Proceedings of the Federated Conference on Computer Science and Information Systems pp. 87–90
DOI: 10.15439/2019F94
ISSN 2300-5963 ACSIS, Vol. 18

Counting Instances of Objects in Color Images Using U-Net Network on Example of Honey Bees

Weronika W. Westwańska
Zespół Szkół Stowarzyszenia Rodzin Katolickich Archidiecezji Katowickiej im. Kardynała Prymasa Augusta Hlonda
ul. Kościuszki 11, 41-500 Chorzów, Poland
Email: mkw.weronika.westwanska@gmail.com

Jerzy S. Respondek
Silesian University of Technology, Institute of Informatics, AEI Faculty,
ul. Akademicka 16, 44-100 Gliwice, Poland
Email: jerzy.respondek@polsl.pl

***Abstract*—This article presents a novel approach to segmentation and counting of objects in color digital images. The objects belong to a certain class, which in this case are honey bees. The authors briefly present existing approaches which use Convolutional Neural Networks to solve the problem of image segmentation and object recognition. The focus however is on application of U-Net convolutional neural network in an environment where knowledge about the object of interest is only limited to its rough, single pixel location. The authors provide full access to the details of the code used to implement the algorithms, as well as the data sets used and results obtained. The results show an encouraging low level of counting error at 14.27% for the best experiment.**

I. Introduction

CONVOLUTIONAL Neural Networks (CNNs) are considered state of the art architectures for object detection and segmentation in color images [1]-[4]. In this article we apply a specific type of CNN, called U-Net CNN (UNETCNN) [5], [6], to count instances of honey bees in color images captured by digital video camera. The dataset was sourced from [7] and is freely available to anyone who wishes to test their own bee counting routines. We are proposing a novel way of preparing data modeling for a UNETCNN, where the only information available, about the object of interest (OOI), is its approximate location, defined as a single point in two dimensional planes. Majority of research in the area of CNNs assumes that an OOI location is provided by a rectangle tightly encompassing its border [4]. We propose to use a circle with its center placed on the OOI. The assumption is: we are not focusing on finding exact boundaries of OOI, but rather on counting instances of the OOI, versus manually provided data in a segmentation set. As we are going to show in this paper, it is not necessary to cover an OOI with a bounding box, to achieve high classification accuracy. Instead we are only considering a location of a pixel lying on the surface of an OOI. The location of the pixel constitutes a center of a small circle, which contains pixels belonging mostly to OOI and partially to the background. This step is called UNETCNN data generation. Following that, we employ further steps: training of UNETCNN, using the trained model for automatic segmentation of OOI, automatic counting of OOI instances. The final step allows computation of the relative error by comparing the number of OOI instances that were detected automatically versus how many of them were manually labelled by a human.

The article is organized as follows. In section II we discuss recent works in the area of adopting neural network to solve the problem of speed and accuracy in image recognition process including bee detection. We also present UNETCNN architecture and give reasons for adapting it for our own solution. Section III describes in details each stage of our experiment and presents the results. In section IV we summarize our work and discuss future directions which could lead to interesting findings.

II. Existing Approaches to Bee Recognition Problem and UNet Characteristic

A. Different approaches to OOI detection

The bee detection problem was analyzed in [8] with various scenarios according to diversity of background characteristic, light intensity, bee size, image segmentation and labelling efficiency etc. The experiment evaluation helped us to decide which aspects of image recognition are the most important in our experiment and suggested the way of training set preparation. It also convinced us that Adam optimizer [9] is a good choice for training our neural network.

In [10] bees' recognition problem is discussed in the field of different methods of object recognition. CNNs are compared with Multi-Layer Perceptron (MLP) models. According to the experiments and results presented in that

This work was supported by Statutory Research funds of Institute of Informatics, Silesian University of Technology, Gliwice, Poland (BK/204/RAU2/2019).

paper, it was found that MLP performance is much worse than CNN (taking into account the same dataset). The author stated that ADAM optimizer [9] gave reliable results in comparison to others. In terms of kernel size it was suggested that choosing it as 5x5 pixels, provides better performance of the model. In [10] it was noticed, that one of the problems in proper classifying the bees is a possible presence of bee shadow, which has got the same shape as the bee.

B. UNet

UNETCNN is a NN architecture designed for image segmentation, characterized by low demand on number of annotated training images, and fast data processing. In [5] the authors use medical images as source of training data and demonstrate state of the art results, compared to manual labelling, making this architecture a de facto standard in medical images segmentation.

A typical UNETCNN consists of two paths. One (the contracting path) is represented by a typical CNN with two operations of convolution and max pooling following one after another. This path reduces spatial information, but provides better information where OOI might be present. The second path (expansive) matches the features extracted in contracting path using a sequence of up-scaling transformations.

A game changing innovation in [5] was the fact that a small set of training images can yield more precise segmentations than larger training sets in other algorithms. For a UNETCNN the training data is sourced from the images by dividing them into smaller windows, and then randomly chosen to be included in a training set.

Another feature of UNETCNN is that the algorithm enables finding the solution not only for diverse data set, but also for relatively similar data. Normally lack of diversity would lead to difficulties in recognizing objects in images which do not present features close to the ones the network was trained on. In order to alleviate this phenomenon and make the results independent of the input data, an excessive data augmentation is used. The network gains the data not only straight from the input data, but also from elastic deformations of the training images. This way the network training process can be invariant to deformations even if the images used in the process do not contain enough OOI. Recent works on UNETCNN [6], [11]–[14] prove that this architecture yields very good results and currently might be the best for solving objects recognition problem not only for 2D but also for 3D data.

In our approach we decided not to segment the training data manually, because of the amount of time it would take. We decided to verify if it would be practical to find a way of solving the problem of manual segmentation, by not providing bounding box or a mask for each object in the image. We decided to segment images based only on single points for each OOI located in any of the training images.

III. The Proposed Solution

As we mentioned above, there are 4 stages applied in OOI counting, each described in detail below. For the purpose of our experiments, we decided to use data set downloaded from [7]. At the time of performing experiments, the dataset contained only 550 manually labelled images. We enhanced this data set with a further 1086 images, manually labelled by us, using custom written software. The algorithms examined in this work are publicly available via [15].

A. Stage 1 – UNETCNN input data generation

As it was commented earlier we decided to adopt a shape of a circle to describe part of an area belonging to each OOI. As the input images are of size 640 x 480 pixels we empirically set the size of the circle to be 16 pixels in radius (which is fully configurable). The idea is that when an OOI is labelled, a point is placed on its surface. We assumed that the pixels lying within the nearest neighborhood of the labelled location belong with a high probability to the OOI itself. We decided to simulate such a neighborhood with a circle of a chosen radius, where the pixels lying more towards the circle's edges are less likely to be part of the OOI itself. A linear probability decrease function is implemented with a minimum probability Pmin declared for the edge of the circle and maximum probability Pmax for the center of the circle. We set these values to 0.99 and 1.0 respectively for circles with radius of 16 pixels, and to 0.80 and 1.0 for circles with radius of 20 pixels (within different sets of experiments – see Table I). Any pixels lying outside of the circle are considered to belong fully to the background. This approach means that the problem becomes a binary classification one, where we set class 0 as background, and class 1 as an OOI. Any pixel in the generated modelling data has a probability associated with it: background and foreground. Both probabilities sum up to 1.0.

After all the images from the dataset were labelled, and the parameters of the data generation decided (such as circle radius, Pmin and Pmax), we could finally create a numeric representation of the modelling data, stored as two separate Python numpy files. The first file described the color RGB channels of the pixels from the labelled images, normalized to values between 0.0 and 1.0. The second file described probability values for non OOI (NOOI) and OOI classes for every pixel, meaning that each pixel has a 2 dimensional feature vector associated with it.

The idea behind such generalization for the shape of an OOI as a circle, was that we were dealing with bees, which are relatively similar in size. We also were using the

fact that NN is working with fuzzy data, where we can assume that a pixel can partially belong to the NOOI and partially to the OOI. Based on that assumption and the fact that training algorithms for NN find local optimal solution we decided to test if this approach worked.

B. Stage 2 – Training UNETCNN model

In previous stage 1 we created 2 sets: one representing samples X, and the second set Y representing corresponding values mapped to probabilities of all the pixels from modelling set. These sets stored on a hard drive are quite large in size, depending on amount of images in the modeling set. The X and Y sets, stored as Python numpy files, were used as a source for training of the UNETCNN model. The model we used was a slight modification from the original, with last layers changed to use 2 instances of RELU layer, followed by a Dropout layer and SoftMax used for classification. This solution was introduced to minimalize the risk of overfitting the network and gave significant improvement according to results of computations. SoftMax layer enabled classification of background and foreground classes and would allow us to use it for classification of more than only two classes in the future.

In order to provide the training data for the UNETCNN, the generated data had to be randomly accessed to retrieve rectangular windows of pixels which contained modelling samples for the OOI and the NOOI classes. We adopted an algorithm, where each modeling image has a total of 80 (configurable value) windows randomly retrieved. Among the 80 windows, a specific amount of windows are considered as OOI windows and the rest are considered as NOOI windows. In order for a window to be OOI based, it has to pass a minimum percentage threshold for OOI pixels count. The amount of OOI windows would vary per image, depending on how many manually labelled OOI samples were present, versus total OOI samples in the modelling set. For example if image IMG1 had 2 OOIs labelled, and image IMG2 had 4 OOIs labelled, then there would have to be 2 times less OOI based windows randomly selected from IMG1 than compared with a number of randomly selected OOI based windows from IMG2. This individual approach was dictated to preserve a balanced number of OOI and NOOI classes in training data. The details of how the modelling set is created from the data generated in Stage 1 are available in [15].

The modelling data collected so far was then further split into training and validation sets (80/20 ratio) to be used in UNETCNN. On average we achieved a decent 96% validation set accuracy.

C. Stage 3 – OOI Segmentation

After the UNETCNN model was trained we could use it to perform OOI detection on images from segmentation set. As it was mentioned in Stage 2, the model operates on assumption that the input tensor used in classification is of certain dimensions. In our experiments the dimensions were Nx32x32x3, where N is the number of windows collected from the segmented image, 32x32 are width and height of the window, and 3 stands for RGB channels normalized to [0,1] interval. The value of 32 is also configurable within the source code, and corresponds to diameter of the OOI modelling circle from Stage 1 [15].

We propose a custom approach for segmenting an input image, where a set of windows meshes is created. A single mesh is started at a specific (x, y) offset from the top left corner of a segmented image. The idea is to cover as much of the image as possible with windows tightly attached to each other, where every window needs to be wholly fitting into the image. The windows and their pixel RGB values would create a set which is then classified by the trained UNETCNN model from stage 2. The results of each mesh's classification were then added into a special matrix with values taken for OOI class at corresponding locations to the mesh pixels. Another matrix is also kept to count how many classifications were computed for each pixel from the segmented image. In the end, when all meshes are classified, the accrued classification results are scaled (using the accumulated classification counts), so that a heat map is created. The meshes created for the segmentation process are generated at a specified step of 2 pixels (configurable value) from coordinates of (0, 0) to (32, 32), where 32 is a size of classification window for UNETCNN.

The fore mentioned heat map is later converted to a binary image, where each pixel is decided to be as a part of the OOI or part of the background, based on the scaled voting produced by meshes.

D. Experiments - Counting OOI

In our experiments we decided to examine how different parameters affect counting error which is expressed as a difference between 100% and a percentage of automatically detected instances of OOI (bees) versus total amount of manually labelled instance of OOI. In Table I we present OOI relative counting error, dependent on modelling size set and window size used for training of U-Net and further segmentation. The error progression is visualized in Fig. 1. Minimum percentage of pixels per window, so that can be considered as an OOI based window, was set as 45 and 50 for Experiment 1 and Experiment 2 respectively. More details can be found in logs provided in [15]. It turns out that the more images are available, the better results. Interestingly, error started dropping dramatically at about 500 images, reaching its

minimum value for full modelling set size of 1096 images with the OOI window size set at 40 pixels (Experiment 1).

TABLE I.
BEES COUNTING ERROR DEPENDING ON AMOUNT OF IMAGES USED IN MODELING SET ALONG WITH WINDOW SIZE

Attempt number	[Modeling images count, window size]	OOI Relative Counting Error
1	[100,40]	75.08%
2	[200, 40]	73.00%
3	[500, 40]	24.23%
4	[1000, 40]	16.83%
5	[1096, 40]	14.27%
6	[100, 32]	75.24%
7	[200, 32]	74.55%
8	[500, 32]	26.35%
9	[1000, 32]	16.87%
10	[1096, 32]	18.17%

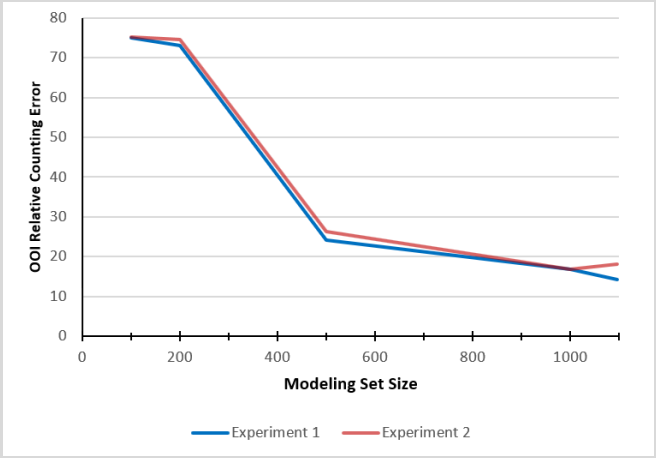

Fig. 1 OOI Counting Relative Error for experiments on parameters set 1and set 2.

IV. CONCLUSIONS

In this article we presented a review of the most recent methods of OOI segmentation and detection. Based on these we chose a UNETCNN architecture which we adapted to a fairly new topic of counting of OOI, based only on their singular locations in the training set. We developed a new approach for generating modeling data, using the trained UNETCNN for segmentation, and further for counting occurrences of OOI in the validation set. We reached a satisfactory level of error at 14.27%, which encourages us to pursue this topic further. The experiments performed show that the training data preparation does not have to be mundane and time consuming and just a few hours spent on the process of labeling images can yield good results, not only in terms of counting error reduction but also decent outcome in OOI segmentation. The Python code which was developed for the purpose of this research is available freely to anyone from [15]. We would like to thank Mr. Jonathan Byrne for making his data set available in [7].

REFERENCES

[1] I. Goodfellow, Y. Bengio, A.Courville, Deep Learning, Cambridge CA: Massachusetts, pp. 321–359, 2016. https://www.deeplearningbook.org/
[2] E.R Davies, Computer Vision. Principles, Algorithms, Applications, Learning, 5th ed., London, pp. 456–462, 2018.
[3] R. Yamashita, M. Nishio, R. Kinh Gian Do, K. Togashi, "Convolutional neural networks: an overview and application in radiology", Insights into Imaging, vol. 9, pp. 611–629, 2018. https://doi.org/10.1007/s13244-018-0639-9
[4] Z. Zhao, P. Zheng, S. Xu, X. Wu, "Object detection with deep learning: A Review", Journal of Latex Class Files, vol. 14, no. 8, 2017 https://doi.org/10.1109/TNNLS.2018.2876865
[5] O. Ronneberger, P. Fischer, T. Brox, "U-Net: convolutional networks for biomedical image segmentation", International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241, 2015. https://doi.org/10.1007/978-3-319-24574-4_28
[6] K. H. Jin, M. T. McCann, E. Froustey and M. Unser, "Deep convolutional neural network for inverse problems in imaging", IEEE Transactions on Image Processing, vol. 26, no. 9, pp. 4509-4522, 2017. https://doi.org/10.1109/TIP.2017.2713099
[7] https://www.kaggle.com/jonathanbyrne/to-bee-or-not-to-bee, accessed on the 1st of February 2019.
[8] M. Kelcey, "Counting bees on a rasp pi with a conv net", 2018. http://matpalm.com/blog/counting_bees
[9] P. Kingma, J. Lei Ba, "ADAM: a method for stochastic optimization", arXiv preprint arXiv:1412.6980, 2014. https://arxiv.org/abs/1412.6980
[10] A. Tiwari, "A deep learning approach to recognizing bees in video analysis of bee traffic", Utah State University All Graduate Theses and Dissertations, 7076, 2018. https://digitalcommons.usu.edu/etd/7076/
[11] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, O. Ronneberger, "3D U-Net: Learning dense volumetric segmentation from sparse annotation", Medical Image Computing and Computer-Assisted Intervention, vol. 9901, pp. 424–432, 2016. https://doi.org/10.1007/978-3-319-46723-8_49
[12] J. Chen, L. Yang, Y. Zhang, M. Alber, D.Z. Chen, "Combining Fully Convolutional and Recurrent Neural Networks for 3D Biomedical Image Segmentation", NIPS'16 Proceedings of the 30th International Conference on Neural Information Processing Systems, pp. 3044–3052, 2016. https://arxiv.org/abs/1609.01006
[13] J.P. Vigueras-Guillén,B. Sari, S.F. Goes, H.G. Lemij, J. van Rooij, K.A. Vermeer, L.J. van Vliet, "Fully convolutional architecture vs sliding-window CNN for corneal endothelium cell segmentation", BMC Biomedical Engineering, vol. 1, 2019. https://doi.org/10.1186/s42490-019-0003-2
[14] S. Baek, Y. He, B.G. Allen, J.M. Buatti, B.J. Smith, K. A. Plichta, et al. "What does AI see? Deep segmentation networks discover biomarkers for lung cancer survival", 2019. https://arxiv.org/abs/1903.11593
[15] https://github.com/WeronikaWestwanska/ToBeOrNotToBee, accessed on the 8th of May 2019.