StrawberrAI: Strawberry Classification using a Convolutional Neural Network

Ivan Liongson
Stanford University
ivanlion@stanford.edu

Erik Luna
Stanford University
eluna1@stanford.edu

Abstract

Strawberry shape and ripeness are critical in agricultural quality control, as it influences consumer preferences and marketability. This project presents a two-step approach to first detecting strawberries and then classifying them by shape and ripeness using a custom Convolutional Neural Network (CNN). We used YOLOv7 to object detect strawberries against a black background found in "Classification and Quantification of Strawberry Fruit Shape". After which, we utilized labeled datasets of shapes and ripenesses, and applied our custom CNN to apply to the individual strawberry images segmented by YOLOv7. We processed raw RGB images to isolate individual strawberries, standardized their dimensions for model, and performed data augmentation to have a balanced distribution across all classes.

Our method involves a CNN architecture for improved accuracy over simpler modeling techniques. Preliminary results demonstrate a performance increase, achieving a test set accuracy of 0.90 after 5 epochs, compared to the baseline K-nearest neighbor classifier accuracy of 0.71.

Our baseline for strawberry detection was the YOLOv7 base model which did not have a strawberry class, and thus 0 accuracy. After finetuning YOLOv7 on our dataset of boxes, we achieved near perfect detection.

After the strawberries were identified and isolated, we extracted those bounding boxes and classified them by shape and ripeness. The shape classification model was trained on the original strawberry extractions, and the ripeness model was trained on another dataset. We combined these to perform a multitask detection and classification. Finally, we evaluated the performance of our pipeline by measuring the accuracy of each individual task as well as the overall efficiency of the end-to-end segmentation and classification process. Our final results demonstrated a 100% detection accuracy with YOLOv7, a 95% accuracy for the shape classifier, and an 85% accuracy for the ripeness classifier.

1. Introduction

The strawberry, known as the "Queen of Fruits", comes in different shapes and sizes as it is picked from the field. Strawberry shape, ripeness, and size are visual features that consumers look into when picking their produce. Strawberry classification by shape is important for quality control, as consumers prefer purchasing more desirable strawberry shapes. Based on their visual appearance and quality, strawberries may be sorted for direct sale to consumers at markets, or sorted for alternatives such as jam production. Shape can even be indicative of different strawberry varieties or suggest the presence of some diseases. While some machines learning models have attempted to automate this process, none have used ConvNets to solve this problem. We plan to create a multitask classifier that performs object detection and classification. We used a dataset that has labeled strawberries by shape and another that has strawberries by ripeness. We finetuned YOLOv7 to detect strawberries in a picture, and from those extracted pictures, we classified them based into 10 shapes and 6 ripeness levels. For our project, classes will range from 1 through 10, with each representing different shapes along a general continuum of wider-than-taller or taller-than-wider. Our baseline model was a sklearn library KNN model for 10 shape classes. Finally, we measure the pipeline's performance through individual task accuracy and an end-to-end segmentation and classification pipeline. For our final results, we achieved a YOLOv7 100% detection accuracy, 95% shape classifier, and a 85% ripeness classifier test accuracy.

2. Related Work

We are classifying strawberries by shape given a dataset of strawberry pictures split into 10 classes based on their geometric qualities and by ripeness based on a dataset of ripeness scores. Our objective is developing a high accuracy finetuned YOLO model, a shape classifier and a ripeness classifier. Much of these tasks of finding strawberries are manual tasks and not much implemented automation exists. We looked into existing work for classifying strawberries by shapes and ripeness to automate the process.

2.1. General machine learning approaches

Furthermore, we reviewed a Random Forest classifier for the strawberries dataset [3]. Another paper also uses Random Forest and SVM classification based on dimensional measurements of strawberry shapes into 9 classes with promising results[5]. These classes were defined by their measurements, including curvature and dimensions and relate to our project since they inspired our 10 class system we chose. Additionally, papers have investigated using CNNs for disease identification but not for shape classification [4]. Compared to previous models focused on measurements taken from the images, the CNN captures smaller granular data like texture which our model could learn from to make decisions on the shape. While these machine learning approaches result in good accuracy, we plan to perform even better through a CNN architecture.

2.2. CNN approaches

A state-of-the-art model, YOLO, is good at varied object detection [7]. In general, this is very accurate and very quick during inference. These quick inference times could could help agricultural farmers look through their strawberries and determine their ripeness and size for picking. Additionally, Current pretrained YOLOs exist, such as that one pretrained on the COCO dataset [6]. A downside to this dataset is that it does not have any strawberry class.

More specifically, a paper looks at strawberry field detection from pictures of plants from a birds-eye view from a drone which are processed with YOLOv2 [1]. They run these images through YOLO to detect strawberries and classify their maturity levels. The maturity classification was 0.88 for a test data set at 2 meters from the ground. This effectively does two tasks: object identification and classification and proves the versatility of YOLO. Some downsides are that this was a less accurate YOLO model and that the data only has three classes: flower, immature and mature ripeness. We are tackling 6 classes of maturity and 9 shapes, so there is less delta between classes. This is also an outdated YOLO model but we can look into using a more recent one, such as v7, which performs better for detection [2]. As a downside, there are no publicly available pretrained weights that locate strawberries.

3. Data

Our data involved two original datasets, one for shapes of strawberries and another for ripeness levels of strawberries. Post generating one of our own after outputting it from YOLO. Further processing of our dataset for the YOLO model is discussed in the Methods section.

Figure 1. Feldmann shape dataset example

3.1. Shape Dataset

Our base data was the "Classification and Quantification of Strawberry Fruit Shape" dataset by Feldmann. [3]. We are working directly from raw RGB images containing one to three different strawberries taken against a black background. Each image is identified by a plot ID and progeny ID number. Feldmann provides a features csv file which contains mathematically extracted features for each strawberry shape in addition to their shape classification labels. After processing, our train, validation, and test counts were were 5440/680/686. The images were JPG in full color RGB as in Fig. 1. Feldmann's original pictures were 3840x2160 and after processing, they were resized to 1000x1000 including black padding for the smaller dimension.

There were around 2,000 original images in this format:

In order to use the labels Feldmann provides, we had to extract individual strawberries from the raw RGB images. Because there are thousands of raw images with each containing multiple strawberries, we wrote a script using the OpenCV library to autmatically rotate images, crop out large unnecessary regions, mask out red objects, and identify contours for strawberry shapes. We use these contours to save every isolated strawberry image, and then manually removed any remaining false positive images. Fig. 1 shows contours and individual extracted fruits on a sample raw image. Each single strawberry has an associated fruit number that uniquely identifies it within the csv. We were able to map strawberries to their fruit number based on their relative x coordinate position within each raw image. Finally, we performed data augmentations to even out the class distributions. As shown by the bar plot in Fig. 2, the largest class had over 800 points while the smallest had around 400. To minimize the destructiveness or interference of our artificial augmentations, we decided to create new data points by horizontally flipping images. We selected a random subset of images to flip based on the number of extra images required to match the largest class size. After data augmentation, we we had equally sized classes. We chose to normalize the image scale since shape classification is not

Figure 2. Ripeness dataset example

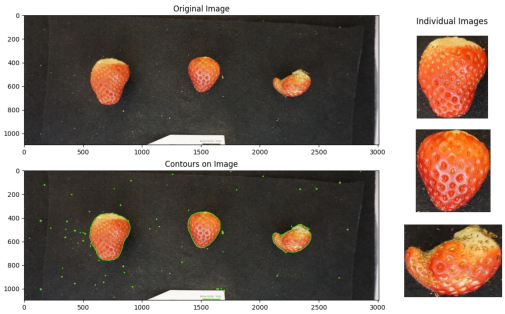

Figure 3. Strawberry contour detection by red color thresholding. Above is the original image, below shows the drawn on contours. Small artifacts were removed by setting a size threshold. On the right are the extracted individual strawberry images.

dependent on the relative size of a given strawberry.

3.2. Ripeness Dataset

We gathered a ripeness dataset called Strawberry-DS which had 247 pictures of plants which a varying number of fruits per picture, totaling over a thousand strawberries. We split for this dataset into the ratio 80/10/10. Likewise, for this ripeness dataset, the original pictures were 3840x2160, and after processing, they were resized to 1000x1000 including black padding on the smaller dimension. This data was gathered by the Central Laboratory for Agricultural Climate in Egypt [**?**].

4. Methods

4.1. Shape and Ripeness Classifiers

To establish a baseline for our shape and ripeness classification tasks, we first trained a K-nearest neighbor (KNN) multiclass classifier using scikit-learn. For the purpose of computational efficiency at evaluation time, we shrunk each image from 1000x1000 to 64x64.

To improve upon our baseline, we created and trained a convolutional neural network (CNN) from scratch for both of the tasks. We reasoned that a CNN could learn more complex features within each image and also better handle images at the full 1000x1000 dimensions. The pre-preprocessed datasets are loaded using 'ImageFolder' from 'torchvision.datasets', which automatically assigns labels based on the name of image's folder. We use the 'DataLoader' method to create iterable data loaders, which Pytorch uses for batching and shuffling. After converting the images to PyTorch tensors, we use the 'transforms' library to normalize the pixel values using the mean and standard deviation of each channel for all the images in the ImageNet dataset.

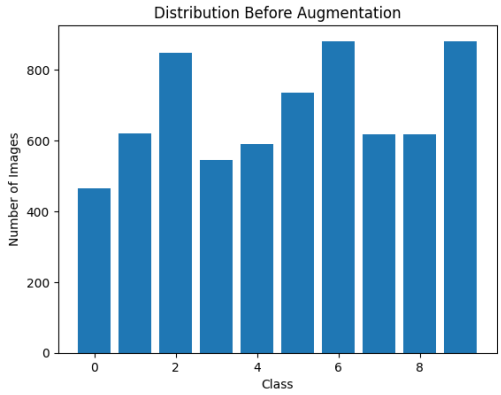

Figure 4. The distribution of the 10 strawberry classes before balancing the dataset. Augmented images from mirroring were added to the smaller classes to match the number of datapoints in the largest class.

Our base convolutional neural network (CNN) architecture used for image classification consists of two sets of convolutional layers each followed by the ReLU activation function and a 2x2 max-pooling layer. We used padding values of 2 first layer and 1 for the second layer to allow the convolution filter to be centered on all input pixels including those at the border. The convolutional layers were followed by a flatten layer to convert into a one-dimensional tensor, and finally a fully connected layer with K units to correspond

We use Stochastic Gradient Descent (SGD) to optimize the model. The learning rate is set to 1×10^{-3} and the momentum to 0.95. Compared to standard gradient descent, SGD is more memory efficient for larger datasets such as our set of thousands of strawberries. Additionally, the stochasticity introduced by SGD helps the model better generalize to unseen samples. Finally, because SGD results in noisier updates, the updates more easily escape local minima and saddle points, plus the model converges faster at the beginning of the training process with the number of updates being made.

We also employ Nesterov momentum beyond the standard gradient descent algorithm, as shown in equations 1 and 2. In general, momentum accelerates training and dampens excessive oscillation in regions of high curvature; i.e., momentum smoothes out the training trajectory. Compared to standard momentum, Nesterov Momentum incorporates a look-ahead mechanism by computing the gradient

at the position where the momentum term takes the parameters. This better dampens oscillations and can also act as an implicit form of regularization to reduce overfitting in noisier data.

$$V_t = \beta V_{t-1} + \alpha \nabla_w L(w - \infty, X, y) \quad (1)$$

$$W = W - V_t \quad (2)$$

We use a standard CNN model training loop by performing the forward pass, calculating the cross-entropy loss between predictions and the true labels, performing backpropgation to calculate the gradients in the backward pass, and then updating the model parameters using the computed gradients. After each epoch, the model's accuracy is evaluated on the validation set by calculating the percentage of correctly classified images. Once model training is complete, we check its performance on the test set as well.

4.2. YOLOv7 Fine Tuning

To perform the image segmentation step to isolate individual strawberries from a raw image of multiple strawberries against a black backward, we decided to use the YOLOv7 image detection model. YOLOv7 is pretrained on labels from the COCO dataset, which had 80 classes; however, because none of the data points were labeled as strawberries, this makes the default model weights unsuitable for our task of strawberry segmentation. Attempting to run inference using the provided 'YOLOv7.pt' weights resulted in output images with no labels or bounding boxes annotated at all.

Thus, we had to fine tune YOLOv7 using our own strawberry image data to adapt the model to our specific task. Specifically, we performed an instance of transfer learning since we are applying the existing YOLO model to a new task, with the reasoning that it is already capable of general object detection even if it does not specifically know how to classify strawberries yet.

To perform the fine tuning/transfer learning process, we used the existing Feldmann dataset. These strawberry group images were unlabeled without bounding boxes, so we leveraged our existing the countour detection algorithm discussed in the preprocessing section and shown in figure 3 to generate the bounding boxes. We simply had to convert the extracted pixel coordinates into the YOLO annotation format, which consists of the x origin, y origin, x width, and y height. All boxes were assigned the same class label since we are only trying to detect strawberries.

The actual fine tuning process itself was done using YOLOv7's provided 'train.py' code. We adapted our dataset to match their specific data formatting and hierarchical requirements. Additionally, we had to define our own .YAML configuration file to define the data structure as well as set the training hyperparameters such as learning rate and momentum, in addition to augmentation parameters to help make the training more robust.

4.3. Full Pipeline: from Segmentation to Classification

After training and testing our classifiers and the fine-tuned YOLOv7 model, our final step was to integrate these models into a single pipeline that would handle detection, segmentation, and the multi-task classification. This process primarily meant extracting the bounding-box labeled images found by YOLOv7, preprocessing the images to match the format needed for our classifiers, and then running the classifiers to output classes for use in our final image labels.

We were able to run image segmentation inference using YOLOv7's provided 'detect.py' script. Our first attempts at integrating YOLO with our classifiers involved modifying this 'detect.py' script and its two-stage classifier code which is intended to further filter on YOLO's labels with a secondary classifier. We reasoned we could replace rather than filter the output labels to show our classifier's result. Although we were able to successfully load and call our models within this framework as well as assign the model labels to bounding boxes, we were concerned by the accuracy of this approach because of the numerous conversions between YOLO's annotation format (origin coordinates, width, and height) and the pixel bounding box itself.

Thus, the ultimate method we settled on was using YOLO's extracted annotations in conjunction with the raw images to cut out the individual strawberries from the raw file before passing into our preprocessing pipeline and calling each model as standard. This also gave us the flexibility of formatting and adjusting bounding box text label positions as needed.

5. Experiments and Results

For our baseline KNN model on the shape dataset, we were able to achieve a training set accuracy of approximately 0.82 and a test set accuracy of 0.71 on Feldmann's 10-class labels. We used k = 5 as the neighbor hyperparameter. We also calculated precision and recall for each of the 10 classes and found noticeably worse metrics for certain classes over others. For example, class 5 had 0.62 and 0.64 for precision and recall respectively. In contrast, class 1 had 0.86 for precision and 0.76 for recall. Meanwhile, the baseline KNN for the ripeness dataset was only able to achieve a training accuracy of 0.67 and a test accuracy of 0.49.

To maximize the performance of our convolutional neural networks, we performed hyperparameter tuning for parameters including learning rate, number of convolutional channels, etc., using the Optuna library. This served as our primary experimental iteration process in an attempt to improve the models. Optuna is a hyperparameter optimization

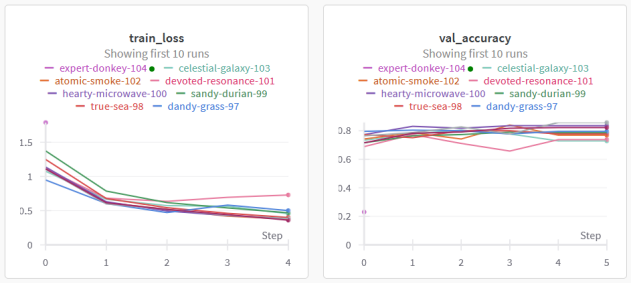

Figure 5. Loss and accuracy plots across OpTuna hyperparameter tuning trials as visualized using the Weights Biases platform.

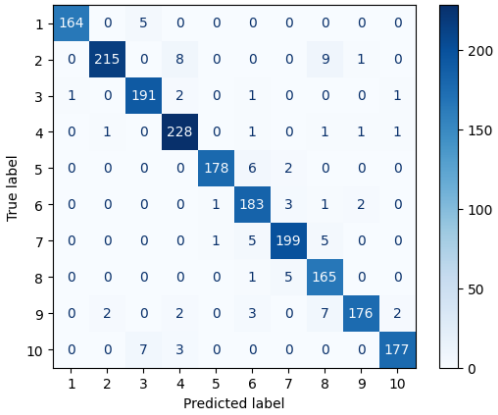

Figure 6. Multi-class confusion matrix showing performance of our CNN on the shape test dataset.

framework[1] that uses the Tree-structured Parzen Estimator (TPE) sampler to better infer the next set of parameters to test. Overall, this results in a more efficient hyperparameter search than with more naive methods such as grid search or random search. Figure 5 shows loss and accuracy plots over time/epoch for each of the OpTuna trials, with each trial testing a new set of hyperparameters. After finding the best parameters, we retrained the model using the new configuration and saved the weights for evaluation as well as input into our final pipeline.

Upon completing hyperparameter tuning and training the models, we were able to achieve a shape classification accuracy of 0.9537 and a ripeness classification accuracy of 0.85, significantly improving upon the baselines established by our KNN classifiers. Figure 6 and figure 7 show the confusion matrices for shape and ripeness classification respectively.

The YOLOv7 fine-tuning process proved straightforward, as only 5 epochs of fine tuning allowed us to achieve near-perfect precision and accuracy on the validation set. Upon loading the stored 'best-model.pt' weights and running our test dataset, we saw no errors in segmentation or drawing the bounding boxes.

[1] https://optuna.readthedocs.io/en/stable/

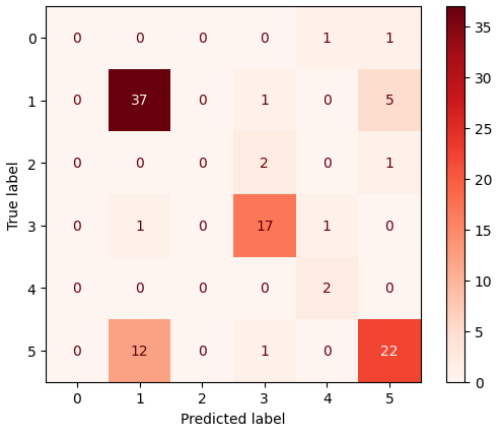

Figure 7. Multi-class confusion matrix showing performance of our CNN on the ripeness test dataset.

Once we saved our best model weight dictionaries, we integrated them into our pipeline and ran inference on both the existing Feldmann multi-strawberry images as well as images we took ourselves. Figure 8 shows our own image of five strawberries, segmented by the fine-tuned YOLOv7 model and classified based on shape and ripeness using our CNN classifiers. Qualitatively, we found that the pipeline worked well on our store-bought fruit. Upon examining the reference class labels, our strawberries appeared to be of similar shape classes and ripeness stages.

6. Summary and Conclusion

This project implemented an end-to-end pipeline classifying strawberries by shape and ripeness using a Convolutional Neural Network (CNN). We utilized YOLOv7 for object detection to isolate individual strawberries in images and then applied custom CNN models to classify the isolated strawberries based on their shape and ripeness. Our approach significantly outperformed baseline KNN models, achieving a shape classification accuracy of 0.9537 and a ripeness classification accuracy of 0.85.

The success of the models demonstrates the significant potential of deep learning for agricultural quality control, providing a reliable tool for automating the classification process. This highlights the potential of deep learning in agricultural applications, offering a tool for automating strawberry shape classification. This automation can greatly improve the efficiency and accuracy of sorting and grading strawberries, benefiting both producers and consumers.

Our future work would focus on enhancing the model to manage more complex scenarios, such as multiple strawberries in a single image and varying degrees of noise from

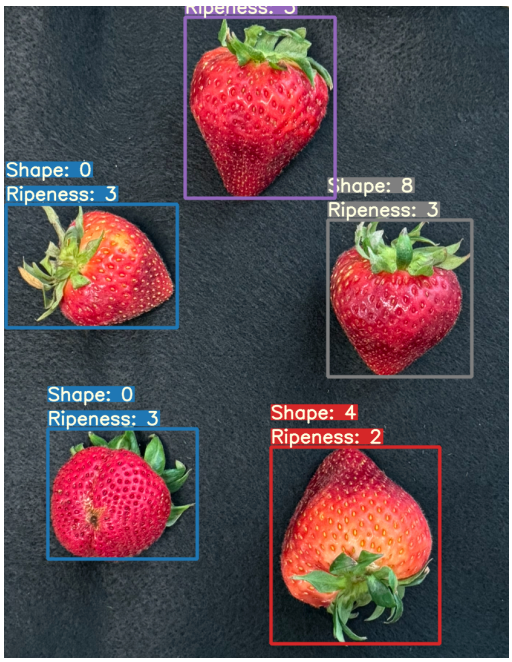

Figure 8. Example final output of the strawberry segmentation and classification pipeline, performed using our own strawberry image.

leaves or other fruits. Additionally, investigating the integration of other deep learning methods and larger datasets could further improve the model's performance and application scope. This kind of research will lead to more intelligent and automated solutions in the field.

References

[1] H. Chen, J. Zhang, K. Xu, and Y. Liu. Deep learning for real-time object detection and classification in surveillance. In *Proceedings of the Journal of Surveillance*, 2021. 2

[2] A. Farhadi et al. Yolov7: Towards efficient and accurate object detection. In *Proceedings of the Advanced Vision and Pattern Recognition Conference*, 2021. 2

[3] M. J. Feldmann. Classification and Quantification of Strawberry Fruit Shape. Zenodo, Sept. 2019. 2

[4] X. Hu, R. Wang, J. Du, Y. Hu, L. Jiao, and T. Xu. Class-attention-based lesion proposal convolutional neural network for strawberry diseases identification. In *Frontiers in Plant Science*, volume 14, page 1091600. Frontiers Media SA, 2023. 2

[5] T. Ishikawa, A. Hayashi, S. Nagamatsu, Y. Kyutoku, I. Dan, T. Wada, K. Oku, Y. Saeki, T. Uto, T. Tanabata, S. Isobe, and N. Kochi. Classification of strawberry fruit shape by machine learning. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume XLII-2, pages 463–470, Riva del Garda, Italy, June 2018. ISPRS TC II Mid-term Symposium "Towards Photogrammetry 2020". This contribution has been peer-reviewed. © Authors 2018. CC BY 4.0 License. 2

[6] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pages 740–755, 2014. 2

[7] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2