Dandelion Image Classification

John Poirier
Department of Electrical Engineering
University of Vermont
Burlington, VT 05405
jlpoirie@uvm.edu

Samuel Starzec
Department of Electrical Engineering
University of Vermont
Burlington, VT 05405
Samuel.Starzec@uvm.edu

Abstract

This project trained several models on a manually collected and annotated field dandelion data set to identify dandelions within test images. Several models were tested, including logistic regression, XGBoost, YOLO, and DETR. While several of the tested models successfully classified images, it was found that the DETR model had the best performance by analyzing accuracy, precision, and recall.

1 Introduction and Problem Definition

Within the world of agriculture, autonomous robotics already exist to maintain large farms. However, there is a gap in the need for small-scale farms. Existing solutions are either too large for a small-scale farm or do not support the variety of crops found on smaller farms. Thus, this project seeks to solve the problem by implementing targeted removal, rather than targeted protection. For this report, the specific problem of identifying dandelions was explored. Machine learning is a good tool to solve this problem, as it will allow for live identification through video feed once a model is trained.

2 Related Work and Bibliography

2.1 Sensor Utilization

The primary type of sensor used in machine vision for this application is image sensing, specifically basic RGB cameras. A patent filed for selective treatment of crops using machine vision to control a system of herbicide application outlines the use of several imaging sensors to accomplish the task [1]. This patent cites the use of color sensors, visible light sensors, infrared sensors, LIDAR, three-dimensional sensors, and several others in order to capture an imaging treatment arrangement. This is a similar but more industrial application of the principles that this project seeks to use. The use of many types of image sensors creates a robust classification system with highly detailed data, but it is beyond the scope of this project. This project only utilizes an RGB+Depth camera. Another patent on hybrid vision systems for crop ground navigation also cites imaging sensors providing imaging to a machine learning model for crop classification [2].

2.2 Detection and Classification Algorithms

Patent [2] also outlines the use of a trained machine learning classifier in conjunction with the imaging sensors to identify targeted crops. Another patent on the detection and management of target vegetation using machine vision specifies the use of a deep learning artificial neural network (DLANN) model to accomplish this [3]. Common public applications of plant classification machine learning models include programs like iNaturalist and Pl@ntNet. iNaturalist lists TensorFlow, an open-source Python software library, as its machine learning package, which runs the model [4]. TensorFlow and similar packages, like PyTorch and Ultralytics, are used in this project for model

training. Both iNaturalist and Pl@ntNet use public databases to train their machine learning models, taking advantage of the scale of open-source projects where the community can include their own data [4], [5]. This project will not use its own public or open-source database but will instead utilize manual data collection for training of the machine learning model in order to closely replicate the view that the RGB camera will see when implemented in real-time. A paper from IEEE cites a convolutional neural network (CNN) for the use of plant classification [6], and another paper, which uses a 3D LiDAR as their machine vision sensor, cites using linear regression, support vector machines, and neural networks [7].

3 Model and Training Algorithm

This project solves a binary classification problem. The models access the collected dandelion data images and manual annotations, train from that data, and then, given a set of test images without annotations, predict if the image contains a dandelion. The predicted classifications are then compared to the manually annotated ground truth of the test images.

3.1 Logistic Regression

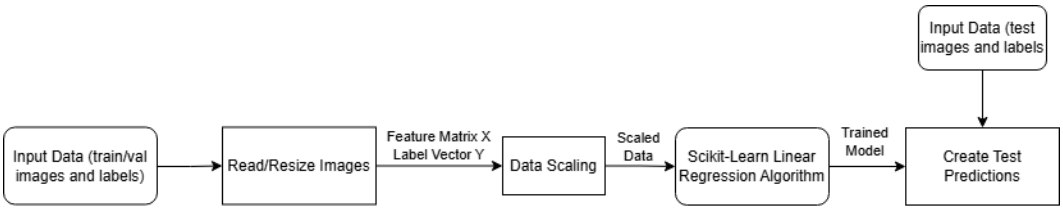


Figure 1: Logistic Regression Block Diagram

The most simple model trained for this project was a logistic regression model. This model utilizes SciKit-Learn's LogisticRegression function, which implements the standard logistic regression logic. This model initializes model weights, computes predictions via equations (1) and (2), measures the cross-entropy loss of the predictions, then repeats this process, iteratively adjusting weights to lower the loss. Once the loss converges, the model weights are saved to be used for testing.

$$z = \mathbf{w}^T \mathbf{x} \quad \text{(Apply Linear Regression Model)} \tag{1}$$

$$p = \frac{1}{1 + e^{-z}} \quad \text{(Apply Sigmoid Activation Function)} \tag{2}$$

K-folds cross-validation was also used on the logistic regression model. SciKit-Learn's StratifiedK-Fold was used for testing. This standard K-Folds algorithm splits the data into *k* number of "folds", which are sequentially used for cross-validation. This algorithm looks at the training image labels, and based on the user-defined number of folds, splits the data into that many folds for each of the classifications (for binary classification, it would create *k* folds for class 0 labels and *k* folds for class 1 labels). The folds are then combined, matching the class 0 and class 1 folds. This new segmented training data is then used to train logistic regression.

3.2 XGBoost

Figure 2: XGBoost Block Diagram

The second model explored was Extreme Gradient Boosting (XGBoost) classifier. XGBoost is a decision tree based learning algorithm. One issue that came up early on in the XGBoost implementation is its inability to process the image data. With that in mind a hybrid convolutional neural

network (CNN) pipeline was developed to convert each image into readable data for the model. As shown in **figure 2**, the pipeline starts by reading the data from the xml file that was exported from CVAT. For each image, binary labels were generated based on whether or not a dandelion was present. The images are then loaded and passed through the ResNet-50 CNN. ResNet-50 is used to extract the features from the images. These features include information such as the texture, shape, color patterns, etc. This creates a feature matrix which is then split into training, validation, and testing sets. XGBoost then begins training with each boost creating a new decision tree that attempts to correct the mistakes of the old. This is what makes it better than other decision tree methods. Early stopping is applied to prevent overfitting. Once completed the model is evaluated for precision, recall, and accuracy. To end the trained booster model is saved for future use.

3.3 You-Only-Look-Once (YOLO)

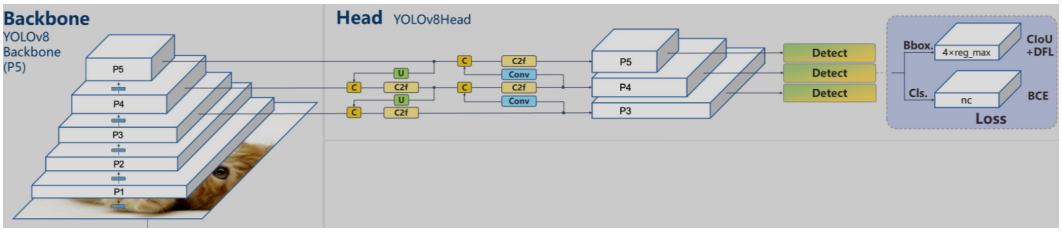


Figure 3: Simplified YOLO Block Diagram [8]

The third model explored was You-Only-Look-Once (YOLO). For this specific project, YOLOv8 was used. A simplified block diagram for the YOLO architecture is shown in **figure 3**. YOLO uses a backbone-neck-head structure. For YOLOv8, the backbone employs a CSPNet CNN designed to extract multi-scale features from data images. This backbone utilizes depthwise separable convolutions and other layers to optimize for speed and accuracy and reduce computational overhead. The neck of the YOLO structure utilizes a version of Path Aggregated Network (PANet) to enhance information flow across feature levels. This is beneficial for detecting objects of varying scale and size. The head of YOLO produces the output predictions, including confidence scores, bounding box coordinates, and class labels. This head uses an anchor-free method for bounding box predictions [9]. Notably, unlike logistic regression and XGBoost, YOLO is capable of predicting the actual bounding boxes of the detected dandelions, not just if the image contains a dandelion. This was not used in analysis for the sake of fair comparison, but is an important aspect of the model.

3.4 Detection Transformer (DETR)

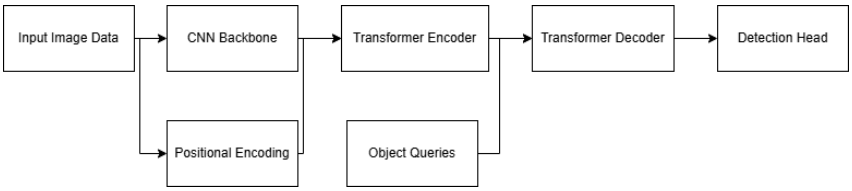


Figure 4: DETR Block Diagram

The final model explored was the Detection Transformer (DETR) model. The DETR architecture follows the flow of the block diagram shown in **figure 4**. DETR starts by passing each training image through a ResNet-50 CNN backbone. The CNN converts the raw image data into a lower-resolution feature map. Positional encodings are then added to preserve the spatial structure of the data, and tell the model where each feature is on the image. The spatial feature is then flattened to be received by the transformer encoder. The transformer encoder uses self-attention to process the image features. This is different from YOLOv8's, as it sees all of the data globally as opposed to YOLOv8's local method. The encoder output is then fed into the transformer decoder. The transformer processes a set of learned object queries that act as an input to the decoder to help it detect objects. The decoder outputs are then passed into the detection head. The detection head consists of a feed-forward network

(FFN) that predicts bounding boxes and class labels for each object query. During model training, the model receives the processed image tensor and COCO annotations to produce a custom dataset class. The final trained model outputs bounding boxes and class predictions without any anchor boxes [10].

4 Dataset

The dataset for this project was a set of manually collected images of dandelions in fields and small farms. These photos were taken with a RealSense d435i RGB+D camera at the UVM Horticulture Farm. In total, 724 images were manually taken of dandelions. Each of these images is 1280x720 pixels, per the camera specifications. The depth images from the d435i camera were unused for this project. The images were then manually classified, using bounding boxes to annotate where dandelions appear in the images. The organization of the labels depends on the model being trained. The total size of the images is approximately 1.3 GB. Although 724 images is not necessarily a large number of images for general image classification applications, since the models are being trained on one specific object, it is sufficient for this project. After splitting the data into training, validation, and testing sets, the sets had 506, 73, and 145 images, respectively.

5 Experimental Evaluation

5.1 Evaluation Methodology

The criteria being evaluated in this project are model accuracy, precision, and recall. These metrics were chosen due to their representative nature for binary classification. Importantly, the most important factor is a high precision value (i.e. low false positives). This is because false positives pose a safety concern, as the robot would attempt to actuate on a target that is not the true desired target. For testing in this project, models were tested on if they correctly predicted whether or not an image had a dandelion. Confusion matrices were used to visualize the performance of the models. Given these metrics being tested, confusion matrices are an ideal representation.

5.2 Results

The confusion matrices of the testing for each of the models are shown in **figure 5**. The metrics of the results are also summarized in **table 1**.

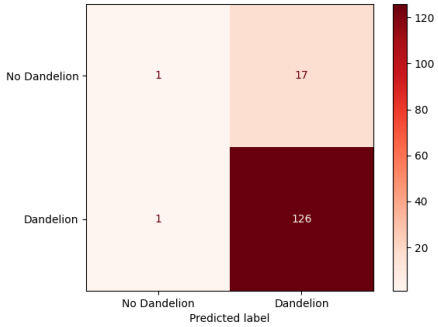
Table 1: Model Evaluations

	Model				
Measure	Logistic Regression	Logistic Regression (with K-Folds)	XGBoost	YOLO	DETR
Accuracy	87.6%	88.5%	91.0%	93.1%	92.5%
Precision	88.1%	89.0%	89.8%	96.1%	98.4%
Recall	99.2%	99.4%	100.0%	96.1%	93.0%

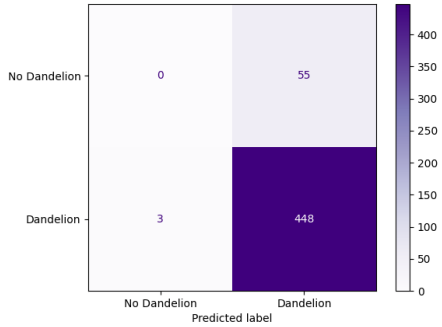
These results show successful classification from all models, with particularly strong performance from YOLO and DETR. As previously noted, it is also important to recognize that, although not tested here, logistic regression and XGBoost are not capable of predicting bounding box locations, while YOLO and DETR are.

5.3 Discussion

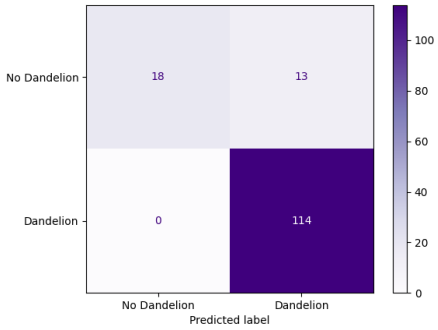
DETR has shown significant improvement when compared to the previous iterations in this project. With the inclusion of backbone freezing, a warm-up phase, more training epochs, more data, and augmentation, it has become the best model at predicting binary classification and the bounding boxes for the dandelions. For the newest iteration, DETR was trained with 120 epochs; for the first 20 of those epochs the backbone was frozen and the next 5 were part of the warm-up phase. During the warm-up phase, the learning rate was gradually increased until its maximum value. This ensured that the model would remain stable for the rest of the training process. Augmentation was used to



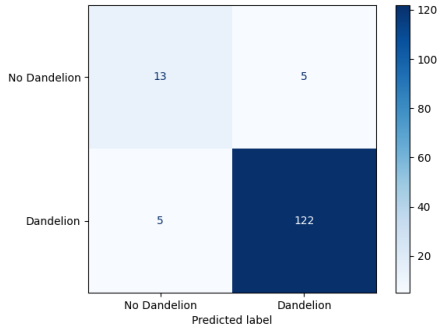
(a) Logistic Regression



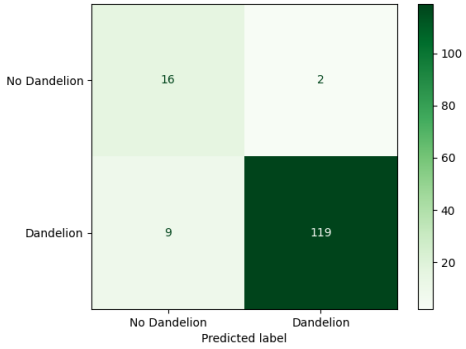
(b) Logistic Regression with K-Folds



(c) XGBoost



(d) YOLO



(e) DETR

Figure 5: Confusion Matrices

change the brightness, contrast, saturation, and hue of the images. As all of the data was captured on the same day on the same field, augmentation was necessary to simulate different weather and situations. Each of the 120 epochs had variations of the same training and validation images.

YOLO still performed strongly, however, when the optimizations to DETR were implemented, it got outpaced. That being said, YOLO also trained significantly faster than DETR and on fewer epochs to get only a slightly worse performance. Augmentation and other optimization methods used in DETR were not implemented into YOLO. If those changes were to be implemented, it is possible that YOLO may outperform DETR.

Linear Regression and XGBoost both performed poorly, however, that was expected given the nature of this project. Since the only data given were RGB values (no data like petal length, leaf width, etc. were available), the simpler linear models were expected to do poorly. Even with implementing the CNN hybrid model for XGBoost, the performance was only marginally improved. Similarly, K-Folds cross-validation only slightly improved the performance of the logistic regression model.

Implementing K-Folds into the other models may provide a small increase in performance, but may not be worth the additional computing power.

6 Next Steps

As DETR was the best model it will undergo more extensive training by expanding the diversity of the dataset. A method of collecting more data includes allowing the autonomous weeding robot to automatically collect data in a field for long periods of time in order to collect samples under different conditions. This can start a semi-autonomous process in which DETR labels the images with bounding boxes, these boxes are then put through human verification. This will expedite the annotation process and generate a much larger dataset in a shorter amount of time. With this new data, the model will also be trained for longer and the training parameters will be experimented with to find the optimum values. Other DETR variants will also be explored and active learning will be incorporated into the model. This means that the images with a high uncertainty will be sent back to be manually labeled and put through training to accelerate the models growth.

Although DETR performed the best in the scope of this project, YOLO still shows promise. Next steps for YOLO would be to implement similar optimizations to the data and training that were implemented for DETR. Additionally, K-Folds and other validation methods can still be tested on both YOLO and DETR. YOLO would also significantly benefit from a larger and more diverse dataset.

7 Code

The code used in this project can be found at the following GitHub Repository.

Note: This repository is the full repository for another project, which this project was a subsection of. All code used for this project are found in the "Images" folder (for the data and data annotations) and the "Machine_Learning" folder. Additionally, some files, namely the YOLO and DETR model files, have been omitted due to GitHub file tracking size restrictions. All scripts and files necessary to recreate these missing files are present in the repository.

8 Student Roles

Poirier's roles include data collection, annotation, and model training and testing. Specifically, Poirier collected the manual data using the RealSense camera and contributed to the annotation and organization of data, manually identifying dandelions in the images. Poirier has primarily worked on the YOLO model, logistic regression model, and K-Folds cross validation.

Starzec's roles include data annotation and model training and testing. Specifically, Starzec contributed to manually annotating data by identifying dandelions in the images and organizing the data as needed. Starzec has primarily worked on the DETR and XGBoost models.

9 Conclusion

From the reported results, it can be concluded that, because of the specificity of the environment and situation of the dataset for this problem, it turns out to be a relatively difficult problem. Most specifically, because the available data only contains RGB images with no features other than channel/pixel values, it becomes difficult for simpler models to identify the dandelions. Both YOLO and DETR have been shown to be strong options for models, even with the remaining room for improvements.

References

[1] Yoav Halevi Itzhak Khait, Alon Klein Orbach. System and method for selective treatment of crops using machine vision. U.S. Patent, 2024. Issued by the United States Patent and Trademark Office.

[2] M·. Hybrid vision system for crop ground navigation. China Patent, 2023. Issued by the China Patent and Trademark Office.

[3] Jialin Yu Arnold W. Schumann, Nathan S. Boyd. Detection and management of target vegetation using machine vision. U.S. Patent, 2018. Issued by the United States Patent and Trademark Office.

[4] iNaturalist. inaturalist computer vision explorations. https://www.inaturalist.org/pages/computer_vision_demo, 2025. Accessed: 2025-02-09.

[5] Pl@ntNet. Open data. https://plantnet.org/open-data/, 2025. Accessed: 2025-02-09.

[6] Salar Razavi Hulya Yalcin. Plant classification using convolutional neural networks. In *2016 Fifth International Conference on Agro-Geoinformatics (Agro-Geoinformatics)*, 2016.

[7] Stefan Laible Karsten Bohlmann Andreas Zell Ulrich Weiss, Peter Biber. Plant species classification using a 3d lidar sensor and machine learning. In *2010 Ninth International Conference on Machine Learning and Applications*, 2010.

[8] Ultralytics. Yolov8 anchor-free bounding box prediction - issue 189. https://github.com/ultralytics/ultralytics/issues/189, 2023. Accessed: 2025-02-09.

[9] Muhammad Yaseen. What is yolov8: An in-depth exploration of the internal features of the next-generation object detector. https://arxiv.org/html/2408.15857v1#bib.bib14, 2024. Accessed: 2025-02-09.

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers, 2020.