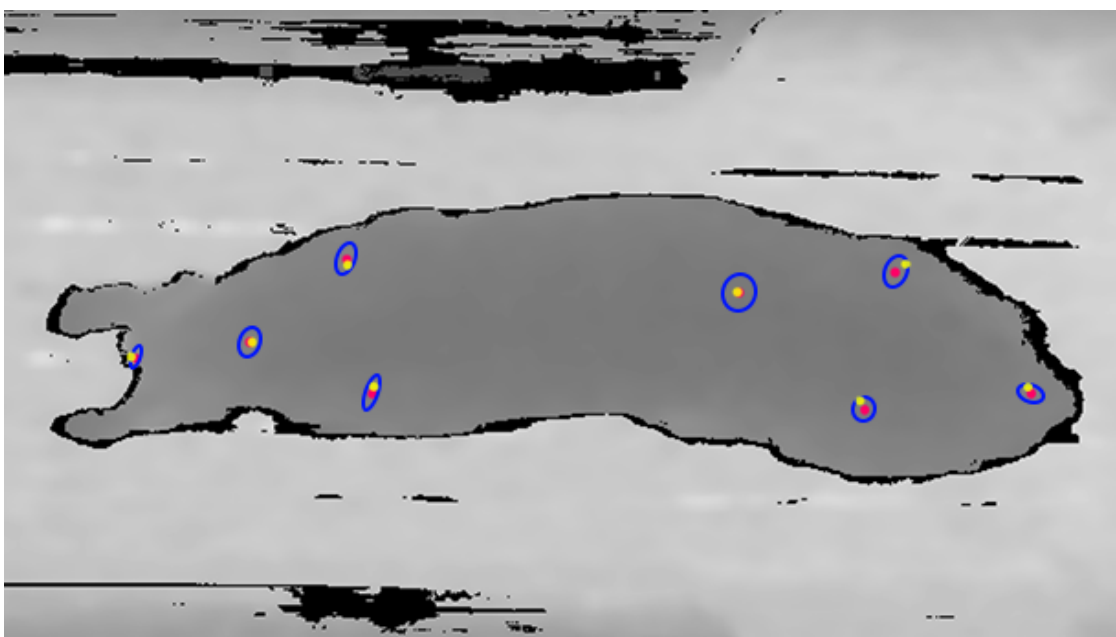
Annotations set	Error in pixels	Error in cm	Miss detection rate
Infrared based joint marker extracted annotations	8.7	2.1	2.4%
Human annotated joint locations	11.2	2.6	2.6%

Table 5.1 shows the average error between these evaluation sets along with the miss detection rate where no joint was detected. The results show that there is an  $8.7px$  error with a 2.4% chance of a missed detection. The performance between the human and generated labels is slightly worse with  $11.2px$  error and 2.7% missed detection rate; due to the error using *transfer labeling* for dataset generation.

Figure 5.11 shows the 99.7% confidence region represented as an ellipse of each joint. Evaluation is made by aligning the sow horizontally using a the angle from last rib to tail as a reference. Then, a vector is made between the ground truth and the predicted joint coordinates and stored. These values are used to calculate the covariance matrix for each joint and deviation is found from the

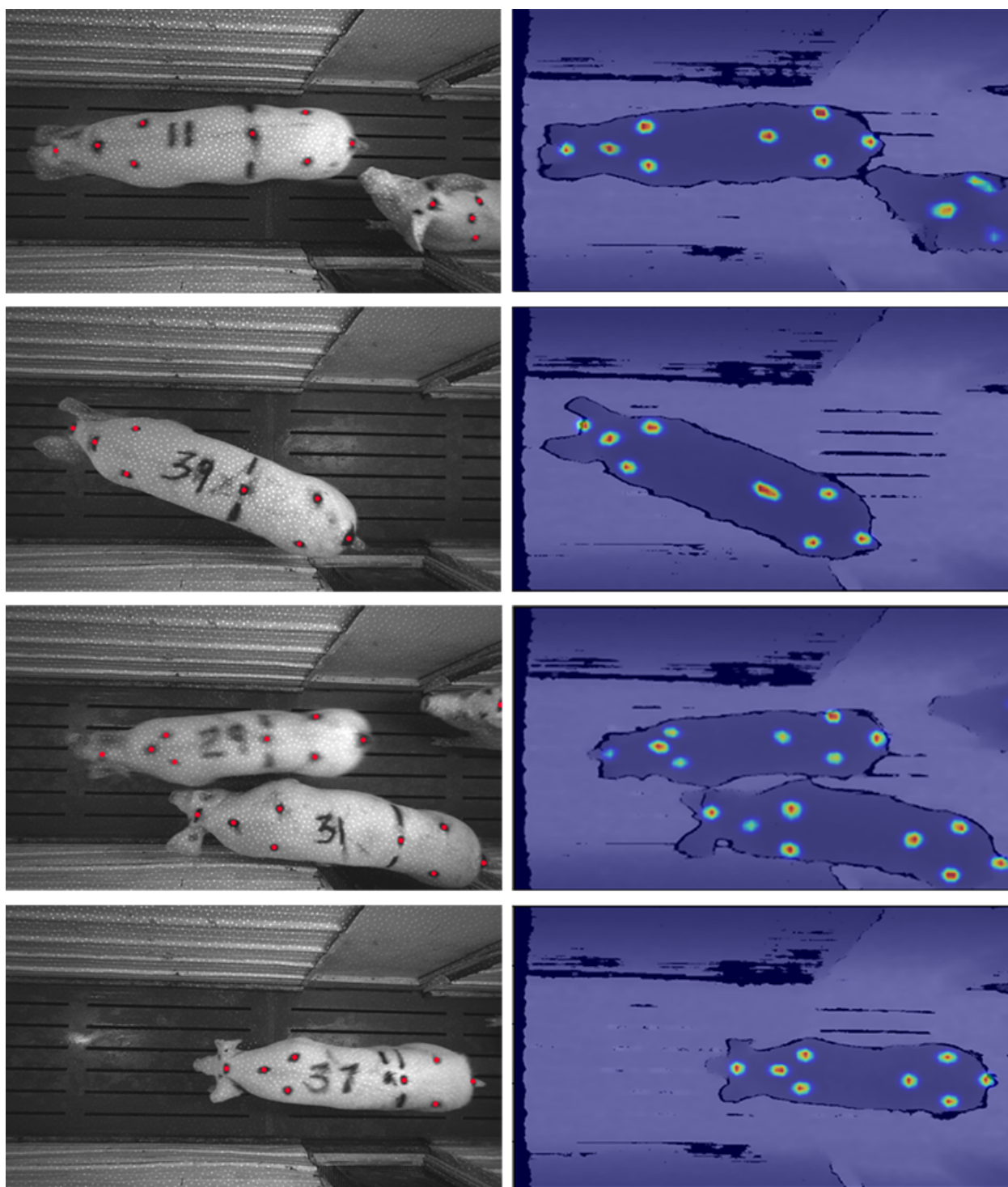


(a) Reference sow infrared image with joint marks and the ground truth joint locations (red).



(b) Sow depth image with the ground truth joint locations (red), predicted joint locations (yellow) and the covariance ellipses (blue) at 3 standard deviations.

**Figure 5.11:** Evaluation of the 99.7% confidence region for predicted joint locations indicated by the blue ellipses.



**Figure 5.12:** Network sample outputs for multiple sow poses. Left is the ground truth infrared image with the labeled joints indicated in red. Right is the compiled network output on the depth image with detected joint locations represented by the heatmap.



square root of the diagonal entries. Results show that there is little variation on the head joint and the most at the last rib location. This is due to a lot of contextual information around the head region and little for the last rib position. Table 5.2 show the deviation values along each axis.

**Table 5.2:** Predicted joint deviation along the axial and lateral axis of the sow for a 99.7% confidence interval.

Joint Name	Axial Deviation ( $\sigma = 3$ )		Lateral Deviation ( $\sigma = 3$ )	
	px	cm	px	cm
Head	6.7	1.6	14.6	3.4
Neck	13.2	3.1	18.4	4.3
L shoulder	8.5	2.0	22.4	5.3
R shoulder	11.6	2.7	19.6	4.6
Last rib	19.7	4.6	21.9	5.2
L hip	13.9	3.3	14.9	3.5
R hip	13.5	3.2	19.5	4.6
Tail	15.6	3.7	10.7	2.5

## 5.6 Summary

The network trained by *transfer labeling* estimates the sow's joint locations from a depth image. This network is based off of the Stacked Hourglass Network by Alejandro Newell [27] with modifications to the input layers and loss function.

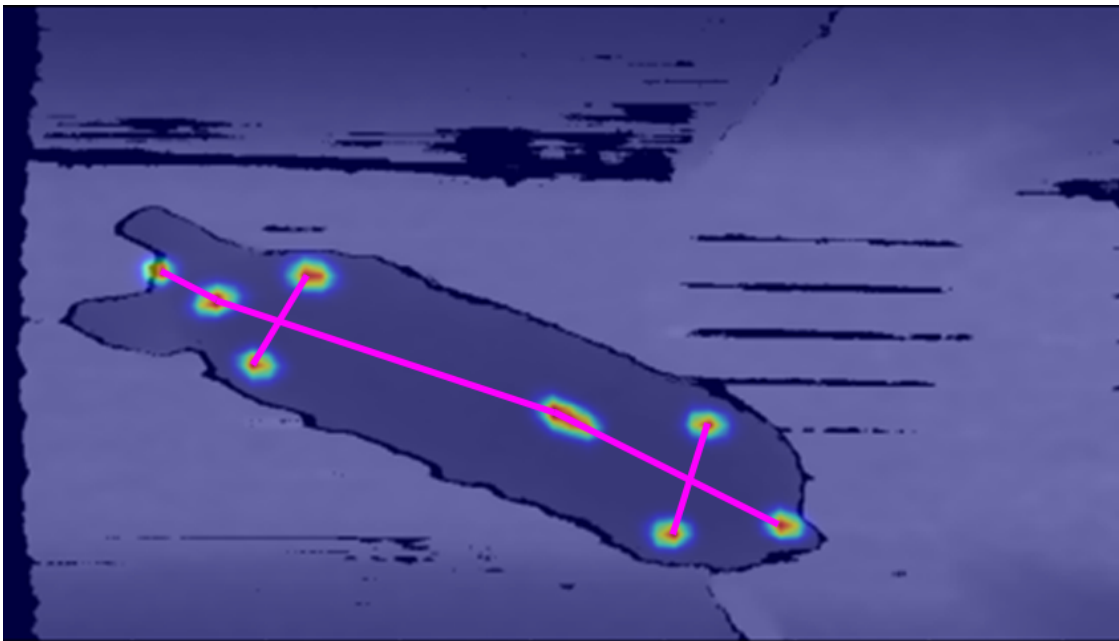
The network uses three images for its input: a scaled and normalized depth image, a set of joint location images, and a weight map. The joint location image is created by using a Gaussian distribution centered at the joint location. The weight map is used to define foreground, regions where the sow is located, and background pixels. Foreground pixels are assigned a value of 1.0 and background pixels a ratio between the number of foreground to background pixels.

The loss function used a weighted cross-entropy loss. The weight factor applied is from the weight map provided. This factor makes the network focus on parameters related to the foreground pixels during training. Thereby converging the network faster and reducing training time.

The results of the network produces joint location prediction images for every joint. The analysis shows that there is an 8.73 pixel (2.05cm) error with a 2.4% miss rate between the joints from the infrared annotations, and 11.24 pixel (2.64cm) error with a 2.6% miss rate between the human-annotated joints.

## 5.7 Alternative Approaches and Future Work

Possible modification to the pose network involves adjusting the architecture to match the original input resolution (848x480px). It might reduce the detection variance in the joint confidence maps since there are more network parameters and higher image resolution, reducing joint location error. Other possibilities for improvement include using a different loss function like KL divergence or mean squared error. These functions optimize the network differently and may produce better results.



**Figure 5.13:** Compiled image of all joints detected overlaid over the input infrared image with a drawn skeleton to represent the full body pose.

The pose network can serve as a foundational platform for sow health detection devices. It can be

seen by compiling the prediction set of images; the joint estimates generate a pose represented by a skeletal structure like in Figure 5.13. This skeletal structure serves as a model to perform various health assessments like kinematic analysis for lameness detection and holistic body condition scoring.

Further research work can be extended outside the swine industry. With the non-invasive aspect of the *SIMKit*, adapting this device and network for a variety of animals such as cows, goats, chickens, and horses can be possibilities in the future. This device can also be used for verification testing of animal pharmaceuticals. By taking evaluations before and post-drug administration can serve as a verification method of drug effectiveness. The utilization of such technology can improve the quality of care animals and provide a way to reduce the number of animals removed from the herd.

## CHAPTER 6

### CONCLUSIONS

Through quantitative evaluation to reduce the incidence of lameness and poor body condition in sows, precision livestock farming can provide solutions to reduce mortality and pre-mature culling. Using this technology, early detection and accurate evaluations of functional morphology can provide an overall health assessment and indicate animals requiring stockperson attention and treatment and perhaps, preventing unnecessary removal of animals from the herd.

The development of the *SIMKit* device enabled the collection of sow information for training artificial intelligence. The *SIMKit* was specifically designed to withstand harsh farm environments while accounting for limited connectivity in rural areas. It is a standalone system that records, encrypts, and saves infrared and depth image data. With this collected information, the creation of a sow pose dataset was created.

The construction of the sow pose dataset was completed through the development of a semi-automated labeling technique called *Transfer Labeling*. This method utilizes pose-annotation markings on sows, only visible in the infrared images, to generate precise labels for depth image sequences. Through the creation of a mark detection network, the marks on the sows are extracted. The use of optical flow propagates user-defined ids to all images in a sequence with two-pixel accuracy. Thereby, generating over 20k image annotated dataset efficiently for a depth-based pose-estimation network.

A sow pose-estimation network was created by adapting the Stacked Hourglass Network, designed for humans, to work with sow depth images. This modified model features a weighted cross-entropy loss function using a custom weight map for the refined learning of joint confidence maps. The trained model was evaluated based on the error between the annotations (human and generated)

and the network assessed locations. The results show it has a 8.7px (2.1cm) error with 2.4% miss detection rate for generated labels and 11.2px (2.6cm) error with 2.6% miss detection rate for human labels. This model produces promising results for sow pose estimates, and with further refinements, can serve as a platform for lameness detection and body condition scoring networks.

Overall, the *SIMKit* device and the depth-based pose-estimation network provide a new pathway into precision livestock farming. The developed hardware serves as a model in how to design environmentally robust farm technology and the network provides an approach to sow functional morphological evaluation. Further research can use these results to investigate automated prediction of lameness factors and body condition scoring. This thesis demonstrates the development of tools using artificial intelligence within the agricultural industry and recommends continued research for health analysis models to improve productivity and animal welfare.

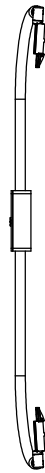
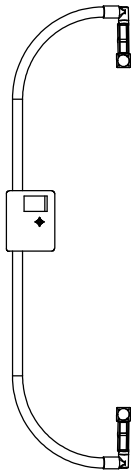
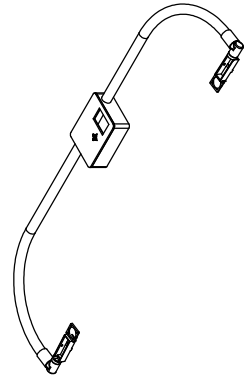
## **APPENDICES**

## **APPENDIX A**

### **SIMKIT - HARDWARE**

#### **A.1 CAD Drawings**

6 5 4 3 2 1

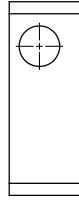
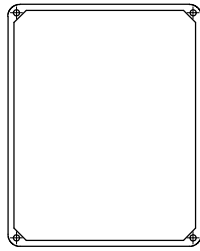
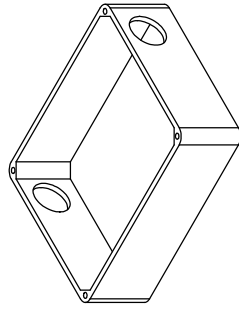


UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION: 1.0	
SURFACE FINISH:		LINEAR:		TOLERANCES:		Steven Yik			
ANGULAR:		NAME		SIGNATURE		DATE		TITLE:	
DRAWN								SIMKIT Device	
CHKD									
APP'D									
MFG									
QA								DWG NO.	
								SIMKIT v1.0	
								A4	
								SCALE: 1:20	
								SHEET 1 OF 1	
								WEIGHT:	

6 5 4 3 2 1



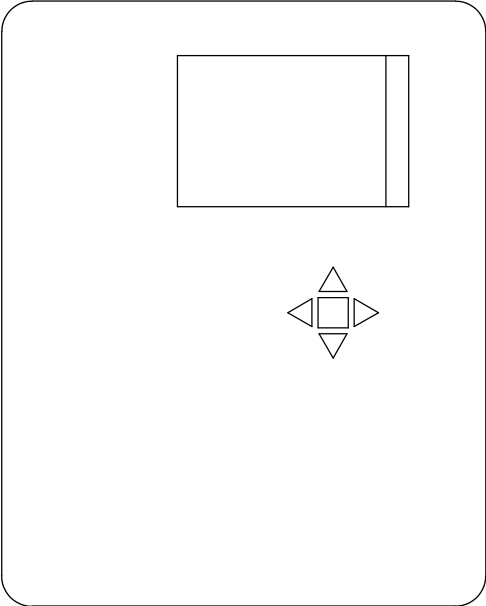

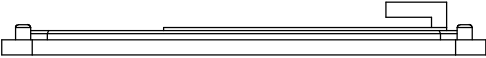
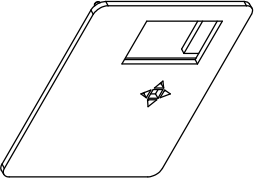




UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS		FINISH:		DEBUR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING		REVISION: 1.0	
TOLERANCES: FRACTIONAL ANGULAR:						Steven Yik			
						TITLE:			
DRAWN		NAME		SIGNATURE		DATE		SIMKIT Box Base	
CHK'D									
APP'VD									
MFG									
Q.A.								DWG NO.	
								Box Base v1.0	
								A4	
								SCALE: 1:5	
								SHEET 1 OF 1	

A vertical number line with tick marks labeled 1, 2, 3, 4, 5, and 6 from top to bottom.

6 5 4 3 2 1

								
UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS			FINISH:		DEBURR AND BREAK SHARP EDGES		DO NOT SCALE DRAWING	
SURFACE FINISH:			TOLERANCES:		DRAWN		STEVEN YIK	
LINEAR:			ANGULAR:		CHKD		TITLE:	
NAME			SIGNATURE		DATE		SIMKIT Box Lid	
DRAWN							DWG NO.	
CHKD							A4	
APP'D							Box Lid v1.0	
MFG							SCALE: 1:2	
QA							SHEET 1 OF 1	
MATERIAL:								
WEIGHT:								

6 5 4 3 2 1

## **A.2 Construction Manual**

---

# SIMKit Construction Manual

---

Steven Yik  
Michigan State University  
College of Electrical and Computer Engineering  
Masters: 2018-2020  
yiksteve@egr.msu.edu  
Feb. 9, 2020  
Manual — Version 1

## Description:

This document contains build instructions for the *SIMKit* Data Collection Device intended use for academic research for Michigan State University and given licences for individuals. Operational instructions are contained in a different document.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Intended Use, Fabrication, Operator . . . . .	3
<b>2</b>	<b>Hardware</b>	<b>3</b>
2.1	Parts and Tools Required . . . . .	3
2.2	Software Setup: . . . . .	4
2.3	Construction Instructions: . . . . .	4
2.4	Reference Images . . . . .	6
<b>3</b>	<b>Installation</b>	<b>7</b>
<b>4</b>	<b>Troubleshooting</b>	<b>7</b>
4.1	Hardware . . . . .	7
4.2	Software . . . . .	8
<b>5</b>	<b>Known Issues</b>	<b>8</b>
5.1	Hardware . . . . .	8
5.2	Software . . . . .	8
	<b>Appendix</b>	<b>9</b>



## 1 Introduction

### 1.1 Intended Use, Fabrication, Operator

This section provides an overview for how the *SIMKit* is used and constructed for different classes of operation.

**Intended Use:** This device is intended for use and operation for academic settings and private licensing for top-down topological data collection using infrared and depth imaging.

Unintended use is not subjected to liable hardware or software consistencies and are not designed to stay within constraints of this manual.

**Fabrication:** The construction of this unit is done with multiple fabrication techniques and are required to know or obtain necessary skills for successful creation.

These qualifications are outlined in the *Construction Prerequisites* section of this document.

**Operator:** The operators of this device should be trained by the fabricator on software and hardware use and maintenance before using.

## 2 Hardware

### 2.1 Parts and Tools Required

#### Parts List:

- 1x - *Nvidia Jetson Nano Developer Kit*
- 2x - *Intel Realsense Depth Camera D435*
- 1x - 32GB Micro SD card
- 1x - 3.5in LCD HDMI Monitor
- 1x - USB3.0 Portable SSD (250GB - 1TB)
- 2x - 3ft USB-C to USB3.0 Cables
- 5x - Push Buttons w/ Assortment of resistors and wire
- 1x - Perf-board (2x3in)
- 2x - 40mm 5v DC Fan
- 1x - 5V/4A AC-DC Power Supply w/ 5.5x2.1 Barrel Plug
- 2x - 3/4in 90 Deg Schedule 40 PVC w/ Belled End
- 1x - 3/4in 10ft Schedule 40 PVC
- 1x - 1Kg roll of PLA (used for 3D printing)
- 2x - 1/4in x 1/2in Bolts
- 4x - 1/4-20 Bolts
- 8x - M2 Screws
- 12x - M2 Nuts

! → Parts list may vary due to construction methods and are subjected to change. For best results, follow the guide carefully and adapt to the situation if problems do occur.

### Recommended Tools:

- Saw
- Dremel
- Drill w/ Bit Set
- Soldering Iron
- Measuring Tape
- 3D Printer (7x8in print bed minimum)
- An Assortment of Screwdrivers and Allen Wrenches
- Hot Glue Gun
- Pliers

## 2.2 Software Setup:

This section is software preparation for the *Nvidia Jetson Nano Development Kit*.

- ! → For access to the list of software procedures for post operating system setup, please send an email to [yiksteve@egr.msu.edu](mailto:yiksteve@egr.msu.edu) for permission to the Git project.

### Board Initialization:

1. Follow the initialization and flashing process on *Nvidia's* website: **Get-Started-Jetson-nano-devkit**
2. Proceed to install the programs in the *Software-Instructions* file in sequential order.

## 2.3 Construction Instructions:

Construction assumes that you have obtained or have access to the tools and materials listed above.

- ! → For access to the 3D print files and software setup scripts, please send an email to [yiksteve@egr.msu.edu](mailto:yiksteve@egr.msu.edu) for permission to the Git project if you did not do so previously.

Follow the instructions below in the order that they are presented. All critical instructions and comments will be denoted with the ! symbol so read carefully before proceeding.

Software Box : 1. 3D print the following parts with quantities:

- 1x - Box\_Base.STL
- 1x - Box\_Top.STL
- 4x - Button\_Arrow.STL
- 1x - Button\_Mid.STL

2. Place items off to the side as these items are used as reference for other parts in the construction process.

-----

Camera Holders : 1. 3D print the following parts with quantities:



- 2x - Camera\_Holder.STL
  - 2x - Clamp.STL
2. Feed the 10ft Schedule 40 pipe through the circular openings in the Software box.
  3. Fit one end of the pipe with the 90 Deg Schedule 40 PVC piece and measure out 5ft from the end of the pipe horizontally.
  - ! → 4. Place the second 90 Deg PVC against the 5ft distance and mark out where the belled portion starts and meets with 10ft PVC pipe. **DO NOT** mark at the end of the belled portion or the length of the pipe will be incorrect.
  5. Cut down the 10ft PVC pipe at the mark.
  - ! → 6. Locate the center of the pipe and create a notch roughly 3.5in in length and 1/4in deep. This notch is to provide an access way for cables to be travel to the cameras without being exposed.
  7. Press fit the Camera\_Holders at the ends of the 90 Deg PVC pieces.
  8. String through the 3ft USB-C to USB-3.0 from the notch to the camera holders such that the USB-C end is located in the camera holder.
  9. Repeat the previous step on the other side of the device.
  - ! → 10. Proceed to do the same process with 2 strands of 18 AWG wire on each end for power lines to the fan shrouds. Leave at least 6in of slack in the software box for easier wiring.
  11. Plug in and secure the *Intel Realsense Cameras* in the camera holder and screw in the 1/4in x 1/2in bolt on the side.
  12. Screw down the 40mm fans in the fan shrouds located at the ends of the camera holders.
  13. Cut and solder the power lines for the fans to the 18 AWG wire previously strung through.

-----

- Software Box Electronics: :
1. Orient the software box in front of you with the pipe along the top edge.
  2. Place the *Nvidia Jetson Nano* board in lower left corner with the ports facing towards the right of the box. Mark the mounting holes and drill out using a 1.6mm drill bit.
  3. Screw down 4 M2 bolts from the bottom side of the box and secure using nuts.
  4. Plug in the cameras on the two lower USB 3.0 ports of on the *Jetson* board.
  5. Plug in the portable SSD in one of the remaining USB 3.0 ports.
  6. Create a board with the simple schematic based on the design in Appendix A of this document.

7. Plug in the HDMI cable into the 3.3in Display and the *Jetson* board.
8. Enclose everything and screw down the lid to of the box with 4x 1/4-20 bolts.

## 2.4 Reference Images

Here are some reference images of the completed device:



Figure 1: Software Box

-----



Figure 2: Camera Holder

-----



Figure 3: Ceiling Mounted Device

### 3 Installation

To install and operate the device, look for a location with the following:

- Well isolated area that is not exposed to the elements
- Location where lighting is consistent during operation
- Ceiling with exposed conduit or easily mountable surfaces
- Mounting height is between 2.5-3.5 meters (8-12ft) from the ground
- Easily accessible outlet location

For setting up:

1. Clip the device along the ceiling conduit using the provided C-clamps on the device.
2. Run an extension cord from a nearby outlet to the device and plug it in
3. The device should turn on and automatically boot to the data-collection interface.

### 4 Troubleshooting

#### 4.1 Hardware

Within a farm setting, any exposed hardware devices are susceptible to corrosion or dust buildup. In the event where this occurs, use antiseptic wipes to clean off the device and re-connect any exposed connections, i.e. cameras. The contained software box should not have any issues regarding withering as it is isolated.

This device is not waterproof. Any direct contact with water will potentially cause issues or complete failure of the device.

## 4.2 Software

The software is run using the system startup scripts ” /.config/autostart/”. If the device does not auto-boot into the user interface, check these scripts and verify its operation.

Hardware malfunctions may prevent the software from working correctly. If the interface is not intractable, i.e through the buttons, it is most-likely a hardware issue.

The terminal on the device will display any software related issues and is easily diagnosed by the diagnostic messages displayed.

## 5 Known Issues

### 5.1 Hardware

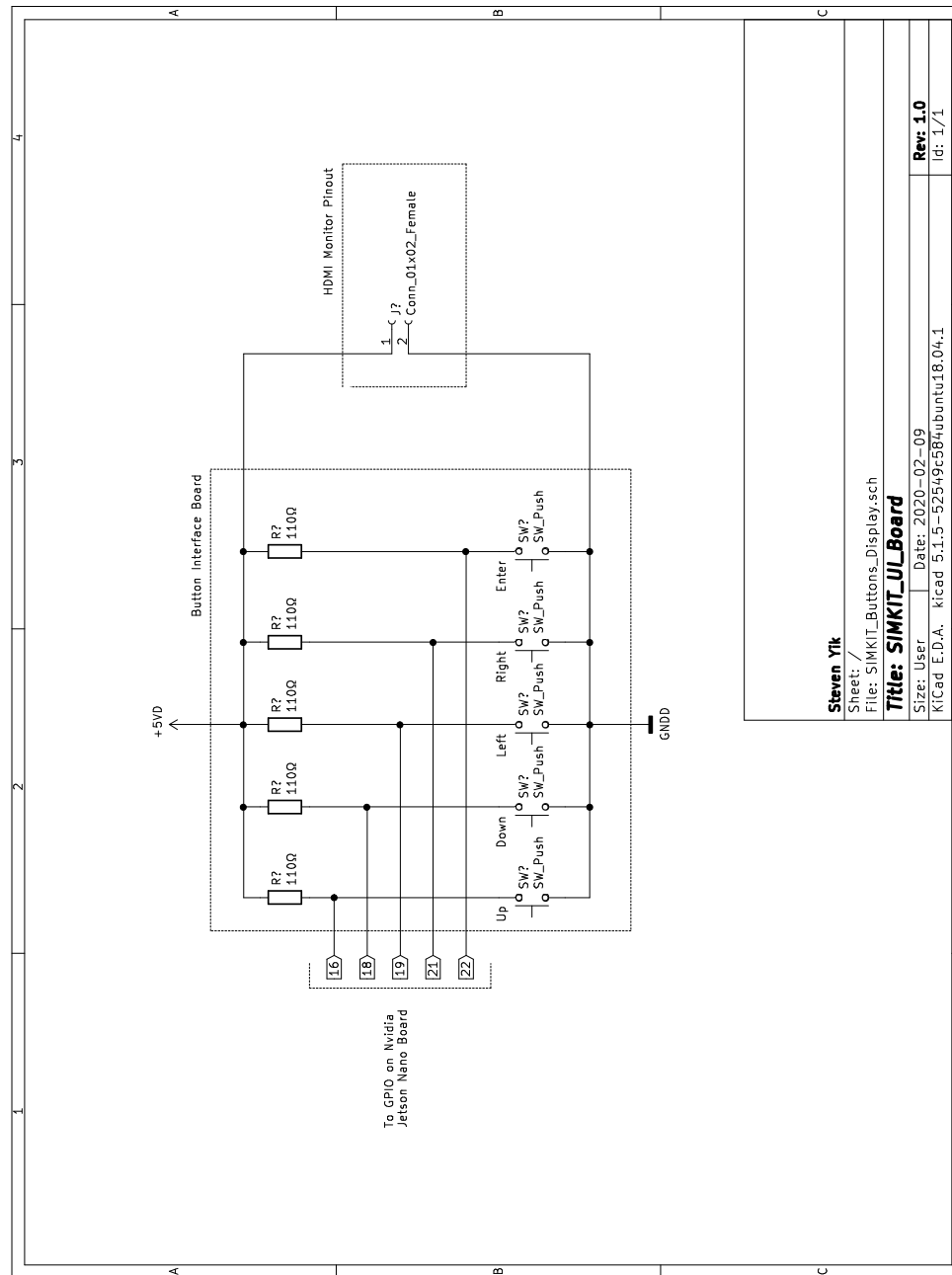
The USB 3.0 ports have a bandwidth limitation which may cause interruptions of power or data through initialization. System may work with 2 cameras in operation, but will always work with one plugged in.

### 5.2 Software

There may be issues with folder creation on the external SSDs. Make sure the external device is mounted and named correctly within the linux fstab table.

## Appendix

### A Electrical Hardware Schematic for Buttons and Display



## **APPENDIX B**

### **SIMKIT - SOFTWARE**

#### **B.1 Instruction Manual**

# Quick SimKit Setup Instructions

## Required Hardware

- 1x SimKit Housing + Computer + Cameras
- 1x Samsung Portable Hard Drive
- 1x Extension Cable

## Setup Instructions

- 1 Take the Samsung Portable Hard Drive and plug it into the SimKit via USB-C cable located on the side of the box, indicated in the picture. (Please send hard drives out for processing of data every 1-3 weeks, with recordings bi-weekly. These can be swapped out with specific drives sent from the research team.)



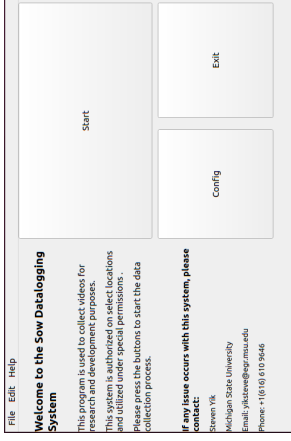
- 2 Make sure the hard drive is securely attached to the SimKit by inserting it inside the coil of wires or taping it down.
- 3 Lightly unscrew the wingnuts located on the backside of the SimKit on vertical support shafts.
- 4 Carefully attach the SimKit to the ceiling along the electrical conduit as a mounting

point, between 2 lighting fixtures, with the metal clamp on the support shaft and tightening the wingnuts to clamp down. See image for reference:



**NOTE:** Place side with power cable closest to the door.

- 5 Plug in power by utilizing the Extension Cable. If necessary, place the power cable above the door, NOT in the hinge portion.
- 6 The computer should automatically boot to the software program after a minute. If not, proceed to power cycle the computer.
- 7 The loaded main menu application looks like the image below. Please use the arrow keys on the box to navigate the menus and the square button to confirm an action.



- 8 Press the **Config** button to adjust the cameras by looking at the display to ensure the image sees the hallway and is relatively centered and level. Proceed to do the same with *camera2*.
- 9 Navigate back the main menu and hover over the **Start** button. Right before sending sows, press the square button to start the data collection process. You can confirm it is operational if the button turns **Green**.
- 10 Once the movement of sows is complete, press the square button once more the stop the program, the button should be **Red**.
- 11 To take down, make sure the button is red (step 10) and unplugging the power cable. Unscrew the wingnuts on the mounting bracket and store. (Send hard drives and repeat steps 1 & 2 for assembly procedure)

## Troubleshooting

- Blue/Black Screen - Reboot the system
- Error Messages - Read the message and follow instructions on the screen
- Does not power on - Please contact *Steven Yik*

Steven Yik

yiksteve@egr.msu.edu | +1(616)610-9646



## **BIBLIOGRAPHY**

## BIBLIOGRAPHY

- [1] JI Alawneh et al. “The effect of clinical lameness on liveweight in a seasonally calving, pasture-fed dairy herd”. In: *Journal of dairy science* 95.2 (2012), pp. 663–669.
- [2] Sukumarannair S Anil, Leena Anil, and John Deen. “Effect of lameness on sow longevity”. In: *Journal of the American Veterinary Medical Association* 235.6 (2009), pp. 734–738.
- [3] Sukumarannair S Anil et al. “Association of inadequate feed intake during lactation with removal of sows from the breeding herd”. In: *Journal of Swine Health and Production* 14.6 (2006), pp. 296–301.
- [4] John L Barron, David J Fleet, and Steven S Beauchemin. “Performance of optical flow techniques”. In: *International journal of computer vision* 12.1 (1994), pp. 43–77.
- [5] Madonna Benjamin and Steven Yik. “Precision livestock farming in swine welfare: a review for swine practitioners”. In: *Animals* 9.4 (2019), p. 133.
- [6] Daniel Berckmans. “Precision livestock farming technologies for welfare management in intensive livestock systems”. In: *Scientific and Technical Review of the Office International des Epizooties* 33.1 (2014), pp. 189–196.
- [7] National Pork Board. “Safe Pork Handling: Pig Behavior, Handling & Fitness.” In: (2014). URL: <https://www.secura.net/secura-erater-vm/pdf-files/prevention-connection/farm--ag/safety-around-farm-animals/pig-behavior-handbook-english.pdf>.
- [8] Magnus Campler et al. “Rubber mat placement in a farrowing and lactation facility: tips and techniques”. In: *Journal of Swine Health and Production* 24.3 (2016), pp. 142–146.
- [9] EJ Clowes et al. “Parturition body size and body protein loss during lactation influence performance during lactation and ovarian function at weaning in first-parity sows”. In: *Journal of Animal Science* 81.6 (2003), pp. 1517–1528.
- [10] V Courboulay and C Foubert. “Testing different methods to evaluate pig welfare on farm”. In: *Animal Welfare* 16.2 (2007), pp. 193–196.
- [11] A De Montis et al. “Encyclopedia of Big Data”. In: (2017). doi: <http://dx.doi.org/10.1007/978-3-319-32001-4>.
- [12] J Deen et al. “Feet first from zinpro: lesion scoring guide”. In: *Zinpro Corporation, Eden Prairie, MN, USA* (2009).
- [13] T Gillespie. *Analysis of sow mortality using big data from PigCHAMP*. 2019.
- [14] Tasha R Gruhot et al. “An economic analysis of sow retention in a United States breed-to-wean system”. In: *Journal of Swine Health and Production* 25.5 (2017), pp. 238–246.

- [15] Simon Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.
- [16] Berthold KP Horn and Brian G Schunck. “Determining optical flow”. In: *Techniques and Applications of Image Understanding*. Vol. 281. International Society for Optics and Photonics. 1981, pp. 319–331.
- [17] Miso Ju et al. “A kinect-based segmentation of touching-pigs for real-time monitoring”. In: *Sensors* 18.6 (2018), p. 1746.
- [18] Mark Knauer. “SOW BODY CONDITION FOR BREEDING SUCCESS”. In: *LONDON SWINE CONFERENCE*. 2018, p. 3.
- [19] Mark Thomas Knauer and David John Baitinger. “The sow body condition caliper”. In: *Applied engineering in agriculture* 31.2 (2015), pp. 175–178.
- [20] Mark Knauer et al. “Physical conditions of cull sows associated with on-farm production records”. In: *Open Journal of Veterinary Medicine* 2.3 (2012), p. 137.
- [21] M Knauer et al. “A descriptive survey of lesions from cull sows harvested at two Midwestern US facilities”. In: *Preventive Veterinary Medicine* 82.3-4 (2007), pp. 198–212.
- [22] Jorgen Kongsro. “Development of a computer vision system to monitor pig locomotion”. In: (2013).
- [23] *Lameness*. URL: <https://www.merriam-webster.com/dictionary/lameness>.
- [24] MS Lund et al. “Detection of quantitative trait loci in Danish Holstein cattle affecting clinical mastitis, somatic cell score, udder conformation traits, and assessment of associated effects on milk yield”. In: *Journal of dairy science* 91.10 (2008), pp. 4028–4036.
- [25] DCJ Main et al. *Repeatability of a lameness scoring system for finishing pigs*. 2000.
- [26] Suresh Neethirajan. “Recent advances in wearable sensors for animal health management”. In: *Sensing and Bio-Sensing Research* 12 (2017), pp. 15–29.
- [27] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked hourglass networks for human pose estimation”. In: *European conference on computer vision*. Springer. 2016, pp. 483–499.
- [28] M Pairis-Garcia and CJ Rademacher. “The Common Swine Industry Audit: Future steps to assure positive on-farm animal welfare utilizing validated, repeatable, and feasible animal-based measures”. In: *Journal of Animal Science* 94 (2016), pp. 45–45.
- [29] Andrea Pezzuolo et al. “On-barn pig weight estimation based on body measurements by a Kinect v1 depth camera”. In: *Computers and Electronics in Agriculture* 148 (2018), pp. 29–36.
- [30] K M Pierdon and D T Parsons. “eliability of a four point sow lameness score across multiple observers”. In: *eliability of a four point sow lameness score across multiple observers*. 2015, p. 57.

- [31] Eric T Psota et al. “Multi-pig part detection and association with a fully-convolutional network”. In: *Sensors* 19.4 (2019), p. 852.
- [32] LV Randall et al. “Low body condition predisposes cattle to lameness: An 8-year study of one dairy herd”. In: *Journal of Dairy Science* 98.6 (2015), pp. 3766–3777.
- [33] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.
- [34] Marie Rhodin et al. “Vertical movement symmetry of the withers in horses with induced forelimb and hindlimb lameness at trot”. In: *Equine veterinary journal* 50.6 (2018), pp. 818–824.
- [35] David E Rumelhart, Geoffrey E Hinton, James L McClelland, et al. “A general framework for parallel distributed processing”. In: *Parallel distributed processing: Explorations in the microstructure of cognition* 1.45-76 (1986), p. 26.
- [36] José C Segura-Correa et al. “Frequency of removal reasons of sows in Southeastern Mexico”. In: *Tropical animal health and production* 43.8 (2011), pp. 1583–1588.
- [37] DJ et al Sprecher, DE Hostetler, and JB Kaneene. “A lameness scoring system that uses posture and gait to predict dairy cattle reproductive performance”. In: *Theriogenology* 47.6 (1997), pp. 1179–1187.
- [38] Sophia Stavrakakis et al. “Validity of the Microsoft Kinect sensor for assessment of normal walking patterns in pigs”. In: *Computers and Electronics in Agriculture* 117 (2015), pp. 1–7.
- [39] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Thirty-first AAAI conference on artificial intelligence*. 2017.
- [40] Tijmen Tieleman and Geoffrey Hinton. “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude”. In: *COURSERA: Neural networks for machine learning* 4.2 (2012), pp. 26–31.
- [41] Statistics USDA. Economics and Market Information System. *USDA*. 2019.
- [42] Ke Wang et al. “A portable and automatic Xtion-based measurement system for pig body size”. In: *Computers and Electronics in Agriculture* 148 (2018), pp. 291–298.
- [43] DM Weary, JM Huzzey, and MAG Von Keyserlingk. “Board-invited review: Using behavior to predict and identify ill health in animals”. In: *Journal of animal science* 87.2 (2009), pp. 770–777.
- [44] Philippe Weinzaepfel et al. “DeepFlow: Large displacement optical flow with deep matching”. In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1385–1392.

- [45] Zoran Zivkovic. “Improved adaptive Gaussian mixture model for background subtraction”. In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004*. Vol. 2. IEEE. 2004, pp. 28–31.
- [46] Zoran Zivkovic and Ferdinand Van Der Heijden. “Efficient adaptive density estimation per image pixel for the task of background subtraction”. In: *Pattern recognition letters* 27.7 (2006), pp. 773–780.