A spatial AI-based agricultural robotic platform for collision avoidance and disease detection in crops

by

Sujith Gunturu

B.Tech., RVR and JC college of Engineering, 2017

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Major Professor
Dr. Arslan Munir

Copyright

Abstract

To derive more consistent measurements through the course of a wheat growing season, this thesis conceives and designs an autonomous robotic platform that perform collision avoidance and disease detection in crops using spatial artificial intelligence (AI). This thesis demonstrates the working of the proposed robotic platform on wheat crop. The main constraint the agronomists have in breeding trials is to not run over the wheat while driving. This limits the freedom of the robot to freely navigate through the wheat field. To overcome this hurdle, we have trained a spatial deep learning model that help navigate the robot freely in the field while avoiding collisions with wheat. To train this model, we have used publicly available databases of prelabeled images of wheat along with the images of wheat that we have collected in the field. We have used YOLO (You Only Look Once) as our deep learning model to detect wheat. Faster R-CNN (Faster-Region-based Convolutional Neural Network) with ResNet-50-FPN (Residual Neural Network-50 Feature Pyramid Network) as backbone is also used to compare the accuracy of YOLO. This allowed 1-3 frames per second (fps) vision for wheat detection in the field. With the robot driving between 2-5 miles per hour in the field, this frame rate of 1-3 fps would only allow the robot to detect its surroundings once every foot or so. To increase the frame rate for real-time robot response to field environments the previous images were used to train the MobileNet single shot detector (SSD) and a new camera, the Luxonis Depth AI Camera, was used for inference in the field. Together the newly trained model and camera could achieve a frame rate of 18-23 fps, fast enough for the robot to sense its surroundings once every 2-3 inches of driving.

Following the discussion of sensing, the autonomous navigation of the robot is next addressed. The new camera allows the robot to determine the distance to sensed objects by using

a stereo camera also embedded with the main AI camera. The stereo camera allows the model to determine the distance of the robot from a particular object. By knowing appropriate distances an algorithm can be written to have the robot maneuver more precisely within its surroundings in the field. When it detects an object in the field the MobileNet SSD sends a binary thresholded distance signal to the robot motion controller. The motion controller can then use this information to make decisions, such as continuing motion, steering, or stopping. With an intention to improve the range of potential robot applications, a classification model is also installed on the robot to locate two of the most common diseases in wheat namely, stem rust and wheat rust. Making it more reasonable for agronomists to find these diseases in a large field.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my professor, mentor, and supervisor Dr. Stephen Welch for his support throughout my graduate studies. I admire him for his knowledge, patience, and clarity of thought. I have learned a lot from him while working with him on his research project.

I thank my major professor, Dr. Arslan Munir, for mentorship for the last two years in course selection. He is one of the best teachers I have studied under. His continuous feedback and help on the thesis put me in the right direction.

I thank Dr. Dan Flippo for allowing me to work on this project. I thank Dr. Lior Shamir for dedicating his valuable time to serve on my committee.

I Would like to thank my colleague Calvin Dahms for this project idea and for using my deep learning model on his robot. I thank my mother, Madhulatha, my father, Sridhar for making my dream of studying in the United States come true. Special thanks to Divya Vani Lakkireddy for her support both personally and professionally. Finally, I would like to thank my mentor Dr. Kaliramesh Siliveru, my friends: Niketa Penumajji, Ramesh Babu Dilli, Samhitha Jammuladinne, Venkat Surya Dasari, Rahul Chandra Karne, Naveen Jonnalagadda, Manoj Pulivarthi, Ramya Kalam, Sowmya Punati, Neha Yerasmus, Rakshith Kumar, Rahulharsha Cheppally for their presence during my ups and downs.

I would also like to thank the National Science Foundation EPSCoR Project entitled “RII Track-2 FEC: Building Field-Based Ecophysiological Genome-to-Phenome Prediction” (Award #1826820) for funding this research.

Dedication

This work is wholeheartedly dedicated to Dr. Stephen Welch for his immense support throughout my 2 years of graduate studies.

Preface

The basis of this research stemmed from my passion for developing better computer vision models for high precision robots used in the Department of Biological and Agricultural Engineering of Kansas State University. As the world is developing more and more machine-oriented, there will be greater demand for autonomous robots, and artificial intelligence.

From the discussion of this thesis, I believe that readers will better understand the various terms in deep learning and methods we used for data collection along with our less than successful attempts that we surmounted while achieving the accuracy and speed the application required.

Chapter 1 - Introduction

Wheat (*Triticum*) is one of the most important staple foods in the temperate world and with the United States producing 8% of the world's total production [1][2][3]. So, there is an absolute need to conduct research on growth of wheat, the properties of the soil in which it is grown, as well as any diseases present. This will give agronomists better ways to predict wheat traits based on its genetics, and help farmers to plan the planting dates. One of the most important parts of wheat research in the state of Kansas is to get soil properties of fields where research wheat is grown. These soil properties are obtained by a Gem 2 electromagnetic induction, EMI, Sensor weighing over 15 lbs., which typically needs to be carried by a human. Ideally, fields should be scanned with such sensors multiple times per week but this is infeasible using human effort alone, A field of size of 50m x 75m would need an effort of 4 hours per day to get the soil properties at specific intervals of time. To overcome this problem, sensor robots are being developed to carry the sensor through a field either operated by a human via a remote control or, ultimately, in a fully autonomous mode. A remote-controlled robot saves the time and energy of agronomists who are trying to obtain the soil properties. Though it seems quite easy to control the robot, one of the most important rules in any wheat plot research is not to damage the crop while operating the robot. As the field size increases, it becomes harder for a human to control it from afar and there is an unacceptable possibility that robot might run over the wheat. One intelligent approach to make these robots not to run over the wheat is to make the robot recognize and stop when it is unable to turn its wheels in time. If the robot detects wheat head far from its location, it is not necessary to stop the robot; instead the robot's wheels can be realigned to steer the robot and avoid a collision (autonomous navigation). Here, the use of spatial AI comes into picture, spatial AI is an ability of an AI system to reason based on not just what it is looking at but also how far away things are

located. When the distance of the detected wheat is lower than the permitted distance, the robot makes an informed decision and makes its velocity equal to 0. Later, the operator can align its wheels in whichever way necessary and continue to perform the research (getting soil properties in this case).

The main contributions of this thesis are:

- Design of a robotic platform that can navigate in crops while avoiding collisions using spatial AI. The platform uses depth/stereo sensing and deep learning models to accomplish collision avoidance and crop (wheat) detection.
- Disease detection and localization in crops (focusing on wheat) using deep learning.

Collision avoidance is achieved by training neural networks such as YOLO, Faster R-CNN with ResNet-50 Feature Pyramid Network (FPN) backbone to detect objects and obtain depth by spatial AI devices such as OpenCV AI camera.

Identifying the diseases present in crops is one of the most important parts of crop research. In this thesis, a classification algorithm is also trained, so that the robot identifies whether the wheat is a healthy wheat, or it suffers from stem or leaf rust. The performance of wheat detection is tested using a few infield images taken by a digital camera. The performance of the disease classification model such as precision, recall, and accuracy are tested using the same images which were taken to test the wheat detection model.

While using deep learning models in real-time, speed and/or throughput are equally important as accuracy. This work also evaluates the performance of the deep learning model on different platforms.

Furthermore, various evaluation metrics like accuracy, precision, and recall are discussed for both deep learning models (i.e., wheat detection models, and disease classification models).

This thesis is organized as follows. Chapter 2 summarizes the previous work related to object detection and deep learning mainly focusing on convolutional neural networks that are used throughout this thesis. Chapter 3 gives various sources of images, augmentation techniques, software, and hardware used, how training images on YOLO, Faster R-CNN with ResNet-50-FPN backbone is done, and finally a mechanism to stop the robot after estimating depth. Chapter 4 gives how a classification model is trained on ResNet-18 and EfficietNet-B4 to classify the images of wheat into three classes which are stem rust, wheat rust, and healthy wheat. Chapter 5 gives further research directions towards building autonomous navigation. Finally, chapter 6 gives various sources of references that are used to obtain information and knowledge.

Chapter 2 - Related Work

Deep learning has played an important role in various fields, such as biology, medicine, agriculture, agronomy, etc. This section discusses previous works in the literature related to computer vision and agriculture.

Redmon et al. [4] presented YOLO (You Only Look Once), a new approach in object detection. YOLO treats object detection as a regression problem which is the main difference between YOLO prior object detection models. YOLO is very fast and processes images at a higher rate than other object detection approaches and gives the best performance in real-time object detection. As illustrated in Redmon et al. [4], YOLO reasons globally about the image when making predictions, and learns generalizable representations of objects. Results in [4] reveal that YOLO produces only half the number of background errors as compared to fast R-CNN. These reasons attracted us to use YOLO as our deep learning model to detect wheat.

Inspired by the work of Ren et al. [6], we have also implemented Faster R-CNN with an FPN backbone for real-time wheat detection in the field. Faster R-CNN achieved state-of-the-art object detection accuracy on the PASCAL VOC 2007, PASCAL VOC 2012 [22], and MS COCO [23] datasets with only 300 proposals per image. These exceptional qualities have attracted us to use this model in our study. Faster R-CNN and Region Proposal Network (RPN) have been used multiple times by entries in image object detection competitions [7]. It has also been observed that this method is not only a cost-efficient solution for practical usage but also an effective way of improving object detection accuracy. Briefly, Faster R-CNN is composed of two modules. The first module is a deep, fully convolutional network that proposes regions, and the second module is the fast R-CNN detector that examines the proposed regions [5]. In this thesis, Faster R-CNN with ResNet-50-FPN [21] is compared the accuracy of YOLO to decide which is best for use in

the final implementation. Both Faster R-CNN and YOLO had almost equal accuracy, but YOLO is faster than Faster R-CNN. A more detailed discussion about the accuracies of the two models is presented in Chapter 3.

Liu et al. [8] proposed a new "single shot detector" (SSD) approach, which discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. They observed that the SSD model is relatively simple to train. The SSD model eliminates proposal generation and subsequent pixel or feature resampling stage and encapsulates all computations in a single network.

The work done by He et al. [9] is most relevant to our requirement and study. They have presented a residual learning framework to ease the training of networks that are substantially deeper than those used in other state-of-the-art models. The extremely deep representations also have a good generalization performance on other recognition tasks that led them to win First Place in the ImageNet [15] detection and localization, along with the COCO [23] detection competitions. This state-of-the-art performance inspired us to adopt ResNet to solve the problem of disease detection in wheat fields.

Amara et al. [11] proposed a deep learning-based approach to automate the process of classifying banana leaf diseases. The authors used the LeNet [12] architecture as a deep convolutional neural network to classify banana sigatoka and speckle. The results showed an accuracy of 98.61% with color images.

Chapter 3 - Wheat Detection and Robot Control Using Depth Sensing and Deep learning

This chapter provides a detailed workflow on training deep neural networks to detect objects and evaluates their performance on real images from the field and finally, gives a mechanism to obtain depth.

3.1 Data Collection Sites

The performance of any real-time deep learning model mainly depends on the nature of the dataset, which is to say, the quantity and quality of the images. The images for training the deep learning model are obtained from sites where the robots are used. The data were collected from the field at Ashland Bottoms KSU Agronomy Research Farm (latitude 39.128 to 39.1284; longitude -96.6157 to -96.6164) in the city of Manhattan, Kansas, United States.

Figure 3.1 Research site (KSU Agronomy farm) in Manhattan Kansas

The figure shown above gives a wide view of the agronomy field where wheat is grown for research purposes as a part of our EPSCoR project. More details about the collected images are discussed in the next section.

3.2 Camera and Data collection, Kaggle Data, and Google Images.

3.2.1 Images from KSU Agronomy farm:

Most of the images for training the deep learning model are from the KSU Agronomy Farm with a digital camera. The training images were taken so as to be like those images the robot would need process. One of the important requirements of the deep learning model is that the image recognition must be done in all stages for wheat, that is, the wheat color can be brown or green depending on the season or time of operation of the robotic platform. Considering this requirement, images of both green and brown wheat were collected.

Figure 3.2 A Sample image from the KSU Agronomy Farm when wheat is in the reproductive state

Figure 3.3 A sample image from the KSU Agronomy Farm when wheat is in the mature state

Figure 3.1 is one of the training images collected on June 4, 2021. This stage of wheat growth is called the reproductive stage. This is the stage where the wheat heads have fully emerged from the stem. Pollination is very quick and takes only 3 to 5 days to complete. Thus, the images are taken in quick succession forming a data set comprising of 1200 images of pollinated wheat with visible heads.

Figure 3.3 is one of the training images collected on June 22, 2021. This stage of the wheat is called “maturity”, or also “hard dough”. In this phase, the plant turns into a straw color and the kernel becomes very hard.

The purpose of collecting these images is so that they can be annotated for training. The process of annotation is discussed in the following sections.

3.2.2 Google Images

The Google Images search engine allowed us to get non-copyrighted images of wheat which are both green and brown. The images were manually scrutinized and downloaded into a folder for model training purposes. The images selected ranged from just a few heads to multiple heads. Priority was placed on images with good resolution and high clarity.

3.2.3 Kaggle

Kaggle, a subsidiary of Google LLC, is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish data sets, explore, and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges. More than 3000 wheat images are taken from Kaggle data to train the deep learning model. Some of these images were selected for and subjected to augmentation prior to their use in training. Augmentation methods are discussed below.

3.3 Hardware

3.3.1 Camera

The camera used for capturing images in the field is a commonly available digital camera from Samsung. The model number is WB35F.

3.3.2 LattePanda

LattePanda Alpha 864s is a high-performance, palm-sized board with low power consumption that runs Windows 10. It is widely used in edge computing, vending, advertising machines, and

industrial automation. This palm-sized machine is molded onto the robot, which is used for running a deep learning model, communicating with the operator, and communicating with the robot's integral components. Key features of LattePanda Alpha are

- Intel Core M3-8100Y, Dual-core, 1.1 - 3.4 GHz
- Intel UHD Graphics 615
- 8 GB Memory
- Integrated Arduino coprocessor ATMEL 32U4

3.3.3 Intel Neural Compute Stick (NCS)

Intel Neural Compute Stick is an accelerator which can support central processing unit (CPU), graphics processing unit (GPU), and a field programmable gate array (FPFA). A neural stick is used to accelerate the deep learning by which the model recognizes objects at many frames per second. A few key features of neural compute stick are:

- **Software:** Intel Distribution of OpenVINO toolkit.
- **Supported operating systems:** Windows, Mac, Ubuntu, etc.
- **Hardware (NCS)**
    1. Processor: Intel Movidius Myriad X Vision Processing Unit (VPU)
    2. Supported frameworks: TensorFlow, Caffe, Apache MXNet, Open Neural Network Exchange (ONNX), PyTorch, and PaddlePaddle via ONNX conversion.

3.3.4 GPU

GPU used for all training purposes in this work is from Google Collaboratory using Colab pro subscription. Google Collaboratory provides Nvidia Tesla K80 GPUs that have a dual-GPU design with 4992 CUDA core (2496 CUDA cores per GPU). Nvidia Tesla K80 has a 24 GB of GDDR5 memory and has a PCI Express (PCIe) interface.

3.3.5 OpenCV AI kit

OpenCV AI Kit with Depth (OAK-D) is a spatial AI powerhouse, capable of simultaneously running advanced neural networks while providing depth from two stereo cameras one on the left and one on the right, and color information from a single 4K camera in the center.

Figure 3.4 Stereo/RGB cameras from OpenCV AI kit

The two outer sensors are for the left and right views forming a stereo pair. The center sensor provides the RGB image used for object detection.

A requirement to use this device is that models must be loaded into it in the form of a blob. This required a formatting conversion step described in Section 3.11.1.

3.4 Preprocessing Steps

3.4.1 Annotations

Data labeling or annotation is a primary step for training deep learning models. This section describes various types of annotations that facilitate deep learning training and provides details on annotations for YOLO.

Bounding Boxes: Bounding boxes are one of the most common annotation techniques used by machine learning practitioners. Rectangles are used to determine the location of the object in the image. A rectangle is defined by the (x, y) coordinates of the upper left corner lower-right corners that form the diagonal representation of a rectangle. One other way of representing bounding boxes is given an (x, y) on the top left, and height, and width of the bounding box**.**

Polygonal Segmentation**:** Because objects are not always rectangular complex polygons can be used to locate the object more precisely.

Semantic Segmentation: In this type of annotation technique where each pixel of an image is given a class, denoting one of the objects that need to be recognized. 3D cuboids: 3D cuboids are like bounding boxes, but it gives additional depth information. 3D cuboids can be used to determine the position and volume of an object.

Key-point and Landmark: In this technique detection of small objects is done by creating dots across the image.

Lines and Splines: Lines and splines are mainly used in autonomous vehicle navigation where the shoulder line, yellow line, and broken yellow lines are the lines that need to be recognized.

YOLO Format for Bounding boxes**:** YOLO uses bounding box annotation written on a .txt file with text filename same as the image. Each bounding box is defined with an object class

number, coordinates of the center, width, and height of the bounding box rectangle. The format of YOLO annotations is <object-class> <x><y><width><height>. The following numbers are the annotations for the image in Figure 3.5.

0 0.106445 0.014160 0.212891 0.028320.
0 0.690430 0.418457 0.138672 0.139648
0 0.206055 0.397461 0.320312 0.160156
0 0.346191 0.049316 0.297852 0.098633
0 0.141113 0.328613 0.282227 0.102539
0 0.846680 0.155762 0.306641 0.131836
0 0.904297 0.631836 0.191406 0.166016
0 0.821289 0.026367 0.154297 0.052734
0 0.975586 0.833496 0.048828 0.159180
0 0.759766 0.789551 0.103516 0.112305
0 0.423340 0.681641 0.379883 0.113281
0 0.582031 0.086914 0.152344 0.140625
0 0.804688 0.801270 0.076172 0.122070
0 0.830566 0.813965 0.083008 0.120117
0 0.949219 0.851562 0.046875 0.244141
0 0.067871 0.136719 0.135742 0.080078
0 0.234375 0.034180 0.232422 0.068359
0 0.354004 0.054199 0.250977 0.108398

Figure 3.5 Bounding boxes over the wheat heads

3.4.2 Augmentations

Image augmentation is the process of taking the images that are already present in the training dataset and manipulating them to create many altered versions of the same image. This provides more images to train and gives a different viewpoint to the classifier. Different viewpoints can represent changes in the saturation, color, crop, and horizontal and vertical flips. The aim of augmentation is to introduce variation whose effect will help avoid model over-fitting. To create augmented data, a PyTorch orientation module named Albumentations is used. Albumentations is a module where all the necessary augmentation steps for augmentations are provided in predefined functions. These functions are applied to the image dataset to create manipulated images and then used for training the deep learning model. A few examples of augmentations applied are SmallestMaxSize**:** Rescales an image so that the minimum size is equal to the given maximum size, keeping the aspect ratio of the initial image. ShiftScaleRotate**:** Randomly assigns transforms: translate, scale, rotate to the input, RandomCrop: Crops a random part of the image, RGBshift**:** Randomly shifts values for each channel of the input RGB image, and RandomBrightnessContrast: Randomly changes brightness and contrast of the input image.

A sample transformation after applying augmentation is shown in Figure 3.6 for an image in the training dataset.

(a) Smallest max size (b) Shift scale rotate (c) Random crop

(a) RGB shift limit (b) Random brightness contrast (c) Hue saturation

Figure 3.6 Augmentations applied to training images

Panel (a) is, for all practical purposes, the original image.

3.5 Training YOLO

The above sections 3.1 to 3.4 described data collection data, annotations, and augmentations. The final and the most crucial part of creating a real-time deep learning model is to train the model. YOLO performs supervised training for object detection. The training is done in Google Collaboratory with backend specs given in section 3.3.4. For every cycle of data collected from KSU Agronomy Farm, YOLO is trained on approximately 2000+ images for 7.679 hours of GPU time. The training is performed on pretrained weights, and the new images were added to the dataset to update the weights. The training and validation splits are given below in percentages.

Table 3.1 Partitioning different data sources

Image Source	Training	Validation	Testing
Kaggle	80	10	10
Google images	80	10	10
KSU Agronomy farm	70	10	20

3.5.1 Training Image Size, Batches, Epochs, Weights

Training is done by setting image size 1024x1024 and batch size 16 and 100 epochs on publicly available weights of large YOLO model.

3.5.2 Metrics

The following subsections describe the performance of the model as training progressed through multiple epochs. The Y-axis of each graph defines the respective metric and the X-axis of defines the epoch number.

3.5.2.1 Mean Average Precision(mAP)

The mAP for a set of detections is the mean of the interpolated AP for each class [25]. This per-class AP is given by the area under the precision/recall (PR) curve for the detections. mAP_0.5 in Figure 3.7 represents mAP at 0.5 IoU (Intersection over Union), mAP_0.5:0.95 represents mAP over different IoUs from 0.5 to 0.95 at step of 0.05.

3.5.2.2 Precision

Precision and Recall are measured to evaluate how well the model performs as explained in the following. Precision is the share of issued predictions that are predicted correctly. Precision is given by $true\ positives\ /\ (true\ positives\ +\ false\ positives)$ where true positives are correct predictions and false positives are number of issued predictions that are incorrect.

3.5.2.3 Recall

Recall is the share of the targets that are predicted correctly. Recall is given by $true\ positives/\ (true\ positives\ +\ false\ negatives)$. Where false negatives are the number of targets that should have been predicted but were missed. From Figure 3.7 we observe that precision, recall increase as the training progresses.

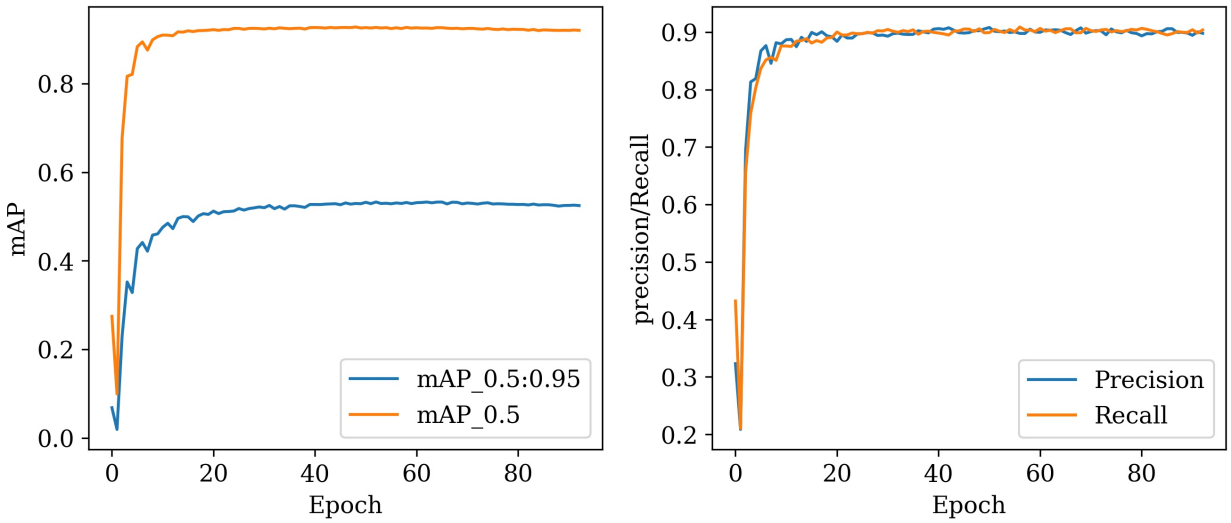

Figure 3.7 Mean average precision, recall, precision

3.5.2.4 Training and validation losses

Box Loss is a loss that measures how tight the predicted bounding boxes are to the ground truth object. **Object loss** is due to wrong-box object IoU prediction. **Class loss** is a loss that measures the correctness of the classification of each predicted bounding box.

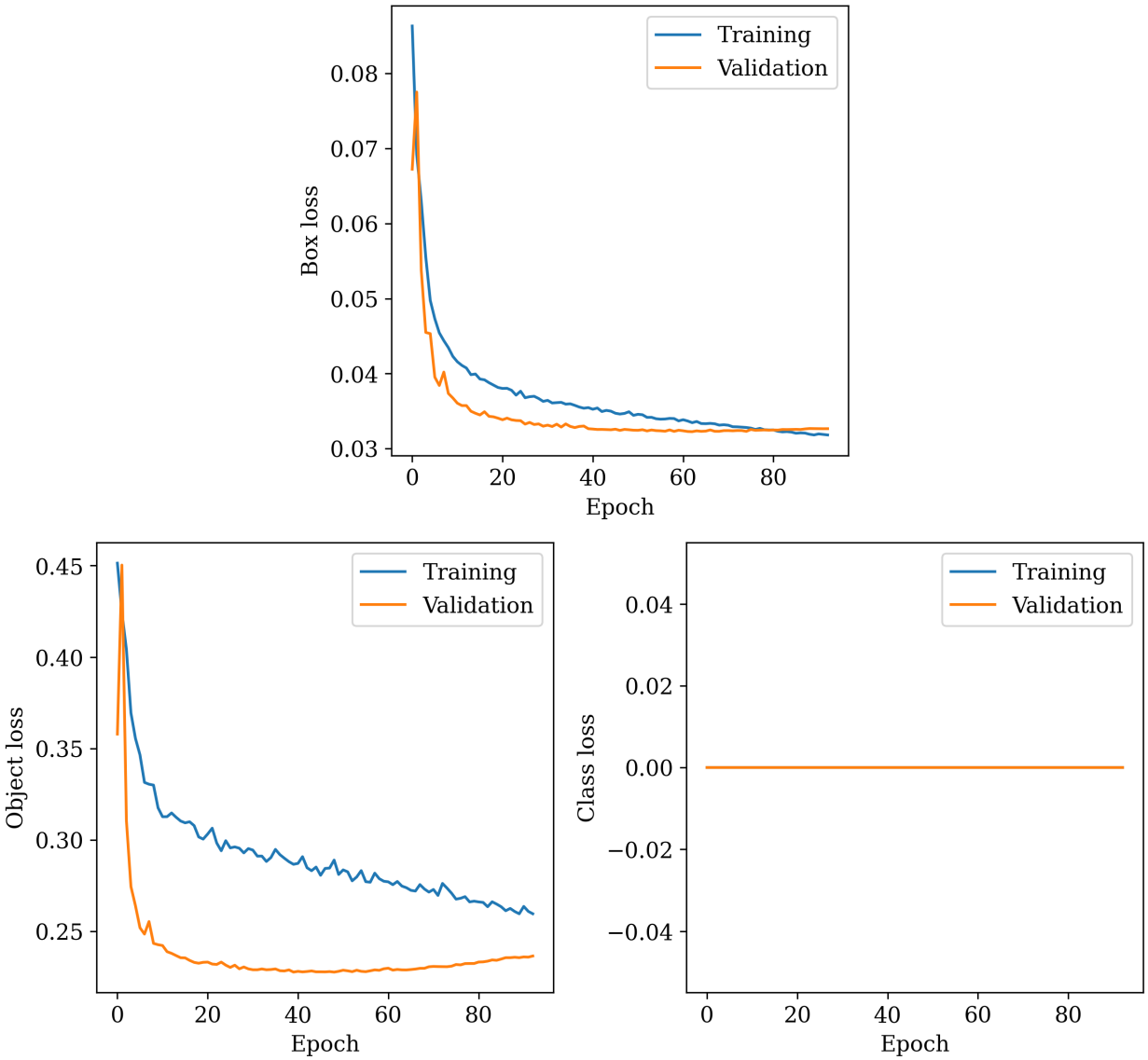

Figure 3.8 Losses during training and validation

From Figure 3.8, we observe that the training losses decrease as the training progresses, and we also observe that there is no increase in validation loss at any point in training which clearly shows that the model has not been overfitted.

3.6 Testing YOLO

To test the performance of YOLO, the model is run on a few images from the dataset. The performance of the model is defined by using the formula.

$$Performance = \frac{Number\ of\ wheat\ heads\ detected}{Total\ number\ of\ wheat\ heads\ present} \qquad \textbf{Equation 3.1}$$

The model achieved a performance of 0.93 on the test dataset. The weights of this model are saved for future training and inference in the field.

3.7 Training and Testing Faster R-CNN with ResNet-50-FPN

The speed of a deep learning model is as important as the accuracy of the model. In anticipation to achieve faster and accurate results, a Faster R-CNN model with ResNet-50-FPN backbone is trained on the collected dataset. This section briefly describes how training and testing of the model are performed.

3.8 Training Faster R-CNN with ResNet-50-FPN

The optimizer used in training this model was stochastic gradient descent (SGD). The learning rate (LR) scheduler used in training this model was step LR. The loss function used in Faster R-CNN is binary cross-entropy in the first state of RPN; in the second stage classification the loss function is normal cross-entropy [6]. Training is performed in Google Collaboratory. The initial training of Faster R-CNN with ResNet-50-FPN took 8 hours of GPU time for 2000+ images.

Object loss comparison between YOLO and Faster R-CNN with ResNet-50-FPN while training is shown below.

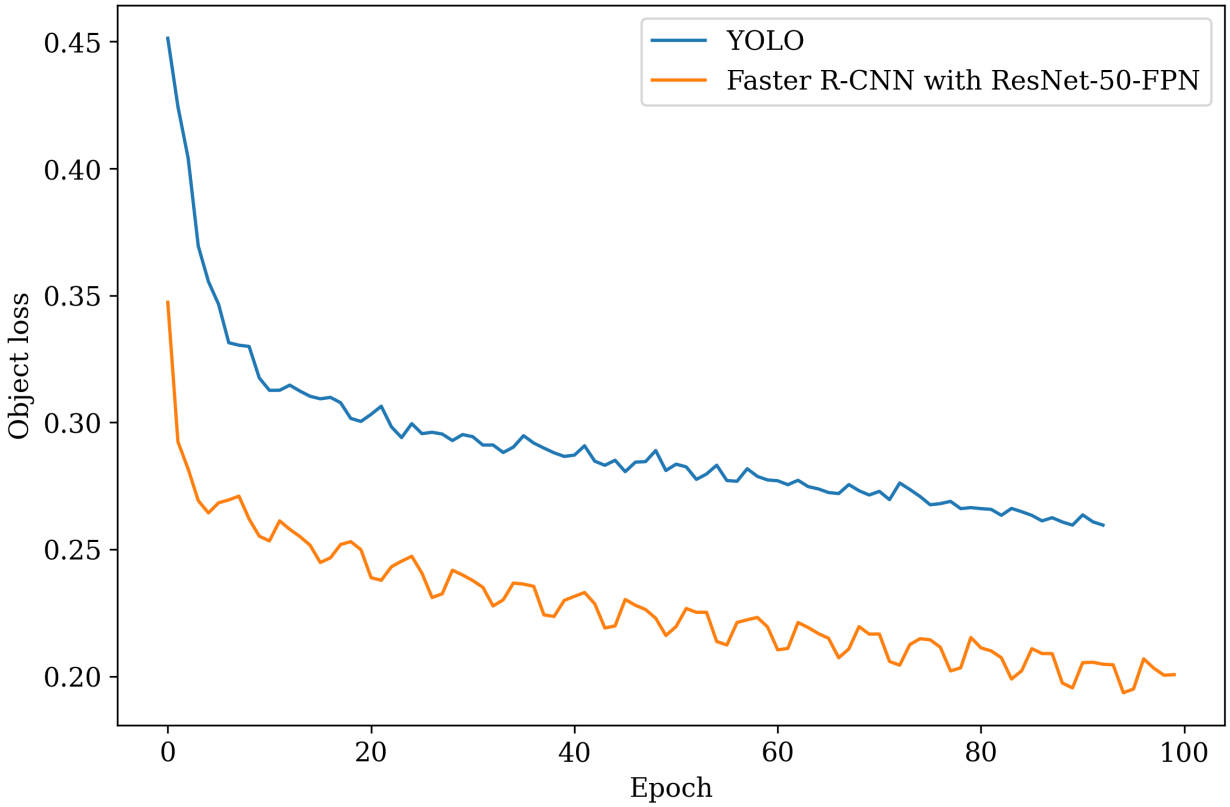

Figure 3.9 Object loss for YOLO and Faster R-CNN with ResNet-50-FPN

3.9 Faster R-CNN Testing

Faster R-CNN with ResNet-50-FPN achieved a performance of 0.90 in identifying the wheat heads. The performance of YOLO is 3% higher compared to Faster R-CNN with ResNet-50-FPN. The metric for performance is given in Equation 3.1. Testing is done on the LattePanda which is the actual embedded computing platform used for real-time detection in the field.

3.10 Comparing Faster R-CNN with YOLO

After training all the images on both YOLO and Faster R-CNN with ResNet-50-FPN, Both the models have achieved almost similar results in performance. YOLO is three times faster than Faster R-CNN with ResNet-50-FPN. Figure 3.10 shows time taken (seconds) by two trained deep learning models to perform inference on various numbers of images with image size 640 x640

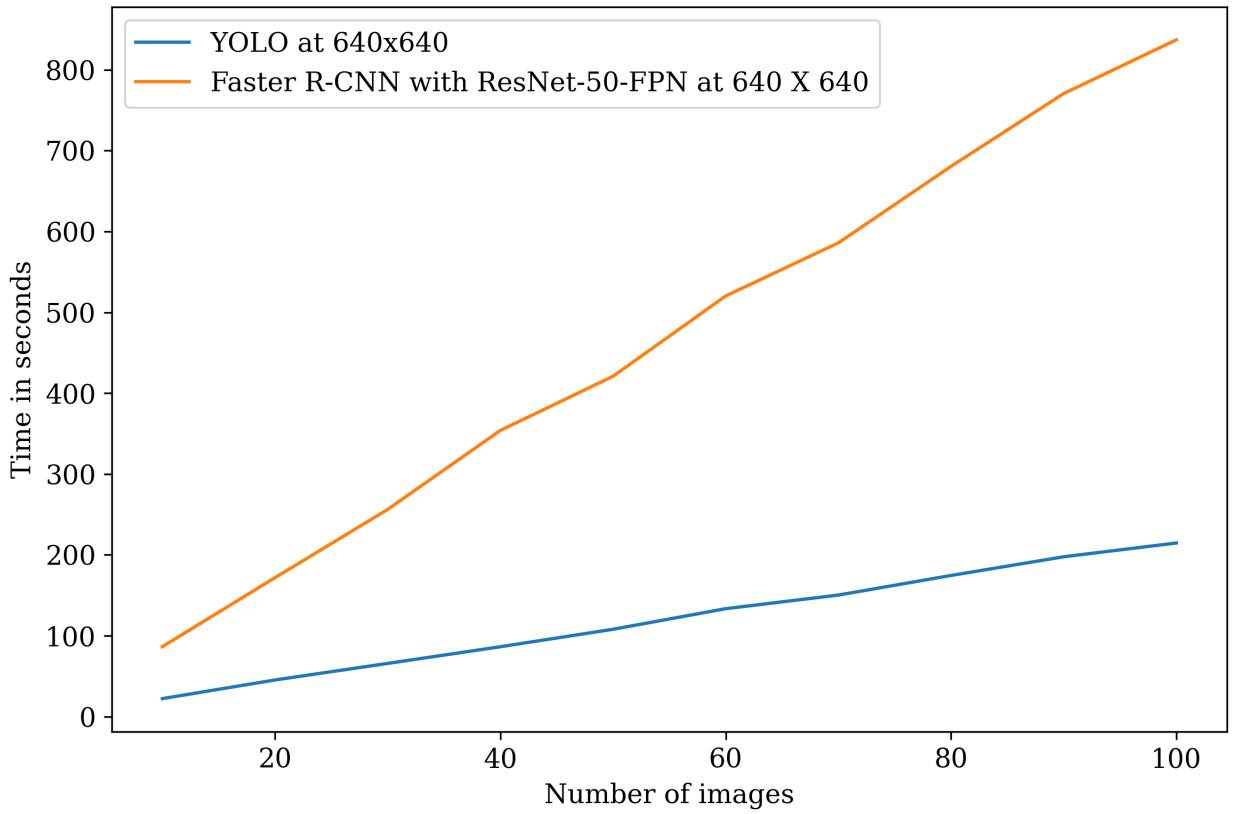

Figure 3.10 Timing comparison of YOLO and Faster R-CNN with ResNet-50-FPN on LattePanda

3.11 Running Deep Learning Model on Intel Neural Compute Stick

As described in the above section 3.4, an Intel Neural Compute stick is used to speed up deep learning frameworks. The Intel Neural Compute Stick works on the openVINO model, Open

VINO is a toolkit that enables running deep learning models on various Intel devices. The weights obtained from training YOLO, Faster R-CNN with ResNet-50-FPN are converted into openVINO model using the compilation framework provided by Intel as shown in Figure 3.1. Frames are captured from OpenCV video capture and inference is then performed on Intel Neural Compute Stick. The model weights and topology are, respectively, described with .bin and .xml files. (If more than one .xml file is required to specify topology a third, .mapping file us used to connect them.) Often inference of deep learning models does not require high-end GPUs and CPUs, and can be done on relatively less expensive CPUs, GPUs, or even a USB stick. However, in this case, the speed of the model increased from 20% to 30% for both deep learning models when the Neural Compute Stick was used.

3.11.1 Workflow for converting into blob

As noted above, this step is necessary in order to use the OpenCV AI kit. The compilation steps are shown in Figure 3.11. ONNX (Open Neural Network Exchange) is a standardized format for representing the weights and other information used in various machine learning models. At the end of the pipeline a blob (binary large object) is a heterogenous collection of diverse binary data stored as a single entity. Blobs are widely used as interfacing methods. In this pipeline the intermediate representation is an ad hoc extraction of the ONNX information designed to facilitate its conversion to the final blob. The first three steps in this pipeline were coded using open source modules available in Python. The last step (also programmed in Python) was done via an online API call to software provided by Luxonis.com [34].

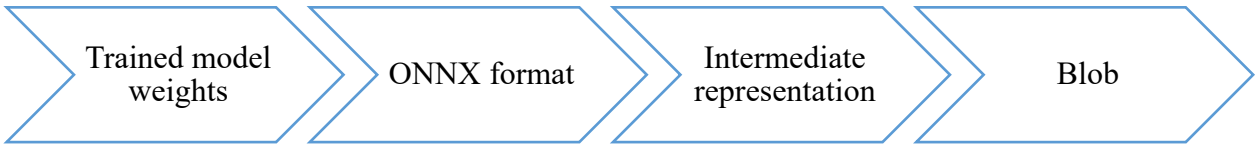

Figure 3.11 Converting into blob

3.12 Depth Sensing

The above sections have described how the object detection is achieved. But the final decision on when to stop the robot is most properly based on the distance between wheat and the robot. Hence, the depth information becomes crucial and is obtained using the stereo camera in the OpenCV AI kit.

3.13 Moving to MobileNet SSD for Inference while Depth Sensing

When the robot is running on the wheat field the size of an image and the view area of the robot can be varied depending upon the needs. After experimenting with advanced deep learning architectures, it is observed that SSD network with MobileNet backbone with an input image size 300x300 gives a higher performance as compared to YOLO and Faster R-CNN with ResNet-50-FPN. Hence, the blob which is obtained from MobileNet SSD is used in the field robot. We have saved the blob and weights from trained YOLO and Faster R-CNN with ResNet-50-FPN models for further research.

3.14 Communicating with the Robot Using PySerial

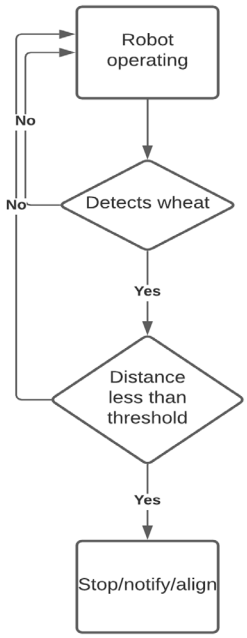

Figure 3.12 Communicating depth information

To stop the robot when it detects wheat at less than the threshold distance, a connection needs to be established between the robot and the Latte Panda board. As discussed earlier the LattePanda has a USB port, which is connected to the Arduino of the robot. When the deep learning model detects wheat, it communicates this fact to the Arduino, which then sends a signal to the robot's speed controller to adjust the speed. A flowchart of this process is shown in Figure 3.12 and the related code snippet is in Figure 3.13. This code snippet is embedded into depth-sensing code for communicating with the robot. In the code "conf" represents the confidence of object detection which can be varied depending on the performance of the model.

```
import os, serial, time, sys

def write_read(x):
    arduino = serial.Serial(port ='COM6',baudrate = 9600,timeout=5)
    arduino.write(bytes(x,'utf-8'))
    time.sleep(.05)
    data = arduino.readline()
    arduino.close()
    return data
wheatheads_count = 0

if conf.item() > 0.90 and distance < THESHOLD_DISTANCE:
    wheatheads_count += 1
    if wheatheads_count > 1:
        num = "H"
        value = write_read(num)
        wheatheads_count = 0
        option = input("enter L to start AND Q to quit")
        if option == "Q":
            num = "L"
            value = write_read(num)
            sys.exit()
        value = write_read(option)
```

Figure 3.13 Code snippet to perform collision avoidance

3.15 Results

The performance of object detection and depth-sensing was tested in the KSU Agronomy Farm where the robot is deployed. The results of object detection are shown in Figure 3.14 and Figure 3.15. The effectiveness of the recognition can be increased by varying the threshold/confidence level of the model.

Figure 3.14 Real-time detections of wheat heads in the field

The depth map of the wheat field from the point of view of the robot is shown in Figure 3.15 below.

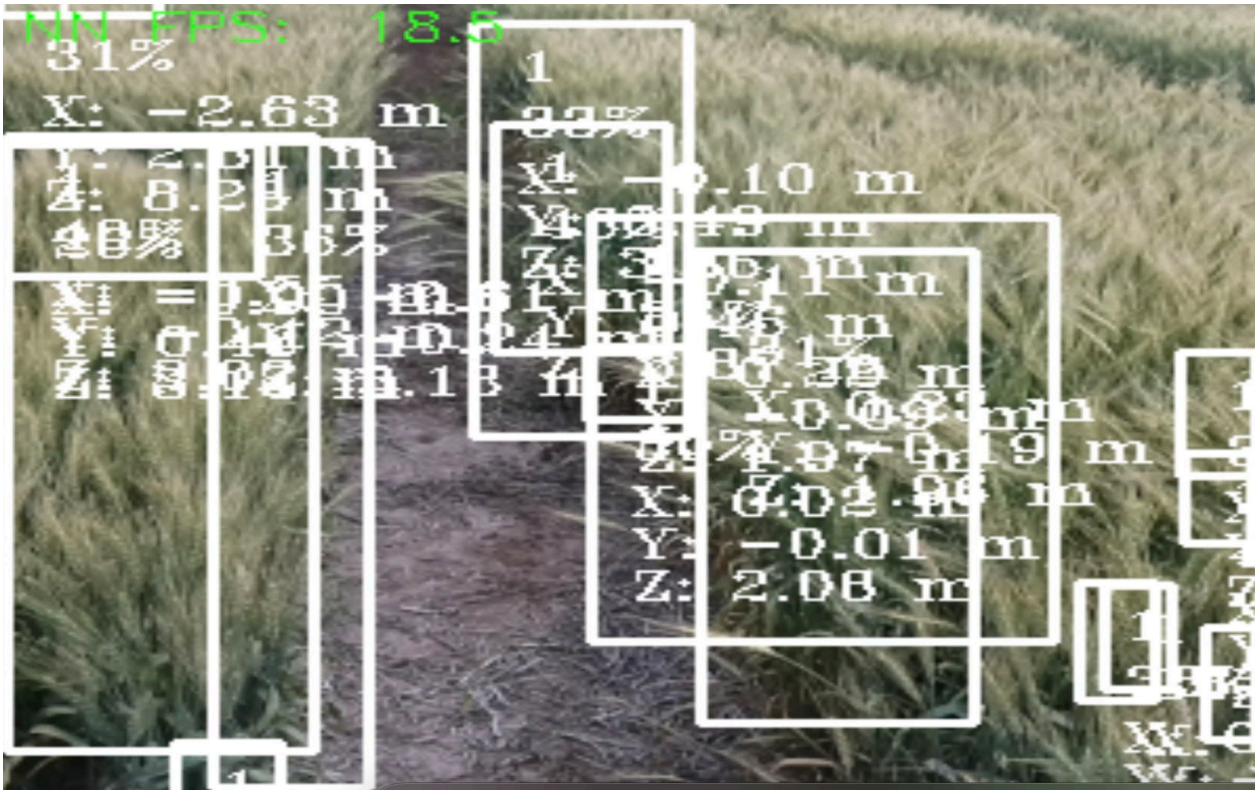

Figure 3.15 Real-time depth sensing of wheat in the field.

For speed the image resolution was set to 300 x 300. The frame rate (upper left in green) is 18.5 fps.

3.16 Benchmarking on LattePanda

The below Figure 3.16 shows the number of images versus time taken in seconds to perform inference. We observe that MobileNet SSD on the OpenCV AI kit at an image size 300x300 has achieved the lowest inference time while Faster R-CNN with ResNet-50-FPN at an image size 640x640 has the highest inference time.

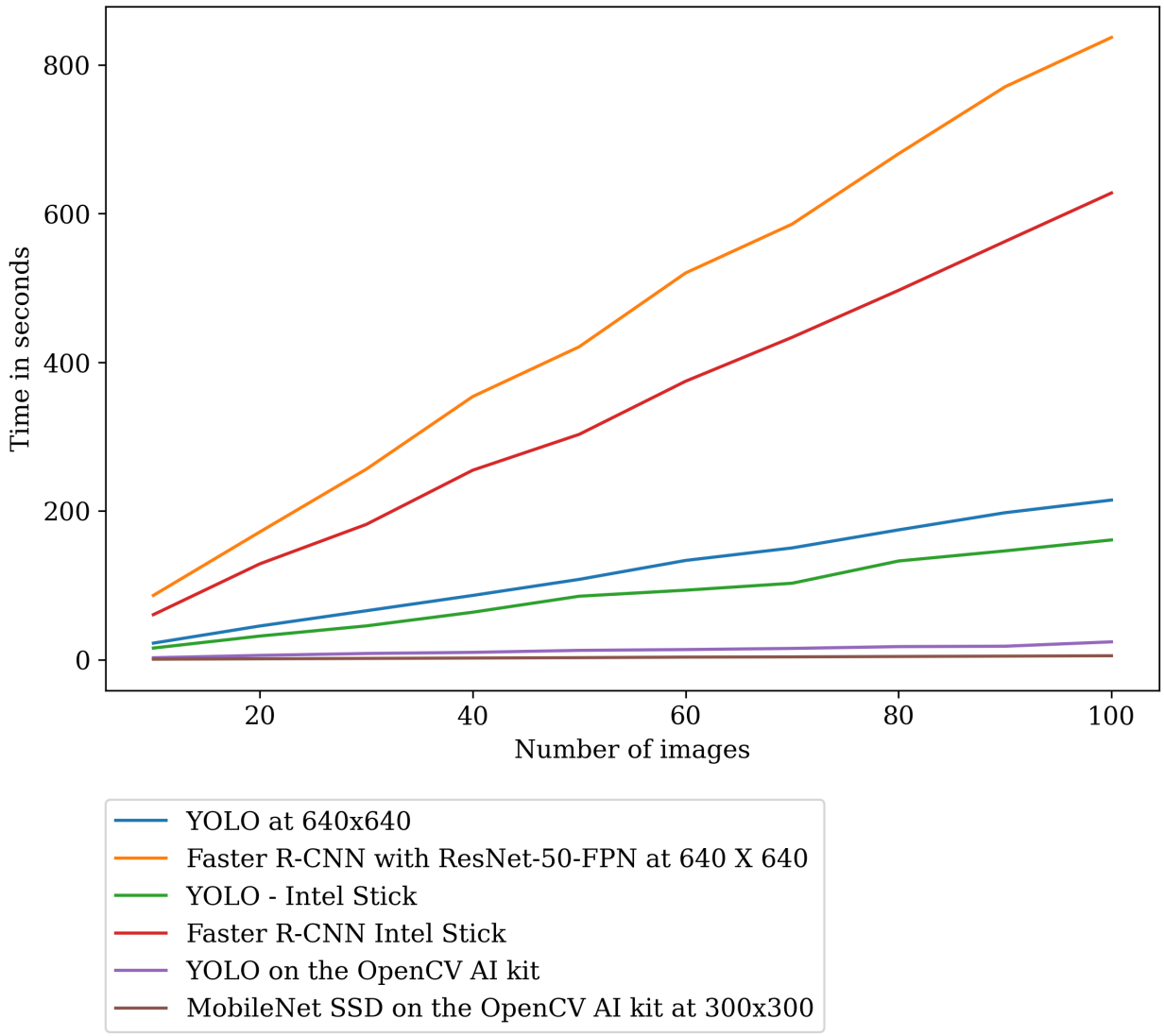

Figure 3.16 Time taken to perform wheat detection on various platforms

Chapter 4 - Disease Detection in Crops Using Deep learning

This chapter provides a detailed workflow on training train deep neural networks to classify images and evaluates their performance on images collected in the field.

4.1 Data Collection

The wheat leaf images were taken with a commonly available camera from mobile devices at the same experimental fields mentioned in Section 3.1. The dataset consists of images of entire leaves and stems. As the data from KSU Agronomy Farm is insufficient to produce an effective deep learning model dataset, different sources of images like Kaggle, and Google images are used to prepare the training data set. Images from Kaggle include three different images of wheat namely, 1) healthy wheat 2) leaf rust and (*Puccinia triticina*) 3) stem rust (*Puccinia graminis*).

4.2 Workflow

Our aim here is to classify if wheat leaf/stem rust is present and if there exists any disease, then, record the location (GPS coordinates) of the robot. The workflow of disease classification is like that for wheat detection with subtle changes. This section elaborates the detection workflow and how the robot uses this detection algorithm to localize the disease by noting down the GPS coordinates. Figure 4.1 depicts the workflow to detect wheat diseases.

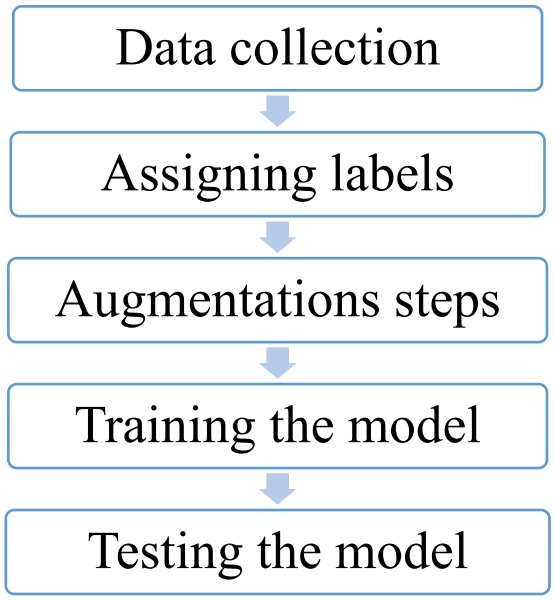

Figure 4.1 Workflow for disease detection

4.3 Approach

To classify an image the training images are labeled with their respective classes that is., 0 for healthy wheat, 1 for images with leaf rust, and 2 for images with stem rust. A csv file containing image_ids and labels is obtained.

image_id	label
03TD19.jpg	0
0LBIWV.jpg	2
0O5BON.jpg	1

The distribution of images in the training set in percentages is 16% healthy wheat, 41% leaf rust, and 43% stem rust. The training dataset is obtained from The Consultative Group on International Agricultural Research (CGIAR) computer vision for Crop Diseases which contains images collected in-field by International Maize and Wheat Improvement Center (CIMMYT) and its partners in Ethiopia and Tanzania.

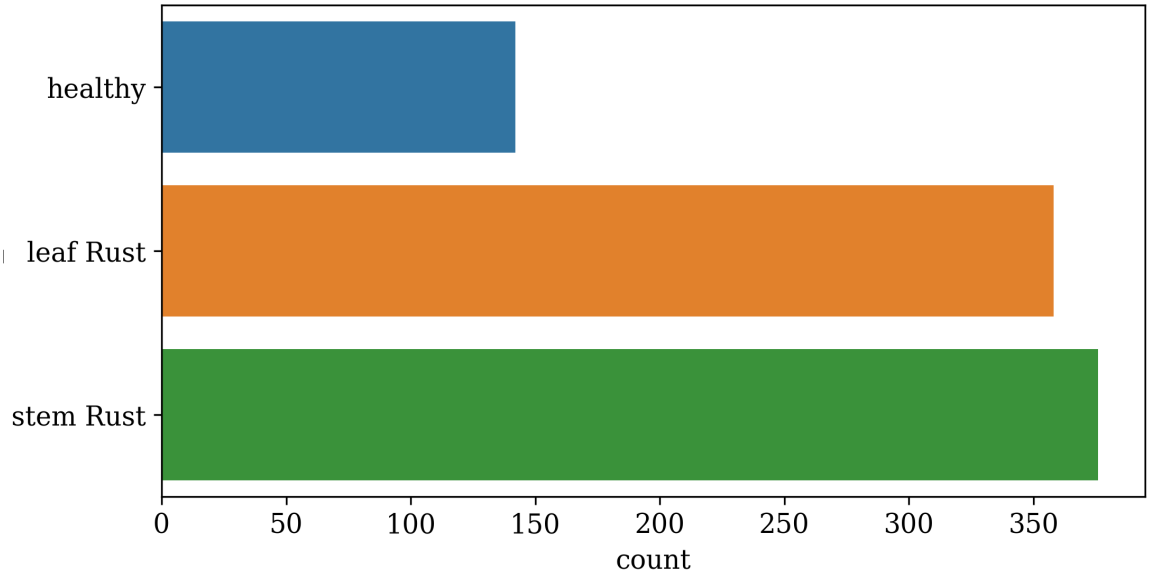

Figure 4.2 Distribution of CGIAR dataset.

Figure 4.3 Sample images for each class in the CGIAR dataset.

The training images are treated with different types of Augmentation techniques like Coursedropout: Removes rectangles randomly from training images, Resize: Resizes the input image to the given height and width.

(a) Original image (b) Shift scale rotate (c) Coarse dropout

Figure 4.4 Augmentation for an image.

4.4 Training

A convolutional neural network Architecture with a pretrained ResNet-18 backbone is used to train the model. This section describes how the training and testing of the wheat disease classification model are done and explains the type of optimizer, scheduler used, and type of loss used during training of the neural network.

4.4.1 Optimizer

The role of the optimizer is to iteratively change the neural network weights until a set is found that maximizes the inference capability. There are two prominent optimizers available 1) Adaptive moment estimation (ADAM), and 2) stochastic gradient descent (SGD). In training this model, the ADAM optimizer was used. It is a method of stochastic optimization that implements an adaptive learning rate. In normal SGD, the learning rate operates in a common fashion across all of the parameters/weights of the model. ADAM selects a separate learning rate for each parameter. Hence, ADAM was preferred over SGD in training this disease classification model.

4.4.2 Loss

Neural networks training requires a loss function to measure improvements during successive epochs. In the context of deep learning, we seek to minimize the loss function. The loss function used in the optimized disease detection model is cross-entropy provided by PyTorch's nn (i.e., neural network) module. Cross-entropy combines both Log SoftMax loss and negative log-likelihood loss.

4.4.3 Learning Rate Scheduler

As the training progresses it is necessary to reduce the learning rate. At a high learning rate, the model possesses high kinetic energy, and hence, explores the parameter space more broadly without becoming trapped in some possibly suboptimal shallow part of the loss function. Later, when the learning rate is made to be slow by lowering the kinetic energy, the parameter vector can settle down in deep, narrow parts of the loss function corresponding to high performance. The learning rate schedule used in training the ResNet-18 was StepLR which decays the learning rate of each parameter group by a given gamma for every given step size.

4.5 Results

For disease classification, we observe the accuracy is from 80 to 90 percent on various test dataset splits. Table 4.1 shows the classification accuracy based on a test split. It should be noted that the model performed equally well for images with dominating background, and images with extraneous objects like human hands and legs. ResNet-18 is 2% more accurate compared to EfficientNet-B4. Table 4.2 gives precision and recall for both classification models.

4.5.1 Accuracy

The metric we have chosen to measure the performance is Accuracy, which is the ratio of the number of samples predicted correctly to the total number of samples

$$Accuracy\ score = \frac{Number\ of\ images\ predicted\ correctly}{Number\ of\ images.} \quad \textbf{Equation 4.1}$$

Table 4.1 Accuracy of the disease detection model.

Model	Train test split percentage	Accuracy
ResNet-18	80-20	90
EfficientNet-B4	80-20	92

Table 4.2 Total precision and recall of the disease detection model

Model	Train test split percentage	Precision	Recall
ResNet-18	80 -20	0.91	0.91
EfficientNet-B4	80-20	0.89	0.89

Table 4.3 Class precision and recall for disease detection model.

Model	Class Precision			Class Recall		
	0	1	2	0	1	2
ResNet-18	1.0	0.88	0.87	1	0.88	0.87
Efficinetnet-B4	1.0	0.85	0.84	1	0.85	0.84

4.5.2 Confusion Matrix

Confusion Matrix The below Figure 4.5 shows the true label and predicted label for the 3 classes. True labels on the y-axis and predicted labels on the x-axis.

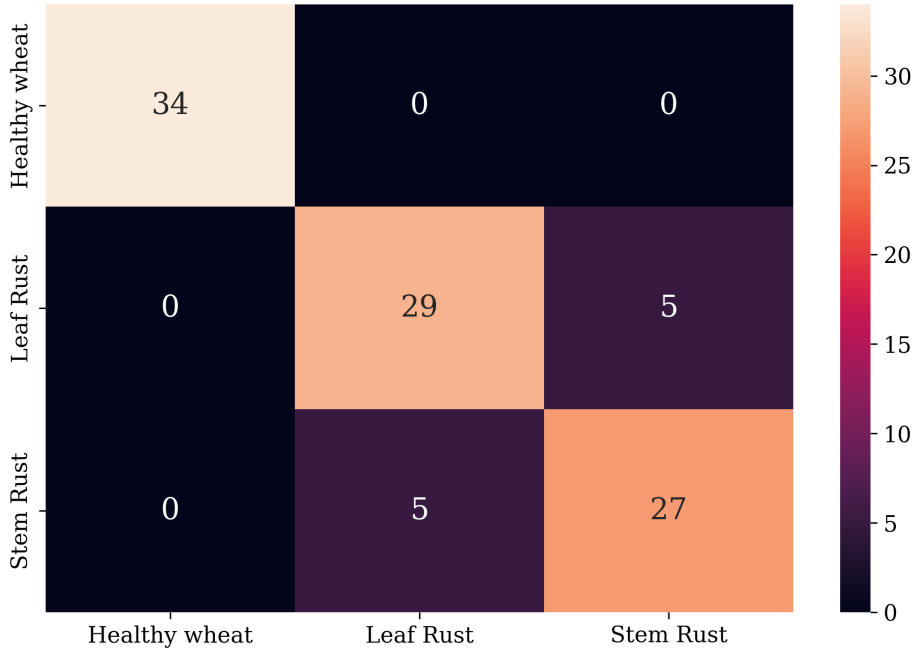

Figure 4.5 Confusion Matrix for ResNet-18.

From the above confusion matrix, it can be understood that the model was very successful in classifying disease and non-diseased wheat since no image of healthy wheat is classified as leaf rust or stem rust and vice-versa.

4.6 Disease Localization

Field robots are equipped with an advanced Global Positioning System (GPS). When disease is detected, locating the affected plants in the field is a matter of recording the GPS reading at that time point. A topic of current interest to plant pathologists is to reduce the cost of disease control by only applying chemical protectants at locations where disease is present. The developments made here are directly supportive of such a strategy.

Chapter 5 - Conclusions and Future Research directions

In this work, a workflow is designed to detect wheat using various state-of-the-art deep learning techniques and different methods are explored in improving the accuracy and speed on a relatively low-cost device. Later, the trained models are combined with stereo depth sensing to detect the distance from the camera to the object. The later part of the work focuses on identifying diseases in wheat crops. The latter work was made easier due to the many learning methods and architectures that were evaluated during the wheat detection efforts. From the broader perspective, this work solves two of the most common artificial intelligence problems which are object detection and classification.

Finally, benchmarking is done on LattePanda for YOLO, Faster R-CCN with ResNet-50-FPN on various platforms like IntelCore-M3, IntelCoreM3 with Intel NCS, and YOLO, MobileNet SSD on IntecoreM3 with OpenCV Ai camera.

The current work includes limitations such as 1) making the robot fully autonomous, and 2) identifying alleys between individual plots in the field (evident in Figure 3.14). These limitations can be eliminated by using segmentation techniques to detect alleys in the field and making the robot fully autonomous using path planning algorithms such as the dynamic window collision approach. Path planning algorithms can be combined with deep learning models to achieve fully autonomous driving.

Chapter 6 -References

1. Asseng, S., Ewert, F., Martre, P., Rötter, R. P., Lobell, D. B., Cammarano, D., ... & Zhu, Y. (2015). Rising temperatures reduce global wheat production. *Nature climate change*, *5*(2), 143-147.
2. Tack, J., Barkley, A., & Nalley, L. L. (2015). Effect of warming temperatures on US wheat yields. *Proceedings of the National Academy of Sciences*, *112*(22), 6931-6936.
3. Ihsan, M. Z., El-Nakhlawy, F. S., Ismail, S. M., & Fahad, S. (2016). Wheat phenological development and growth studies as affected by drought and late season high temperature stress under arid environment. *Frontiers in Plant Science*, *7*, 795.
4. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
5. Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).
6. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-CNN: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*, 91-99.
7. ImageNet Large Scale Visual Recognition Challenge 2015 (ILSVRC2015). Available: https://image-net.org/challenges/LSVRC/2015/index.php
8. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.
9. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
10. David, E., Madec, S., Sadeghi-Tehran, P., Aasen, H., Zheng, B., Liu, S., ... & Guo, W. (2020). Global Wheat Head Detection (GWHD) dataset: a large and diverse dataset of high-resolution RGB-labelled images to develop and benchmark wheat head detection methods. *Plant Phenomics*, *2020*.

11. Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification. Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband.
12. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.
13. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. International journal of computer vision, 88(2), 303-338.
14. Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. International journal of computer vision, 111(1), 98-136.
15. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252.
16. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., … Chintala, S. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32 (pp. 8024–8035). Curran Associates, Inc. Retrieved from http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
17. Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: fast and flexible image augmentations. Information, 11(2), 125.
18. Salton, G., & McGill, M. J. (1983). Introduction to modern information retrieval. mcgraw-hill.
19. Ting K.M. (2011) Precision and Recall. In: Sammut C., Webb G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_652
20. Tan, M., & Le, Q. (2019, May). Efficientnet: Rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

21. Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2117-2125).
22. Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338.
23. Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.
24. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.
25. Robertson, S. (2008, July). A new interpretation of average precision. In Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval (pp. 689-690).
26. Wheat Growth Stages, Available: https://prairiecalifornian.com/wheat-growth-stages/
27. LattePanda Alpha 864s (Win10 Pro activated) – Tiny Ultimate Windows / Linux Device, available: https://www.dfrobot.com/product-1729.html
28. Intel® Neural Compute Stick, Available:https://www.intel.com/content/www/us/en/developer/tools/neural-computestick/overview.html
29. OpenCV AI Kit: OAK—D, Available: https://store.opencv.ai/products/oak-d
30. Biboswan roy, optim.Adam vs optim.SGD. Let's dive in. https://medium.com/@Biboswan98/optim-adam-vs-optim-sgd-lets-dive-in-8dbf1890fbdc
31. Precision, Recall, F1-Score for Object Detection - Back to the ML Basics. Available: https://www.linkedin.com/pulse/precision-recall-f1-score-object-detection-back-ml-basics-felix/
32. YOLO V3 Explained. Available: https://towardsdatascience.com/yolo-v3-explained-ff5b850390f

33. CGIAR Computer Vision for Crop Disease. Available: https://www.kaggle.com/shadabhussain/cgiar-computer-vision-for-crop-disease
34. Blob Converter Available: http://luxonis.com:8080/