A Deep Learning solution for on-ear corn kernel counting.

Computer Vision and Cognitive Systems 2021/2022
Professor: Lamberto Ballan

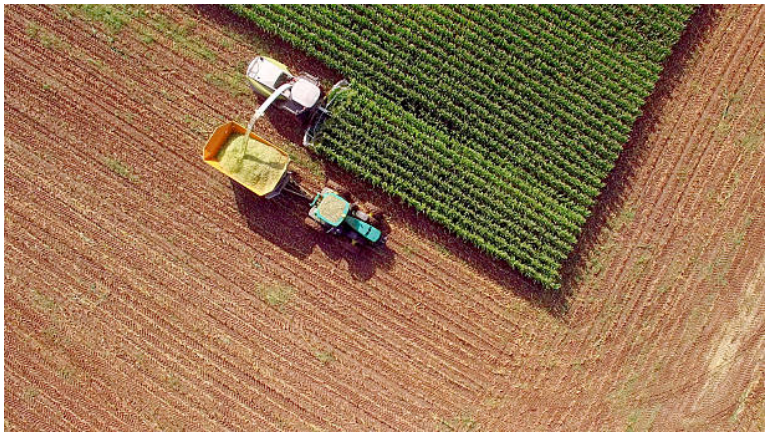
Hilario Capettini Croatto
September 27, 2022

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Overview

Overview

Corn Yield Estimation



Corn yield estimation is an activity yearly performed by farmers to organise storage and transportation.

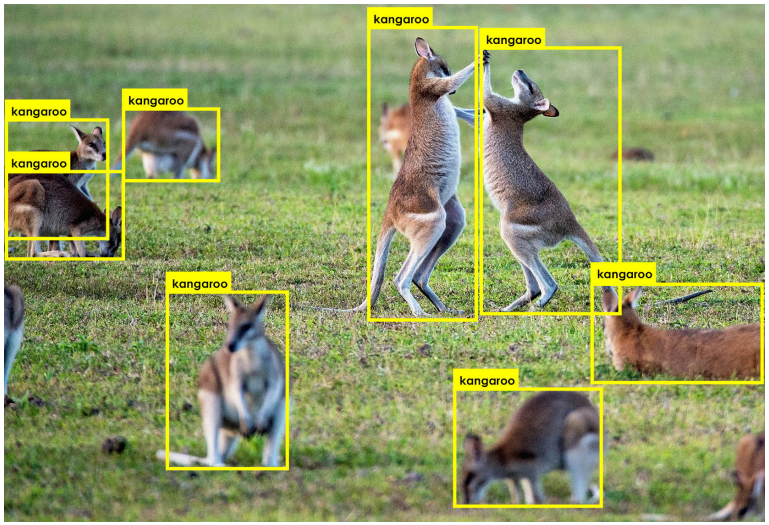
- Estimate the number of plants per hectare.
- **Estimate the number of kernels by plant.**

Corn Kernels Counting

The process of corn kernel counting is a time demanding activity and also prone to errors:

- Each ear has $200 - 600$ kernels.
- A farmer takes around 5 samples pro Hectare.

Object Counting



To count the number of objects present in an image there are different approaches based on deep learning:

- Regression-based
- Detection-based
- **Density maps generation**

Density maps generation

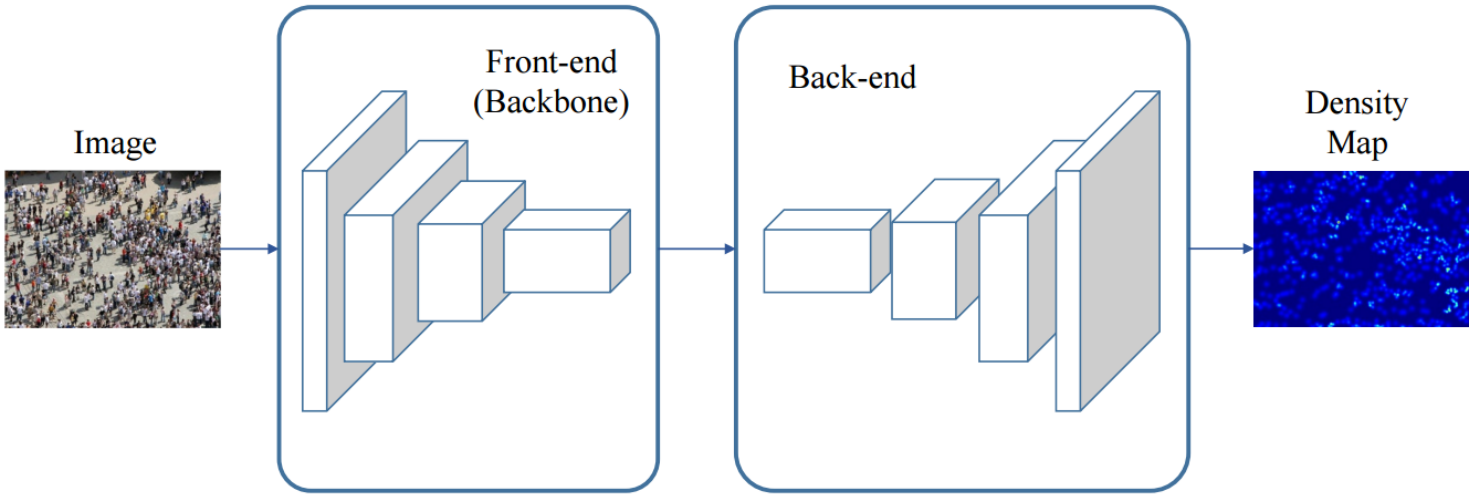


Figure: General pipeline for density maps generation, [Rong and Li, 2020].

To train these models we need to generate the labels properly, the ground truth density maps.
Once the density map was inferred, the count is a simple integration of it.

Overview

Related Works

Most of the works use the same front-end structure but exploit different strategies for the expansive path. All of them are variations of a U-Net.

- [Shen et al., 2018] uses U-Net structure with GAN loss.
- [Huynh et al., 2019] uses U-Net for multitask solving.
- [Valloli and Mehta, 2019] uses reinforcement branch to converge faster.

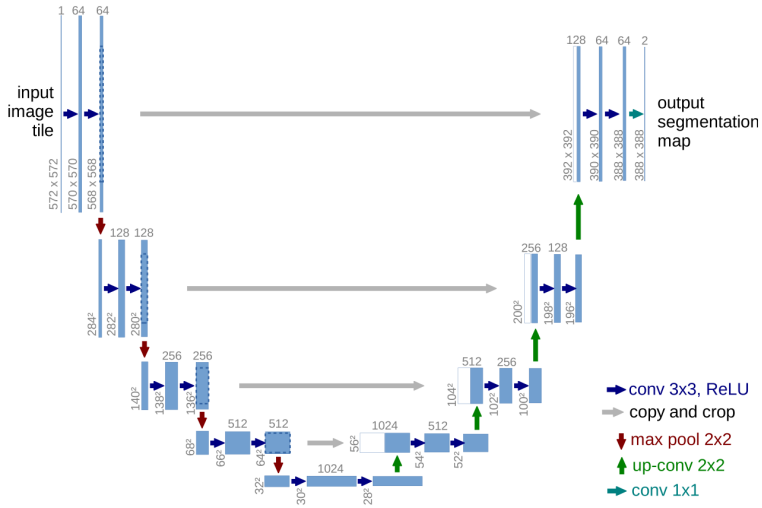


Figure: Original U-Net [Ronneberger et al., 2015]

Related works

In particular [Khaki et al., 2021] tackled the problem of corn kernel counting using an encoder-decoder model which combines feature maps from the decoding phase:

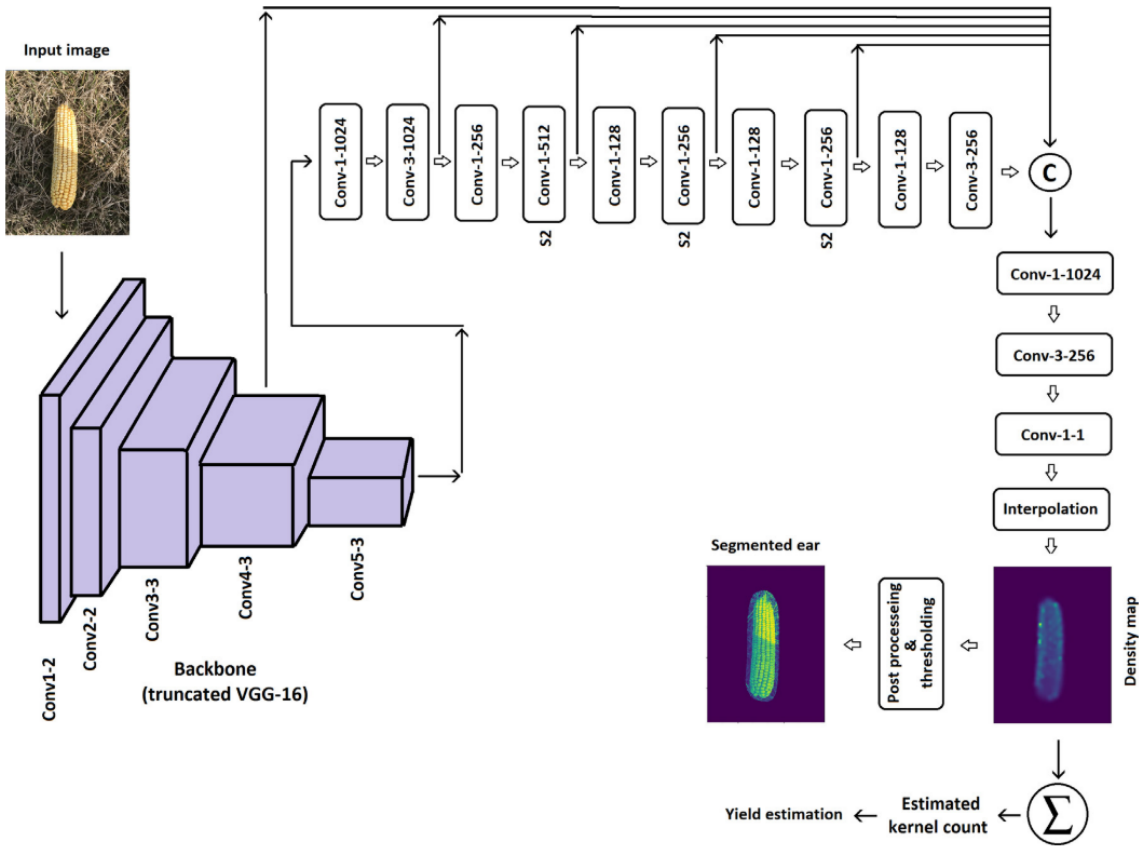


Figure: [Khaki et al., 2021].

Overview

Corn Kernel Counting Dataset

Corn Kernel Counting Dataset was collected by Hobbs et al. in 2021 [Hobbs et al., 2021]. It was collected to perform the counting task using detection based approach. The dataset is highly detailed and was annotated using COCO.

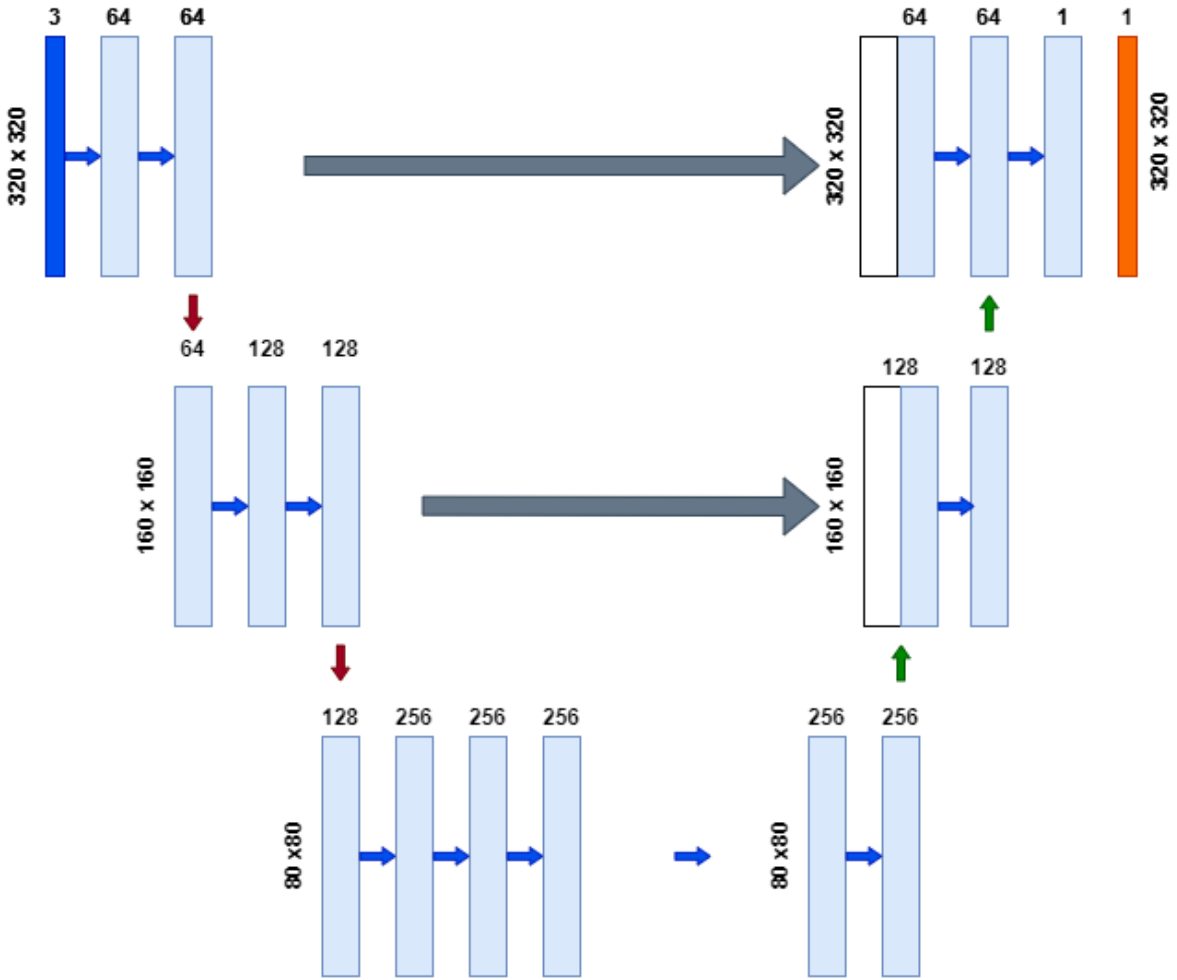
Dataset	Number of images	Min	Max	Avg	Total
Base	313	6	802	226	54809
Narrow	60	232	346	294	14138

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

Overview

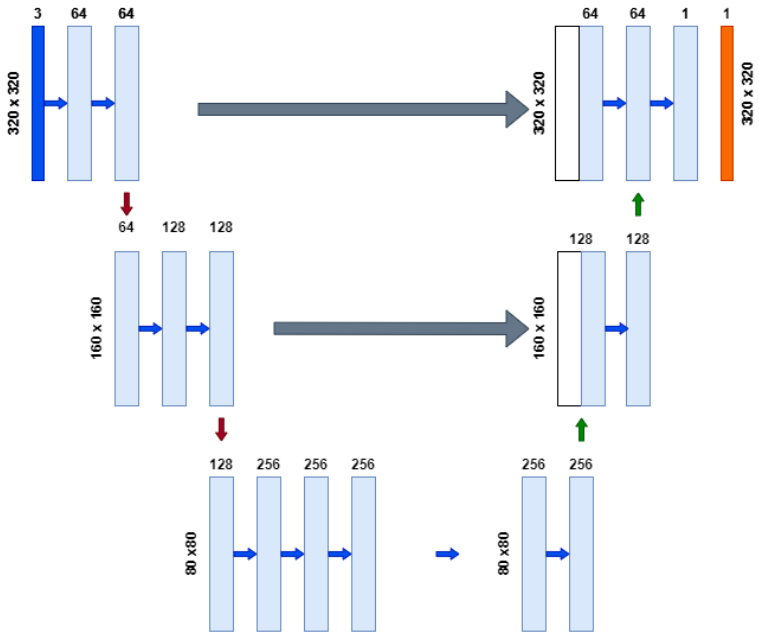
Network Architecture

U-Net was initially proposed by [Ronneberger et al., 2015] to perform segmentation of biomedical images.



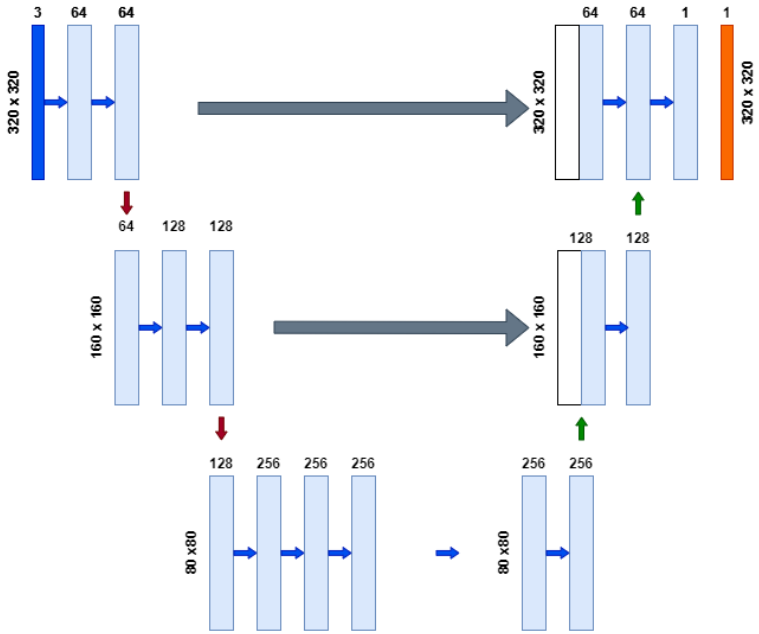
Network Architecture: Encoding Phase

- Pre-trained VGG16 backbone
- kernel size 3×3
- Downsampling image size sequence $[320, 160, 80]$
- Downsampling number of feature maps sequence $[3, 64, 128, 256]$



Network Architecture: Decoding phase

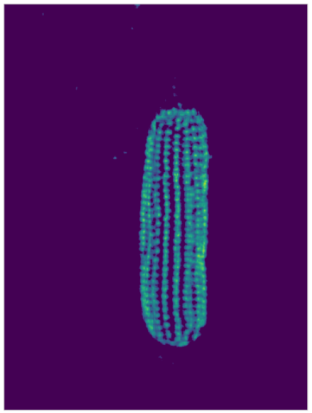
- Up-sampling using nearest neighbours
- Appending of the encoding maps
- Convolution using 3×3 filters
- Identity activation

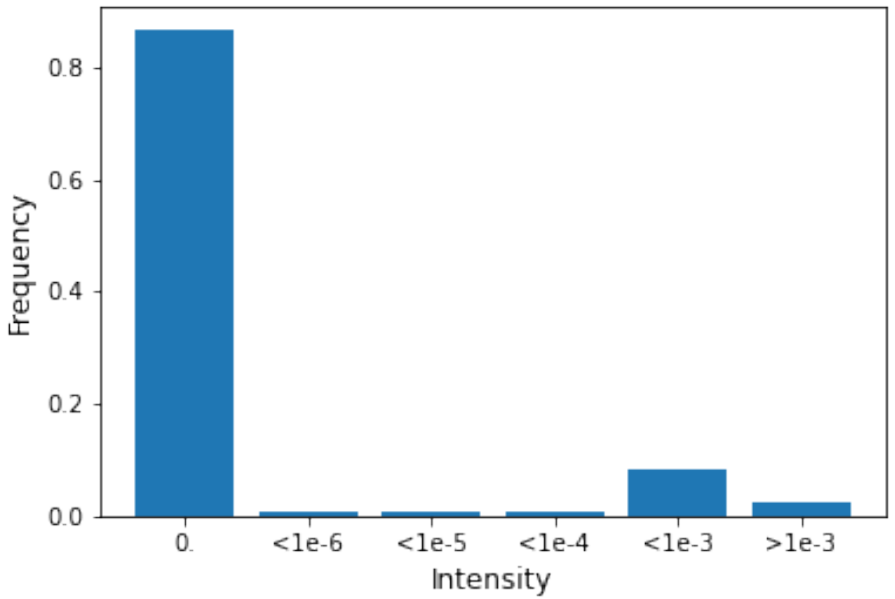


All the code was developed using Pytorch, the network was produced using the package PyTorch Segmentation Models [Iakubovskii, 2019].

Ground Truth Generation

The ground truth density maps are obtained by performing a convolution of an array containing the position of the kernels with a Gaussian kernel G_σ. The obtained label displays the corn kernel distribution map



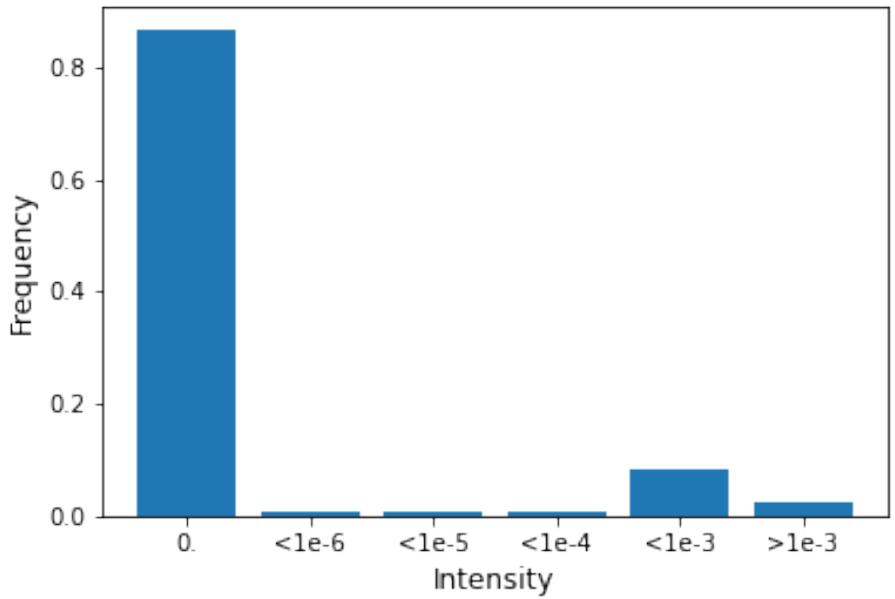


Ground Truth Generation

The ground truth density maps are obtained by performing a convolution of an array containing the position of the kernels with a Gaussian kernel G_σ. The obtained label displays the corn kernel distribution map

Parameters:

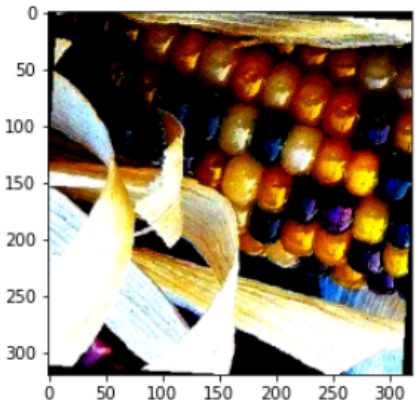
- $\sigma = 12$
- $threshold = 1e-4$
- $k = 1e5$

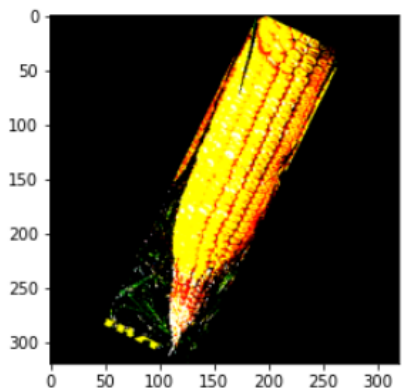


Images pre-processing

As the backbone of the U-Net was trained on ImageNet we have to cero center our corn images to ImageNet standard.
Data Augmentation:

- Crop random sections of 320×320 pixels
- Geometrical transformations (Rotations and Flips)
- Bright contrasts
- Other image transforms had negative effects (Hue saturation)





Training details

- First train on the base dataset
- Then fine tune on the narrow dataset
- 64% for training, 16% for validation and 20% for testing.
- Adam optimizer and batch size $= 10$
- The backbone weights are frozen
- On Base dataset:
  - $lr = 1e-5$
  - maximum of 200 epochs
- On Narrow dataset:
  - $lr = 1e-6$
  - maximum of 100 epochs

The used loss, Mean Squared Error:

$$MSE = \frac{1}{M} \sum_{m=1}^{M} \sum_{x,y=0} |D_m(x,y) - \hat{D}_m(x,y)|^2$$

Evaluation details

As we trained on patches of the original images, to count properly we have to split the image into windows and predict on each of them, the process is:

- zero-padding to split the image in an integer number M of patches of size 320×320.
- Perform an inference for each patch
- Build the final map attaching the predicted density map patches together and cropping it to the original image size.
- **apply a threshold to remove small values**

Evaluation details

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:

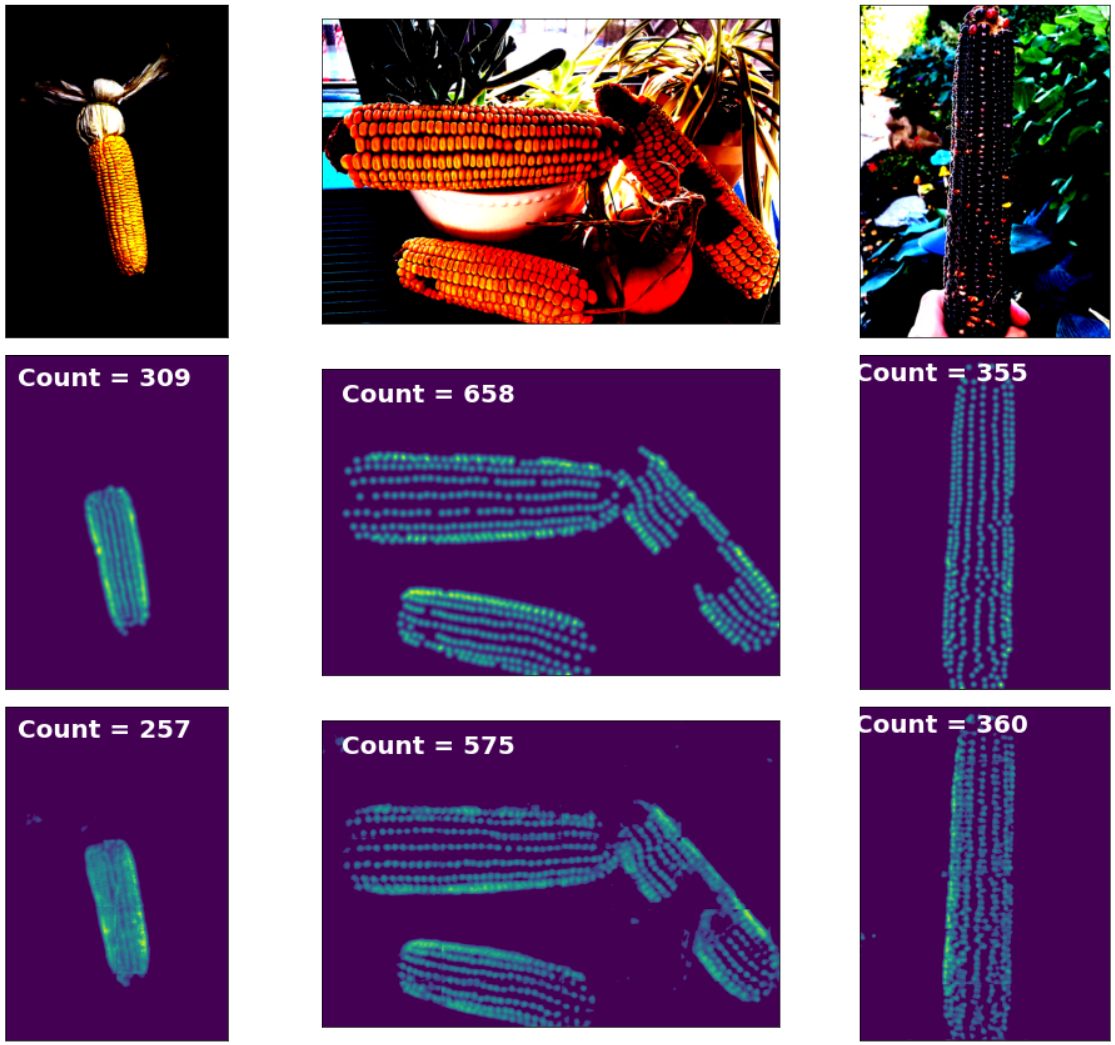
$$MAE = \frac{1}{N}\sum_{i=1}^{N}|C_i^{GT} - C_i^{pred}|$$

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

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

Overview

Training on Base Dataset

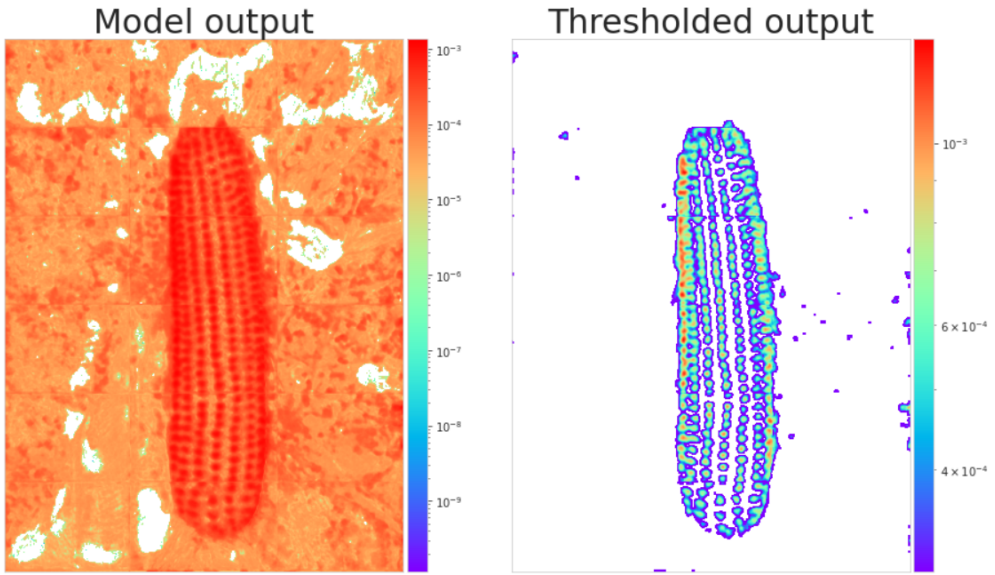


Training on Base Dataset

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

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

The importance of the threshold



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

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

Fine tuning on narrow dataset

Metric	
MAE	46
RMSE	51
MAPE	0.1

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

Comparison with other approaches

Metric	Capettini	Khaki et al.
MAE	46	41
RMSE	51	60

Table: Comparison between my work and work done by Khaki et al.

Overview

Conclusions

.

- I managed to implement a deep learning pipeline to count on ear corn kernels.
- The results are comparable to the ones obtained by other approaches.
- I managed to exploit the data at disposal to train the model.
- The obtained maps are useful not only for the counting purpose but they also preserve location information which could be useful for other tasks.

Thanks for your attention!

References I

Hobbs, J., Khachatryan, V., Anandan, B. S., Hovhannisyan, H., and Wilson, D. (2021).
Broad dataset and methods for counting and localization of on-ear corn kernels.
Frontiers in Robotics and AI, 8:627009.

Huynh, V.-S., Tran, V.-H., and Huang, C.-C. (2019).
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.

Iakubovskii, P. (2019).
Segmentation models pytorch.
`https://github.com/qubvel/segmentation_models.pytorch`.

References II

Khaki, S., Pham, H., Han, Y., Kuhl, A., Kent, W., and Wang, L. (2021).
Deepcorn: A semi-supervised deep learning method for high-throughput image-based corn kernel counting and yield estimation.
Knowledge-Based Systems, 218:106874.

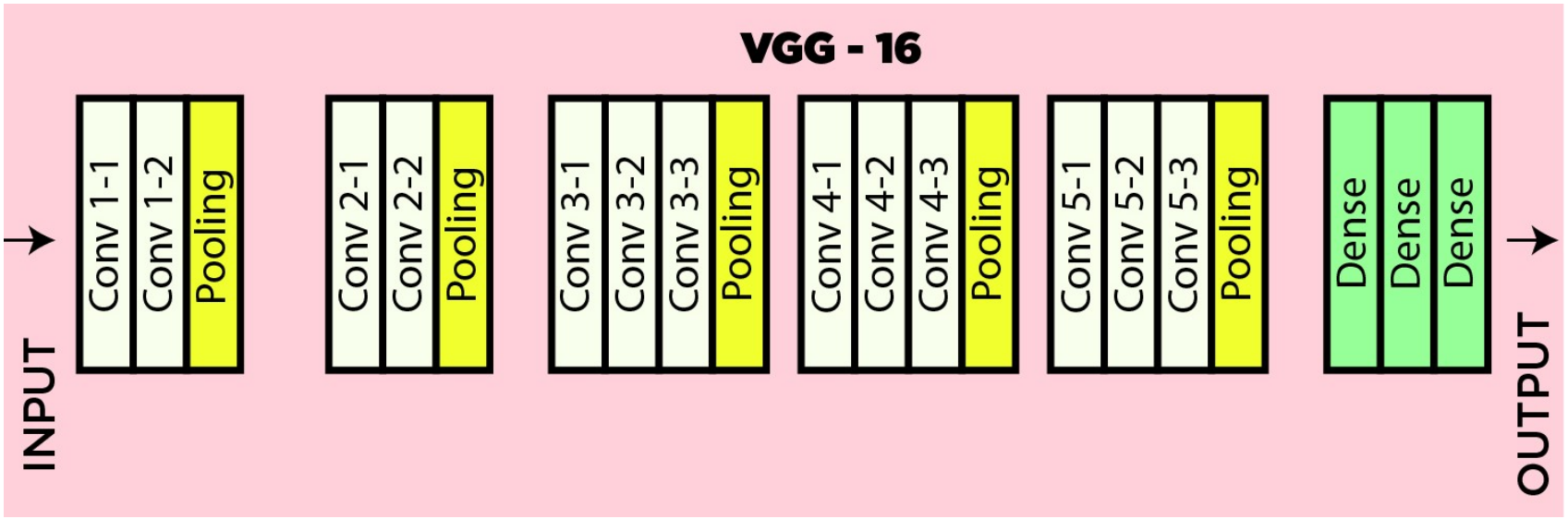
Rong, L. and Li, C. (2020).
A strong baseline for crowd counting and unsupervised people localization.
arXiv preprint arXiv:2011.03725.

Ronneberger, O., Fischer, P., and Brox, T. (2015).
U-net: Convolutional networks for biomedical image segmentation.
In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

References III


Shen, Z., Xu, Y., Ni, B., Wang, M., Hu, J., and Yang, X. (2018).
Crowd counting via adversarial cross-scale consistency pursuit.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5245–5254.


Valloli, V. K. and Mehta, K. (2019).
W-net: Reinforced u-net for density map estimation.
arXiv preprint arXiv:1903.11249.


VGG - 16
INPUT
Conv 1-1
Conv 1-2
Pooling
Conv 2-1
Conv 2-2
Pooling
Conv 3-1
Conv 3-2
Conv 3-3
Pooling
Conv 4-1
Conv 4-2
Conv 4-3
Pooling
Conv 5-1
Conv 5-2
Conv 5-3
Pooling
Dense
Dense
Dense
OUTPUT