CounTX model for berry counting

https://github.com/fabiopoiesi/CounTX_Berry

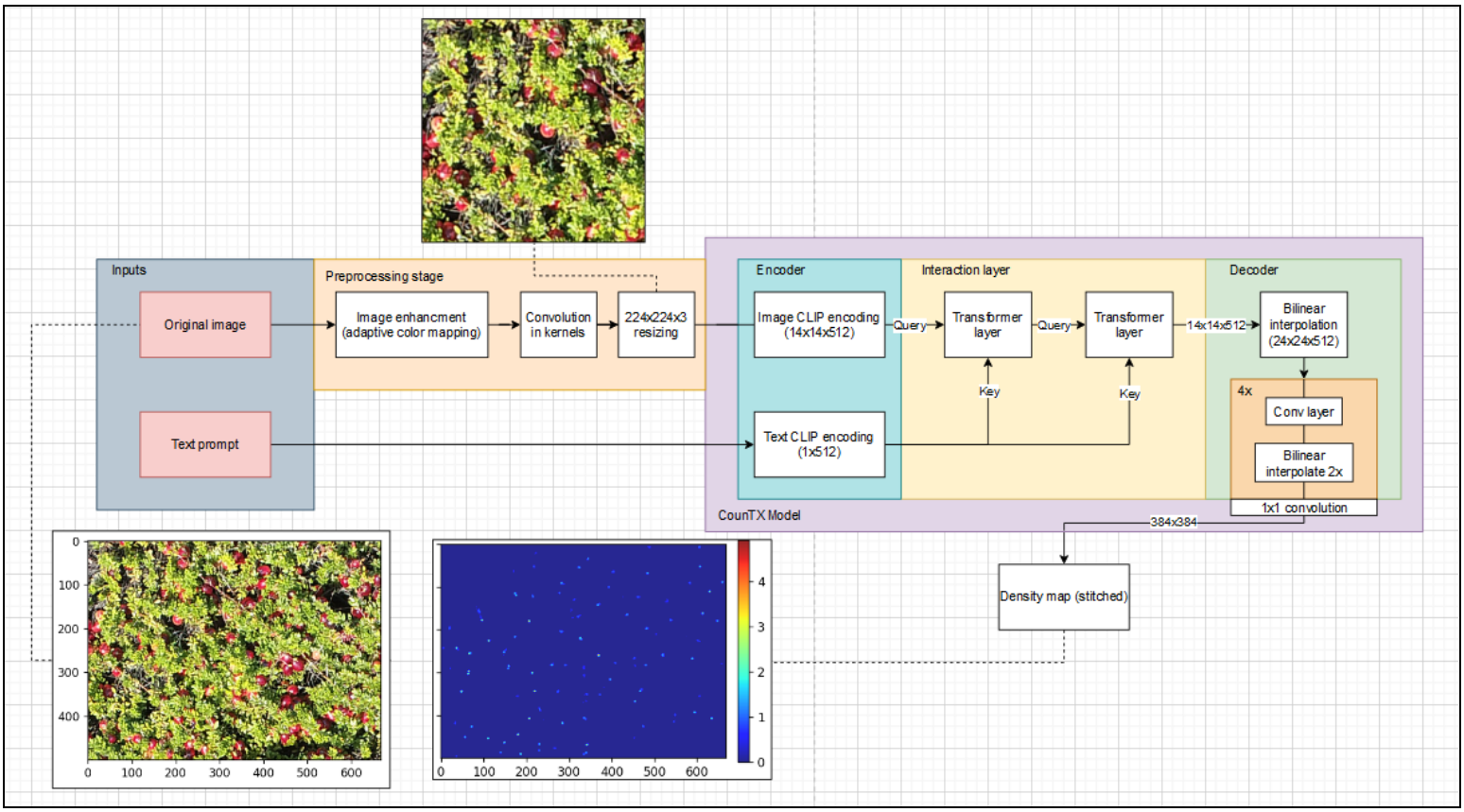
Alessandro Giustina

Proposed approach

To facilitate the integration of the CounTX model into a modular work pipeline, input and output modules were developed to streamline the process. In addressing the constraint of the input image size, which is limited to a 224x224 square RGB image, a purpose-built convolution layer was designed specifically to partition the image into smaller sections called kernels. These kernels are then processed individually by the model, utilizing a predetermined stride and kernel size.

After computing the density maps for each kernel, the output module takes into consideration the kernel size, stride, and image dimensions to stitch the resulting 384x384 fixed output maps. Through this transformation, the output module remaps the maps onto a new pixel-space, accurately generating the corresponding density map.

The concern regarding the overlap was mitigated by employing a maximum union approach, wherein the highest values from the two kernels were amalgamated.



CounTX model limitations

This document aims to provide a comprehensive analysis of the limitations that were encountered while trying to implement the CounTX model.

CounTX is a model designed for open-world object counting in images, with the target object class specified through a text description. It utilizes a single-stage architecture with a transformer decoder counting head, leveraging pre-trained joint text-image CLIP representations. Unlike previous approaches, CounTX is class-agnostic and can count instances of any object class given an image and its corresponding text description [12].
One of the advertised features of this model was the ease of finding the final count by just integrating over the density map; as will be explored in section 4 this feature is quite unusable and most of the times just plain wrong.

In this section we will explore the limitations found:

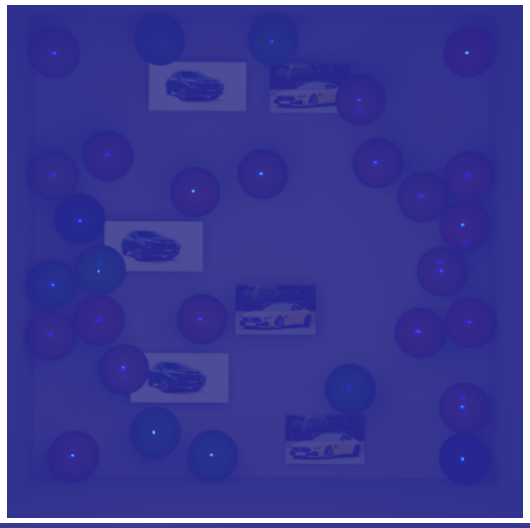
1. Query-output mismatch
2. Division bias
3. Scale inconsistency
4. Wrong integration count

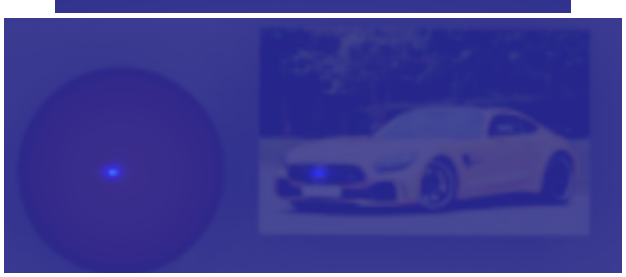
1) Query-output mismatch

Through testing it is possible to see how the text doesn't matter to the model if it doesn't find clearly the specified class and, as fallback, just searches for the most prominent object in the image. This happens a lot when the background is noisy but it can be seen also when it isn't.

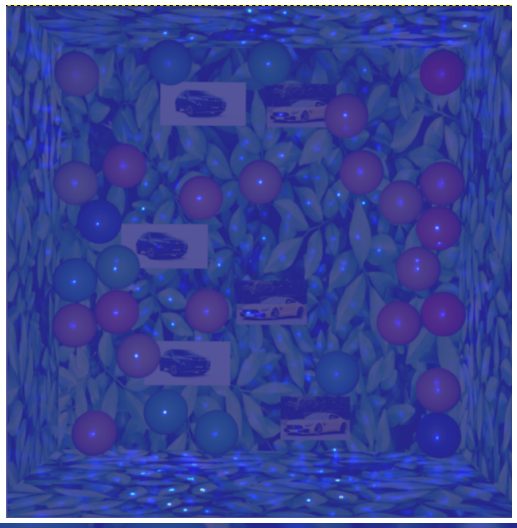
Examples of images fed to the network with the prompt: "the number of balls".
It is clear how in image [2] the model sees the leaves as objects to be counted and even mistakes a car for a ball while in image [1] the model sees the balls but there is still insecurity.

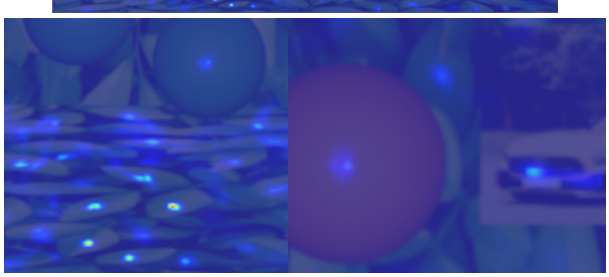
[1]





[2]





In addition, when presented with an empty prompt, the CounTX model exhibits an interesting behavior. Instead of producing an empty density map, it generates peaks of high density on seemingly random objects within the image, presumably the ones with higher relative prominence as shown by the examples below.

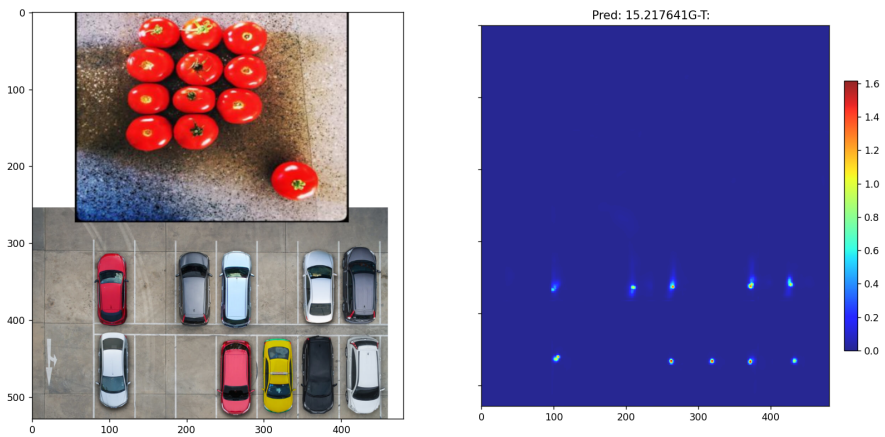
In conclusion the observations from the testing highlight the limitations and potential biases of the model when it comes to counting objects. The model's performance can be influenced by the complexity of the background, the prominence of objects, and the clarity of the specified class.

2) Division bias

When presented with multiple classes that aren't clearly divided, the model tends to encounter issues and struggles to accurately distinguish and count the objects. The lack of clear boundaries between different classes in an image poses a challenge for the model's classification capabilities. It may result in confusion and ambiguity as the model tries to identify and assign objects to specific classes. This difficulty becomes more pronounced when the background is noisy or when there is overlapping and intermingling of different objects and different classes. In such cases, as explored before, the model may exhibit a tendency to focus on the most prominent or visually dominant objects, leading to inaccuracies and misclassifications.

Here is an example of this very issue. When the two classes are clearly divided with each a clean background and no other classes [3] the model performs very well in classifying/counting the objects, but as the examples [1] and [2] prove, when intermingling or when noise is added to the images/background, the results completely differ from reality.

[3] Kernel size 600px “the number of cars”.

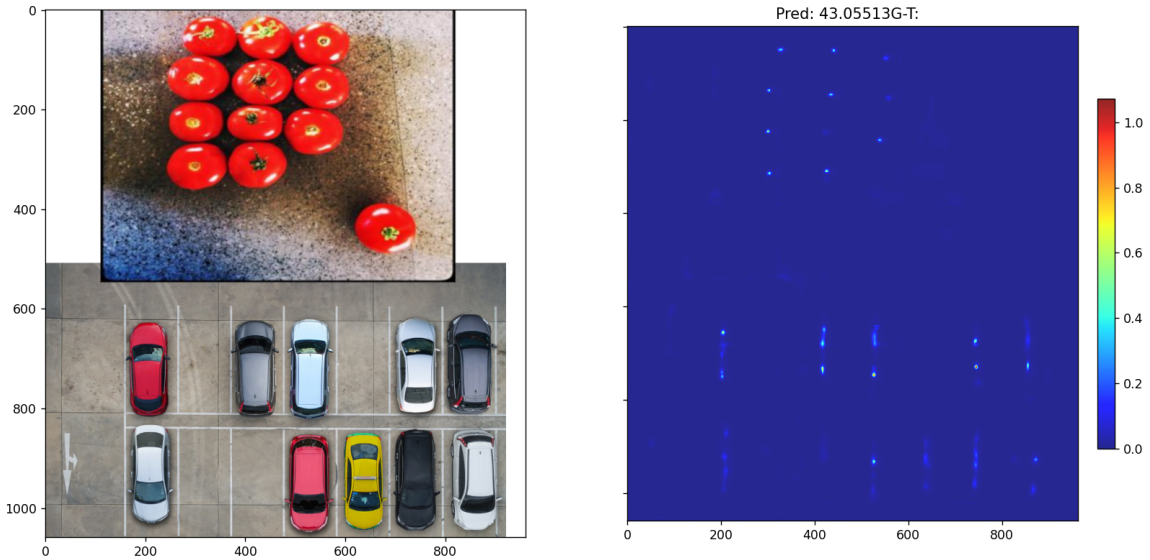


3) Scale inconsistency

In addition to the aforementioned challenges, another issue encountered during the evaluation pertains to the inconsistency of classification exhibited by the model at different scales. It became apparent that the model lacks scale invariance, indicating that its performance is significantly influenced by variations in object sizes present within the images. This lack of scale invariance poses a considerable limitation, as the model's ability to accurately classify objects becomes compromised when confronted with size disparities. Notably, the model appears to be more adept at recognizing and classifying objects of certain pixel sizes, while struggling with objects of differing scales. This disparity in performance across scales suggests a need for enhanced robustness and adaptability of the model to accommodate variations in object sizes and effectively generalize its classification capabilities across different scales; a solution for this very issue will be discussed in the next section by using an adaptive kernel size.

The issue of inconsistency in classification at different scales is evident when examining the comparative analysis of example [3] and example [4]. In both instances, the model was presented with the identical prompt of "the number of cars," albeit with varying kernel sizes. Notably, when a kernel size of 600px was employed, the model demonstrated proficiency in accurately identifying the intended objects. However, when the kernel size was reduced to 320px, the model encountered significant challenges in discerning between tomatoes and cars, resulting in misclassifications.

[4] Kernel size 320px “the number of cars”

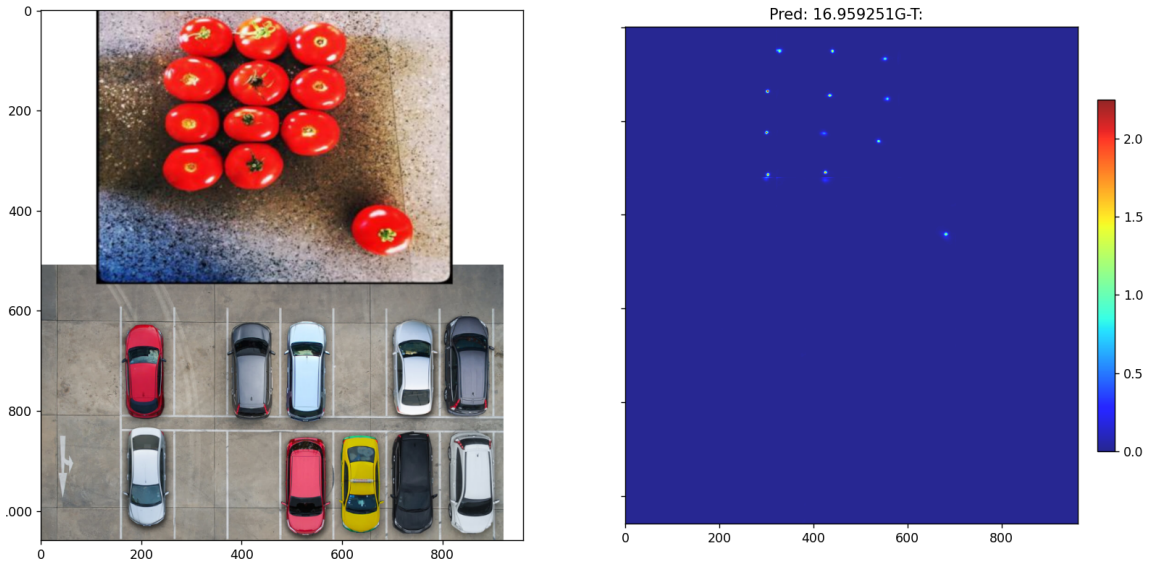


4) Wrong integration count

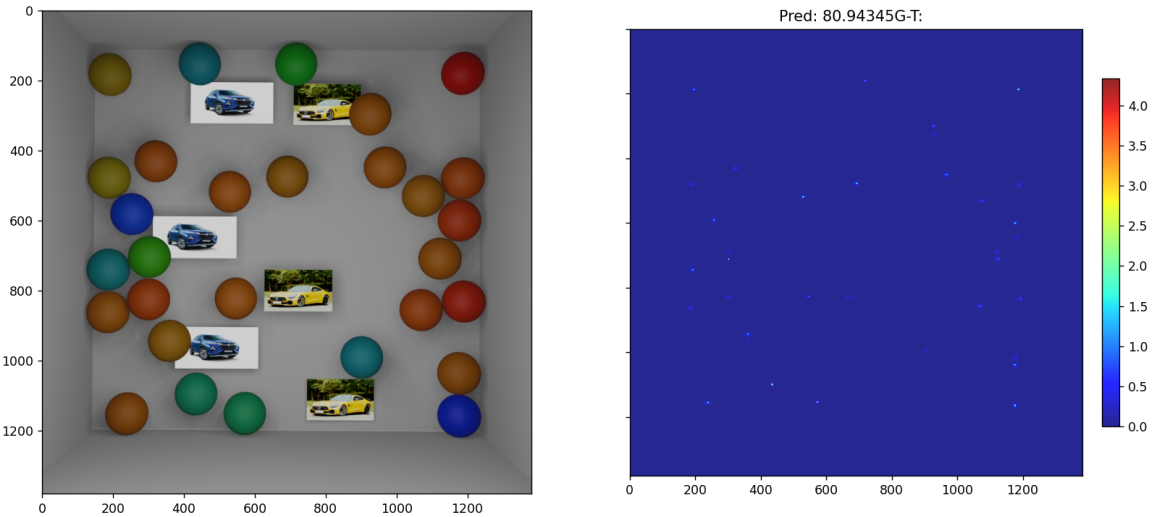
During my experimentation, I encountered limited success in achieving accurate object counting integration. I was able to accomplish this goal only once, and it was under specific conditions. This success occurred when working with a single object class against a plain white background or with a carefully segmented and clean image in a horizontal layout. However, I noticed that even the presence of minimal noise had a significant impact on the model's performance. The model's functionality was compromised, and its ability to accurately count and identify objects was affected. This observation highlights the model's sensitivity to noise and the need for further research and development to improve its robustness and resilience in the presence of visual disturbances.

For example even when the model did distinguish between the classes the map failed to add up to the correct value like in image [3], [5] or [6] because the peaks either don’t add up to one or are way too big.

[5] Kernel size 320px “the number of tomatoes”



[6] Kernel size 320px “the number of balls”



Using CounTX to count berries with the CRAID dataset

CounTX_berry framework creation

The framework employed in this implementation begins by processing input images through a series of steps to prepare them for analysis. This involves creating an adaptive color mapping and subsequently dividing the images into square sections. These sections are then resized to a standardized dimension of 224x224 before being passed into the model for further processing. In the subsequent stages, the kernels used in the analysis have a predetermined stride, and the overlapping regions are fused using a max function. The resulting output is then subjected to a thresholder, which sets all values below a certain threshold to zero. Next, the modified image is inputted into a cluster finder, which counts and filters clusters based on their size to eliminate noise. If required, the clusters can be indexed to determine the total number. The diagram below provides a visual representation of this process.

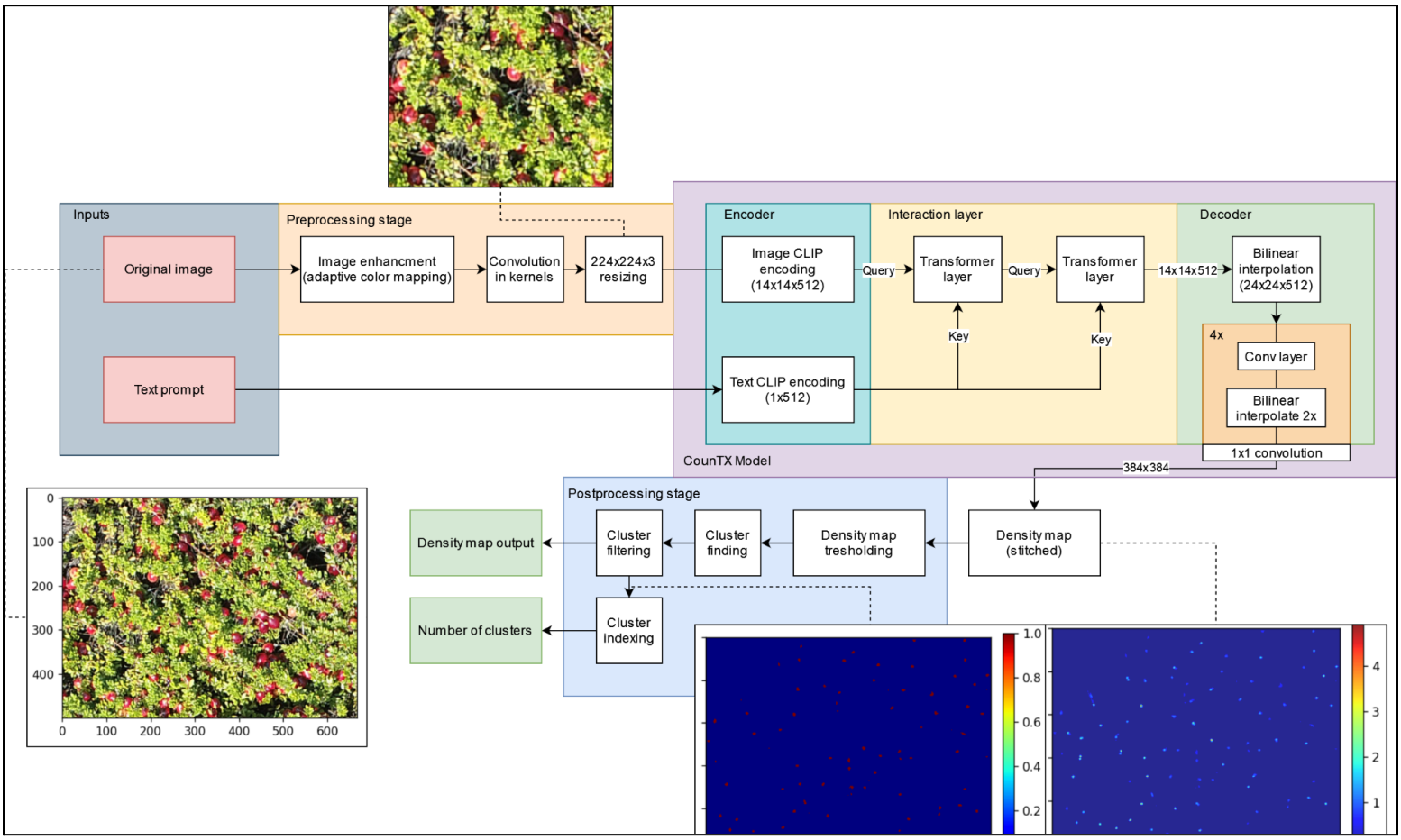
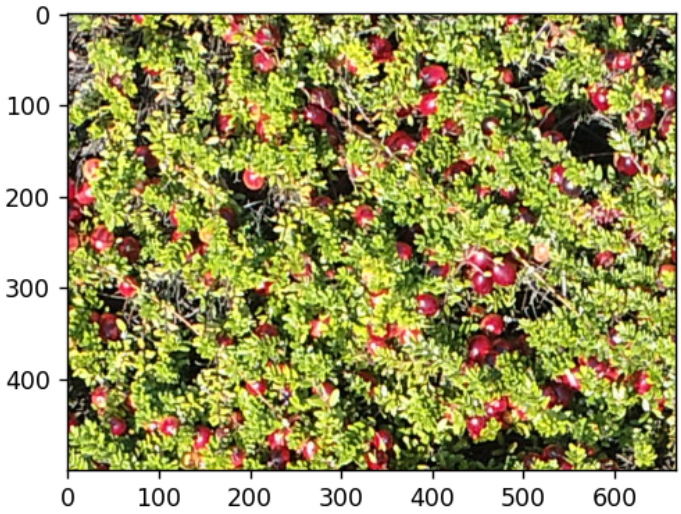
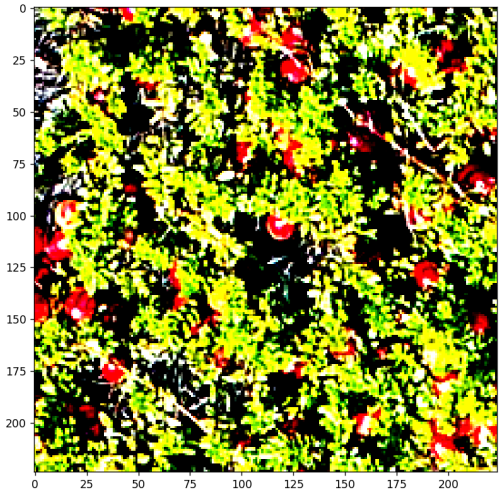


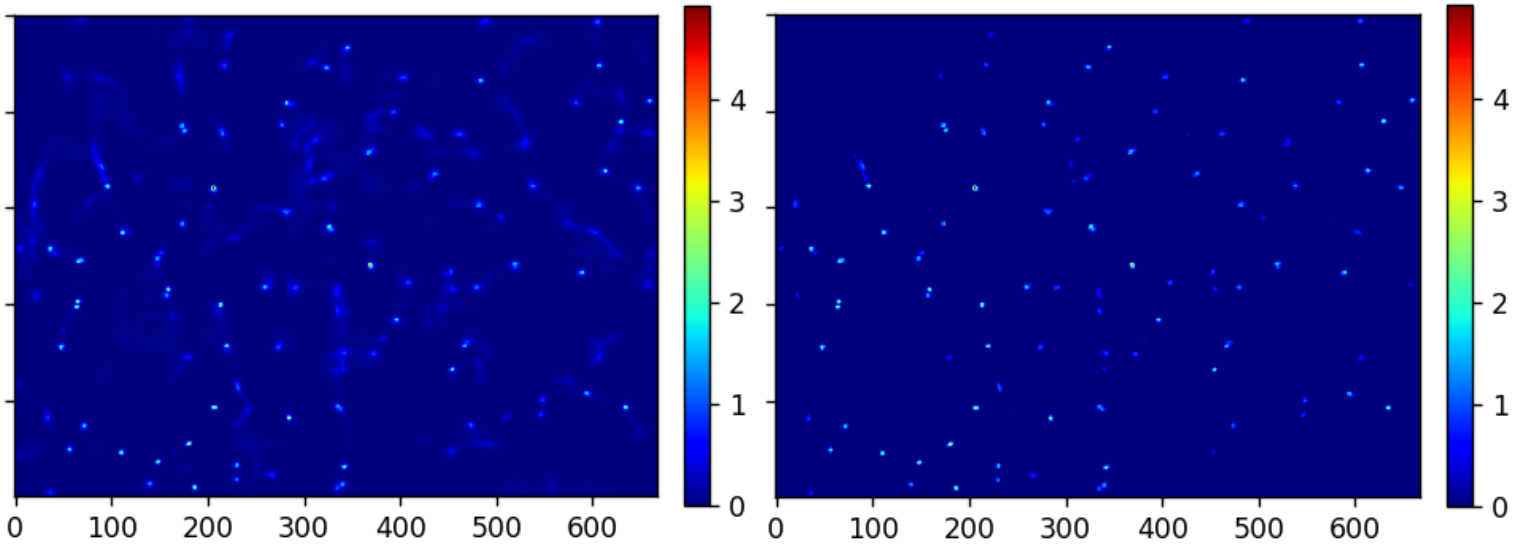
Image pipeline

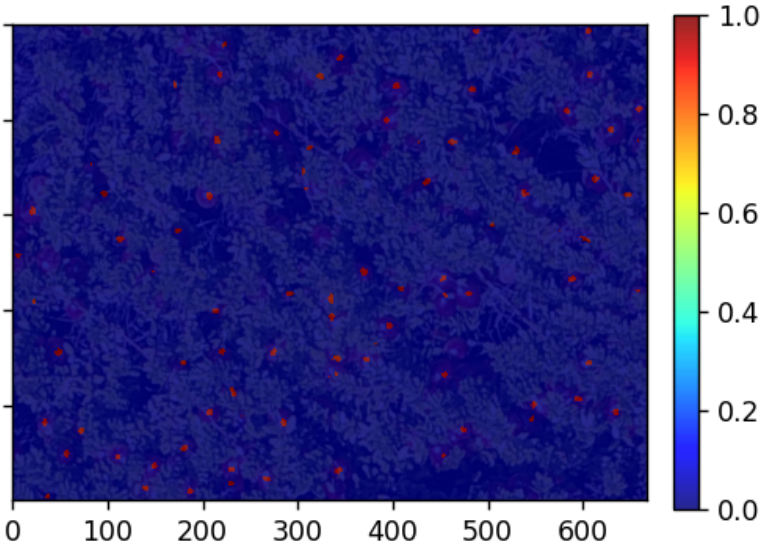
The image is cut, normalized and resized to 244x244





Then it is passed through the model and then a threshold is applied before it is passed through the clusterfinder which finds the clusters in the map.



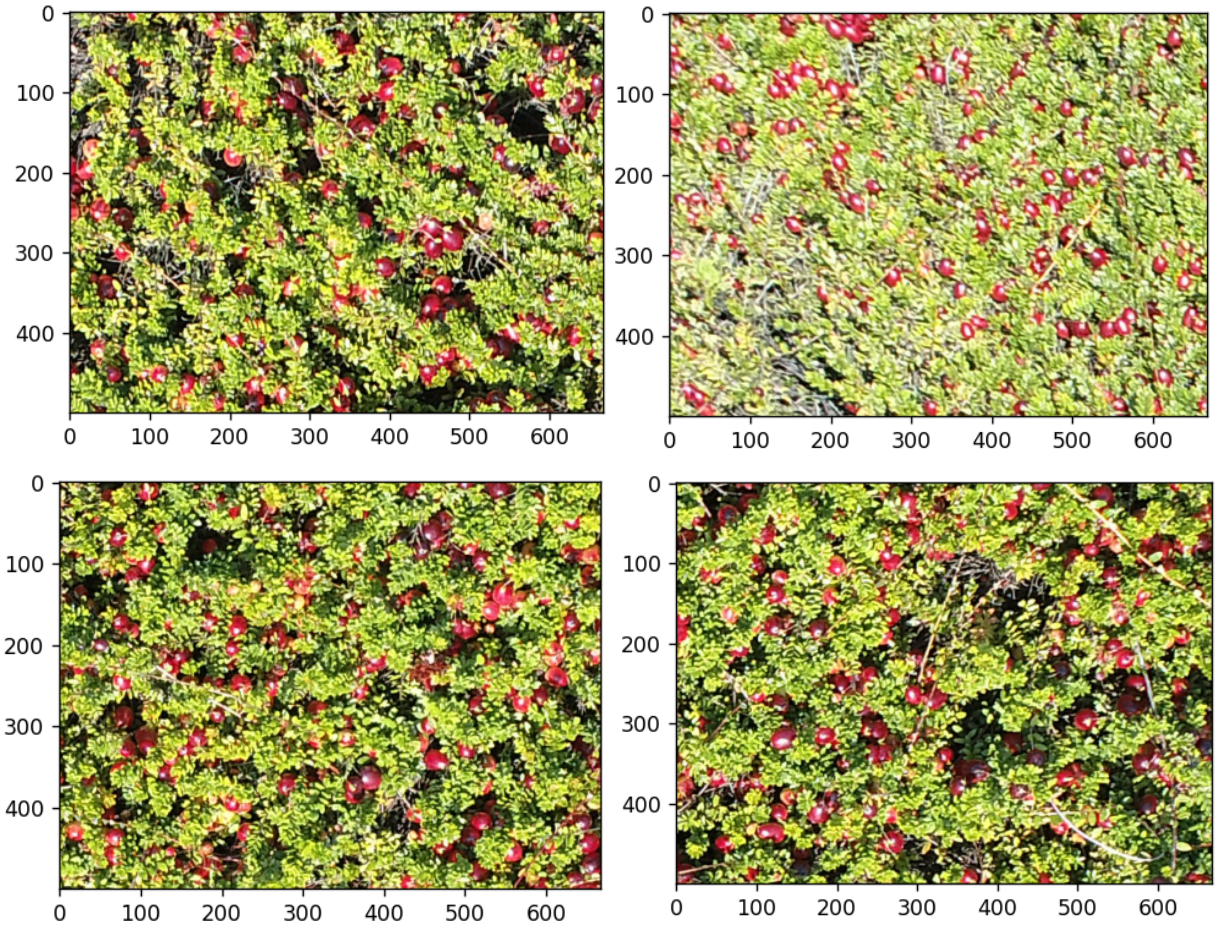


CRAID dataset

In order to assess the model's performance and optimize specific parameters such as the threshold, kernel size, maximum cluster size, and stride amount, the CRAID dataset was utilized. The complete dataset comprises 21,436 cranberry images, each sized 456 x 608 pixels. These images were captured using a Phantom 4 drone from various altitudes, with camera settings manually fixed at 100 ISO, 1/240 shutter speed, and 5.0 F-Stop. Consequently, a considerable number of images are either overexposed or underexposed, necessitating normalization prior to inputting them into the model.

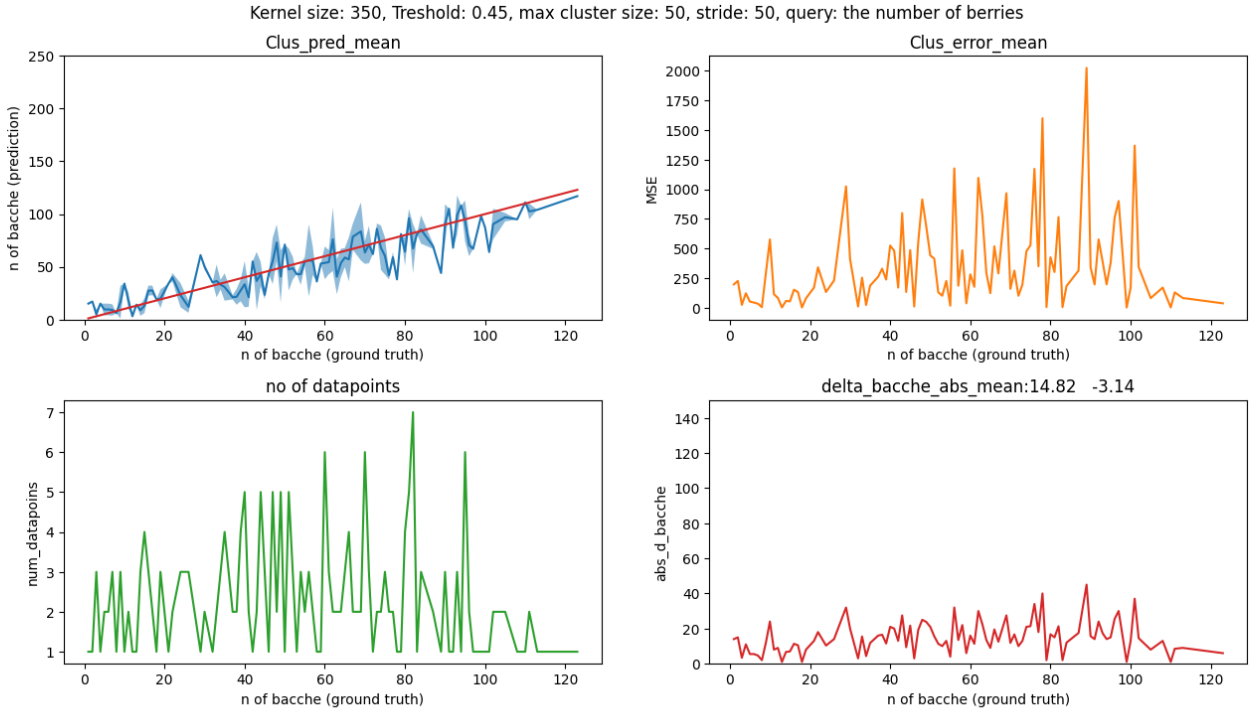
On average, each image in the CRAID dataset contains 39.22 cranberries. For evaluation purposes, a subset of 2830 images was selected, from this subset three sub-subsets were created.

This implementation faces two major challenges with the dataset. Firstly, when the ground-truth values exceed 90, the model struggles to recognize the berries as they appear too small in the images. **This issue could be resolved by employing an adaptive kernel size and threshold based on the drone's altitude**, although this information is not provided in the dataset. Secondly, the presence of noisy backgrounds poses a significant problem for the model. Consequently, determining the appropriate parameters to filter out this noise and obtain usable images proved to be quite challenging.
The following examples showcase images from the dataset along with the results achieved through fine-tuning the parameters in the training set.

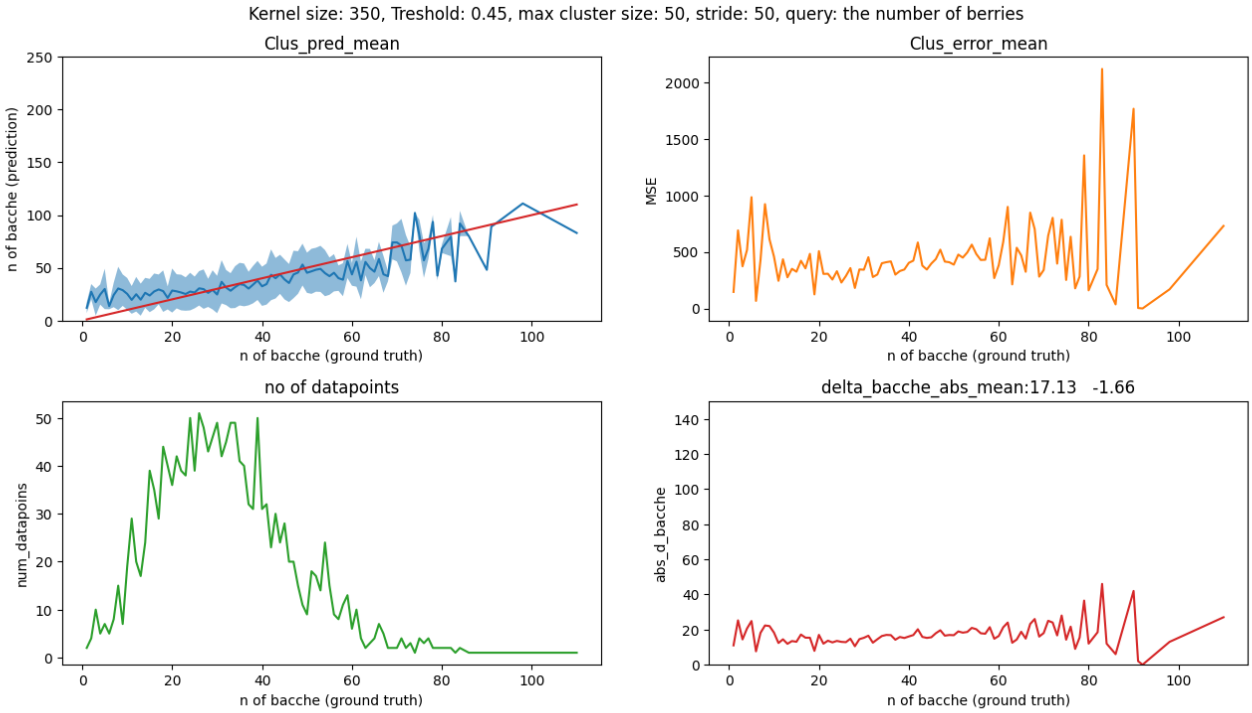


Model results

Model evaluated on the validation subset (224 images):



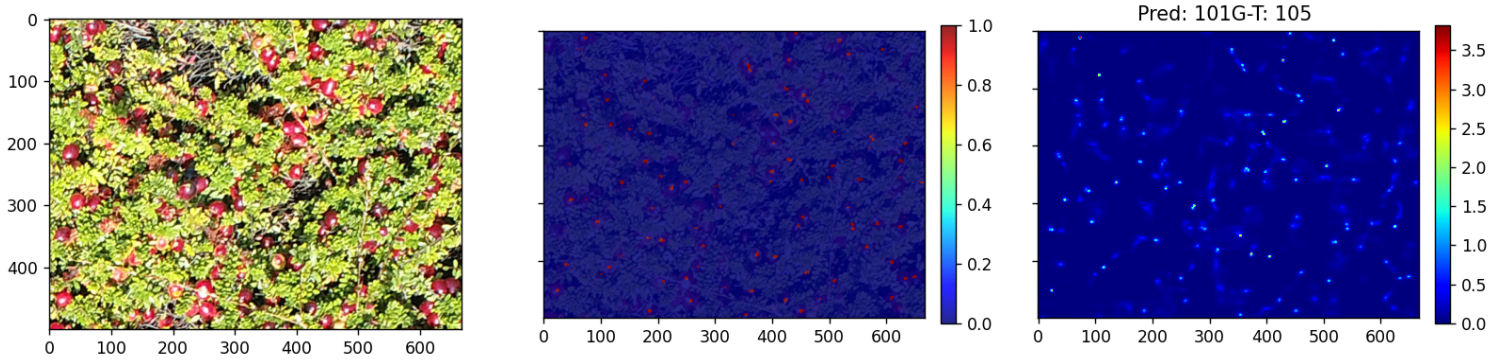
Model evaluated on the training subset (used to tweak the hyperparameters, 1677 images):

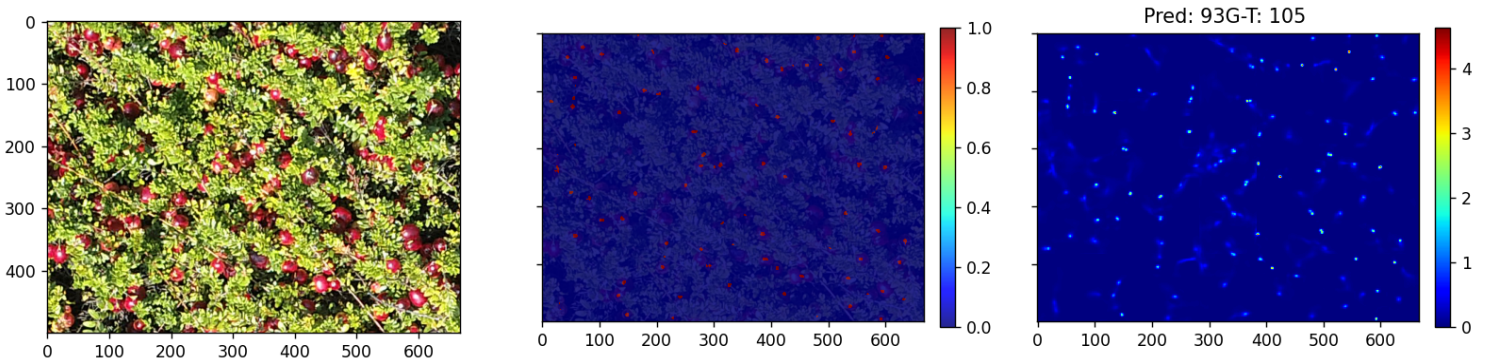


Results

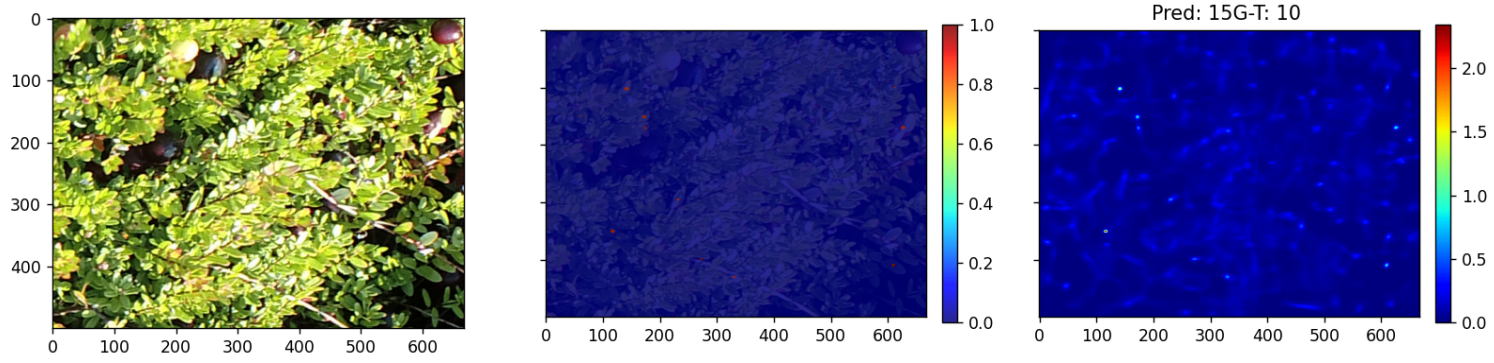
All of these results have been collected with the model parameters as such: kernel-size=350, stride=[50,50], threshold=0.45, max_size=50.

In these two first examples two images with a high number of berries were analyzed. Both of the density maps are quite noisy but the tresholder and the subsequent filtering helped to remove the majority of misclassifications.

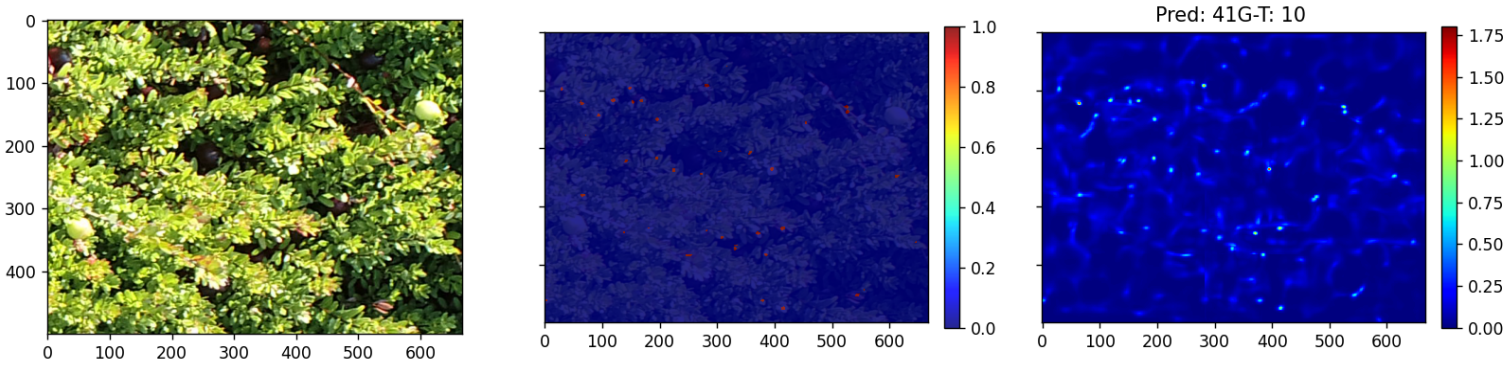




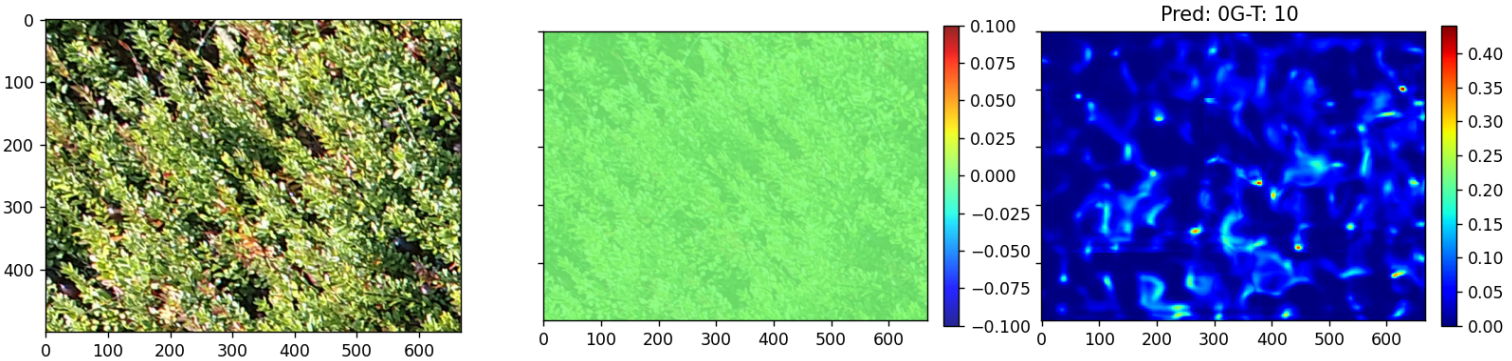
In the subsequent example, the model demonstrates commendable proficiency in classifying the berries despite the scarcity of visible instances. However, it is essential to acknowledge that even though the model performed admirably overall, the noise floor remains relatively elevated. As evidenced by the central-left region of the image, the model still encountered a misclassification scenario wherein a leaf was erroneously identified as a berry.



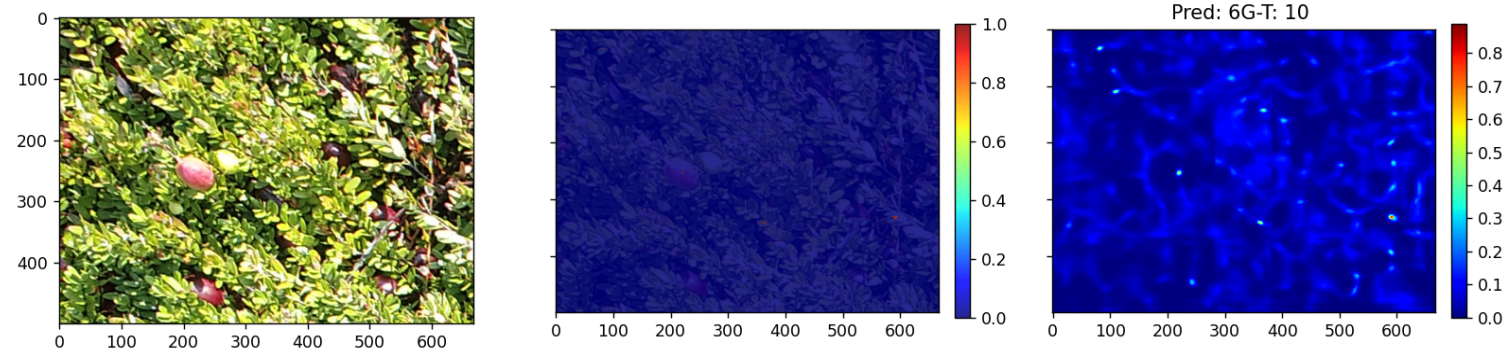
In the following visual representations, it becomes evident that the model exhibits certain patterns when confronted with specific scenarios. Specifically, when encountering indistinct berry images, the absence of berries, or instances where the berries appear disproportionately large, the model produces a considerable amount of noise. Additionally, the model exhibits occasional divergences from the input prompt, leading to the misclassification of numerous leaves as berries.

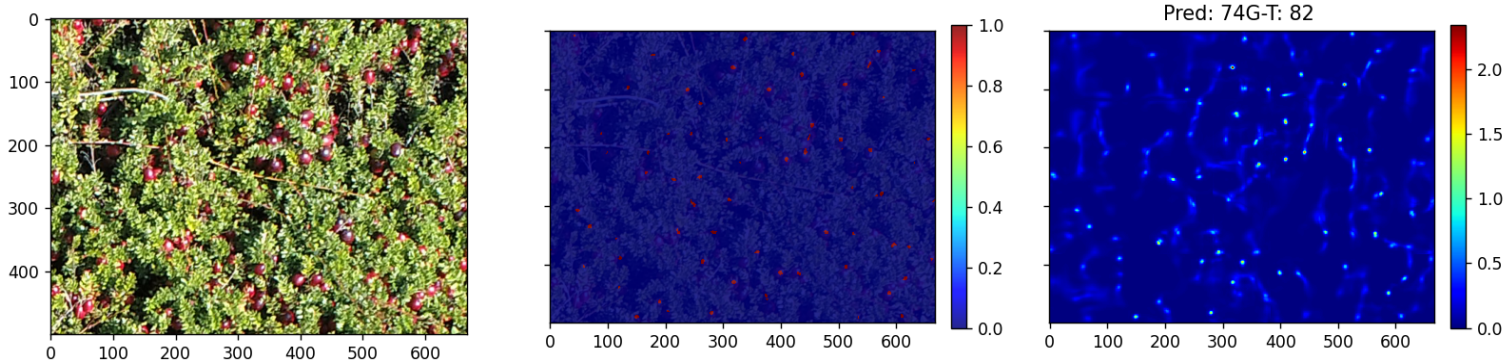


In this example the noise level was so high that the cluster finder failed to detect anything.



These are other examples of the model classifying correctly in difficult situations like with overexposed images, or with a lot of intermingling of berries.



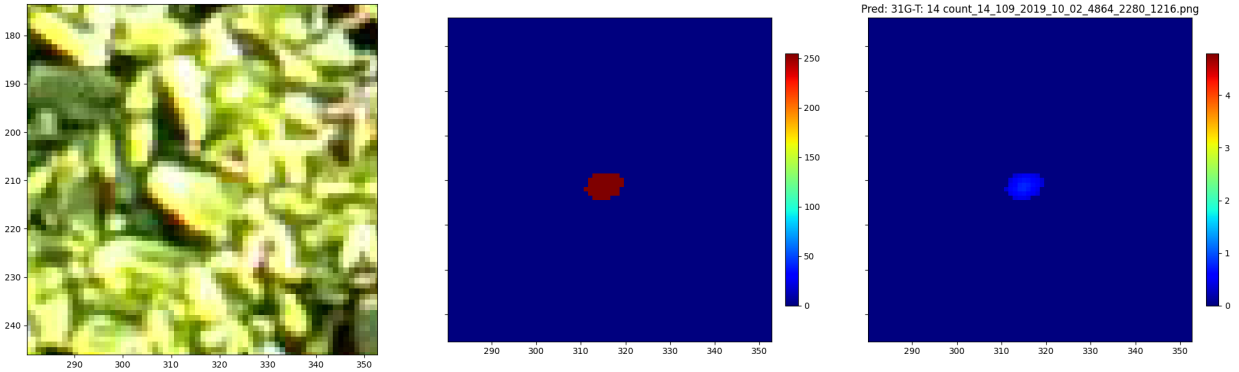


Results analysis

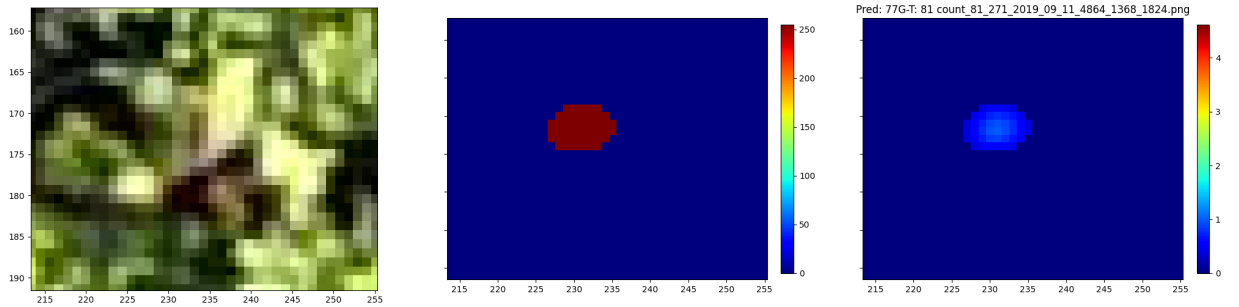
The density maps outputted from the model, while displaying a considerable level of noise, did not hinder the potential of the framework to accurately classify berries. This was achieved through the implementation of the thresholder technique, followed by the subsequent cluster filtering process, which effectively reduced the majority of misclassifications. Notably, the model showcased good proficiency in identifying berries in instances with limited visible berries. However, it is essential to acknowledge that the noise floor remained relatively high, leading to occasional misclassifications, as evidenced by the above instance where a leaf was mistakenly identified as a berry or by the examples below. Nevertheless, the overall effectiveness of the thresholding and filtering methods in removing misclassifications highlights their significance in enhancing the accuracy of berry detection in dense berry-populated images. These findings exemplify the potential of post-processing techniques as valuable tools for noise reduction and augmenting the model's performance in challenging scenarios as will be more in depth explored in the next section.

Examples of model misclassifications for various reasons, other examples can be found in the appendix:

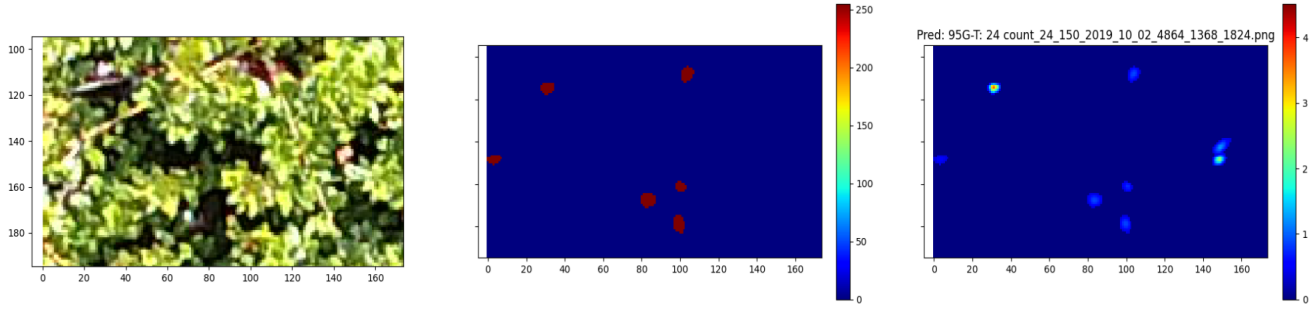
Very prominent feature.



Blurriness and sligh tendency to red.



Very noisy and blurry image.

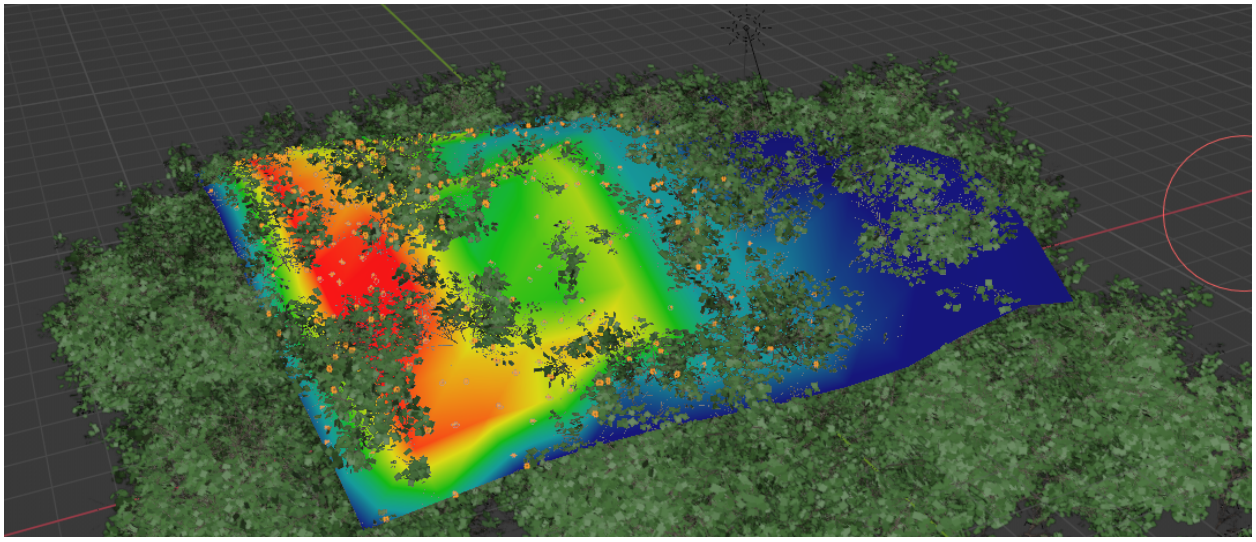


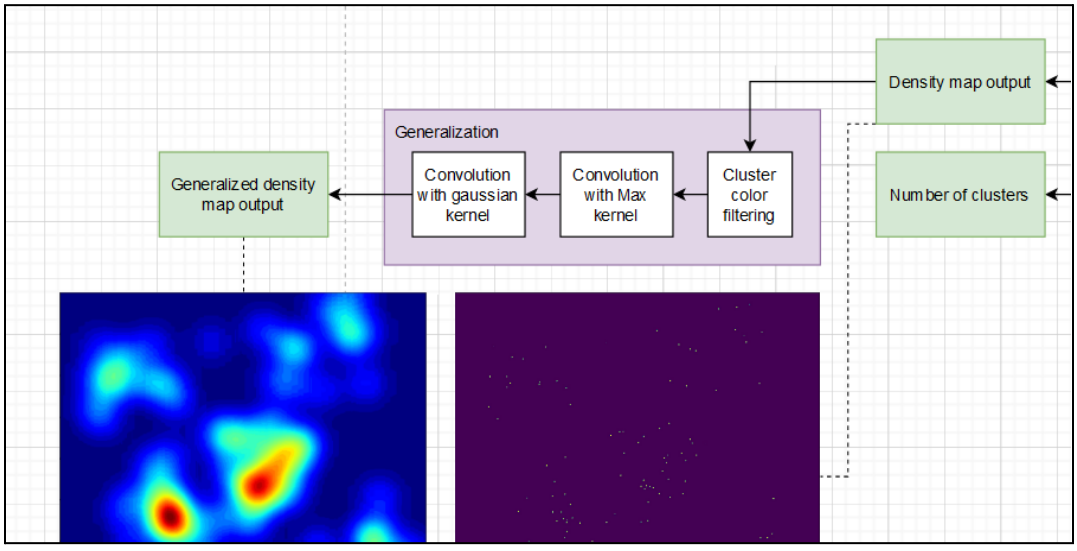
Further improvements of the framework for a general density estimation application

General experiments:

The procedure for determining the density distribution in images lacking a uniform berry distribution is very similar to the counting process. However, there are two additional steps involved. Firstly, a cluster-filter layer is incorporated to verify whether the cluster contains red elements. Secondly, a final rescale and convolution operation is performed to obtain a more comprehensive overview. The upstream model remains the same but utilizes a distinct kernel to accommodate the change in resolution.

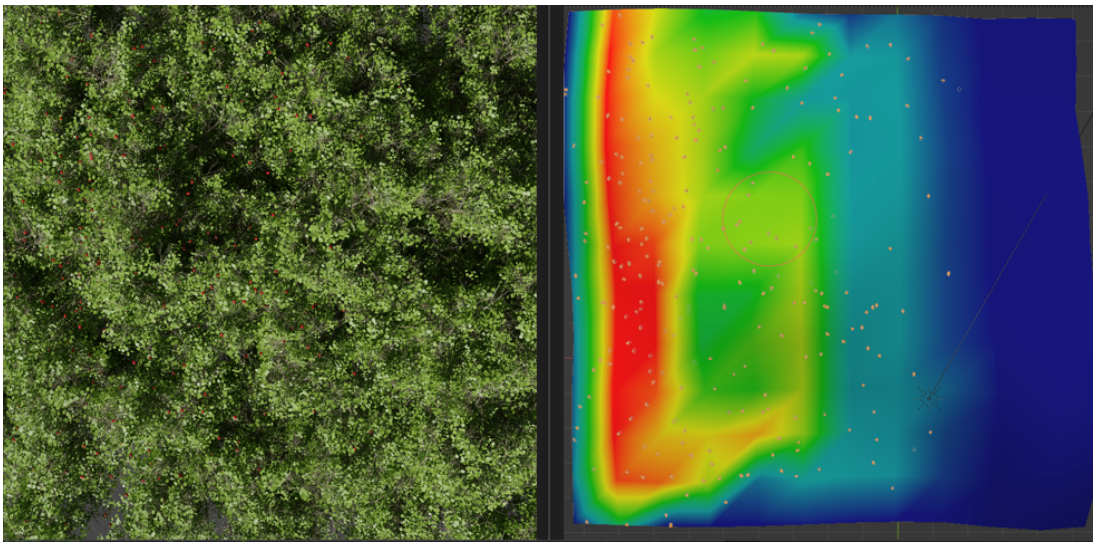
To assess the effectiveness of this model, a new dataset must be generated from scratch since the existing CRAID dataset exhibits uniform distribution across all images. To achieve this, a custom hair emitter has been designed within a Blender scene, allowing for manipulation of bias towards various density patterns. The following examples illustrate three distinct density distribution scenarios that were made to test different aspects of the module:



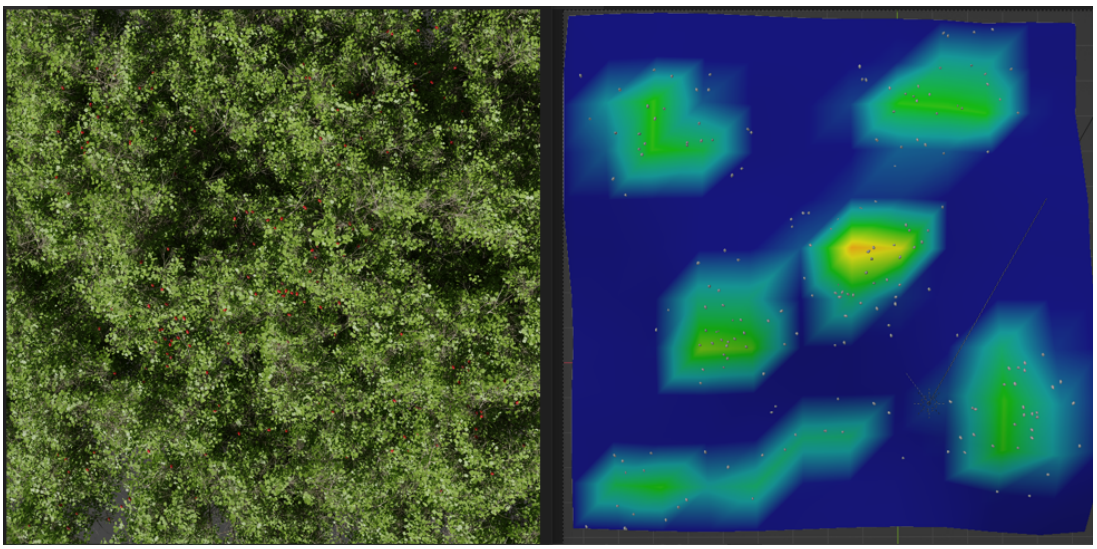


Renders

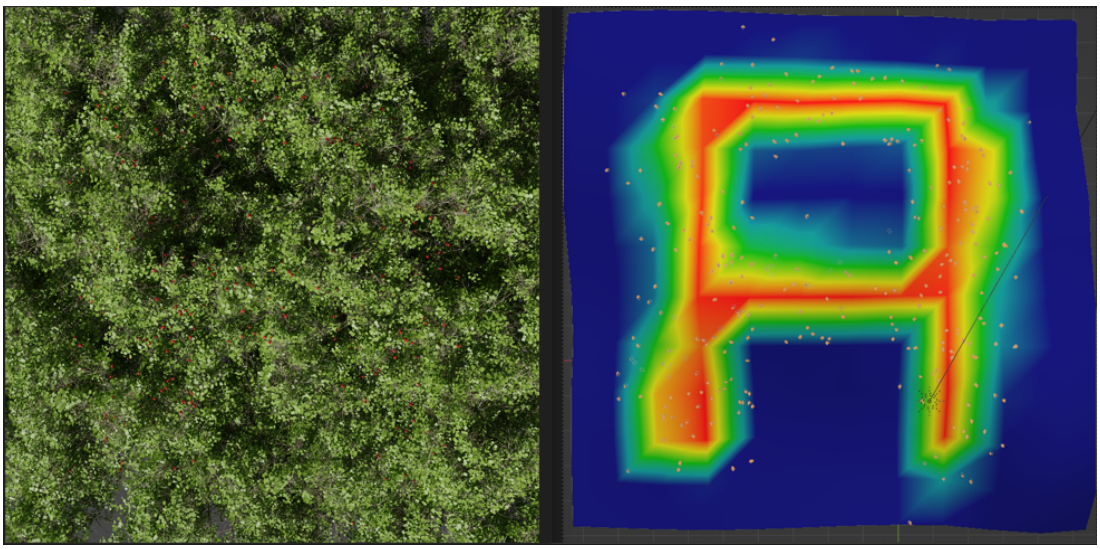
Even gradient distribution, render (left) weightmap (right):



Uneven “cluster” distribution:

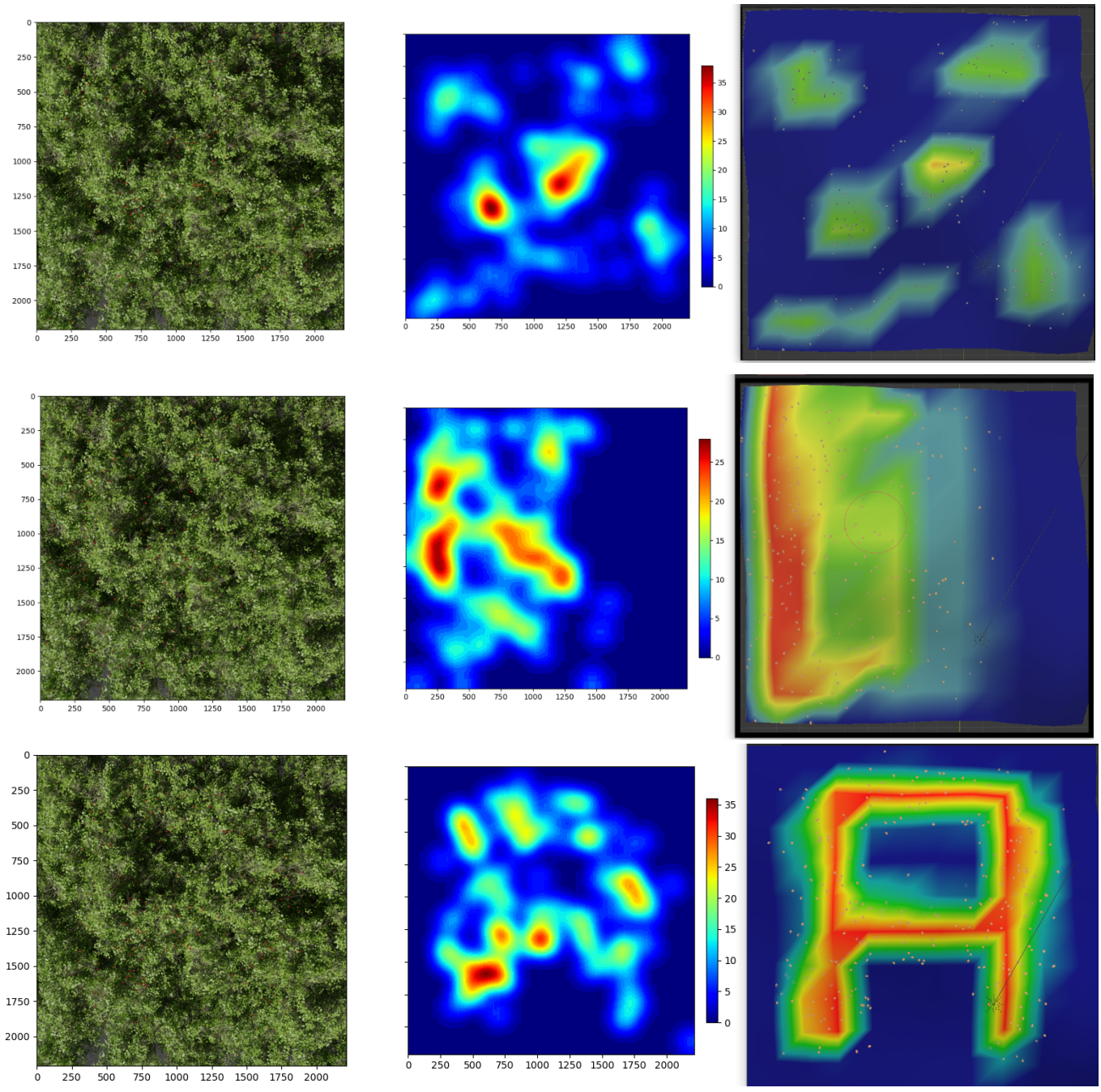


“A” distribution:



Results

The results have been generated with very minimal change to the base framework parameters found for the CRAID dataset with a fixed kernel-size of 1000.

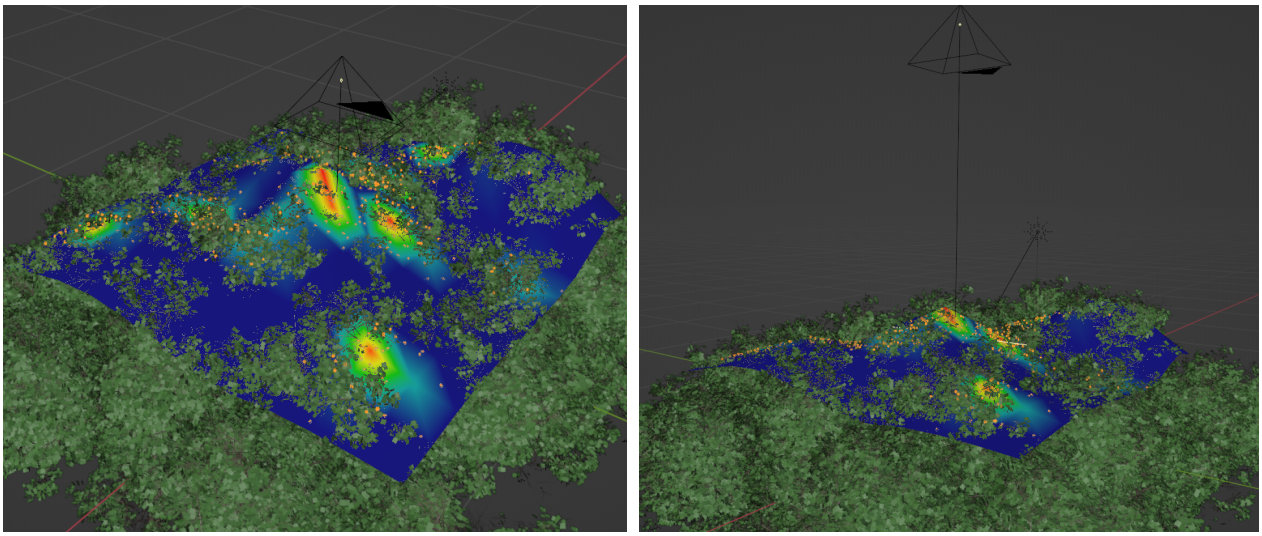


The model demonstrates a notable performance across all three distributions, owing to the inclusion of cluster color filtering as well as the incorporation of the two convolution layers that contribute to the generalization of the output.

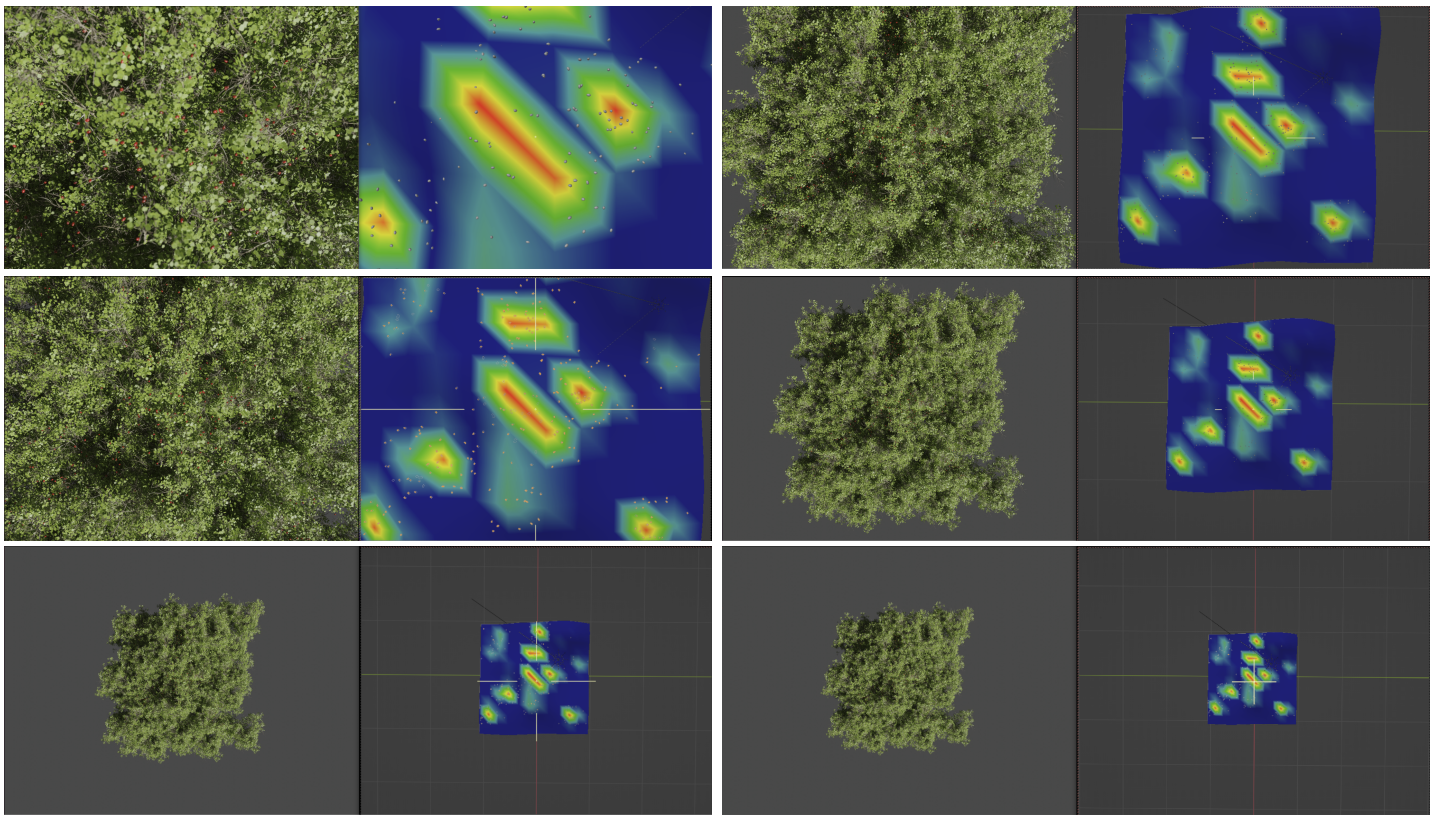
Creation of custom scenes to account for specific drone camera and viewing angles

To optimize the model for the specific task of counting berries using a drone, a virtual camera was created to closely match the camera specifications of the DJI Mavic M3 drone. The camera on the drone has a 20-megapixel resolution, a 4/3" sensor size, a 24mm focal length, an aperture of f/2.8 and a minimum focal length of 1m. Once the scene and camera settings were configured, renders were captured at different altitudes, ranging from 0 to 5 meters with intervals of 0.5 meters [7]. The berries have been set to have an average size of 1.4cm.

From this a correlation to the height can be found between the best kernel for the model input, the gaussian and the max filtering.



[7] The renders (left) with their corresponding weight map used on the emitter (right)

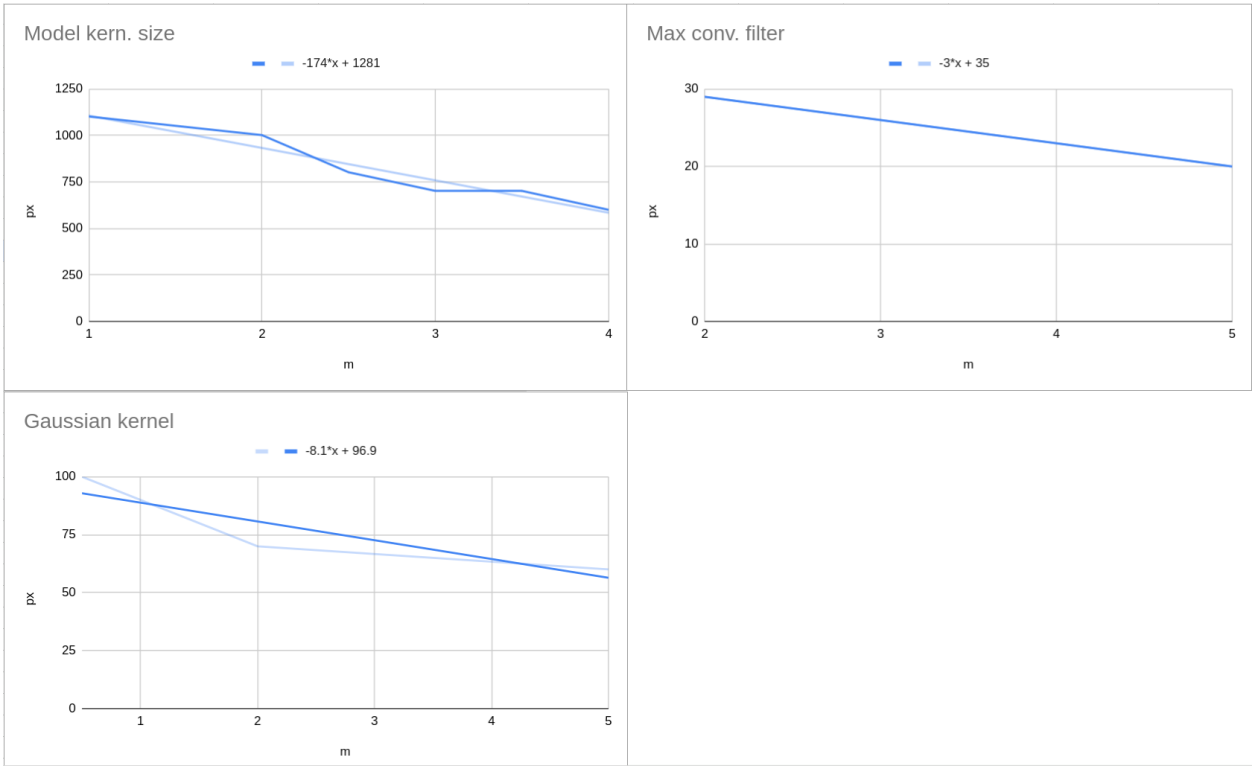


Results

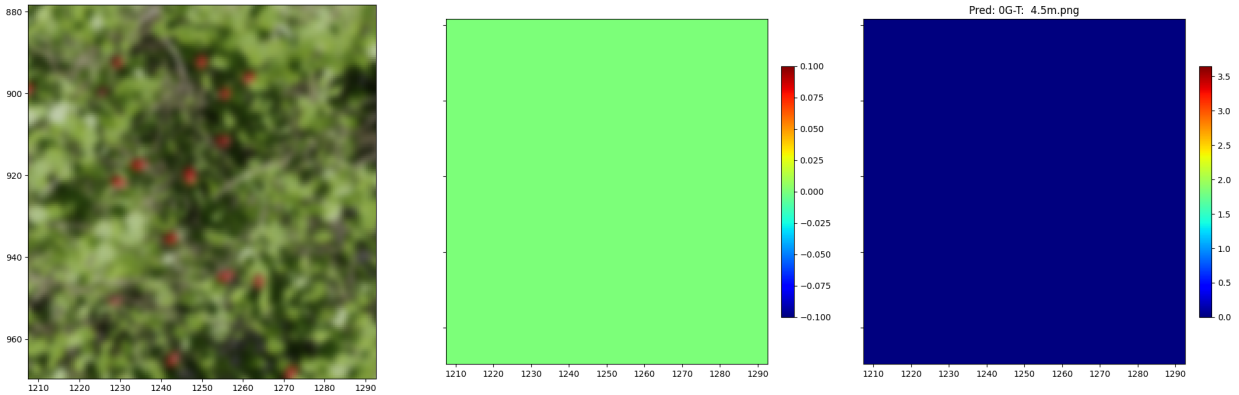
From the renders produced at various distances we identified, through testing, three linear functions that enable the model to autonomously determine kernel sizes for the three convolutions applied to the image: the kernel used for the model, the Max kernel for cluster expansion, and the Gaussian kernel for density visualization.

With the aid of these three functions, the model, tailored to this specific camera, can generate results independently if given the height, as showcased in the following pages.

To assess the framework's robustness, we conducted tests without altering any parameters, applying a 2x2 Gaussian convolution to introduce noise and simulate real-world scenarios. Remarkably, this noise had minimal impact on the model's performance at lower altitudes. At higher altitudes above 4m, the model cannot be faulted for not detecting berries, as they were too small to be discerned effectively [8].



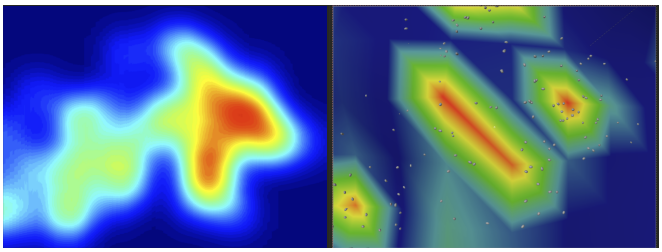
[8] Detail of the berries at 4.5m.



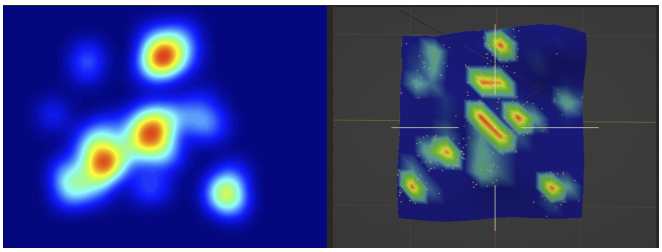
No blur

Appendix [10]

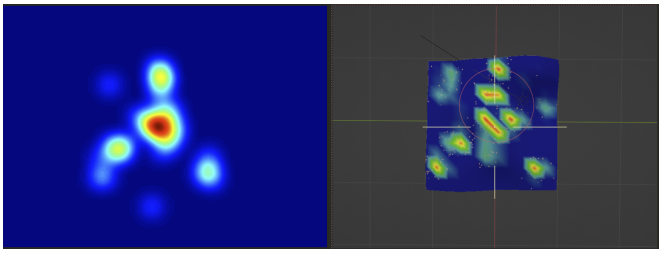
0.5m



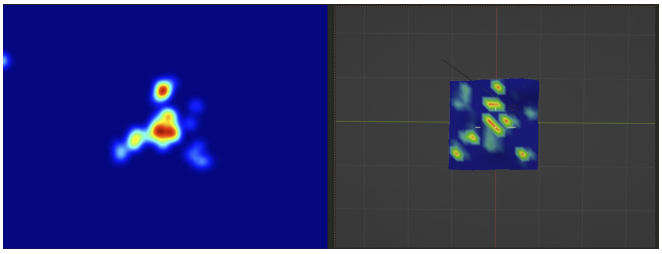
2m



3m



4.5m

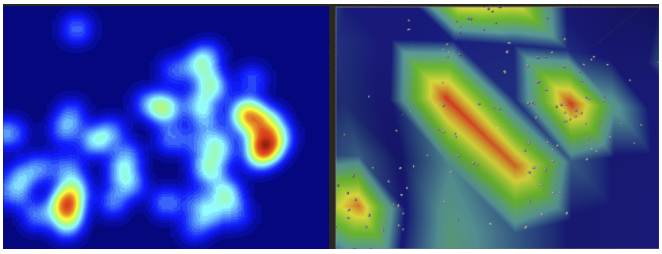


Blur

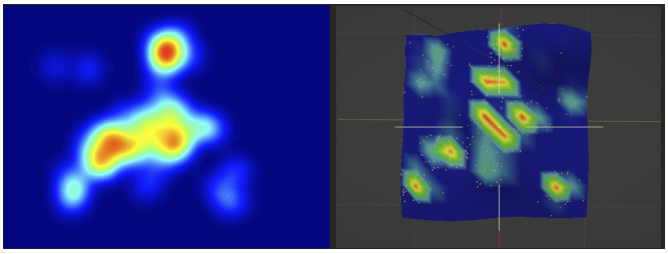
Appendix [11]

As mentioned above the model can't detect anything above 4m because as can be seen [8] the berries are way too small to be detected correctly.

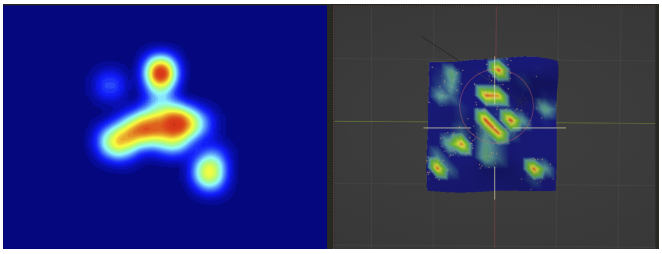
0.5m



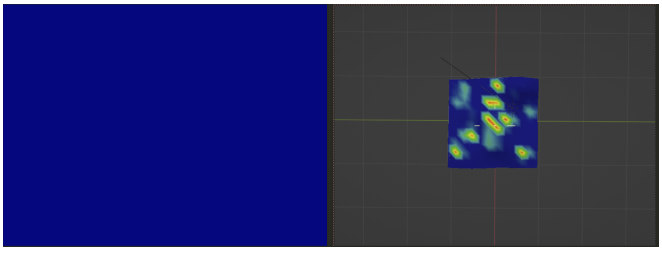
2m



3m



4.5m



In short, the use of adaptive kernel sizes in the model helps it generalize better at different altitudes. This adaptability allows the model to adjust its focus and capture features effectively, leading to improved performance in various environments. Additionally, the model's parameters can be easily tweaked to accommodate different situations and camera specifications, making it versatile and applicable to real-world scenarios.

Conclusion

In conclusion, the CounTX model presents itself as a completely viable solution for detecting, positioning, and counting objects classifiable by the CLIP model, despite its notable limitations.

Without the integration of the framework, the zero-shot model would be unusable on its own. The challenges faced in accurately counting berries within an extremely noisy environment were effectively overcome by incorporating supplementary modules surrounding the central framework.

These added modules significantly enhanced both the model's performance and its overall robustness. Moreover, the demonstrated generalization capabilities of the model in applications involving plants and berries were evident in its strong performance on specially designed test renders representing an unfamiliar environment; not to mention the remarkable accuracy that it demonstrated on the CRAID dataset.

In its entirety, the CounTX model, helped by the comprehensive framework and the modules built around it, presents significant promise in effectively addressing a wide range of object detection and counting tasks.

Citations

Amini-Naieni, Niki, et al. "Open-world Text-specified Object Counting." *arXiv preprint arXiv:2306.01851* (2023).

Full size diagram

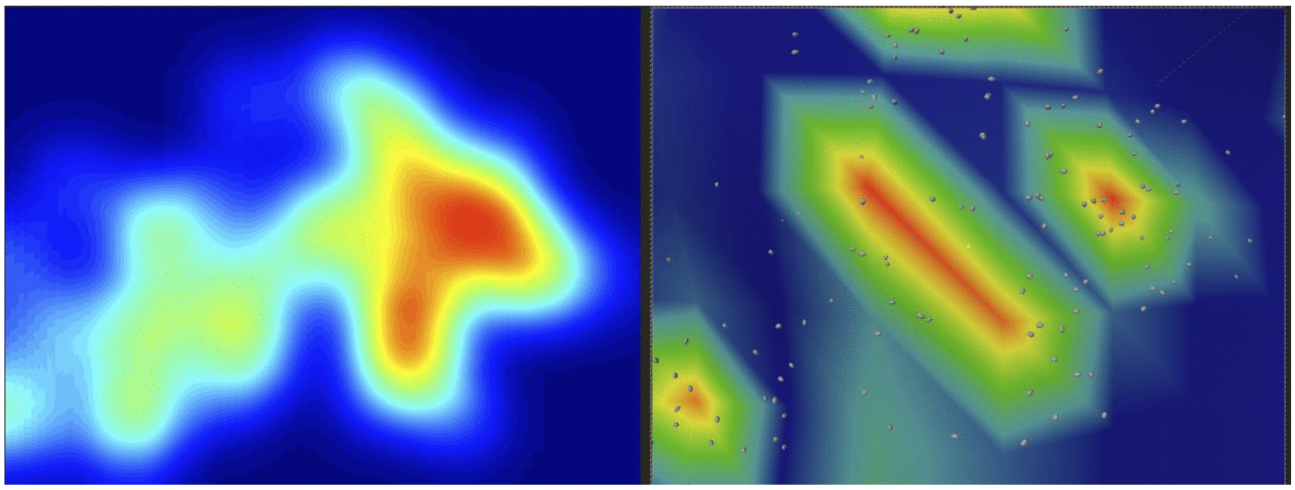
Appendix

Gifs and images

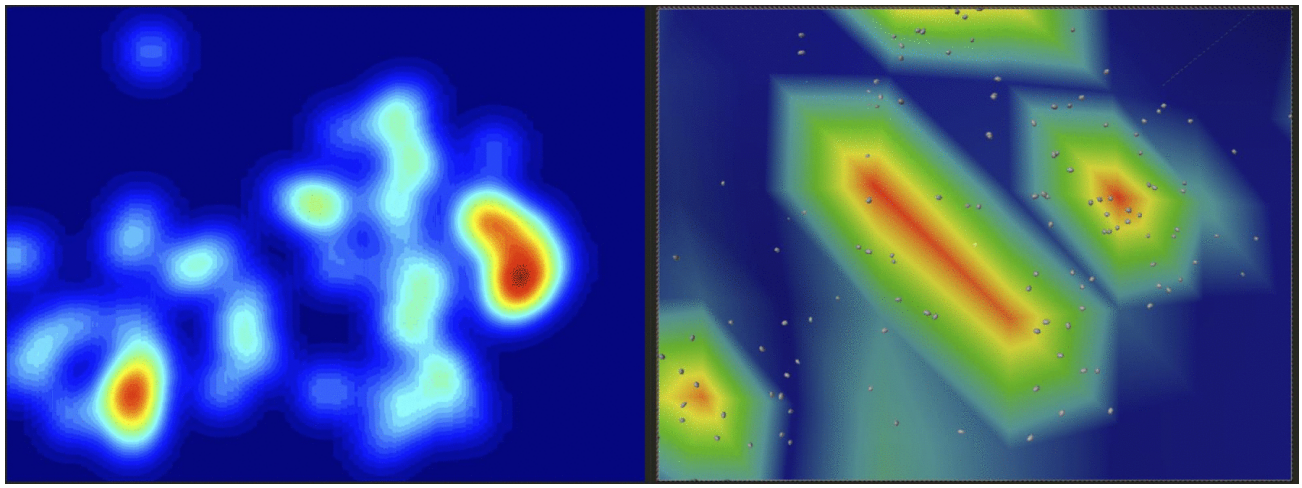
[9] (Gif) Difference between different kernel sizes and heights

2m

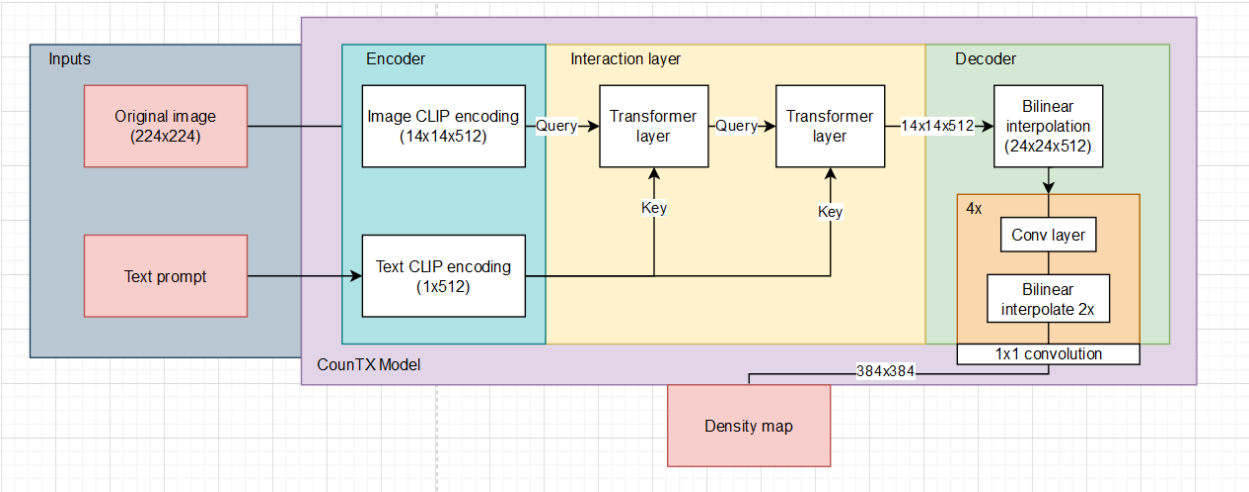
[10] (Gif) Image sequence ranging from 0.5 to 5 meters without gaussian blur.



[11] (Gif) Image sequence ranging from 0.5 to 5 meters with gaussian blur.



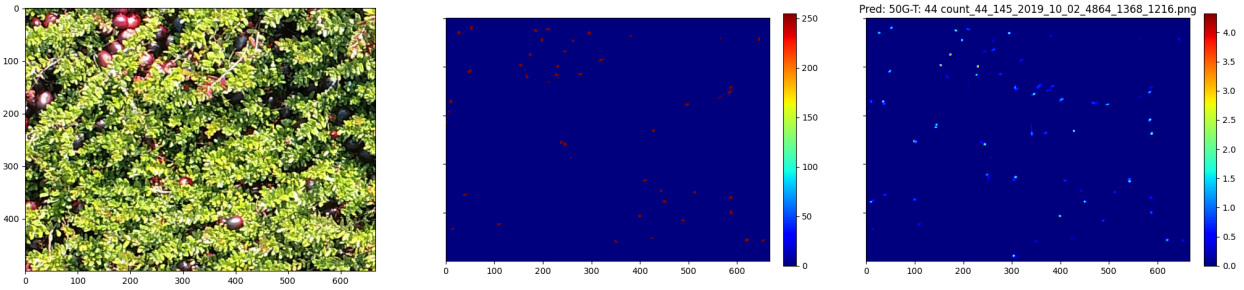
[12] CounTX model



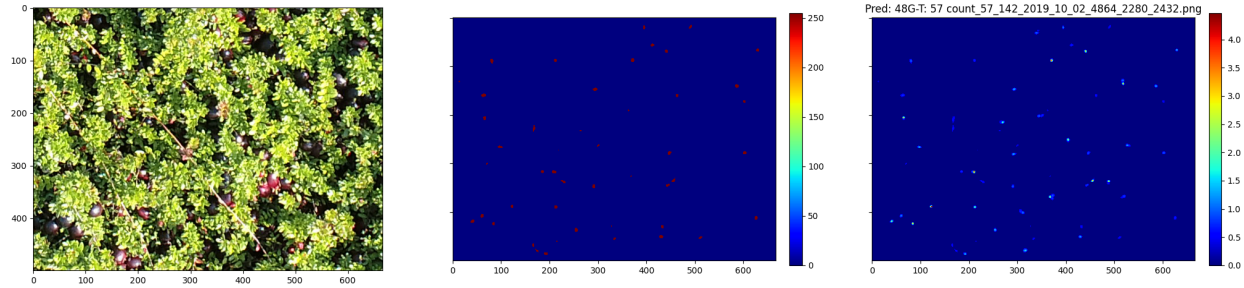
Further CRAID examples

Positive

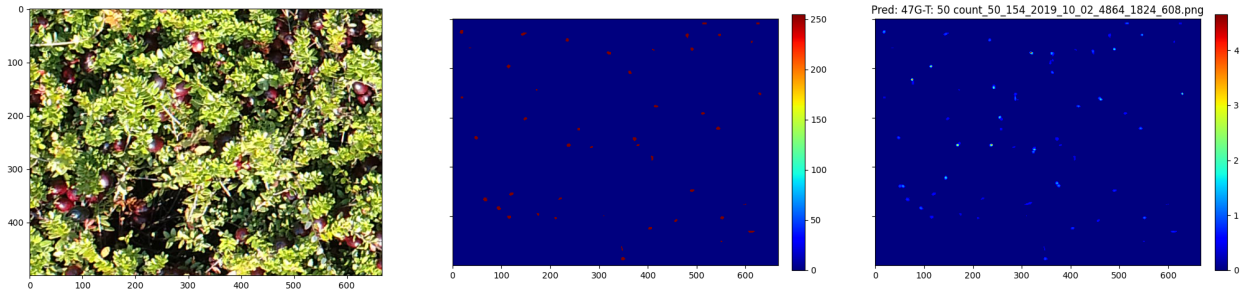
[13] Very clear density map and good cluster positioning



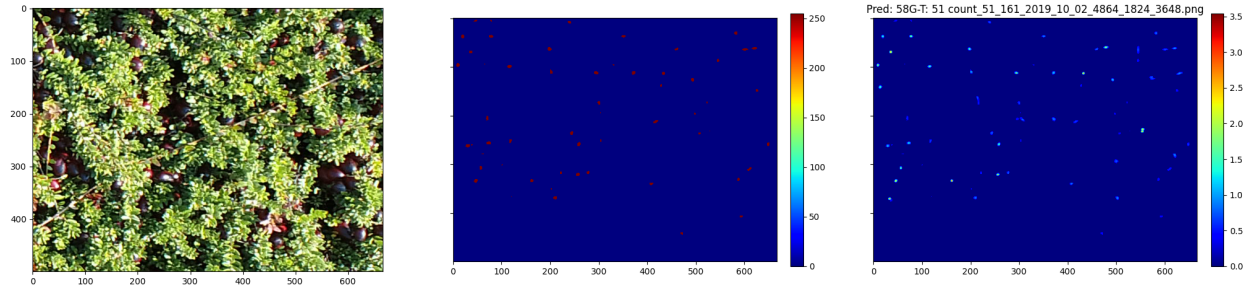
[14] Good density map and clusters.



[15] Clean density map and good positioning on an image with a high dynamic range.

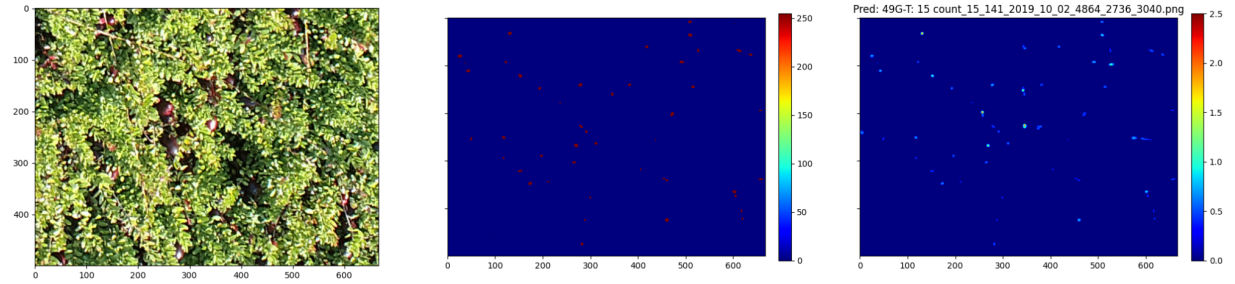


[16] Good berry finding in an image with a lot of intermingling and shadows.

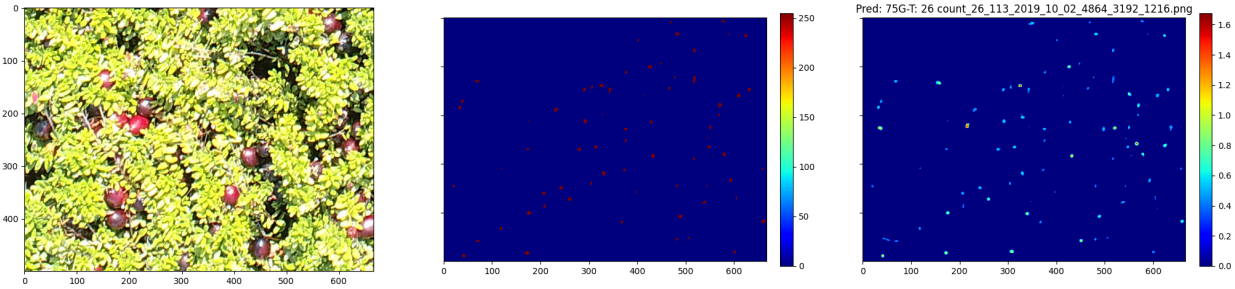


Negative

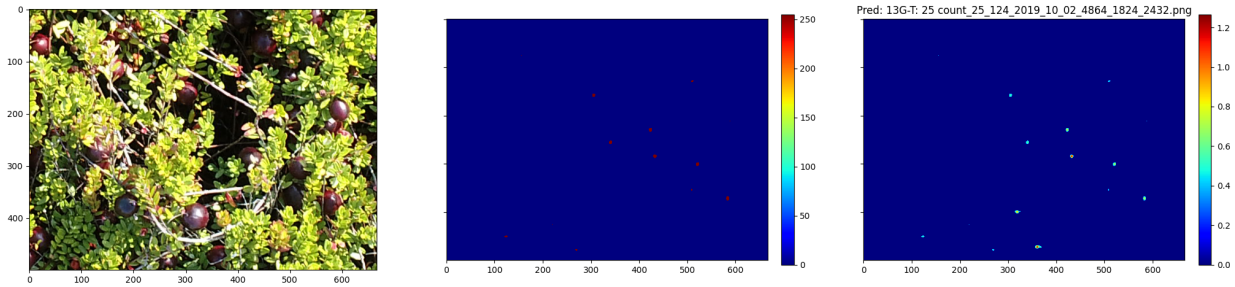
[17] Leaves misclassification given probably by the noise created by the contrast of the shadows.



[18] Misclassification given by the overexposure of the image.



[19] Misclassification given by the size of the berries and leaves contrast.



[20] Misclassification given by the contrast of the shadows.