EE292D Final Project: Grape Bunch Counting

Christos Polzak, Harry Chen

June 11, 2024

1 Motivation and Challenges

From an agricultural perspective, the purpose of counting berries throughout their development is clear: knowing how many berries are growing is valuable for predicting and estimating yield (Liu et al. (2020)). However, manually counting berries is evidently an extremely time-consuming and inefficient process, which perfectly poses an opportunity to use machine learning (ML) techniques to automatically count berries via an image processing pipeline. However, attempting to count berries automatically is a challenging task itself. Major roadblocks include the high density of the objects of interest within small visual and pixel spaces; large amounts of occlusion and color overlap between berries and due to surrounding leaves; variations in the size, color, and shape of the berries, both between species and throughout development; variations in lighting; and variations in the positions, size, scale, and density of different berry clusters, including in images that may have multiple clusters.

2 Methodology

To address the challenges posed in the previous section, we adopt several strategies. Firstly, we reduce the scope of the problem to focus on a much smaller subset of mature berries, namely, darkly-colored grapes (black and red). We hypothesized that grapes would be relatively easy to count, as individual instances have a smooth surface (as compared to, for instance, blackberries, where a single berry can be composed of many lumps) and are consistently a highly distinct color from the leaves surrounding them. Furthermore, grapes provided the most images available to us to generate a dataset. We then further reduced the scope to cases where the berry bunches are well-segmented, both for practical reasons and because it would be reasonable to run a berry bunch segmenting pipeline followed by our individual berry bunch counting algorithm in an actual deployed environment. Secondly, we leverage developments in the relatively well-studied crowd counting task, where the model is tasked with counting the number of people (usually by face) in an image. This provides several benefits: firstly, the field is more developed, likely producing better ways of approaching the counting task; secondly, the existence of larger and higher-quality datasets, as the task is more popular; and thirdly, the task is already intertwined with the limitations of edge hardware.

2.1 Model Architecture and Task Formulation

Drawing from crowd counting, we use crowd-counting P2PNet (point-to-point net), which was proposed by Song et al. (2021) and achieves state-of-the-art performance on the popular ShanghaiTech crowd counting benchmark, as well as having available released code. Under their approach, individual objects are tracked by assigning each one a center point, and the model is encouraged to place the correct number of points in the correct locations. They found this approach to be more robust and effective than other common (and often more indirect) dense object counting strategies, such as approximating from density maps or using segmentation to count objects; several works in the agricultural counting space adopt such strategies (Liu et al. (2018), Chen et al. (2017), Farjon et al. (2023)). An additional benefit of using P2PNet is the relatively simple labeling approach, which makes it easier to generate a custom dataset for a new domain.

In particular, P2PNet uses the following strategy (visualised in Figure 1). First, the backbone of a VGG-16 model is used to generate convolutional image features, which are then upsampled by a factor of two.

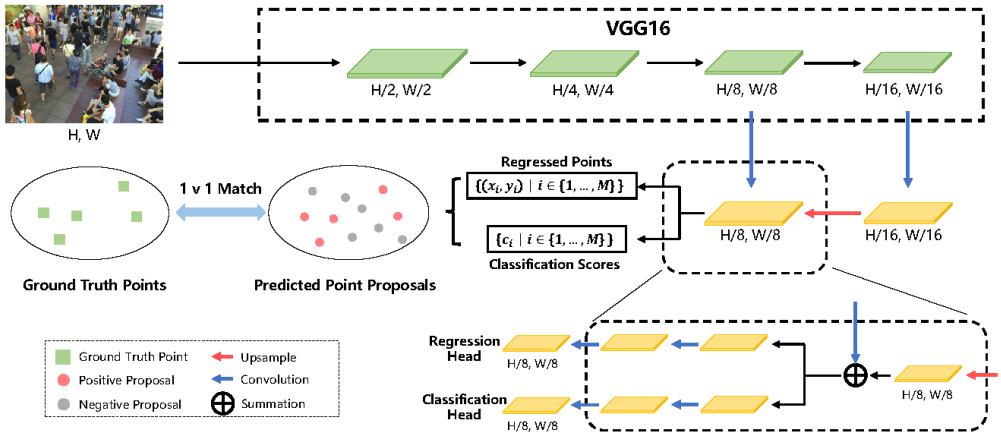

Figure 1: Visualization of P2PNet architecture

Next, a set of anchor points for the image are defined at regularly spaced intervals along the original image. Then, the upsampled image features are fed through two heads, one for classification and one for regression. For each anchor point, the classification head predicts whether the point is adjacent to the object of interest for counting, and the regression head predicts the coordinate shift between the default anchor point location and the center-point of the object. During training, the Hungarian matching algorithm Kuhn (1955) is used to match the regressed points to the ground truth points in the image, and the model is penalized both for incorrect classification and regression of its proposed points. At evaluation time, the predicted center points are simply the shifted anchor points with a positive classification class, and the number of objects counted is the number of positive anchor points.

2.2 Data and Training Approach

To train our model, we first generate and curate a dataset of 200 hand-labeled images of grape bunches in a variety of shapes, lighting conditions, and naturalistic backgrounds. Specifically, we pull images from grape-associated classes ("Grapevine family", "Grape") of the OpenImages V7 dataset created by Kuznetsova et al. (2020). We then use a self-built labeling tool to generate crops of individual bunches of grapes from those larger images, and then we individually label the center point of each healthy dark grape that is completely or partially visible in the image, discarding the image crops where the individual grapes are difficult to discern even as a human. We do not label the green or shriveled grapes; in the case of occluded grapes, we attempt to approximate a reasonably centered point that falls within the visible part of the grape (later figures in this analysis will demonstrate what the final labeling and cropping looks like).

2.2.1 Data Pre-Processing

During training, we use several data augmentations drawing from the original work of P2PNet: firstly, we perform random horizontal flipping (critically, this requires flipping the point labels as well), randomly take 128 by 128 crops of the images (adjusting the ground truth coordinates accordingly), and random scaling from 0.7 to 1.3. At test time, we simply resize the image to the nearest factor of 128 (rounded down). In all cases, we use the standard ImageNet normalization values.

In addition to these transformations from P2PNet, we also attempted to employ random jitter to brightness and contrast, since our images vary greatly in their lighting conditions and grape resolution, arguably much more than the relatively consistent lighting present in human crowd datasets.

2.2.2 Evaluation

Due to the currently limited size of our dataset, we only create a validation set by randomly holding out 20% of the data (meaning there are only two splits, one for training and one for testing); to preserve experimental

Modification	Train MAE	Train MSE	Val MAE	Val MSE
P2PNet Hyperparams	38.41	46.02	41.85	52.50
+ Reduced Anchor Points	9.69	15.47	11.45	16.12
+ Class Cost	6.40	9.45	9.58	12.12
+ Jitter Augmentation	5.38	7.88	7.88	10.51

Table 1: Ablation Study of Training Augmentations

integrity, we consider the "final" model from an experiment to be the one with lowest training loss. Our model's final evaluation is then based on the MAE of the predicted number of grapes relative to ground-truth. We additionally report MSE for reference; in our case, the two metrics consistently agree on which models perform better regardless.

2.3 Hardware Speed-Ups

With images of appropriate resolution, our model runs at decent, sub-second speeds on Raspberry Pi hardware. However, in a full pipeline, the model would need to handle multiple grape bunches from a single larger image—an image that would also need to be segmented into grape clusters first. For this reason, we investigate the use of 8-bit static quantization across the entire P2PNet model.

To do this, we use the ImageNet validation set as calibration data to determine the quantization bounds, then perform static quantization from 32-bit float to 8-bit int representation. We use the onnx built-in quantization function 'quantize_static' and quantize both the weight and activation. The calibration method we use is MinMax. One caveat is that during static quantization, the input size needs to be fixed. In other words, during inference, the input should also be reshaped to the size used during calibration. This is to ensure that the calibration parameters are consistent with the input. In our experiments, this size is 256 x 128.

3 Results and Discussion

3.1 Hyperparameter Ablation

In using P2PNet, we find that there are several key hyperparameters to tune that drastically affect the model's performance on grape counting. We report an ablation analysis of their effects in Table 1; we discuss them further here. In all cases, we start from the pre-trained P2PNet as much as possible, but fully finetune the entire architecture.

By far the most important parameter to tune is the number of anchor points, which is controlled by the number of anchor points per row and column of the image. In the crowd-counting case, densities can be on the order of thousands of people per image, whereas in our case, the most densely packed grapes per bunch peak at ranges around 100-150 grapes per bunch. Performance drastically improves from complete randomness to some semblance of reason simply by reducing the number of anchor points per row and column. This makes sense as the reduction in anchor points effectively resolves an extreme class imbalance for the classification head. Note that since the number of anchor points is different, we can no longer use the regression and classification head weights from the original P2PNet.

Another important parameter to tune is the weight of the classification loss relative to the regression loss. In our experiments, we find it is valuable to double the weight of the classification loss. We ultimately care more about the number of grapes in the image rather than their exact location, and we find that weighting the classification arm to be higher results in better performance.

Lastly, as discussed previously, given the range of actual color representations and depths of focus of the different grape images, we introduce brightness and contrast jitter (plus-minus of 0.3) as a new image augmentation. We find that this further improves the overall performance of the model.

3.2 Qualitative Analysis

By the end of our adjustments, we have a capable model that can consistently predict grape counts well—for reference, the average number of grapes per image was around 17, so an MAE of 7.8 is respectably reasonable performance. As a visualization of how well the model does, we visualize a comparison of predicted and actual (human-counted) grape counts in Figure 2.

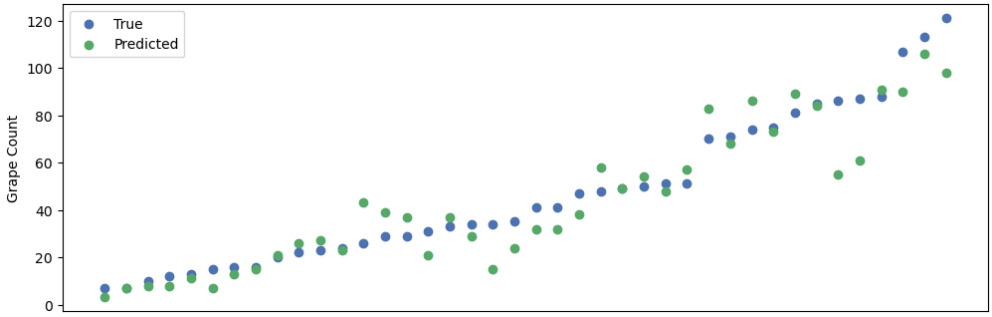

Figure 2: Predicted VS Actual Grape Count on Validation Data

However, there is still room for improvement. As shown in Figure 3, the model struggles with undercounting in cases of very dense but low-resolution grapes; it also struggles to distinguish blurred, dark background objects from shadowed or background grapes labeled positively. While the undercounting is arguably a case of an unreasonable input, the overcounting case certainly should perform better.

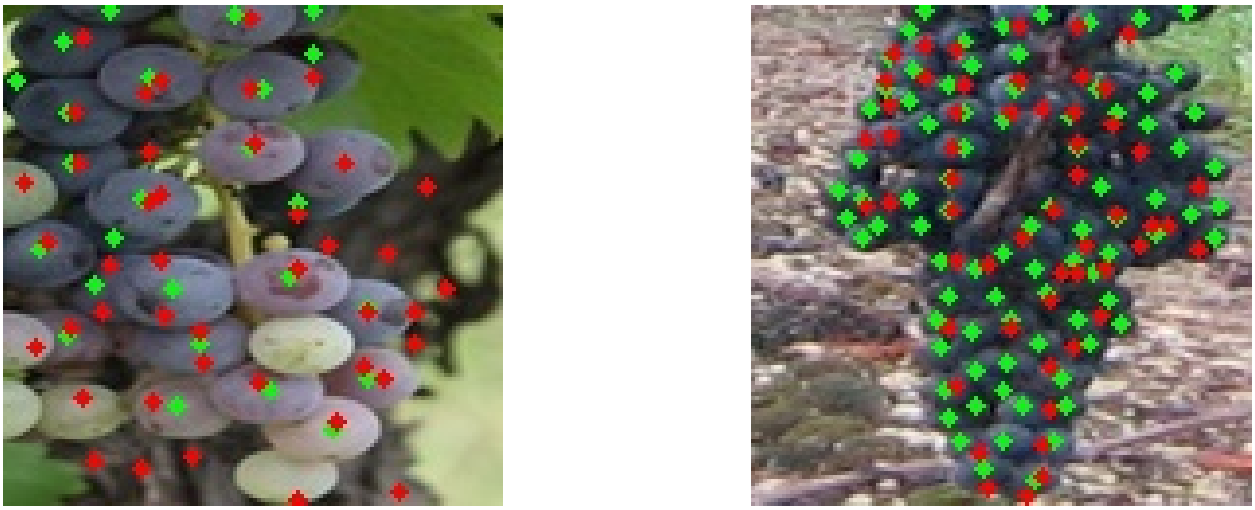

Figure 3: Example failure cases — red is ground truth and green is predicted

3.3 Hardware Deployment

We divided the P2P network into multiple sub-modules that are processed serially and evaluated their runtime on the Raspberry Pi accordingly. The modules are the VGG16 backbone, feature pyramid network (the upsampling step), regression head, and classification head. The total runtime for a 896x384 image is 9.43 seconds. The VGG16 backbone takes the majority of the time at 7.4 seconds. The feature pyramid network, regression head, and classification head take 1.27, 0.41 and 0.34 seconds accordingly. After static quantization, the inference time were cut down to sub 1 second. On the grape dataset we created, we were able to run 12x faster after 8-bit int quantization, from close to 1 second per image to below 0.1 seconds per image. In Figure 4, we show the predicted output from the quantized (left) and unquantized model (right) running on Raspberry Pi. In Figure 5, we show the runtime improvement of the quantized model. We observe that the model predicts similar amounts of grapes after quantization showing the redundancy of the original network.

Figure 4: Examples of outputs predicted by quantized (left) and unquantized (right) model

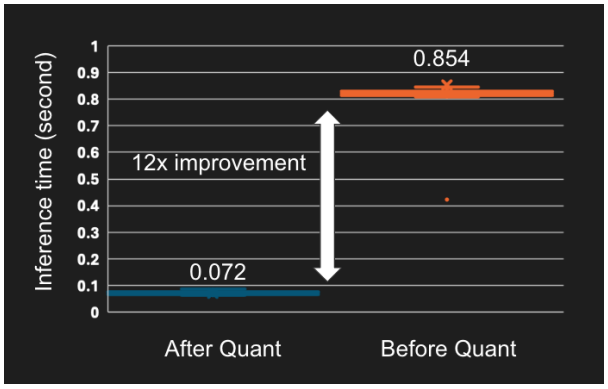

Figure 5: 12x runtime improvement with the quantized model

4 Future Work

The most important future step toward practical deployment would be the introduction of a grape segmenter into the pipeline. However, further improvements could be made with this model as well—firstly, the quantized model currrently requires a fixed input size, and allowing for more flexibility with either preprocessing or a better quantization strategy would be an effective way to improve the model.

On the grape counting side, further refinements could and should be undoubtedly made to increase available training data, either through more clever augmentations or simply collecting more data. This would likely help with the observed undercounting and overcounting cases, which seem likely due to the

model underfitting or not having enough data on understanding occluded grapes properly. Alternatively, perhaps a different approach to labeling might be best (i.e. not including blurrier grapes in the background as part of the count, as this likely is at least part of what causes cases of overcounting). Further work could also be done to curate the data more carefully to help avoid the difficult and easy-to-undercount data.

Another interesting strategy might be to bias the frequency of anchor points in particular directions to account for the fact that grape bunches are typically taller than they are wide, as are the grapes themselves.

5 Work Distribution and Code

Our code is available on Github. Christos focused on hyperparameter tuning and built the labeling tool. Harry focused on quantization. We both worked through the initial literature review, and we both suffered through manual grape labeling.

References

Steven W. Chen, Shreyas S. Shivakumar, Sandeep Dcunha, Jnaneshwar Das, Edidiong Okon, Chao Qu, Camillo J. Taylor, and Vijay Kumar. Counting apples and oranges with deep learning: A data-driven approach. *IEEE Robotics and Automation Letters*, 2(2):781–788, 2017. doi: 10.1109/LRA.2017.2651944.

Guy Farjon, Liu Huijun, and Yael Edan. Deep-learning-based counting methods, datasets, and applications in agriculture – a review, 2023.

Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955. URL `https://api.semanticscholar.org/CorpusID:9426884`.

Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *IJCV*, 2020.

Scarlett Liu, Xiangdong Zeng, and Mark Whitty. A vision-based robust grape berry counting algorithm for fast calibration-free bunch weight estimation in the field. *Computers and Electronics in Agriculture*, 173:105360, 2020. ISSN 0168-1699. doi: https://doi.org/10.1016/j.compag.2020.105360. URL `https://www.sciencedirect.com/science/article/pii/S0168169919326432`.

Xu Liu, Steven W. Chen, Shreyas Aditya, Nivedha Sivakumar, Sandeep Dcunha, Chao Qu, Camillo J. Taylor, Jnaneshwar Das, and Vijay Kumar. Robust fruit counting: Combining deep learning, tracking, and structure from motion. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1045–1052, 2018. doi: 10.1109/IROS.2018.8594239.

Qingyu Song, Changan Wang, Zhengkai Jiang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yang Wu. Rethinking counting and localization in crowds:a purely point-based framework, 2021.