COMPARISON OF OBJECT DETECTION METHODS FOR CORN DAMAGE ASSESSMENT USING DEEP LEARNING

A. Hamidisepehr, S. V. Mirnezami, J. K. Ward

HIGHLIGHTS

- Corn damage detection was possible using advanced deep learning and computer vision techniques trained with images of simulated corn lodging.
- RetinaNet and YOLOv2 both worked well at identifying regions of lodged corn.
- Automating crop damage identification could provide useful information to producers and other stakeholders from visual-band UAS imagery.

ABSTRACT. *Severe weather events can cause large financial losses to farmers. Detailed information on the location and severity of damage will assist farmers, insurance companies, and disaster response agencies in making wise post-damage decisions. The goal of this study was a proof-of-concept to detect areas of damaged corn from aerial imagery using computer vision and deep learning techniques. A specific objective was to compare existing object detection algorithms to determine which is best suited for corn damage detection. Simulated corn lodging was used to create a training and analysis data set. An unmanned aerial system equipped with an RGB camera was used for image acquisition. Three popular object detectors (Faster R-CNN, YOLOv2, and RetinaNet) were assessed for their ability to detect damaged areas. Average precision (AP) was used to compare object detectors. RetinaNet and YOLOv2 demonstrated robust capability for corn damage identification, with AP ranging from 98.43% to 73.24% and from 97.0% to 55.99%, respectively, across all conditions. Faster R-CNN did not perform as well as the other two models, with AP between 77.29% and 14.47% for all conditions. Detecting corn damage at later growth stages was more difficult for all three object detectors.*

Keywords. *Computer vision, Faster R-CNN, RetinaNet, Severe weather, Smart farming, YOLO.*

Digital farming techniques are becoming increasingly useful as advances in sensing, data processing, and analytics are becoming more accessible. Crop producers whose operations are exposed to severe weather can use data-driven approaches to assess the impact of acute crop damage events and to respond faster and with more information than manual detection of crop damage. The need for digital agriculture tools after severe weather events was painfully evident to producers along the U.S. Atlantic seaboard, who were exposed to multiple consecutive severe hurricane seasons. Tropical weather events in U.S. coastal states like North Carolina most often occur during harvest season (Cangialosi, 2017; Stewart and Berg, 2019) (fig. 1a). The 2018 hurricanes Florence and Michael inflicted over a billion dollars in losses to North Carolina's agriculture industry. Even when acute weather events are less impactful, wildlife can damage crops and create financial losses late in the growing season when, for example, black bears feed on corn to add fat prior to winter (fig. 1b). New tools are needed that can detect and quantify crop damage late in the growing season near to harvest.

Current methods for detecting and reporting crop damage are manual and visual. After a severe weather event, growers report that damage has occurred, and then different stakeholders, such as insurance adjusters, Extension agents, or disaster response agencies, survey the damage by walking through or driving along the damaged field. Geospatial information, such as field boundaries, damaged area boundaries, or geo-tagged descriptive images, may also be collected.

Detailed information on the presence and severity of crop damage would help producers make the decision to harvest the crop or file an insurance claim on the damaged field. Insurance providers and emergency response agencies could use crop damage information to estimate indemnity commitments and report damage faster, which could increase the effectiveness of the support for communities impacted by severe weather. Manned aircraft and satellites have been used for collecting remotely sensed data but are limited by cost and spatial or temporal resolution (Hamidisepehr et al.,

Submitted for review on 10 November 2019 as manuscript number ITSC 13791; approved for publication as a Research Article by the Information Technology, Sensors, & Control Systems Community of ASABE on 31 July 2020.

The authors are **Ali Hamidisepehr,** Postdoctoral Research Scholar, Department of Biological and Agricultural Engineering, North Carolina State University, Raleigh, North Carolina; **Seyed V. Mirnezami,** Graduate Research Assistant, Department of Mechanical Engineering, Iowa State University, Ames, Iowa; **Jason K. Ward,** Assistant Professor, Department of Biological and Agricultural Engineering, North Carolina State University, Raleigh, North Carolina. **Corresponding author:** Ali Hamidisepehr, 3100 Faucette Drive, North Carolina State University, Raleigh, NC 27695; phone: 859-420-7211; e-mail: ali.hamidisepehr@gmail.com.

  ISSN 2151-0032 https://doi.org/10.13031/trans.13791

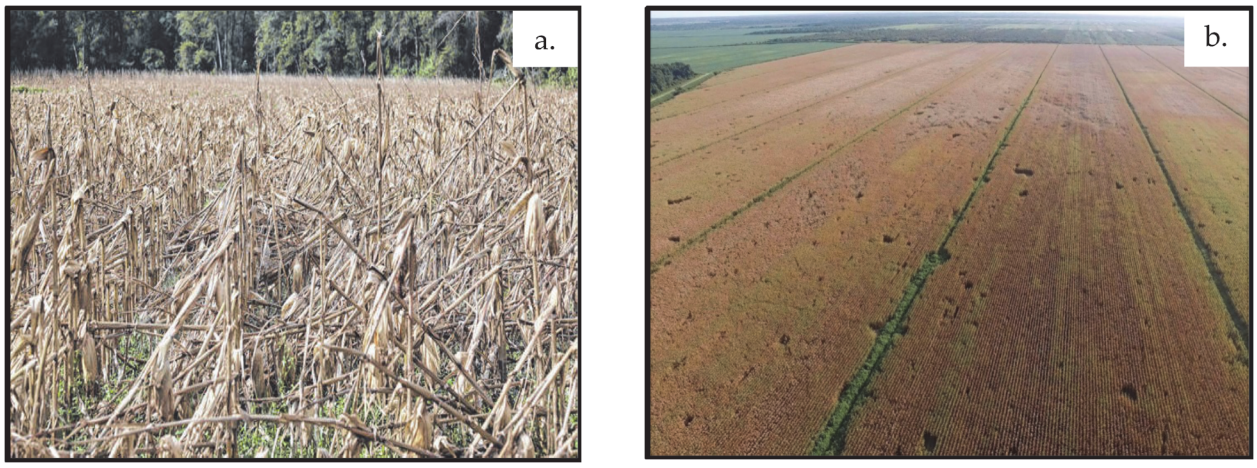

Figure 1. Examples of crop damage caused by (a) Hurricane Florence (source: Sherry Matthews, *Sampson County Independent*, 19 Sept., 2018) and (b) black bears that flattened corn stalks while feeding on corn in eastern North Carolina (*NC Blackbear Newsletter*, 31 Aug., 2018).

2017). Unmanned aerial systems (UAS) have become popular remote sensing tools in digital agriculture because they enable growers to acquire precise information about their fields at specific times of interest (Hamidisepehr and Sama, 2018a). A wide variety of sensors, including hyperspectral, multispectral, and thermal imaging, can be deployed to measure different field parameters (Hamidisepehr and Sama, 2018b). The red, green, and blue (RGB) color bands provided by a standard digital camera are the most recognized data to end-users and provide images in the human visual range at an affordable price. When combined with a UAS platform, RGB digital imaging can provide a high spatial resolution survey of a field (Mahajan et al., 2015).

Traditional image processing methods use manually extracted target data and static methods for analysis (Ma et al., 2019; Vibhute and Bodhe, 2012; Zhou et al., 2019). However, as dataset size and analysis complexity have increased, traditional image processing methods have become less effective, or they fail in robustly processing large datasets and complex images (Kamilaris and Prenafeta-Boldú, 2018). Computer vision has significantly improved the power of image processing tools. Computer vision uses algorithms to address various tasks such as image detection (Jayas et al., 2000), segmentation (Sammouda et al., 2014), and classification (Krizhevsky et al., 2012). There are several examples of using computer vision in digital agriculture, including weed detection (Lu et al., 2017), disease detection (Mohanty et al., 2016), plant recognition (Reyes et al., 2015), plant phenotyping (Falk et al., 2020), fruit counting (Rahnemoonfar and Sheppard, 2017), and crop classification (Rebetez et al., 2016).

Object detection is a computer vision technique that is applied for detecting specific objects in images. Face detection (Kazemi and Sullivan, 2014) and pedestrian detection (Li et al., 2016) are the most well-developed applications of this technique. The most common object detection methods include Faster Region-based Convolutional Neural Network (Faster R-CNN) (Ren et al., 2015), You Only Look Once Version 2 (YOLOv2) (Redmon and Farhadi, 2017), and RetinaNet (Lin et al., 2017). Each of these methods performs differently in different applications. YOLOv2 and RetinaNet can perform faster than Faster R-CNN because they implement a single-stage detection process. Additionally, they showed higher performance on standard datasets (Kamilaris and Prenafeta-Boldú, 2018; Zheng et al., 2018). However, Faster R-CNN has been a popular method for several applications due to its ease of use (Kamilaris and Prenafeta-Boldú, 2018).

In most deep learning applications, it is common to use a pre-developed computer vision model trained on a standard dataset. Using a technique called transfer learning, an existing object detection algorithm can be trained to a desired target with fewer images. The neural network is trained to identify general features first, and in later layers the neural network is trained to the specific target object. Collecting a large enough dataset for developing a custom deep learning method would be difficult, time-consuming, a nearly impossible for most users focused on application. Existing feature extraction methods, such as those mentioned previously, can be leveraged from models trained on standard datasets, and object detection is then fine-tuned to the desired target (Kamilaris and Prenafeta-Boldú, 2018).

Agricultural applications of object detection are becoming more common. A real-time vegetable detection system was developed using deep learning networks (Zheng et al., 2018). Multiple object detectors were selected to recognize tomato and cucumber at different stages. Among all the advanced detectors selected, YOLOv2 had the highest performance in terms of model average precision (AP). AP is an index that incorporates the ability of a model to make correct classifications and the ability to find all relevant objects. Koirala et al. (2019) tested these main detection algorithms to detect mangos in images. By optimizing YOLOv2, they developed a new algorithm that exhibited improved detection performance. RetinaNet was also tested in vineyards to detect Esca disease and obtained AP of 70% (Rançon et al., 2019).

The combination of computer vision, deep learning, and powerful hardware has demonstrated success in digital agriculture projects. Using computer vision techniques along with UAS imagery allows precise assessment of field conditions (Tripicchio et al., 2015). Nolan et al. (2015) used computer vision techniques and a UAS to delineate vine rows automatically and proved the efficiency of the system in commercial vineyards. An autonomous UAS with onboard computer vision capabilities was developed by Alsalam et

al. (2017). That system provided promising results for weed detection and color detection for fruit sorting; however, the authors highly recommended using machine learning methods instead of the traditional image processing method (i.e., target detection software) for a more robust weed detection system.

Identifying crop damage caused by severe weather conditions or other stressors via remote sensing has been also tested. For instance, the structure-from-motion photogrammetry method was applied to detect lodging in maize. In this approach, a 3D map needs to be created after stitching individual and overlapping images collected with RGB and NIR cameras (Chu et al., 2017). Multispectral imaging was used to detect crop hail damage using vegetation indices; detection was more precise in cases with more severe damage or when the imagery was acquired soon after the damage occurred (Zhou et al., 2016). Crop damage was assessed using satellite imagery after a frost, but low spatial and temporal resolution made future applications of the technique unlikely (Silleos et al., 2002).

Despite the progress in recent years, crop damage identification has room for improvement and thereby provides an opportunity to deploy deep learning and object detection methods for detecting and analyzing crop damage data with more complexity and at different stages of growth. In this study, the ability of object detection methods to detect lodging in late-season corn at different stages of senescence was analyzed. Individual methods were compared in terms of their prediction power. The specific objectives were:

1. To compare object detector performance for damage identification in lodged corn.
2. To assess the best model and best growth stage for damage detection in lodged corn.

Materials and Methods

Field Location And Plot Layout

The field study site was located near Goldsboro, North Carolina, on the Cherry Research Station. Corn hybrid Augusta 5065, with 115-day relative maturity (RM), was planted on 19 April 2018. The seed population was 80,000 seeds ha^{-1} (32,000 seeds $acre^{-1}$) on 76.2 cm (30 in.) row spacings. Plots were established in strips six rows wide and 46 m (150 ft) long for a total area of ~210 m^2 per treatment strip. The treatment areas were rectangular. The shape of the simulated treatment area has no bearing on the training and detection of damaged regions within a field. Even if the simulated lodging area has a complex shape, the training annotation and object detection bounding boxes are rectangular and parallel (or near parallel) to the image boundaries. Treatments were not replicated because the goal of this study was to determine if corn crop damage could be reliably detected in imagery and to compare object detection methods.

Field Damage Simulation

Eleven treatments were created to simulate two different corn damage modes in five different late-season growth stages, along with a control. The two corn damage modes were used to represent different weather impacts on corn crop lodging near harvest, and standing stalks were broken either immediately under the first ear or at ground level. The simulated damage was created by manually breaking every corn stalk in the plot designated by the treatment plan. The two corn damage modes were used to increase the robustness of the detection algorithm for different types of damage but were not intended, in this study, to be segmented. More data are needed at each damage level in order to successfully segment the two modes. Collecting additional data in subsequent seasons will give us the chance to detect the damage level.

Time effects were referenced to crop physical maturity as estimated by hybrid RM: stage 1 was two weeks prior to the week of RM, stage 2 was one week prior, stage 3 was the week of RM, stage 4 was the week after RM, and stage 5 was two weeks after RM. The treatments were chosen to add diversity to the object detection model due to physical crop differences and the impact of crop senescence over time. There was only one treatment by time combination because this project was an initial assessment to determine if physical corn damage could be reliably detected in imagery and to compare object detection methods.

Image Acquisition

UAS (M600 Pro, DJI, Shenzen, China) imagery was collected using an RGB camera (Zenmuse X5, DJI, Shenzen, China). Imagery was collected at 92 m above ground level at a ground sampling distance of 2.25 cm per pixel. Imagery was collected on the same day immediately after that growth stage's treatments were applied (table 1). Figure 2 shows example images of the farm at different growth stages. The images on the left were collected when the camera angle was parallel to the image boundaries. The images on the right were collected when the UAS was executing a turn. As described later, images captured while turning were filtered to create different datasets.

Data Preprocessing and Augmentation

Images that did not contain a damaged region (known as negative images) or that were collected before the UAS reached the desired altitude were excluded. This was done because the number of negative images was substantially higher than the number of images with damaged regions (known as positive images). Even in images with damaged regions, most of the image area was negative for target objects. Negative images would not enhance the performance of the object detection model and would cause longer training time. The goal was to assess the object detection methods, not the detection of areas without target objects. False positive results could be adequately assessed from the images containing target objects.

Table 1. Date and days after planting (DAP) of treatment and data collection events during the 2018 growing season.

Activity	Date	DAP
Planting	19 April	0
Stage 1	3 August	106
Stage 2	10 August	113
Stage 3	17 August	120
Stage 4	24 August	127
Stage 5	31 August	134

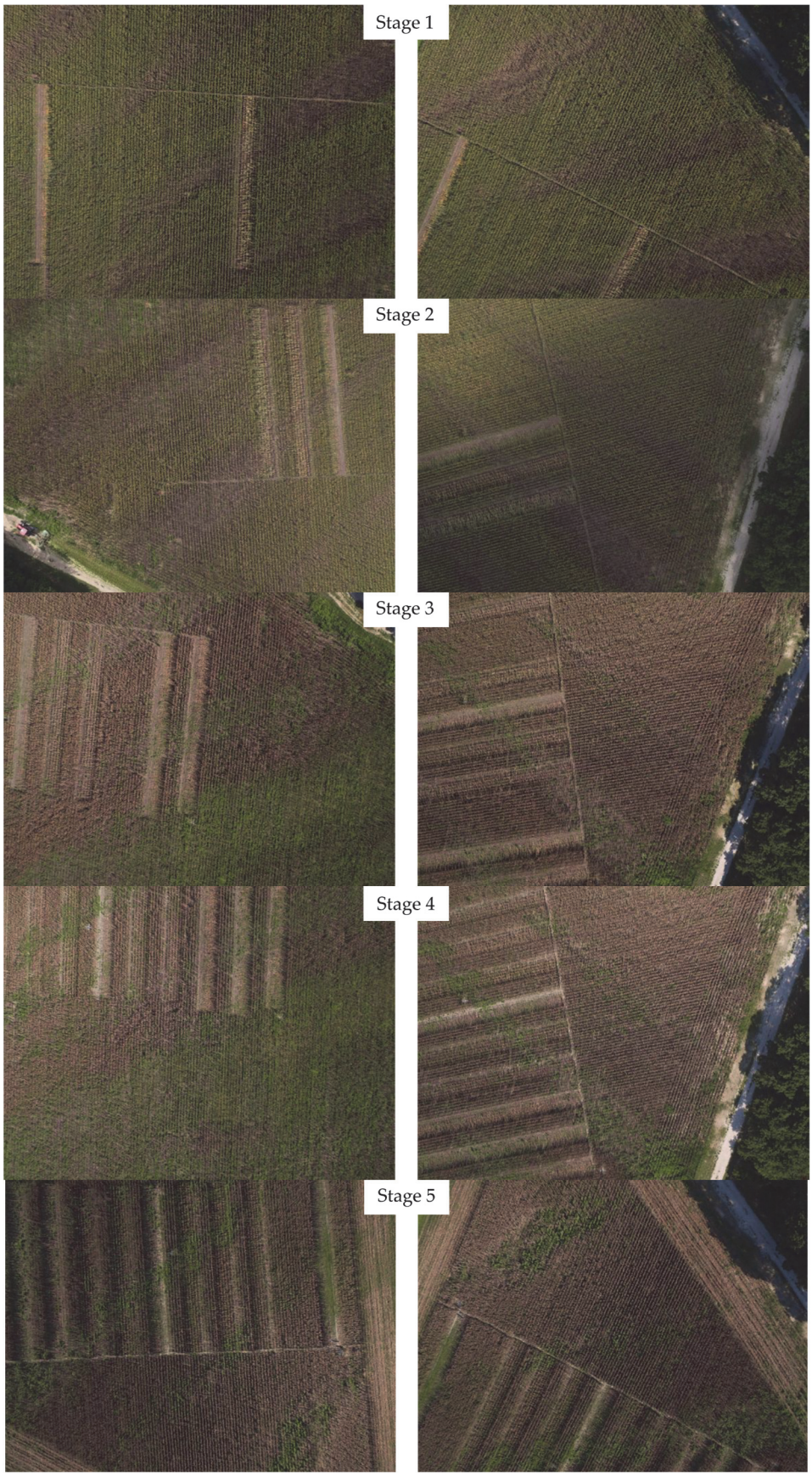

Figure 2. Examples of aerial imagery at different growth stages: (left) plots generally parallel to image boundaries and (right) plots not parallel to image boundaries. During preprocessing, images with plots not parallel to image boundaries were excluded from filtered datasets.

In this study, "model" refers to a successfully trained object detector using one of the three architectures at a particular growth stage. For example, the YOLOv2 stage 1 model was an object detector built using YOLOv2 trained using annotated images from the stage 1 data collection event only. An additional model was developed for each architecture based on all training images from all growth stages to determine if this approach would provide greater prediction power because of the increased diversity in the imagery data.

A limitation with the tested object detectors was that input images could be annotated for training or identifying target objects only with rectangular areas whose borders were parallel (or near parallel) to the image borders. A complex shape could still be identified with a rectangle during training and testing; however, the object's orientation and dimensions could create complexity. A region whose major axis was not parallel to the image borders increased the area that was incorrectly labeled as the target object because the negative space outside the target region but within the labeled region was included in the object classification. In this field study, the damaged regions were rectangular, and the UAS flight path was almost parallel to the major axis of the rectangular damaged region.

During each flight, multiple images were taken at the end of each flight line while the UAS was turning. To analyze the impact of these images on model performance, an additional model was created for each stage in which the imagery was filtered to exclude images of the plots whose major axes were not approximately parallel to the image boundaries. In total, twelve models were generated for testing each object detector: one model for each growth stage from filtered imagery, one model for each growth state from unfiltered imagery, and two aggregate models using all filtered and unfiltered imagery. The number of images in each model with and without filtering is shown in table 2.

Creating simulated damaged regions, collecting imagery data, and data annotation are very labor-intensive. To make maximum use of the collected imagery, additional data were derived from the original data using data augmentation techniques. New images were generated randomly during the training process by rotating, cropping, changing color, and resizing the original images. Data augmentation was provided as a built-in feature of all the object detectors and was used to compensate for the limited dataset. The data augmentation hyperparameters were left at their default values. The original images were resized to 12.5% of their original size (from 4600 × 3400 pixels to 570 × 430 pixels) to make the training process faster and the hardware usage more efficient.

Table 2. Numbers of images and bounding boxes at different growth stages with and without filtering.

Growth Stage	Number of Images		Number of Bounding Boxes	
	Without Filtering	After Filtering	Without Filtering	After Filtering
1	80	67	112	96
2	92	80	267	234
3	103	88	540	472
4	103	87	681	588
5	100	81	670	567
Total	478	403	2270	1957

Data Annotation

To prepare the image datasets for training, the damaged areas in individual images were labeled with rectangular bounding boxes. The different object detectors required different formats for labeling the images. For the Faster R-CNN models, images were labeled using Image Labeler, a MATLAB application. A MATLAB script was used to convert the Faster R-CNN labels into the format required by RetinaNet. YOLOv2 required a distinct format; therefore, LabelImg was used for annotation. Figure 3 shows a flowchart of the corn damage detection system starting with data collection and annotation. The feature extractors were chosen based on the detector method. ResNet50, VGG16, and Darknet19 were chosen for Faster R-CNN, RetinaNet, and YOLO, respectively, similar to Zheng et al. (2018).

Training and Testing

The complete object detection algorithm consisted of feature extraction followed by object identification. Feature extraction identified non-redundant areas of interest in the imagery. Object detection then identified the contents of the areas of interest based on the training data used to develop the algorithm. The dataset for each model was randomly divided to use 85% of the images for training and 15% for testing, as in Hamidisepehr (2018). Three different training and testing datasets were randomly selected, and the performance measures were obtained by calculating an average from running three different data selections. Hyperparameters are variables needed to fine-tune the weights from pre-developed models like ResNet and VGG before applying a learning algorithm to a custom dataset. Hyperparameters, including the number of epochs, batch size, and learning rate, were adjusted for each training dataset to achieve the highest model performance in terms of accuracy. Initial values were set for each model based on default values suggested by Lin et al. (2017). Training and testing were done using MATLAB 2019a for the Faster R-CNN models and Python 3.5 for the RetinaNet and YOLO models.

Epochs represent the number of times that the entire training dataset passes through a neural network. Increasing the number of epochs results in greater prediction power. However, an excessively large number of epochs increases the training time with no performance improvement (Amiri et al., 2017). Model performance was quantified by calculating the total loss, which combined the error in bounding box location, size, and confidence (i.e., the probability of the desired object existing in the bounding box). The amount of loss was monitored after each epoch, and training ceased if the total loss did not marginally decrease. Batch size is the number of images presented to the learning algorithm in one pass. Increasing the batch size usually results in higher prediction power, although an extremely large batch size can cause memory errors. A small batch size, which is needed for less powerful hardware, introduces undesirable noise, which prevents the training process from converging to an optimal value (Rhu et al., 2016). The learning rate is the rate of training. Similar to batch size, increasing the learning rate can improve the model performance but can also cause memory errors depending on the hardware capacity (Dauphin et al., 2015). The number of epochs, the batch size, and

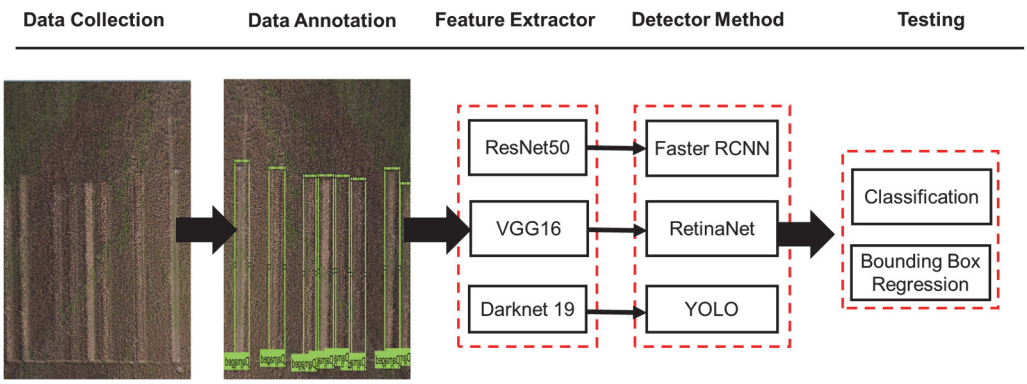

Figure 3. Structure of corn damage recognition system based on deep learning and computer vision. The architecture of the system includes the input image, damage annotation, feature extraction, training, and detection of corn damage in the image. All bounding boxes are labeled as "damaged" in the Data Annotation step.

Table 3. Hyperparameter settings based on preliminary tests.

Model	Epochs	Batch Size	Learning Rate
Faster R-CNN	200	1	10^{-3}
YOLOv2	200	2	10^{-5}
RetinaNet	200	4	10^{-5}

the learning rate were determined empirically to obtain an optimal solution with high performance and minimal training time without memory error. Table 3 shows the hyperparameter settings for each object detection model.

Computing Hardware

Training object detection models using deep learning algorithms is computationally intensive and requires the use of advanced graphical processing unit (GPU) technology (Kirk and Wen-Mei, 2016). Training was initially attempted using nodes equipped with NVIDIA Tesla P100 and NVIDIA GeForce GTX 1080 GPUs. These GPUs provided sufficient resources for training the Faster R-CNN object detection models but struggled to complete the YOLOv2 and RetinaNet models due to the larger batch sizes. Access was provided to computation nodes with more capable NVIDIA Tesla V100 GPUs, which were used to train all the models.

Evaluation

To assess the performance of each model created at different growth stages and with different object detectors, three evaluation metrics were selected: precision, recall, and AP. Precision represents the model performance based on the ratio of the number of correct damage detections to the total number of incorrect and correct damage detections. Recall is the ratio of the number of correct damage detections to the total actual damaged regions in the field. Precision and recall are calculated based on prediction parameters, including true positives, false positives, and false negatives. It is common to prioritize the ultimate goal to minimize either the false positives or the false negatives based on the fault tolerance in a specific project. For example, minimizing false negatives is more important in disease diagnosis, while minimizing false positives is critical for spam detection. The F1 score combines precision and recall for a specific class. The F1 score can be interpreted as a weighted average of the precision and recall, where the best value is 1 and the worst value is 0. The equations for the performance metrics are presented below:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F1 score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

where TP (true positives) is the number of correctly detected damaged regions, FP (false positives) is the number of undamaged areas detected as damaged, and FN (false negatives) is the number of missed damaged regions. The goal of each model was to maximize TP and minimize FN.

Intersection over union (IoU) measures how much the predicted boundary of the damaged region overlaps with the ground truth. A 50% overlap between actual and predicted objects was considered a "match" or a TP object. In this way, the numbers of TP, FP, and FN were counted. With a large enough sample size, the area under the precision-recall curve equals AP. The AP is an index that incorporates the detector's ability to make correct classifications (precision) and its ability to find all relevant objects (recall) (Everingham and Winn, 2012; Henderson and Ferrari, 2016; Koenig et al., 2015).

Results

The aim of this study was to distinguish damaged areas from the undamaged background. Different hyperparameter values were tested to efficiently train each model while avoiding memory errors and obtaining the highest performance. The computational hardware was able to train all object detection models without errors, except the YOLOv2 model for unfiltered images from the entire season. Even at the smallest batch size, that model did not complete the training process due to out-of-memory errors. On average, each model required about 3 h to complete training. The YOLOv2 models generally completed training fastest, while the RetinaNet and Faster R-CNN models were slower, which agrees with the findings reported by Redmon and Farhadi (2018).

Precision-Recall Curve

Precision and recall were calculated for all predictions in the order of their confidence. A precision-recall curve was plotted for each model. The precision-recall curves demonstrate the tradeoff between precision and recall for varying detection thresholds. Maintaining precision at 1.0 or increasing precision indicates that the predictions were correct. Recall values of 1.0 at the endpoint of the curve indicate that all objects were detected and there were no missed objects. An ideal model returns accurate predictions (high precision), as well as detecting all positive objects (high recall). Figure 4 shows the precision-recall curves for each model. To obtain a precision-recall curve for each model, all predictions were ranked based on the confidence level that the model had for detecting that object regardless of whether it was a true or false prediction. The precision value decreased if a prediction was incorrect, while the precision increased for correct decisions. In most cases, objects with top ranks were correct predictions, which is why all curves started at 1.0 precision.

Recall increased after each correct prediction and remained constant in the case of incorrect predictions. Ideally, precision remained high for all recall values, and the curve reached 1.0 recall. For the Faster R-CNN models (figs. 4a and 4b), recall values did not reach 1.0 across all growth stages. At least 20% of the damaged areas were not detected. Maximum precision was not maintained for all models, both filtered (figs. 4a, 4c, and 4e) and unfiltered (figs. 4b, 4d, and 4f), which indicates incorrect predictions even with relatively high confidence. YOLOv2, especially after filtering, and RetinaNet remained at high precision at different recall values in the different growth stages. Among all growth stages, stage 5 had the most inaccuracies across the different object detectors because of the complexity in the dataset due

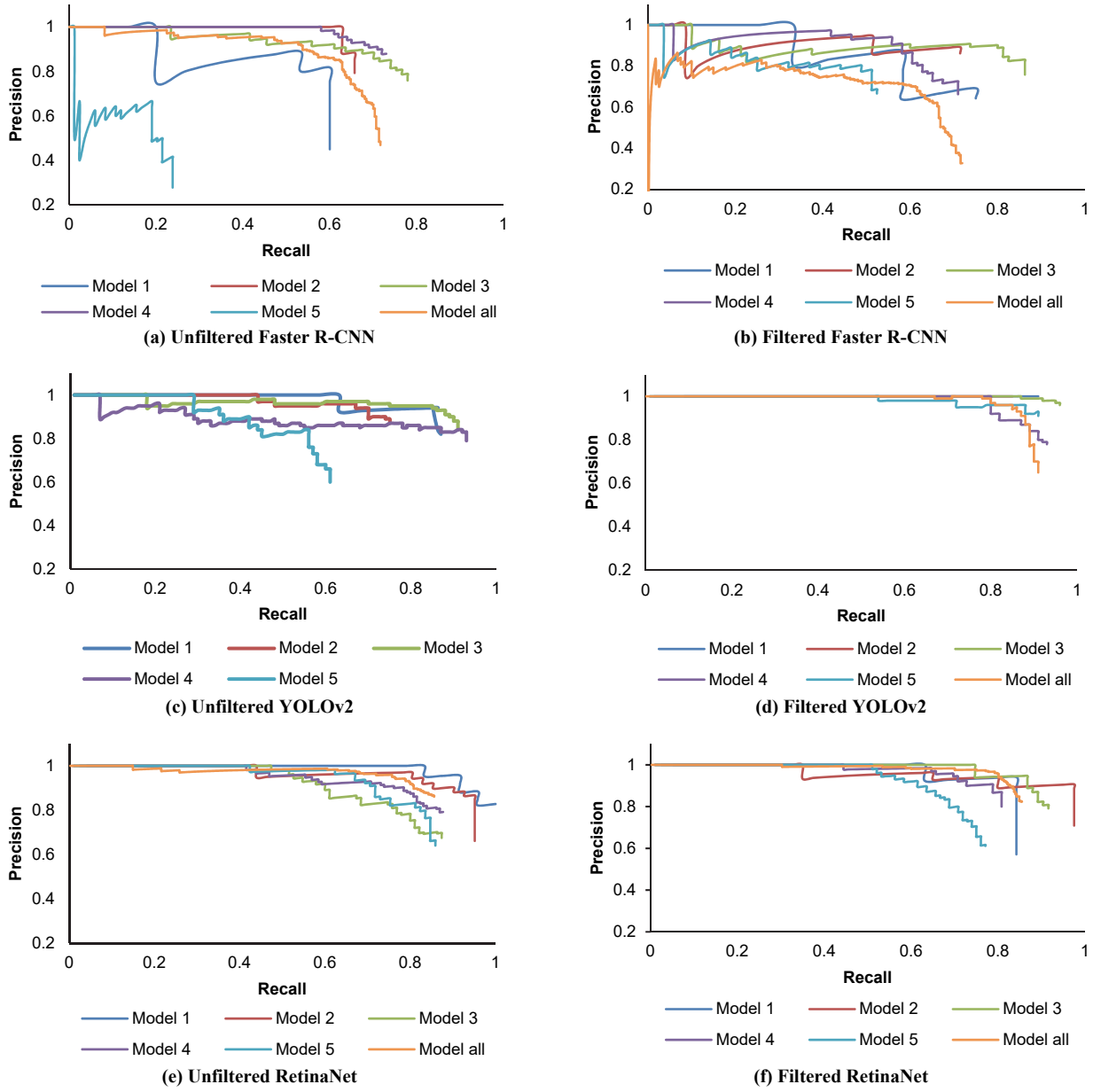

Figure 4. Precision-recall curves at different growth stages for unfiltered and filtered Faster R-CNN, YOLOv2, and RetinaNet models.

to the number of objects, the temporal effect, and the closeness of the objects. The other growth stages provided more accurate models with a slight difference in their overall performance.

Figure 5 shows the bounding boxes for ground truth and for predicted corn damage on the same image for visual inspection. The ground truth bounding boxes are the original annotated testing regions. The colors used to identify the ground truth and detected damaged areas are different for the three algorithms because different labeling and visualization tools were used; color descriptions are included in the figure caption. The images represent one sample from the testing dataset in stage 3 as an example. The numbers above the boxes for the RetinaNet and Faster R-CNN models (figs. 5a and 5c) are the confidence levels for the detection. YOLOv2 provided this information as well, but it was not shown on images. RetinaNet and YOLOv2 (figs. 5a and 5b) detected all damaged areas in the image. Figure 5a shows the impact of regions that are not parallel to the image borders. The farthest left annotated and detected regions cross two treatment zones. Even with annotated regions that include negative space, the model still largely detected the damaged corn. YOLOv2 (fig. 5b) includes a false positive in the bottom left corner of the image. Faster R-CNN (fig. 5c) failed to detect two damaged areas, as indicated by the two bounding boxes with no numbers.

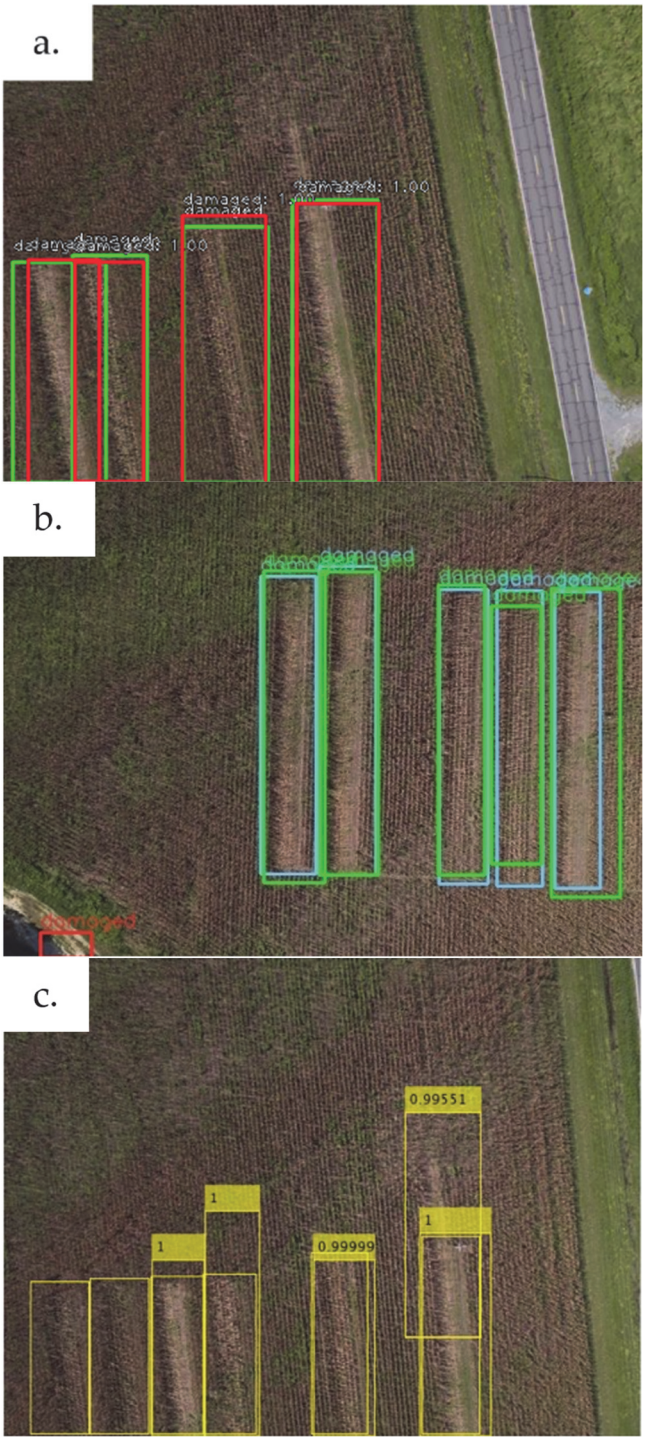

Figure 5. Bounding boxes for ground truth and for predicted corn damage on images from: (a) RetinaNet (red for ground truth, green for prediction, and including confidence level and "damaged" tag for prediction boxes), (b) YOLOv2 (blue for ground truth, green for prediction, red for false positive, and including "damaged" tag), and (c) Faster R-CNN (boxes without a confidence level indicate ground truth, and boxes with a confidence level indicate prediction).

Average Precision

AP is used as an index to measure the overall capability of a model to detect predefined objects. The area under each precision-recall curve from the different models was computed on the test datasets to obtain the AP (table 4). Overall, Faster R-CNN showed lower precision for the different growth stages compared to the two other detectors. The models created with a filtered dataset, which eliminated turning images from the training and testing datasets, were expected to have higher AP values. Less preprocessing on the incoming data is preferred so that the algorithm can be trusted to robustly detect corn damage without additional data management. RetinaNet provided consistently high precision for the different growth stages. Filtering out the turning images did not improve the detection accuracy because the unfiltered models were provided with larger and more diverse datasets. RetinaNet also did not have difficulty in detecting damaged areas, even in turning images, which is an extra benefit of using its models for this type of agricultural target. However, filtering images before training the YOLOv2 models showed an improvement in model performance in most cases. The only inconsistency in the YOLOv2 models was for model 1, which had higher detection power before filtering; this could be due to the low number of images for growth stage 1. Because the datasets were not very large, especially in the early growth stages, the precision can vary to some extent by changing the number of images in the testing dataset. In most cases, model 5 had lower precision compared to the other models with the same object detector. This could be due to the high number of damaged areas within a small distance of each other, which made it complicated for the detector to differentiate between undamaged and damaged areas. In addition, as time passed, weeds emerged in damaged areas that had been created in earlier growth stages (fig. 6).

Overall, both YOLOv2 and RetinaNet were capable of accomplishing the task with promising results, and their AP values varied only slightly after training. This difference can be more noticeable when the training dataset is very small.

Table 4. Average precision of models with different object detectors at different growth stages (%).

Growth Stage	Faster R-CNN		YOLOv2		RetinaNet	
Model	Filtered	Not Filtered	Filtered	Not Filtered	Filtered	Not Filtered
1	66.04	53.82	90.91	97	82.97	98.43
2	64.83	65.38	85.37	73.76	93.87	92.63
3	77.29	73.56	95.5	89	90.2	81.39
4	64.99	72.08	91.28	83.49	79.26	83.97
5	43.98	14.47	90.14	55.99	73.24	82.37
All	52.81	65.93	89.9	N/A[a]	84.24	83.68

[a] YOLOv2 unable to successfully complete training without memory error.

Figure 6. Examples of errors in detection in late growth stages.

Adding images with more variability can enhance the predictive power of a model. The differences between filtered and unfiltered AP results were caused by rotated images. The impact of image rotation was not completely generalizable across all object detectors and models. For YOLOv2, all models performed better when filtered for rotation, except for model 1, which was based on the earliest growth stage and had the least number of bounding boxes. As the number of bounding boxes and the variability among target areas in the images increased, the impact of rotation was more pronounced, and AP decreased. The greatest difference occurred for model 5, for the last growth stage, with a difference in AP of 34.2 percentage points. The least difference in AP occurred for model 1, with a difference of 6.1 percentage points. The AP values for RetinaNet showed less variability between the filtered and unfiltered datasets, and less repeatability in which datasets produced better results. Model 5 produced the greatest difference between filtered and unfiltered datasets (9.07 percentage points), but the unfiltered images performed better than the filtered images. The least difference occurred for model 2 (1.24 percentage points), and the filtered images improved the AP. All of the tested object detection algorithms implement pooling layers in their feature extraction process. The data used in training and the models themselves can satisfactorily handle the differences caused by scale, which should be minimal in this application.

Confusion Matrix

A confusion matrix describes the performance of a model with a testing dataset for which the true values are known. Tables 5 and 6 show the confusion matrices for the most precise models, RetinaNet at stage 1 and YOLOv2 at stage 1, in addition to the evaluation metrics calculated based on TP, FP, and FN. Based on the matrices for RetinaNet and YOLOv2 at stage 1, there were no false negatives (FN), which means that all damaged areas were detected and there were no missing objects. There were 5 and 6 false positives (FP), respectively, which were most likely due to changes in the ambient light, which creates complexity in differentiating the target object from the background. By adding more images from more flights, the detection error can be reduced. In addition, because the non-damaged areas cannot be quantified, true negatives (TN) are identified as N/A (for "not applicable").

Table 5. Testing dataset confusion matrix for RetinaNet at stage 1.

	Positive	Negative
True	24	N/A
False	5	0

TP Rate	Relative FP Rate	Accuracy	F1 Score
0.83	0.17	0.83	0.91

Table 6. Testing dataset confusion matrix for YOLOv2 at stage 1.

	Positive	Negative
True	23	N/A
False	6	0

TP Rate	Relative FP Rate	Accuracy	F1 Score
0.80	0.2	0.8	0.89

Discussion

Little attention has been paid to analytical tools for monitoring corn near physiological maturity until a severe weather event destroys fields and prevents them from being harvested. New data-driven tools, such as deep learning and

computer vision, can be deployed to identify the presence, and eventually the severity, of corn damage caused by severe weather. The resulting data can identify the areas of damage to be reported to farm owners, insurance companies, response agencies, or other relevant parties. These tools could enhance the awareness and resiliency of growers at the time of natural disasters. Corn damage could be detected in near real-time and provide meaningful information immediately after data collection. The expected end-point of this project is an on-line or edge-of-field data processing tool to which georeferenced imagery of a field area with suspected corn damage is uploaded. A corn damage object detection model is then applied to the uploaded images, and a report is generated that describes the percentage of the field that is damaged.

In the current study, all of the training and testing targets were rectangular areas of simulated corn lodging. Severe weather events typically produce damaged areas of complex shape. The objectives of this study were focused on assessing the feasibility of computer vision techniques to detect corn lodging, and simulating the damage in rectangular areas was the best approach available. Adding imagery from actual weather events or simulated damage of complex shapes and patterns will be necessary to develop more comprehensive models for identifying corn lodging caused by natural events. The current results indicate that such research and model development is warranted.

Zheng et al. (2018) compared different object detectors for identifying vegetable crops. Two of their tested object detectors were also evaluated in this study: Faster R-CNN and YOLOv2. In that earlier study, YOLOv2 performed better than Faster R-CNN, which is similar to the results of this study even though there were differences in conditions and imaging platforms. Zhou et al. (2016) used UAS imagery and vegetative indices from multispectral sensors to estimate hail damage in potato crops. Their results indicated that damage estimates were more accurate when they were closer in time to the damage event, which is supported by the results presented here. Silleos et al. (2002) also applied vegetative indices to detect crop damage but used space-based sensors. They indicated that their method could identify fields that needed further inspection but could not detect discrete crop damage due to the low resolution. The method described in this study identified specific damaged areas within a field. Kerkech et al. (2018) reported that UAS-based RGB images and deep learning approaches were useful for detecting diseases in vineyards. They also mentioned data shortage, which was also a concern in this study and can be addressed in the future by collecting more samples and enriching the image datasets.

Chu et al. (2017) used UAS imagery, an index-based technique, and structure-from-motion to determine the damage severity, with $R^2 = 0.50$. The results presented here suggest that computer vision and deep learning techniques could provide more accurate predictions of damaged areas from 2D images without 3D reconstruction and processing. In their study, Chu et al. (2017) largely focused on small plot areas used for phenotyping, rather than the large field areas targeted in this study. Zhao et al. (2019) developed a model to assess rice lodging based on a deep learning UNet architecture and UAS imagery. They randomly divided a single image into many small samples to generate the training dataset. In contrast, in our study, the data were collected at different stages during the growing season to consider temporal effects on the training and testing datasets. In summary, previous work by other researchers indicates the need for crop damage detection and reveals a gap in high-resolution automated detection, which can be filled by computer vision and deep learning techniques that can provide more accurate predictions of crop damage.

Limitations

A challenge identified across the object detectors compared in this study was the decrease in performance as the corn grew, entered senescence, and became desiccated. The experimental design used in this study allowed natural variations and treatment variations to be captured. Visual observation of the corn indicated that most of the field was green at stage 1, and the damaged areas were brown in color. At stage 5, most of the field was brown, along with the damaged areas, and there was less variation in the imagery to allow damaged areas to be distinguished from non-damaged areas. This finding suggests that automated corn damage detection will perform best if damage occurs earlier in the growing season and that data collection as soon as possible after the damage event will improve the accuracy of damage detection. Beyond typical seasonal variation, weeds began growing in the plots where damage were simulated, which made segmentation more difficult because the non-target plants created challenges for the object detection models. This finding is important because such weed growth would occur under actual conditions and will need to be managed to increase damage detection accuracy.

Conclusions

The main goal of this study was to test the feasibility of remotely detecting areas of lodged corn using a UAS and an affordable, easy-to-access RGB sensor. Computer vision tools and deep learning algorithms were assessed for their ability to identify corn damage. Three advanced object detectors, pre-developed for non-agricultural applications, were deployed. They were trained to identify corn damage in images at different late-season growth stages. RetinaNet and YOLOv2 demonstrated robust capability for corn damage identification, with AP values across all conditions ranging from 98.43% to 73.24% and from 97.0% to 55.99%, respectively. Faster R-CNN did not perform as well as the other two object detectors, with AP between 77.29% and 14.47% for all conditions. Filtering out images that were taken while the UAS was turning improved the YOLOv2 models, and the maximum improvement was 34.15 AP percentage points. The RetinaNet models were least sensitive to rotated images, and the best results were not consistent between including and excluding the rotated images. Minimizing data preprocessing prior to training is preferred, and RetinaNet performed the best of the three object detection approaches without filtering. Corn lodging was best detected before crop senescence and desiccation were advanced, and

the performance of all three object detectors decreased as the crop advanced in age.

Future Work

In future work, the current dataset will be extended by creating additional simulated data and including imagery caused by natural events. A replicated, randomized complete block study with the same treatments presented previously will be established at two locations. The additional data should allow data segmentation as well as corn damage detection. Different damage modes or severity levels should be detectable when a more diverse dataset is created.

References

Alsalam, B. H., Morton, K., Campbell, D., & Gonzalez, F. (2017). Autonomous UAV with vision based on-board decision making for remote sensing and precision agriculture. *Proc. IEEE Aerospace Conf.* Piscataway, NJ: IEEE. https://doi.org/10.1109/AERO.2017.7943593

Amiri, H., Miller, T., & Savova, G. (2017). Repeat before forgetting: Spaced repetition for efficient and effective training of neural networks. *Proc. Conf. on Empirical Methods in Natural Language Processing* (pp. 2401-2410). Stroudsburg, PA: Association for Computational Linguistics. https://doi.org/10.18653/v1/D17-1255

Cangialosi, J. P. (2017). Tropical cyclone report: Tropical storm Arlene. Miami, FL: NOAA National Hurricane Center. Retrieved from https://www.nhc.noaa.gov/data/tcr/AL012017_Arlene.pdf

Chu, T., Starek, M. J., Brewer, M. J., Murray, S. C., & Pruter, L. S. (2017). Assessing lodging severity over an experimental maize (*Zea mays* L.) field using UAS images. *Remote Sensing, 9*(9), 923. https://doi.org/10.3390/rs9090923

Dauphin, Y., De Vries, H., & Bengio, Y. (2015). Equilibrated adaptive learning rates for non-convex optimization. *Advances in Neural Information Processing Systems 28 (NIPS 2015)* (pp. 1504-1512).

Everingham, M., & Winn, J. (2012). The PASCAL visual object classes challenge 2012 (VOC2012) development kit. Retrieved from http://host.robots.ox.ac.uk/pascal/VOC/voc2012/devkit_doc.pdf

Falk, K. G., Jubery, T. Z., Mirnezami, S. V., Parmley, K. A., Sarkar, S., Singh, A., ... Singh, A. K. (2020). Computer vision and machine learning enabled soybean root phenotyping pipeline. *Plant Methods, 16*(1), 5. https://doi.org/10.1186/s13007-019-0550-5

Hamidisepehr, A. (2018). Classifying soil moisture content using reflectance-based remote sensing. PhD diss. Lexington, KY: University of Kentucky, Department of Biosystems and Agricultural Engineering.

Hamidisepehr, A., & Sama, M. (2018a). A low-cost method for collecting hyperspectral measurements from a small unmanned aircraft system. *Proc. SPIE, 10664*. https://doi.org/10.1117/12.2305934

Hamidisepehr, A., Hamidisepehr, A., Sama, M. P., & Sama, M. P. (2018b). Moisture content classification of soil and stalk residue samples from spectral data using machine learning. *Trans. ASABE, 62*(1), 1-8. https://doi.org/10.13031/trans.12744

Hamidisepehr, A., Sama, M. P., Turner, A. P., & Wendroth, O. O. (2017). A method for reflectance index wavelength selection from moisture-controlled soil and crop residue samples. *Trans. ASABE, 60*(5), 1479-1487. https://doi.org/10.13031/trans.12172

Henderson, P., & Ferrari, V. (2016). End-to-end training of object class detectors for mean average precision. *Proc. Asian Conf. on Computer Vision (ACCV 2016)* (pp. 198-213). https://doi.org/10.1007/978-3-319-54193-8_13

Jayas, D. S., Paliwal, J., & Visen, N. S. (2000). Review paper (AE - automation and emerging technologies): Multi-layer neural networks for image analysis of agricultural products. *J. Agric. Eng. Res., 77*(2), 119-128. https://doi.org/10.1006/jaer.2000.0559

Kamilaris, A., & Prenafeta-Boldú, F. X. (2018). Deep learning in agriculture: A survey. *Comput. Electron. Agric., 147*, 70-90. https://doi.org/10.1016/j.compag.2018.02.016

Kazemi, V., & Sullivan, J. (2014). One millisecond face alignment with an ensemble of regression trees. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (pp. 1867-1874). Piscataway, NJ: IEEE. https://doi.org/10.1109/CVPR.2014.241

Kerkech, M., Hafiane, A., & Canals, R. (2018). Deep leaning approach with colorimetric spaces and vegetation indices for vine diseases detection in UAV images. *Comput. Electron. Agric., 155*, 237-243. https://doi.org/10.1016/j.compag.2018.10.006

Kirk, D. B., & Wen-Mei, W. H. (2016). *Programming massively parallel processors: A hands-on approach*. Burlington, MA: Morgan Kaufmann.

Koenig, K., Hofle, B., Hämmerle, M., Jarmer, T., Siegmann, B., & Lilienthal, H. (2015). Comparative classification analysis of post-harvest growth detection from terrestrial LiDAR point clouds in precision agriculture. *ISPRS J. Photogramm. Remote Sens., 104*, 112-125. https://doi.org/10.1016/j.isprsjprs.2015.03.003

Koirala, A., Walsh, K. B., Wang, Z., & McCarthy, C. (2019). Deep learning for real-time fruit detection and orchard fruit load estimation: Benchmarking of 'MangoYOLO'. *Prec. Agric., 20*(6), 1107-1135. https://doi.org/10.1007/s11119-019-09642-0

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NIPS 2012)* (pp. 1097-1105).

Li, H., Wu, Z., & Zhang, J. (2016). Pedestrian detection based on deep learning model. *Proc. 9th Intl. Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)* (pp. 796-800). Piscataway, NJ: IEEE. https://doi.org/10.1109/CISP-BMEI.2016.7852818

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)* (pp. 2999-3007). Piscataway, NJ: IEEE. https://doi.org/10.1109/ICCV.2017.324

Lu, H., Fu, X., Liu, C., Li, L.-G., He, Y.-X., & Li, N.-W. (2017). Cultivated land information extraction in UAV imagery based on deep convolutional neural network and transfer learning. *J. Mountain Sci., 14*(4), 731-741. https://doi.org/10.1007/s11629-016-3950-2

Ma, L., Shi, Y., Siemianowski, O., Yuan, B., Egner, T. K., Mirnezami, S. V., ... Cademartiri, L. (2019). Hydrogel-based transparent soils for root phenotyping in vivo. *Proc. Natl. Acad. Sci., 116*(22), 11063-11068. https://doi.org/10.1073/pnas.1820334116

Mahajan, S., Das, A., & Sardana, H. K. (2015). Image acquisition techniques for assessment of legume quality. *Trends Food Sci. Tech., 42*(2), 116-133. https://doi.org/10.1016/j.tifs.2015.01.001

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection. *Front. Plant Sci., 7*, article 1419. https://doi.org/10.3389/fpls.2016.01419

Nolan, A. P., Park, S., Fuentes, S., Ryu, D., & Chung, H. (2015). Automated detection and segmentation of vine rows using high-resolution UAS imagery in a commercial vineyard. *Proc. 21st*

Intl. Congress on Modelling and Simulation (pp. 1406-1412). Modelling and Simulation Society of Australia and New Zealand.

Rahnemoonfar, M., & Sheppard, C. (2017). Deep count: Fruit counting based on deep simulated learning. *Sensors, 17*(4), 905. https://doi.org/10.3390/s17040905

Rançon, F., Bombrun, L., Keresztes, B., & Germain, C. (2019). Comparison of sift encoded and deep learning features for the classification and detection of Esca disease in Bordeaux vineyards. *Remote Sensing, 11*(1), article 1. https://doi.org/10.3390/rs11010001

Rebetez, J., Satizábal, H. F., Mota, M., Noll, D., Buchi, L., Wendling, M., ... Burgos, S. (2016). Augmenting a convolutional neural network with local histograms: A case study in crop classification from high-resolution UAV imagery. *Proc. European Symp. on Artifical Neural Networks, Computational Intelligence, and Machine Learning* (pp. 515-520). Retrieved from https://www.elen.ucl.ac.be/Proceedings/esann/esannpdf/es2016-74.pdf

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, faster, stronger. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition* (pp. 7263-7271). Piscataway, NJ: IEEE. https://doi.org/10.1109/CVPR.2017.690

Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. Retrieved from https://arxiv.org/abs/1804.02767

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 (NIPS 2015)* (pp. 91-992).

Reyes, A. K., Caicedo, J. C., & Camargo, J. E. (2015). Fine-tuning deep convolutional networks for plant recognition. In *CLEF2015 Working Notes*. *CEUR Workshop Proc., 1391*, 467-475. Retrieved from http://ceur-ws.org/Vol-1391/121-CR.pdf

Rhu, M., Gimelshein, N., Clemons, J., Zulfiqar, A., & Keckler, S. W. (2016). vDNN: Virtualized deep neural networks for scalable, memory-efficient neural network design. *Proc. 49th Annual IEEE/ACM Intl. Symp. on Microarchitecture (MICRO)*. https://doi.org/10.1109/MICRO.2016.7783721

Sammouda, R., Adgaba, N., Touir, A., & Al-Ghamdi, A. (2014). Agriculture satellite image segmentation using a modified artificial Hopfield neural network. *Comput. Human Behavior, 30*, 436-441. https://doi.org/10.1016/j.chb.2013.06.025

Silleos, N., Perakis, K., & Petsanis, G. (2002). Assessment of crop damage using space remote sensing and GIS. *Intl. J. Remote Sensing, 23*(3), 417-427. https://doi.org/10.1080/01431160110040026

Stewart, S. R., & Berg, R. (2019). Tropical cyclone report: Hurricane Florence. Miami, FL: NOAA National Hurricane Center. Retrieved from https://www.nhc.noaa.gov/data/tcr/AL062018_Florence.pdf

Tripicchio, P., Satler, M., Dabisias, G., Ruffaldi, E., & Avizzano, C. A. (2015). Towards smart farming and sustainable agriculture with drones. *Proc. Intl. Conf. on Intelligent Environments* (pp. 140-143). Piscataway, NJ: IEEE. https://doi.org/10.1109/IE.2015.29

Vibhute, A., & Bodhe, S. K. (2012). Applications of image processing in agriculture: A survey. *Intl. J. Comput. Appl., 52*(2), 34-40. https://doi.org/10.5120/8176-1495

Zhao, X., Yuan, Y., Song, M., Ding, Y., Lin, F., Liang, D., & Zhang, D. (2019). Use of unmanned aerial vehicle imagery and deep learning UNet to extract rice lodging. *Sensors, 19*(18), article 3859. https://doi.org/10.3390/s19183859

Zheng, Y., Kong, J., Jin, X., Su, T., Nie, M., & Bai, Y. (2018). Real-time vegetables recognition system based on deep learning network for agricultural robots. *Proc. Chinese Automation Congress (CAC)* (pp. 2223-2228). https://doi.org/10.1109/CAC.2018.8623610

Zhou, J., Pavek, M. J., Shelton, S. C., Holden, Z. J., & Sankaran, S. (2016). Aerial multispectral imaging for crop hail damage assessment in potato. *Comput. Electron. Agric., 127*, 406-412. https://doi.org/10.1016/j.compag.2016.06.019

Zhou, Y., Srinivasan, S., Mirnezami, S. V., Kusmec, A., Fu, Q., Attigala, L., ... Schnable, P. S. (2019). Semiautomated feature extraction from RGB images for sorghum panicle architecture GWAS. *Plant Physiol., 179*(1), 24-37. https://doi.org/10.1104/pp.18.00974