Segmentation through patch classification: a Neural Network approach to detect Posidonia oceanica in underwater images

Antoni Burguera[a]

[a] *antoni.burguera@uib.es*
Departament de Matemàtiques i Informàtica - Universitat de les Illes Balears
Ctra. Valldemossa Km. 7.5, 07122 Palma - Illes Balears - Spain

Abstract

This paper focuses on the detection of Posidonia oceanica in underwater images. The input image is split into a set of patches that are classified as depicting Posidonia or not. Two different Neural Networks are proposed to perform the classification. A region growing algorithm able to accurately detect the contours of the Posidonia oceanica from the output of the classifier is also described.

The experimental results, performed using images gathered in coastal areas of Mallorca, show that our proposal surpasses previous studies based on Machine Learning, being superior in some cases to Deep Learning methods. The advantages in terms of computational requirements, which are crucial in underwater robotics, are also highlighted.

Keywords: Posidonia oceanica, Underwater vision, Underwater robotics, Neural Network

1. Introduction

The *Posidonia oceanica* (PO), a seagrass that is endemic to the Mediterranean, protects the shoreline against erosion by forming large underwater meadows where many organisms live. Being a plant, PO also absorbs carbon thus increasing the water quality. As a matter of fact, PO is considered a priority natural habitat by the European Commission's directive 92/43/CEE.

Nowadays, PO monitoring is mainly performed by human divers, who photograph the meadows and use markers to measure their extension [9].

These approaches are inaccurate and slow as well as limited in time by the scuba air tanks. Some researchers proposed the use of multi-spectral satellite imagery [6] or acoustic bathymetry [7] to map PO meadows but these approaches tend to fail when it comes to distinguish PO from other plants or algae. Even though the use of *Autonomous Underwater Vehicles* (AUV) endowed with cameras [10] has proved to solve these problems, the literature on automatic visual PO detection is scarce.

One of the first attempts [3] to automatically detect PO in underwater images used Law's filters to classify the image texture by means of *Logistic Model Trees* (LMT). Other studies [2] trained a *Support Vector Machine* (SVM) using the convolution of the image with a set of Gabor filters or a *Convolutional Neural Network* (CNN) not relying on pre-defined filters [4]. The use of CNN evolved into a *Deep Learning* (DL) approach [5] able to segment PO images with very high accuracy.

In spite of their high detection rates, these studies have some problems. For example, approaches based on classical *Machine Learning* (ML) [3, 2] rely on the extraction of a pre-defined set of image features prior to the detection. This leads to an increase in processing time and reduces the system flexibility. Approaches based on DL [5] do not require fixed feature sets but have large training times and memory requirements, and relatively high prediction times due the the size of the *Neural Network* (NN).

Overall, these studies have shown that the most distinctive features of PO are texture, and to a lesser extent, colour. This suggests that PO can be detected in small image patches as long as they contain enough texture. By taking advantage of this, we can divide the input image in small patches, analyze each of them separatedly and decide if they depict PO or not. That is, our proposal is to turn a complex and time consuming image segmentation into a set of simple and computationally cheap classifications. In order to avoid the need for fixed feature extraction, our proposal is based on NN. However, contrarily to DL, we propose two simple NN that lead to fast detections. A region growing algorithm in charge of refining the patch-based classification is also described.

2. Models

Our proposal to detect PO in underwater images is to decompose the input image into small patches and decide if each of them depicts PO or not.

Let us denote by *PO* the class of the patches depicting PO and *nonPO* the class of the patches not depicting PO.

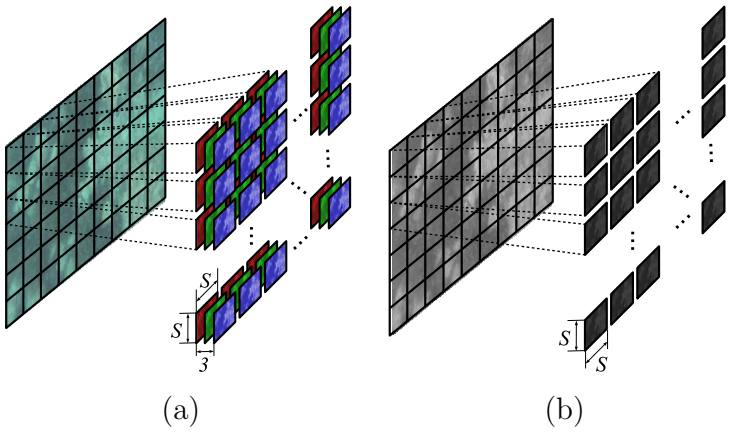

Figure 1: Patch creation. (a) Using color images. (b) Using grayscale images.

The first step is, thus, to split the input image into a set of non overlapping patches of $S \times S$ pixels each. This means that a patch will be an $S \times S \times 3$ matrix if the input image is RGB (Figure 1-a) or an $S \times S$ matrix in case of grayscale input images (Figure 1-b). In both cases, all matrix values are in the interval [0,1].

Each patch matrix constitutes a data sample to be used by our classification system. In this paper we propose two simple NN to perform the classification. These two NN, called Model I and Model II, will be experimentally assessed and are described next.

The Model I has two parts. The first, in charge of feature extraction, is composed of two convolutional layers followed by a pooling layer. The first convolutional layer performs 32 sets of 2×2 convolutions and uses ReLU as activation function. The second convolutional layer performs 64 sets of 2×2 convolutions and also uses ReLU as activation function. Finally, the pooling layer, which is in charge of reducing the dimensionality, is a *MaxPooling* with a pool of size 2×2.

The second part performs the classification. To this end, the output of the pooling layer is flattened and used to feed a dense layer with 128 units with a ReLU activation function. This layer is followed by the output layer, which has one node for each class (*PO* and *nonPO*) and uses *SoftMax* as activation function. Figure 2 illustrates the whole NN.

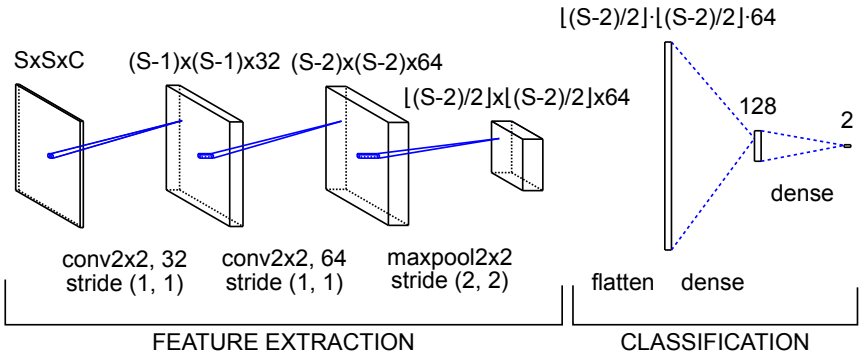

Figure 2: The Neural Network (Model I)

The Model II is, basically, the result of removing the convolutional part from Model I. Thus, it has the simplest possible structure for a NN: one input layer and one output layer without hidden layers. To make it even more simple, the number of nodes in the input layer has been reduced to 64. The remaining hyperparameters are the same used in the non convolutional part of Model I. Also, given the simplicity of Model II, the patches have to be flattened before entering the input layer. The goal of Model II is to test if PO detection can be achieved from the patch itself without the need for feature extraction.

In both models, a cross-entropy loss function and an Adam optimizer are used to train the network. A 20% of the training data is used as the validation set, which allows to fine-tune the number of epochs.

3. Pixel refinement

Since the described NN classifies each patch as *PO* or *nonPO*, the result of processing a whole image can be seen as a low resolution segmentation. This kind of segmentation can be sufficient for a wide range of applications such as computing the PO coverage in medium to large environments or building large scale mosaics depicting the presence of PO. However, if more resolution is required, the following pixel refinement algorithm [2] can be used.

The input of this algorithm is the set L of labels, one per patch, provided by the NN and the image I to which the labels belong. The output is a mask M, which is an image of the same resolution that I whose pixels are white if the corresponding pixel in I depicts PO or black otherwise.

The algorithm first builds an initial guess of M by setting to white or black its pixels depending on the corresponding label in L being PO or $nonPO$. Afterwards, the countours in M are iteratively shrunk or grown depending on the corresponding color in I being closer to the average color of PO regions or to the average color of non PO regions.

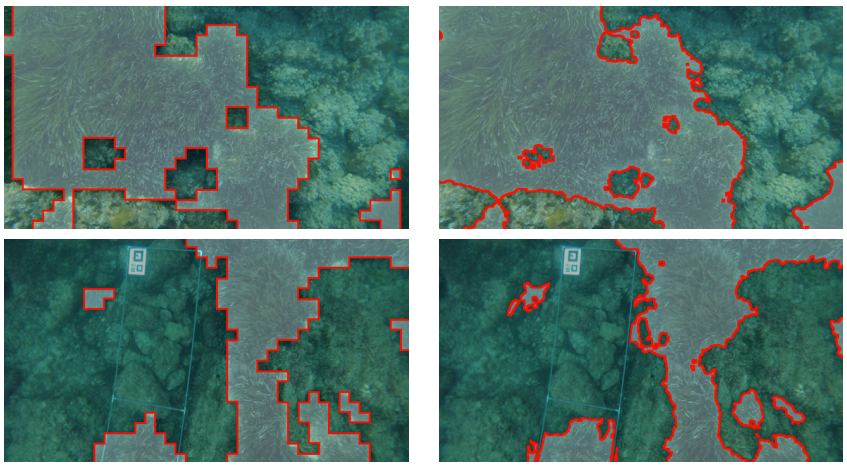

Figure 3: Example of pixel refinement. The contours of the mask have been enhanced to ease visualization. Left column: patch-level classification. Right column: refined classification.

Figure 3 shows some examples of the pixel refinement. A C++ implementation of this algorithm able to do the pixel refinement in video rate is available at `https://github.com/aburguera/POD_STANDALONE`.

4. Experimental results

Our proposal has been tested with three different datasets gathered in several coastal areas of Mallorca with an AUV endowed with bottom looking cameras. Dataset A is composed of 159 images gathered in clear water with healthy PO. Dataset B contains 171 images obtained in turbid water and involving mostly dying and dead PO. In Dataset C there are 69 images depicting a wide range of illumination conditions and PO states.

The ground truth was manually created for each image in the form of one mask per image. Figure 4-b shows the manually created mask corresponding to the image in Figure 4-a. Since the proposed NNs classify image patches, the ground truth has to be adapted to assign a single label PO or $nonPO$

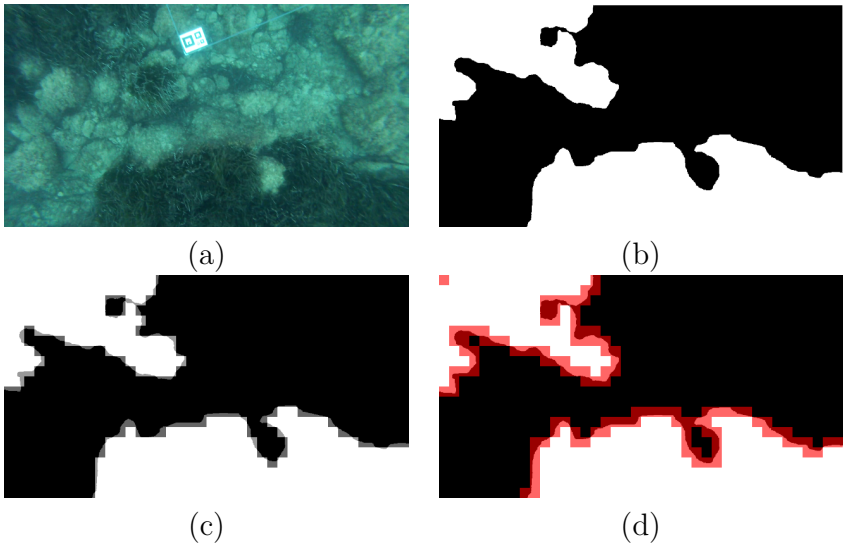

Figure 4: Ground truth construction. (a) Data image. (b) Manually constructed mask. (c) Majority ground truth overlaid to the mask. (d) Binary ground truth overlaid to the mask. The discarded patches are shown in red.

to each patch. To this end, two different ground truths have been created. First, the *majority* ground truth, illustrated in Figure 4-c, assigns a label to a patch depending on the class of the majority of the pixels in the patch. The second ground truth is called *binary* and assigns a label only if all the pixels in the patch belong to the corresponding class. Otherwise, the patch is discarded. Figure 4-d exemplifies this ground truth.

The parameters that will be experimentally tested are the following. First, the image resolution (IR). Prior to training and testing, the input image will be scaled to a different resolution. We scale the image width to a specific value and compute the height that keeps the original aspect ratio. The tested image widths are 160, 320 and 640 pixels. In all cases, the patch size is chosen to produce 40 patches per row. The second parameter, CLR, decides whether to use color information or not. If color is used, each patch has R, G and B channels. If it is not used, the patch has a single channel. The third parameter is the labelling type (LT), which is explained next.

Labelling type 1 means that the majority ground truth is used both to train and to test the system. Labelling type 2 means that the binary ground truth is used during training and testing. Accordingly, type 1 uses all the patches whilst type 2 discards the patches whose pixels do not all belong to

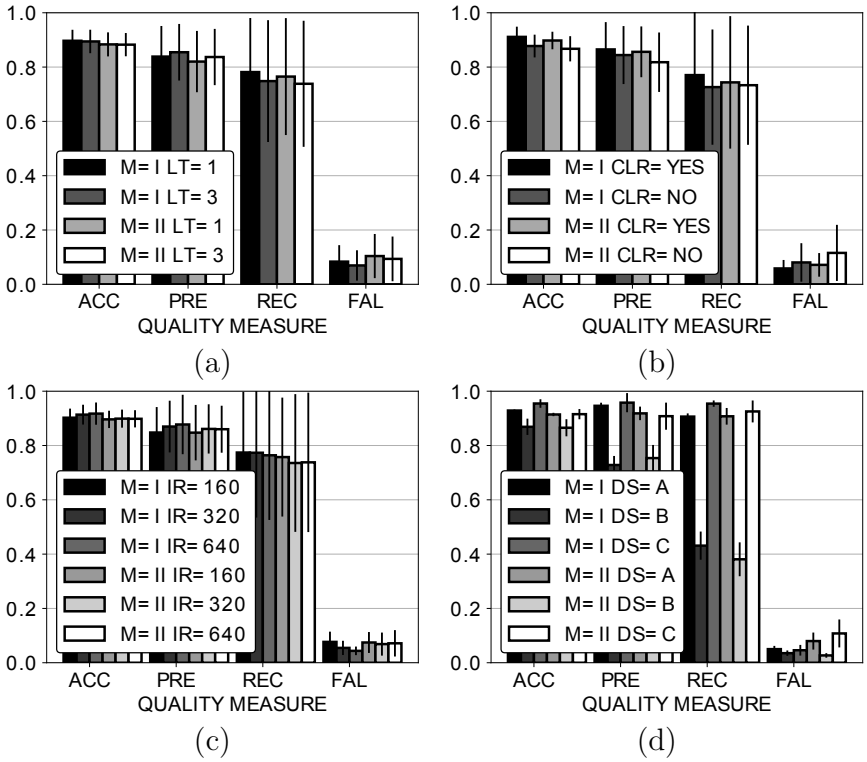

Figure 5: Evaluation of different parameters. (a) Label type (LT). (b) Use of color (CLR). (c) Image resolution (IR). (d) Dataset (DS).

the same class. To provide a more practical application of the binary ground truth, labelling type 3 is defined. In this case, training is performed using the binary ground truth. However, to test the system all the patches are used and validated by comparing the NN output to the majority ground truth. Roughly speaking, types 1 and 3 are those that correspond to real usage of the NN to detect PO whilst type 2 is provided for comparison purposes.

All the possible combinations of these parameters have been tested for each dataset and model and evaluated in terms of mean and standard deviation of accuracy, precision, recall and fallout using a K-Fold ($K = 5$) cross validation schema. The mean and the standard deviation of the training and the prediction times per image have also been measured. The whole results, involving 108 parameter combinations, can be downloaded at `https://www.doi.org/10.13140/RG.2.2.30886.98888`

Table 1 summarizes the results by showing the means for all possible

combinations of parameters over all the datasets. As it can be observed, both models exhibit very high accuracies, precisions and recalls and Model I seems to provide slightly better results. Except for the time consumption, the best results always appear when using labelling type 2. Since this labelling discards some patches, it cannot be used to process full images. Thus, let us focus in labelling types 1 and 3.

Figure 5-a shows the results for each model and labelling type LT. As it can be observed, LT=3 leads to better precision and fallout than LT=1, to almost the same accuracy and to worse recall. Since the standard deviation of the recall is quite large, there is more uncertainty in this parameter and, thus, the best overall LT is LT=3. That is, the best option is to train the system only with patches that are only contain PO or do not contain PO at all.

Let us now analyze the effects of the color using only LT=3. The results, summarized in Figure 5-b, show that the use of color patches leads to better results in accuracy, precision and recall that the use of grayscale patches. Thus, the best option is to use color images.

As for the effects of the image resolution (IR), the results involving only LT=3 and color images are summarized in Figure 5-c. In this case, Model I accuracy, precision and fallout improve with the image resolution, whilst Model II results are not conclusive. Since, up to this point, Model I leads to better results, let us focus on Model I performance and select IR=640.

Now that we have chosen the best possible parameters, let us use these parameters and disaggregate the results in the three datasets as shown in Figure 5-d. As it can be observed, Datasets A and C lead to similar results, but Dataset B leads to particularly bad performance in terms of precision and recall. This is mainly due to the kind of PO displayed in this dataset. In this case, images contain both living and dead PO, but in the ground truth we only labelled as *PO* the living one. Since it is really difficult even for a human to distinguish between both types, it is reasonable that both Model I and Model II fail with this dataset. Nevertheless, the accuracy is still really high with both models (86.2% and 86.5% respectively), so even with Dataset B our proposal works reasonably well.

In [5] a deep learning approach to PO detection was presented, and the results were compared to different algorithms based on classical machine learning (ML-SVM), simple CNN and DL. One of the datasets used in that paper is called the *extra test set* and is similar to our Dataset C. Accordingly, let us focus on Dataset C so we can compare our results with those provided

in the mentioned paper.

According to Table 2, which summarizes the results, Model I surpasses Model II for all the quality measures. Nevertheless, it is remarkable that Model II, which is a simple NN without convolutional layers, is able to surpass the 90% in accuracy, precision and recall. It can also be observed that Model I surpasses all the other tested approaches in accuracy, precision, recall and fallout, except for VGG16-FCN8. This deep neural network is only surpassed by ours in terms of recall. However, it is also remarkable for a simple CNN to display results similar to those of a deep neural network specifically trained for PO detection.

Let us now focus on the computational requirements. We have measured the time consumption of our Python implementation using Keras and Tensorflow with an Intel Core i7 machine at 3.1GHz without GPU support. Model I is able to classify PO at 1.5 FPS and Model II reaches the 3.6 FPS. According to [5], the VGG16-FCN8 is able to process the images at 0.42 FPS using a hardware similar to ours. Thus, both Model I and Model II surpass the deep neural network in terms of computational speed.

There is another aspect to be taken into account: the memory usage. Deep neural network usually require large amounts of memory to store all the weight matrices resulting of the training. Whereas networks such as U-Net, SegNet or VGG16-FCN8 use hundreds of MB, all the weight matrices that define Model I fit in 1.6MB and Model II requires less than 1 MB. Even though the storage may not be a problem when using a desktop computer, it is when it comes to underwater vehicles with limited payload and power supply.

Overall, our proposal does not reach the accuracies, precisions, fallouts and recalls of properly trained deep neural network though it provides similar results. However, it clearly surpasses the deep learning approach when it comes to training and prediction times as well as in memory requirements. Thus, our proposal provides a fair trade-off between quality and computational cost. In particular, Model I reaches exceptionally good results at a reasonable frame rate.

5. Conclusion

This paper focuses on the detection of PO in images gathered by an underwater robot using a bottom looking camera. To achieve this goal, the input image is divided into a set of patches that are classified as depicting

PO or not depicting it. Two different NN have been proposed to perform the classification: Model I, which is a CNN, and Model II which is a simplified version of Model I without the convolutional layers. A region growing algorithm able to accurately detect the contours of PO from the output of the NN is also discussed.

The experimental results performed using real data gathered by an AUV in coastal areas of Mallorca populated with PO show that Model I and Model II reach very high detection rates, surpassing previous studies based on classical ML and simple CNN. Moreover, our proposal provides results similar, even superior in some cases, to DL methods and surpasses them in terms of detection speed.

The two main conclusions are the following. First, simple NNs with some pre-processing lead to results comparable to those of DL when it comes to texture classification. Second, the success of Model II shows that a featureless approach directly operating on the image pixels without any intermediate convolutional layer may be sufficient for most applications.

Both the quality of the results and the reduced requirements in terms of computational power or memory usage make our approach particularly suitable to be used in an AUV, since these robots are usually limited in terms of computational capabilities.

Funding

This work is partially supported by Ministry of Economy and Competitiveness under contract DPI2017-86372-C3-3-R (AEI,FEDER,UE).

Declarations of interest: None

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[2] Francisco Bonin-Font, Antoni Burguera, and Jose Luis Lisani. Visual Discrimination and Large Area Mapping of Posidonia Oceanica Using a Lightweight AUV. *IEEE Access*, 5:24479–24494, 2017.

[3] Francisco Bonin-Font, Miquel Massot, and Gabriel Oliver. Towards Visual Detection, Mapping and Quantification of Posidonia Oceanica using a Lightweight AUV. *IFAC-PapersOnLine*, 49(23):500–505, 2016.

[4] Yolanda Gonzalez-Cid, Antoni Burguera, Francisco Bonin-Font, and Alejandro Matamoros. Machine learning and deep learning strategies to identify Posidonia meadows in underwater images. In *OCEANS 2017 - Aberdeen*, volume 2017-Octob, pages 1–5, 2017.

[5] Miguel Martin-Abadal, Eric Guerrero-Font, Francisco Bonin-Font, and Yolanda Gonzalez-Cid. Deep Semantic Segmentation in an AUV for Online Posidonia Oceanica Meadows Identification. *IEEE Access*, 6:60956–60967, 2018.

[6] R. Matarrese, M. Acquaro, A. Morea, K. Tijani, and M. T. Chiaradia. Applications of remote sensing techniques for mapping posidonia oceanica meadows. In *International Geoscience and Remote Sensing Symposium (IGARSS)*, volume 4, 2008.

[7] Monica Montefalcone, Alessio Rovere, Valeriano Parravicini, Giancarlo Albertelli, Carla Morri, and Carlo Nike Bianchi. Evaluating change in seagrass meadows: A time-framed comparison of Side Scan Sonar maps. *Aquatic Botany*, 104(4):204–212, 2013.

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science*, volume 9351, pages 234–241, 2015.

[9] D. Scaradozzi, G. Conte, G. P. De Capua, L. Sorbi, C. Luciani, P. G. De Cecco, and A. Sorci. Innovative technology for studying growth areas of Posidonia oceanica. In *2009 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems, EESMS 2009*, pages 71–75, 2009.

[10] A. Vasilijevic, N. Miskovic, Z. Vukic, and F. Mandic. Monitoring of seagrass by lightweight AUV: A Posidonia oceanica case study surrounding Murter island of Croatia. In *2014 22nd Mediterranean Conference on Control and Automation, MED 2014*, pages 758–763, 2014.

Table 1: Results of the K-Fold cross-validation of all the datasets for all possible parameter values. For each quality measure, the best value is depicted in bold and the worst is shown in italics. The times express the time spend to process one full image. The tags I and II denote Model I and Model II.

PARAMETERS			**ACCURACY**		**PRECISION**		**RECALL**		**FALLOUT**		**TRAIN TIME**		**PREDICT TIME**	
IR	**CLR**	**LT**	**I**	**II**	**I**	**II**	**I**	**II**	**I**	**II**	**I**	**II**	**I**	**II**
160	YES	1	0.90	0.90	0.85	0.83	0.78	0.79	0.07	0.08	0.61s	0.55s	0.19s	0.23s
		2	0.94	0.93	**0.93**	0.85	**0.95**	**0.82**	0.07	0.06	0.55s	0.52s	0.19s	0.23s
		3	0.90	0.90	0.85	0.85	0.77	0.76	0.08	0.07	0.62s	0.54s	0.23s	0.25s
	NO	1	0.87	0.86	*0.80*	*0.80*	0.78	0.76	*0.13*	*0.14*	0.50s	0.47s	**0.13s**	**0.18s**
		2	0.89	0.89	0.83	0.81	0.79	0.80	0.10	0.12	**0.47s**	**0.45s**	0.14s	0.19s
		3	*0.87*	*0.86*	0.83	0.81	0.72	0.75	0.10	0.13	0.51s	0.47s	0.16s	0.21s
320	YES	1	0.92	0.90	0.85	0.83	0.81	0.78	0.06	0.09	1.29s	0.74s	0.38s	0.33s
		2	0.95	0.93	0.88	0.86	0.83	0.80	0.04	0.05	1.23s	0.70s	0.40s	0.34s
		3	0.91	0.90	0.87	0.86	0.77	0.74	0.05	0.07	1.25s	0.72s	0.44s	0.36s
	NO	1	0.88	0.87	0.83	0.80	0.76	0.76	0.10	0.12	1.13s	0.62s	0.31s	0.27s
		2	0.91	0.90	0.84	0.83	0.79	0.79	0.07	0.09	1.12s	0.59s	0.33s	0.28s
		3	0.88	0.87	0.85	0.82	*0.72*	0.73	0.08	0.11	1.09s	0.61s	0.34s	0.30s
640	YES	1	0.92	0.90	0.86	0.85	0.79	0.75	0.05	0.07	2.81s	0.58s	0.31s	0.20s
		2	**0.95**	**0.93**	0.89	**0.88**	0.81	0.78	**0.03**	**0.05**	2.49s	0.55s	0.30s	0.21s
		3	0.92	0.90	0.88	0.86	0.76	0.74	0.04	0.07	2.46s	0.57s	0.33s	0.23s
	NO	1	0.89	0.87	0.84	0.81	0.77	0.75	0.08	0.12	*3.06s*	*0.84s*	*0.47s*	0.39s
		2	0.93	0.91	0.87	0.83	0.79	0.79	0.05	0.10	2.37s	0.81s	0.23s	*0.43s*
		3	0.89	0.87	0.85	0.83	0.73	*0.72*	0.06	0.10	2.29s	0.83s	0.26s	0.43s

METHOD	ACCURACY	PRECISION	RECALL	FALLOUT
ML-SVM [2]	89.1%	87.1%	94.9%	18.0%
CNN [4]	62.2%	81.0%	31.9%	7.5%
U-Net [8]	93.1%	93.9%	92.1%	6.0%
SegNet [1]	90.9%	90.4%	91.5%	9.7%
VGG16-FCN8 [5]	96.1%	97.2%	95.0%	2.8%
Model I	**95.5%**	**95.8%**	**95.4%**	**4.6%**
Model II	**91.5%**	**90.8%**	**92.5%**	**10.8%**

Table 2: Comparison with other existing approaches to detect PO. Results for the methods other than Model I and Model II were obtained in [5].