Deep Learning Method for On-Ear Corn Kernel Counting

Hilario Gabriel Capettini Croatto
hilariogabriel.capettinicroatto@studenti.unipd.it

Abstract

Crop monitoring and yield estimation provide farmers with important information to take decisions. To do that farmers have to manually count the number of kernels on ears which is a time consuming and prone to error activity given that each ear has between $200 - 600$ *kernels. In this work we propose the usage of deep learning for counting the on-ear kernels by using images that can be taken with a phone. In particular we focus in the usage of a U-Net architecture with a VGG16 backbone for feature extraction. As publicly available data is scarce we first train on a wide dataset containing corn ears in different contexts and then fine tune on a small dataset containing pictures that a farmer could take. We show that after the tuning of the parameters the model achieve state of the art models results. We also observe that the proposed model produce accurate density maps that can be used for kernel localisation.*

1. Introduction

Corn yield estimation is an important task for farmers, the assessment of the field production allow them to forecast future sales, to manage grain storage and transport requirements. Corn yield is driven both by optimising the number of plants per area as well as the number of full, mature kernels on an ear. Depending on the variety of corn, each ear may have 400–900 kernels when fully developed; manually counting each kernel is slow, inaccurate, and labour intensive. The automation of this counting process would provide farmers with substantial speed and accuracy improvements.

The field of object counting is a wide one in Computer Vision, the aim is that of estimating the number of target objects presents in a still image or video frames. It has been successfully applied to different scenarios such as counting cars on a highway or estimating the number of people in a crowd.

In general we can identify three main approaches to solve this task: *regression-based* techniques try to map low level extracted features to the number of objects in the image; *detection-based* approaches aim at finding the right-sized moving window able to detect a single object; and finally *density-map generation* methods first produce a density estimate of the objects of interest in the image (density map) and the count is simply obtained from an integration of this map. The first two techniques are less effective when the objects of interest are partially occluded, that is why most up to date works in crowd counting have been focused on density-map estimations.

Because corn ear images present kernels that are partially occluded on the borders, in this work we focus on the density-map estimation technique. Given that the target here is the correct translation of an image into a density map, this approach strongly relies on Fully Convolutional Networks (FCN). In particular we decided to go for the fully convolutional encoder-decoder structure of U-Net which is a characteristic architecture of segmentation tasks. After exploring the impact of different parameters in the ground truth generation and in the model itself, we managed to obtain state of the art results.

2. Related work

In general the problem of object counting in dense scenes is focused in crowd counting and it is there where new ideas are tested. The general trend, as reported by Rong et al. [7], is that of using an encoder-decoder structure where the front end is a pre trained CNN. This pre trained backbone usually is VGG-16 [10] as shown in the work done Li et al. [5]. This network is considerably deep with 16 layers. The convolution filters are small (3×3) and profits from pooling layers to reduce the size of the feature maps.

U-Nets [8] were first introduced to solve segmentation tasks in biomedical images, but are currently being used to solve the crowd counting problem too. Shen et al. [9] made use of a U-Net structure with an adversarial loss to produce high-quality density maps. Huynh et al. [2] put forward an inception U-Net-based multi-task learning for crowd counting, density map-generating, and density level classification. In [12], the authors proposed U-Net-like architecture called W-Net, which applies a reinforcement branch to improve the crowd counting accuracy and converge quicker. For crowd counting these approaches are justified because images can be extremely dense, which is not the case for

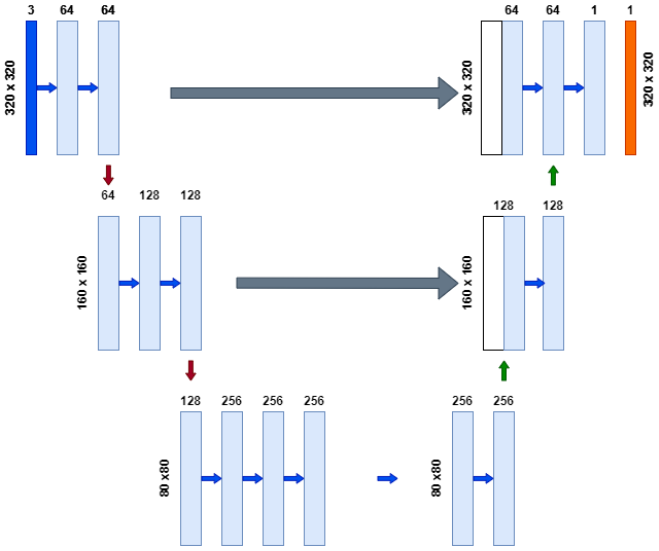

Figure 1. Scheme for the U-Net implemented for this project. Blue arrows represent convolutions, red arrows represent max pooling, green arrows are upsamplings and grey arrows are copy and crops from encoder to decoder.

corn images were our main problem is the occlusion and not the density.

In particular a few works deal with counting corn kernels on ear, Khaki et al. [4] propose DeepCorn which is composed by pre trained VGG-16 encoder for feature extraction. In the decoding phase they propose the merging of feature maps (obtained in this phase, not from the encoding) from different scales to make the model robust against scale and perspective changes. While using U-Net we are taking a similar approach but using feature maps from the encoding phase instead (see figure 1).

3. Dataset

The Corn Kernel Counting Dataset has been collected and published by Hobbs et al in 2021 [1], at the moment of writing this work it is the only publicly available dataset of this kind that we found. It is constituted by 402 images of corn ears, containing over $113,000$ individual kernel segmentation masks and the associated bounding boxes in COCO format. This dataset is highly varied including images with different resolutions, sizes, number of ears, total number of kernels, background, lightning and also corn varieties (see figure 2).

The dataset is divided in three sections, *base* dataset contains the bulk of the data (313 images). These images have a wide range of appearances and may have multiple ears. *Narrow* dataset consist of 60 images, all of them containing a single ear of yellow feed corn located roughly in the centre of the image and vertically oriented. Figure 2 display examples of both sub datasets. Finally the *many* dataset has

Dataset	Number of images	Min	Max	Avg	Total
Base	313	6	802	226	54809
Narrow	60	232	346	294	14138

Table 1. Statistics for the datasets we used for this work.

images with at least 4 ears, a large number total kernels, occlusion and shadows. In the present work we decided to only use the first two subsets. The idea is to train the algorithm using the *base* dataset which contain most of the images and then fine tune the algorithm on the *narrow* dataset which contains images that are more adequate for the counting purpose.

Figure 2. A few examples of both sub datasets are shown, images from the base dataset on the left column and images from the narrow dataset on the right.

4. Method

4.1. Network architecture

Most of the works use a fully convolutional encoder decoder structure where the image is compressed so the key features are extracted and then they are expanded again generating the density map. A famous network with this structure is U-Net [8] first presented as a network for semantic segmentation for biomedical images. The encoding path (here implemented using a truncated VGG-16) captures features at different scales of the images by using a traditional stack of convolutional and max pooling layers, in particular we use the first seven convolutions increasing the number

of feature maps with the sequence $[3, 64, 128, 256]$, a the same time the max pooling layers help to reduce the feature maps size at half the original size $[320, 160, 80]$. The decoder path is a symmetric expanding counterpart that uses nearest neighbor interpolation for upsampling then the corresponding feature map of the encoder layer are appended and finally we are ready to apply the convolutions reducing the number of feature maps (see figure 1). At the end, the output passes another convolution layer with the number of feature maps being one, finally applying the identity activation we obtain our density map. The result is a u-shaped convolutional network that offers a solution for good localisation and counting.

Our proposed network uses the first nine convolutions of a VGG-16 [10] trained on the ImageNet dataset as the backbone for feature extraction. Originally proposed for image classification, the VGG-16 network stacks convolutional layers with a fixed kernel size of 3×3. Even though, the VGG-16 network was originally designed to process input images with size of 224×224, we use image inputs with size 320×320 because the proposed network can potentially learn more fine-grained patterns with higher resolution input images [11].

4.2. Ground truth generation

As previously described the dataset does not contain the kernel point like positions $\mathbf{x}$ but a mask build with the coordinates of the border of each kernel, so for each kernel i in an image I we have a set of points $\{\mathbf{k}_1, \mathbf{k}_2, ..., \mathbf{k}_P\}$ where P is the number of points describing the kernel mask; as we are interested in the centre of the kernel we assume it is located in the centre of mass of the mask and we estimate it as:

$$\mathbf{x} = \frac{1}{P} \sum_{p=1}^{P} \mathbf{k}_p, \tag{1}$$

doing this we obtain the position $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$ for all the kernels in an image.

After that, we are able to generate the ground truth density map for the kernels positions, to do it we convolve the image with normalised Gaussian kernels G_σ centred at annotated kernel positions $\mathbf{x_i}$. The obtained label displays the corn kernel distribution map

$$D(\mathbf{x}) = \sum_{i=1}^{N} \delta(\mathbf{x} - \mathbf{x}_i) G_\theta. \tag{2}$$

The σ parameter which is the standard deviation of the Gaussian filter should be picked as the approximate average distance between kernels in the images. As our images are so heterogeneous and this distance does not remain constant among images we decided to explore different values and to keep the one that output the best results on the *narrow* dataset. examples of the produced density maps are shown in figure 3.

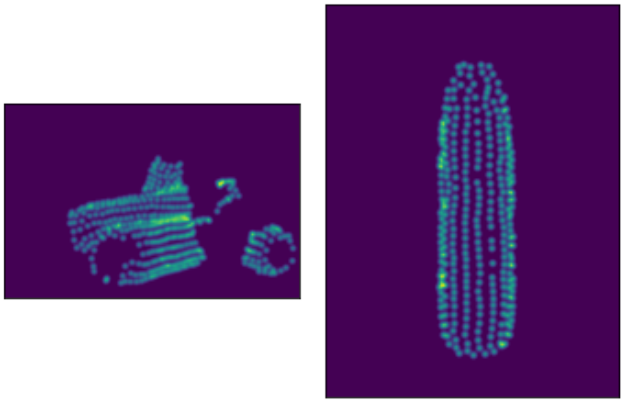

Figure 3. Density maps corresponding to the first two images in figure 2, produced with $\sigma = 12$.

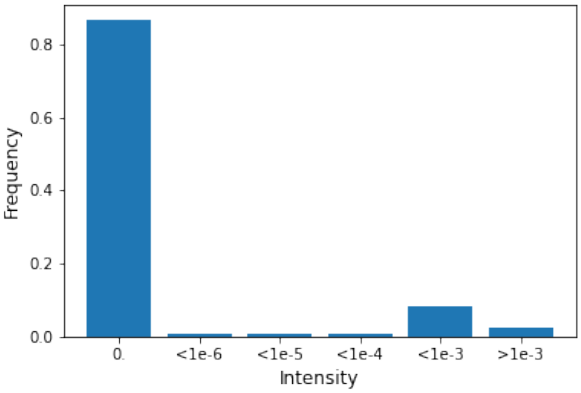

Figure 4. Pixel intensity distribution for the generated density maps.

Because the Gaussian is infinite and therefore can extend well outside the original mask boundaries, we truncated all values below $1e^{-4}$ and then renormalized the map so that it sums to the number of kernels presents in the image.

It should also be kept in mind that Gaussian kernel filtering generates low pixel values as can be seen in the distribution in figure 4. This could lead to gradient vanishing issues in the training process [7]. To tackle this problem we introduce an enhancing factor k whose effects were investigated in this work.

4.3. Images pre-processing and Data Augmentation

As already mentioned the U-Net backbone was pre-trained on the ImageNet dataset, so our training data was zero-centered with respect to it. Because the amount of data for training was small we decided to augment it by training on cropped sections of the images of a size 320×320. As our dataset contains images of different sizes, a few of them smaller than the requested for the crop, we apply zero-padding so all the images have at least the required size for the crop.

Apart from training on cropped sections of the images we also implemented more traditional augmentations using the tochvision [6] package. We implemented random rotations, vertical flipping, horizontal flipping and bright contrasts on a per image, per epoch basis.

4.4. Training details

We divided the datasets into three subsets in the following proportions: 64% training, 16% validation and 20% testing.

The models used for the density map estimation used a U-Net architecture with VGG-16 pretrained on the ImageNet dataset as backbone. We used Adam optimization with a learning rate of $1e^{-5}$, a batch size of 10 and a maximum of 200 epochs for the initial training on the *base* section of the dataset and a learning rate of $1e^{-6}$, a batch size of 10 and a maximum of 100 epochs for the fine tuning on the narrow dataset.

The loss function used for training was the pixel Mean Squared Error (pixel MSE), which returns the squared Euclidean distance between the ground-truth density map D and the predicted density map $\hat{D}$:

$$MSE = \frac{1}{M}\sum_{m=1}^{M} |D(x_m) + \hat{D}(x_m)|^2 \tag{3}$$

All models were constructed using PyTorch, and PyTorch Segmentation Models [3]. The training of the models was done using a Google Cloud machine equipped with a NVIDIA Tesla T4 GPU.

4.5. Evaluation details

Because the network was trained using image patches, in order to evaluate the full image to density map translation we processed the test dataset as follows:

- we apply zero-padding to reach a test image size big enough to split it in an integer number M of patches of size 320×320 (see figure 5).
- we perform an inference for each of the M patches of the image

Figure 5. Image from the test set. After being padded, for each of the red squares of 320×320 pixels, a predicted density map patch was obtained.

- we build the final map attaching the predicted density map patches together and cropping it to the original image size.

Following this process we end up with a density map that perfectly overlaps the image and represents the kernel density distribution.

Finally to evaluate the models performance we used the Mean Absolute Error (MAE), the Root Mean Absolute Error (RMSE) and the Mean Absolute Percentage Error (MAPE), which are defined as follows:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |C_i^{GT} - C_i^{pred}| \tag{4}$$

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} |C_i^{GT} - C_i^{pred}|^2} \tag{5}$$

$$MAPE = \frac{1}{N}\sum_{i=1}^{N} \frac{|C_i^{GT} - C_i^{pred}|}{C_i^{GT}} \tag{6}$$

where N,C_i^{GT},C_i^{pred} denote the number of test images, the ground truth counting and the predicted countings for the ith image, respectively. The counting are obtained by direct sumation over the pixels values $z_{h,w}$:

$$c_i = \frac{1}{k}\sum_{h=1}^{H}\sum_{w=1}^{W} z_{h,w} \tag{7}$$

the enhancing factor k has to be taken into account to retrieve the original count.

We noticed that the network is not always able to return the zero value where no kernel is present and it turns into small values that affect the final count. That is why we

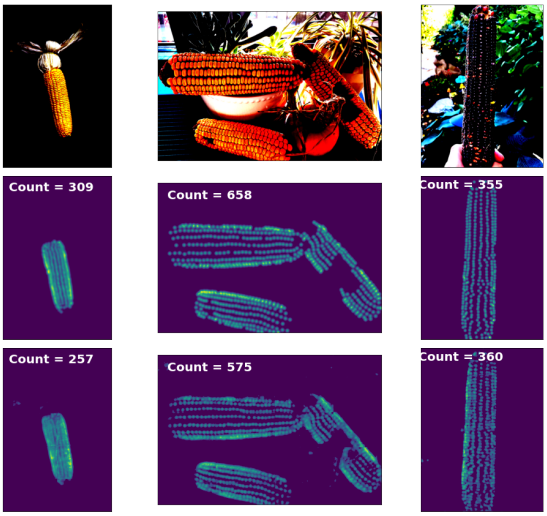

Figure 6. Predictions done using the model trained on the base dataset (from top to bottom: images, ground truth, predicted map), the model correctly identifies the kernels and the densities.

decide to put a threshold on the estimated density map to zero-out regions where the value of the density map is very small [4], this threshold was the same as the one used for the ground truth generation $1e-4$.

5. Experiments

In this section we describe the experiments we performed and the obtained results.

5.1. Training on base dataset:

As already stated, our target was to produce accurate predictions of the number of kernels in images with only one ear. However due to the fact that narrow dataset was so small we decided to first train the model on base dataset which contains a huge variety of images. In general we observe that the model learnt to recognise the kernels belonging to ears and correctly assign a a higher intensity to these regions in figure 6 we observe that the identification is done for ears of different colours, different scales and also recognise kernels that belong to the ear only.

When the metrics are analysed (table 2) we observe that the model performance is rather poor, on the base dataset while slightly better on the narrow dataset. Something we expected given the more simple images.

5.2. The importance of the threshold.

As already mentioned in subsection 4.5 after the predicted maps are obtained we apply a threshold to the values i.e. all values below $1e-4$ are converted to zero. The application of this post processing technique proved to have an enormous impact in the final metrics improving them around a $\%30$ (table 3). This huge impact can be

Metric	Base dataset	Narrow dataset
MAE	114	83
RMSE	155	85
MAPE	0.53	0.27

Table 2. MAE, RMSE and MAPE metrics for model trained on base dataset.

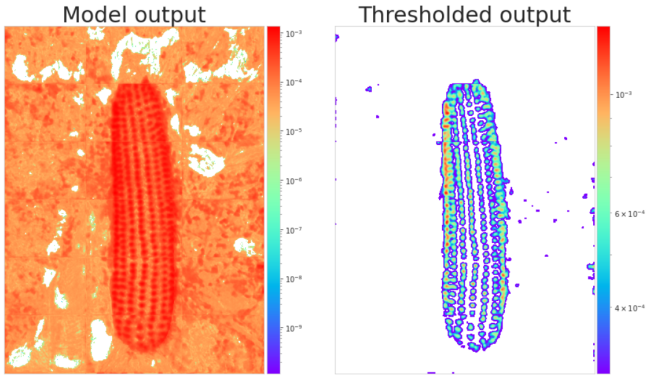

Figure 7. In this figure we show a predicted map before and after applying the threshold. The plot was done using a logarithmic scale for the colours.

Metric	Non thresholded	Thresholded
MAE	114	83
RMSE	155	132
MAPE	0.53	0.53

Table 3. MAE, RMSE and MAPE metrics on base dataset. The variations in the metrics before and after the threshold was applied.

Metric	
MAE	46
RMSE	51
MAPE	0.1

Table 4. MAE, RMSE and MAPE metrics for model fine tuned on base dataset.

explained by two reasons, the first one is that our model is not so good at producing zero values as outputs for the density map when no feature is present, this can be observed in figure 7 where the density map is plotted using logarithmic scale and we can clearly observe a blanket of small values. The second reason is that our images have big spaces with no ear, so at the end the presence of so many small values has a huge impact in the final sum.

5.3. Fine tuning on narrow dataset:

After the initial training on the base dataset we decided to perform a fine tunning on the narrow dataset, in table 4 we can observe the results on narrow test sub-dataset.

Figure 8 depicts some examples of the predicted density

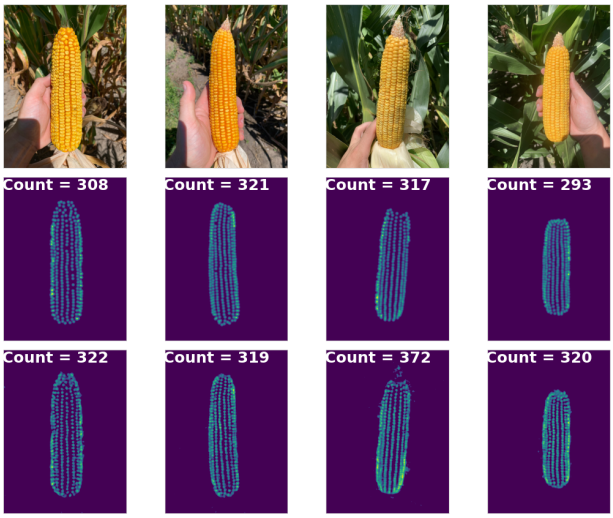

Figure 8. Visual results of our proposed method on narrow test dataset where the first row shows the input images, the second row the ground truth and the last row the prediction.

maps and the corresponding counts. Apart from the precision here obtained it is worth noticing that the maps are perfectly detecting the position of the kernels, so this model could also be useful for kernel localisation given that we are not only retrieving the amount of kernels but also the locations. This information could be useful for seed producers who take care in the geometric distribution of kernels in a ear.

5.4. Comparison with other approaches:

In the work presented by Khaki et al [4] they compare their algorithm (DeepCorn) with others using their dataset (110 corn ears similar to our narrow dataset). Unfortunately their dataset is not publicly available and a direct comparison with our algorithm is not strictly possible. However we observe that their predictions return a MAE $= 41$ which is around the results we are obtaining in our narrow dataset after the fine tuning (MAE $= 46$).

6. Conclusion

In this report we faced the object counting problem in the situation of a dense scenario where the objects are overlapping between them and a direct identification is hard. This problem was solved by means of Deep Learning tools, in particular using a modern encoder-decoder architecture called U-Net which allowed us to estimate the density maps of the images. By integrating these maps we obtained the number of kernels in the images presenting state of the art results. These maps are useful not only for the counting purpose but they also preserve location information which could be useful for other tasks.

In a future work we would like to investigate the usage of attention maps in order to help the model to concentrate in the region of the image where kernels are present given that a huge part of the image does not contain important information.

References

[1] Jennifer Hobbs, Vachik Khachatryan, Barathwaj S Anandan, Harutyun Hovhannisyan, and David Wilson. Broad dataset and methods for counting and localization of on-ear corn kernels. *Frontiers in Robotics and AI*, 8:627009, 2021.

[2] Van-Su Huynh, Vu-Hoang Tran, and Ching-Chun Huang. Iuml: Inception u-net based multi-task learning for density level classification and crowd density estimation. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3019–3024. IEEE, 2019.

[3] Pavel Iakubovskii. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2019.

[4] Saeed Khaki, Hieu Pham, Ye Han, Andy Kuhl, Wade Kent, and Lizhi Wang. Deepcorn: A semi-supervised deep learning method for high-throughput image-based corn kernel counting and yield estimation. *Knowledge-Based Systems*, 218:106874, 2021.

[5] Yuhong Li, Xiaofan Zhang, and Deming Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1091–1100, 2018.

[6] Sébastien Marcel and Yann Rodriguez. Torchvision the machine-vision package of torch. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1485–1488, 2010.

[7] Liangzi Rong and Chunping Li. A strong baseline for crowd counting and unsupervised people localization. *arXiv preprint arXiv:2011.03725*, 2020.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[9] Zan Shen, Yi Xu, Bingbing Ni, Minsi Wang, Jianguo Hu, and Xiaokang Yang. Crowd counting via adversarial cross-scale consistency pursuit. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5245–5254, 2018.

[10] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[11] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[12] Varun Kannadi Valloli and Kinal Mehta. W-net: Reinforced u-net for density map estimation. *arXiv preprint arXiv:1903.11249*, 2019.