Danmarks Tekniske Universitet

DTU

Special Course: Spectral Analysis of Microbiological Images

Final Report

Author

Youyang Shen - s202535

January 21, 2022

Contents

1 Introduction

Bacterial plating and counting procedures are a must in the microbiology laboratory workflow: colony counting and rapid plating methods are used by more and more microbiologists in their daily routines.

Microbiologists need to solve the tedious problem of counting colonies, which are sometimes too small and can also be masked by the specific color of the culture medium or the enumeration itself is slow and long. These daily routines require a better and more efficient way to optimize the processes employed. Now colony counting is mainly conducted manually at Als Denmark A/S (Als). The work is labour consuming, so it is valuable to develop an automated colony counting method based on image analysis technique.

The goal of the project is to build a deep learning model and assess its performance of colony counting on a spectral image dataset produced by VideometerLite. In this paper, we build two spectral image datasets and propose a U-net model to do the segmentation. Two different loss functions are implemented during the training. 8-connected components analysis is used to count the colonies on the predicted images. Finally, we use precision, recall and F_1 score to evaluate the performance of the created models. Experimental results show that U-net model is suitable for colony counting tasks with highest F_1 score 88.50%. [1]

2 Related Work

Colony counting A deep learning pipeline is implemented for colony segmentation and classification[1]. The segmentation task is done by U-net and the classification one by Resnet-34 network. Another network designed for object detection can also be an alternative solution to colony counting[2]. Each of the obtained segments can be assigned to a class, depending on the number of colonies it contains, from 1 to 6, or labeled as an outlier (the seventh class) if, instead of colonies, contains bubbles, dust or dirt on the agar. In this way, the colony can be counted after the network predict the colony segments. In this work, there is only one type of colony and most of the colonies are separated. A segmentation network U-net is built to extract the colonies from the background and blob analysis is used on the output of the network for colony counting analysis.

nCDA Normalized canonical discriminant analyses (nCDA) are used for discrimination between the colonies and background. The nCDA is a supervised model based on multispectral imaging (MSI) transformation of the images, in order to minimize the distance to observations within classes and to maximize the distance to observations between classes. The nCDA is done by Videometer software.

U-net[3] is a classic convolutional neural network that was developed for biomedical image segmentation. In [4][5], U-net is proved to be robust for segmenting hyperspectral image dataset.

3 Dataset

The dataset contains 66 spectral images, 18 of which are blood agar(BA) protocol and 44 plate count agar(PCA) protocol. For PCA, colonies in the whole petri dish should be counted, while for BA protocol only colonies in the inner dark area of the petri dish are valid counts. Figure 1 and 2 shows an sample image of PCA and BA protocol respectively. Both images are pseudo RBG view mode in Videometer software.

[1] The code is available at https://github.com/YouyangShen/U-net-for-spectral-images.

All images are taken by VideometerLite which is a portable spectral imaging device that can illuminate the item with 7 different bands(Violet 405nm, Blue 460nm, Cyan 525nm, Amber 590, Red 621nm, Red 660nm and NIR 850nm). The images are of size 1520×1520 pixels with 7 channels.

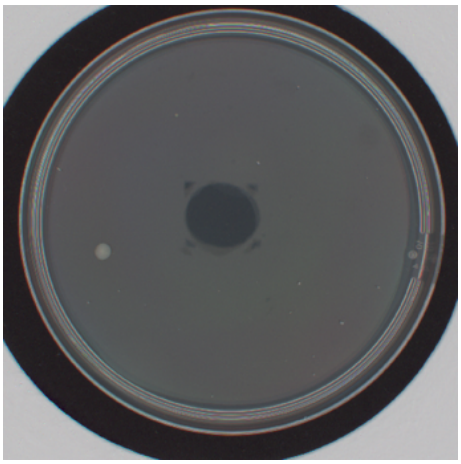

Figure 1: A sample of PCA

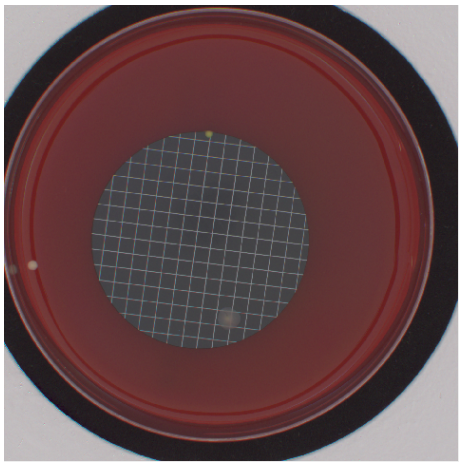

Figure 2: A sample of BA

Preprocessing The aim of preprocessing is to find the region of interest(ROI) in the images. In *PCA* protocol, the ROI is the circular area in the petri dish. In *BA* protocol, the ROI is the inner circular dark area. For each protocol, the ROI is of the same size, so Hough transformation is used to detect the circular ROI. The range of radius of Hough transformation for *PCA* is between 620 and 640 pixels and the range for *BA* is between 340 and 370 pixels. In Figure 3 and 4, the red circles are the result of ROI creation by Hough transformation.

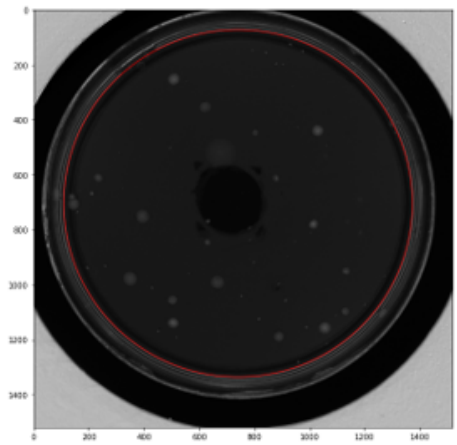

Figure 3: ROI in PCA

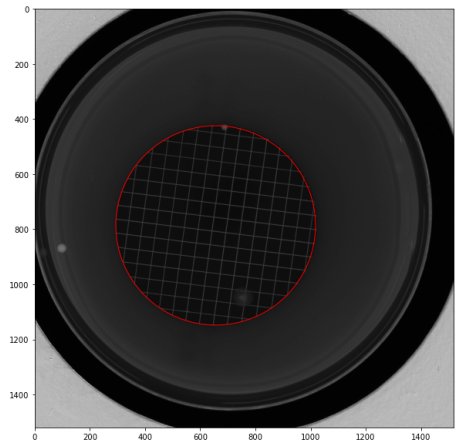

Figure 4: ROI in BA

Ground truth As the dataset is a new one, the ground truth needs to be labelled for training and validation. The image labelling is done in VideometerLab manually after preprocessing.

The ground truth labelling of the images is done in following steps:

1) Input the ROI masked images to VideometerLab;
2) Colony pixels are chosen as layer 1 and background pixels are chosen as layer 2;
3) Run nCDA for the image and output the segmentation masks;
4) Correct the colonies on the masks manually and save the masks as ground truth.

4 Model

Architecture: The network we implement is a U-net which takes a $7 \times 1520 \times 1520$ image as input and segments it into 2 layers, the colony and the background. The size of the network is adjusted to fit the image size. Figure[5] illustrates our network architecture. It first goes down 4 times using a Conv2d layer as we implement it in Pytorch, then goes up 4 times using a ConvTranspose2d layer. The padding is 1 during these transformation. Each time it goes down, the height and width of the image shrinks by a half by MaxPool2d layer. Similarly, each time it goes up, the image size increases 4 times. The number of output channels of each layer is 28, 56, 112, 224 for downsampling and 224, 112, 56, 28 for upsampling respectively. The output is a $2 \times 1520 \times 1520$ image.

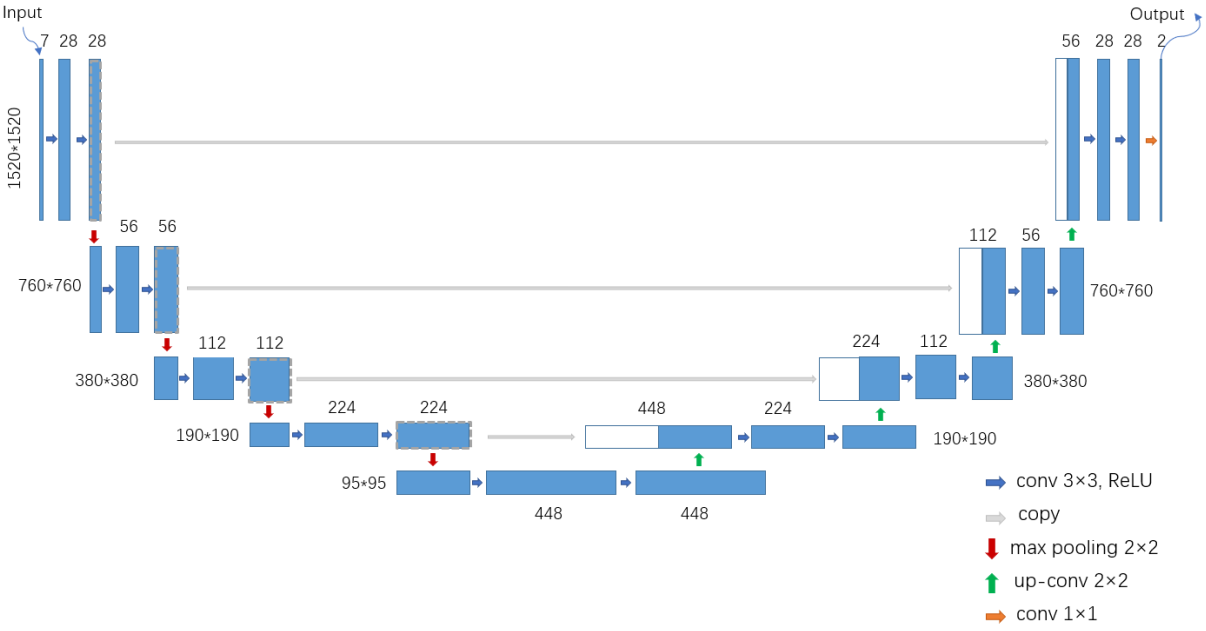

Figure 5: Modified U-net architecture

Loss function: For each model, two different loss functions are implemented, the cross-entropy loss and dice efficient loss.

A. Cross-Entropy[6] is defined as a measure of the difference between two probability distributions for a given random variable or set of events. It is widely used for classification objective, and as segmentation is pixel level classification it works well. It is defined as:

$$L_{BCE}(y, \hat{y}) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})) \tag{1}$$

$\hat{y}$ is the predicted value by the prediction model.

B. Dice Loss[6] The Dice coefficient is widely used metric in computer vision community to calculate the similarity between two images. Later in 2016, it has also been adapted as loss function known as Dice Loss.

$$DL(y, \hat{p}) = 1 - \frac{2y\hat{p} + 1}{y + \hat{p} + 1} \tag{2}$$

Here, 1 is added in numerator and denominator to ensure that the function is not undefined in edge case scenarios such as when $y = \hat{p} = 0$.

5 Experiments

5.1 Training details

Models are created based on two protocol dataset respectively. 90% of the dataset are for training and 10% of the dataset are for validation. Data augmentation is applied to the dataset with the following transformations: horizontal flip, vertical flip, Gaussian blurring and random scaling from 1 to 1.5. The same transformation are applied to the training masks.

The network is trained with 300 epochs. Because the size of the image is too large, the maximum batch size is at most 2. Otherwise there's not enough memory when running on GPU. The learning rate is set at 0.0001. An Adam optimizer is used during the training. Using a HPC node with Tesla V100 16 GB, it takes 5 hours for 300 epochs on average.

Figure[6] shows the result of training and validation loss of one of the models, others are quite similar. In the first 50 epochs, training loss drops dramatically while validation loss oscillates. After 50 epochs, training loss converges as well as validation loss. It can be seen that the training is effective but 300 epochs might be redundant for such a dataset and batch size.

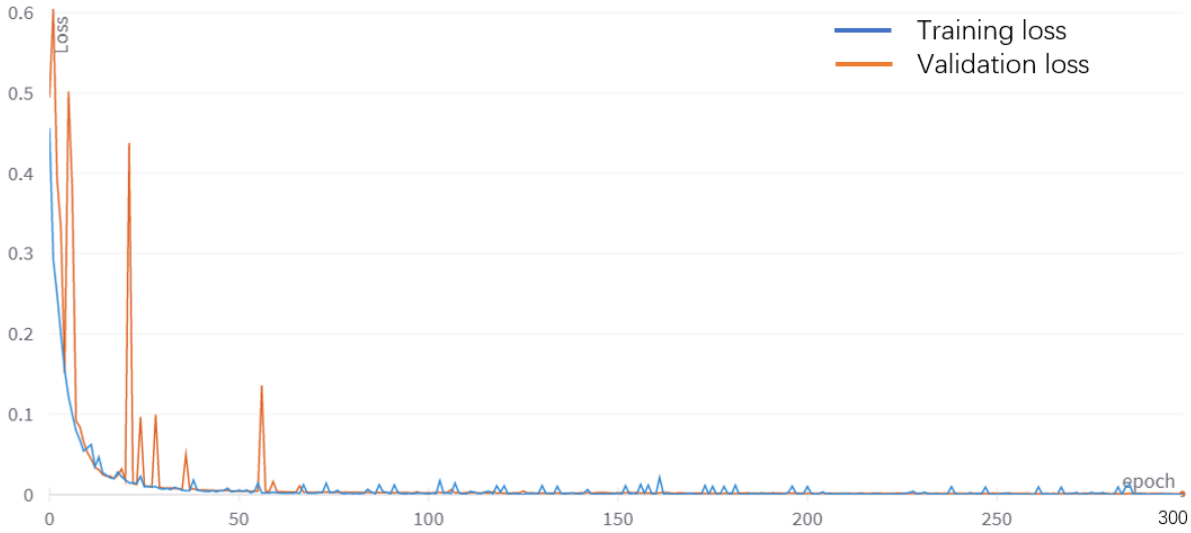

Figure 6: Loss example; Training loss: blue line; Validation loss: orange line;

5.2 Metrics

Several metrics have been introduced to analyze the performance of the model. Among them are IoU, precision, recall and F_1 score.

IoU Intersection over union (IoU)(3) is one of the most widely used metrics for the quantitative analysis of the segmentation results. It is calculated by the intersection of ground truth and prediction devided by their union as shown in Equation 3, where Y is the area in ground truth and $\widehat{Y}$ in prediction.

$$IoU = \frac{Y \cap \widehat{Y}}{Y \cup \widehat{Y}} \tag{3}$$

Because IoU is for analyzing the performance of segmentation, it can not represent the colony counting accuracy. So we use precision, recall and F_1 score instead. To calculate these metrics, we define true positive, false positive and falase negative as below.

First, *ConnectedComponentsWithStats* from Opencv[7] is used to compute the number of separated blobs in both ground truth and predicted images. After running the 8-connected components analysis, the centroids and the number of colonies are computed and stored.

Based on the location of colonies, an algorithm is designed to compare colonies between the ground truth and predicted images. Assume A is a detected colony in predicted images and (x_A, y_A) is the centroid of it. If the pixel of (x_A, y_A) in ground truth is 1, it's regarded as True Positive, otherwise it's False Positive. Assume B is a real colony in ground truth and (x_B, y_B) is the centroid of it, if the pixel of (x_B, y_B) in predicted image is 0, it is False Negative.

Precision Precision is the percentage of true positive among all positive predictions. In our case, it indicates how many colonies identified by the network are indeed a colony.

$$\text{Precision} = \frac{tp}{tp + fp} \tag{4}$$

Recall Recall is the percentage of true positive among all positive ground truth. It means how many real colonies are detected.

$$\text{Recall} = \frac{tp}{tp + fn} \tag{5}$$

F_1 **score** The F_1 score is the harmonic mean of the precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{6}$$

5.3 Result

We report the quantitative results on two datasets in Table 1 and 2 with respect to IoU, precision, recall and F_1 score. The baseline is computed by nCDA method using VideometerLab.

The first block of the results shows U-net with dice loss has highest IoU on PCA dataset and U-net with cross-entropy loss and data augmentation has highest on BA. PCA IoU is higher than BA. Because the size of BA dataset is too small.

In Table 2, U-net with cross entropy and data augmentation has highest F_1 for BA and U-net with dice loss for PCA. Our U-net models perform favorably against the baseline on both datasets, highlighting the effectiveness of our approach.

IoU	BA	PCA
Baseline	32.36%	69.75%
U-net Dice Loss	32.99%	**83.00%**
U-net Dice Loss with augmentation	29.17%	81.66%
U-net Cross-entropy Loss	54.25%	82.42%
U-net Cross-entropy Loss with augmentation	**57.44%**	81.68%

Table 1: IoU

5.4 Discussion

It can be seen augmentation doesn't necessarily improve the model. After some research, we find that we didn't do the augmentation in a proper way, in which the dataset is extended as 4 times the

Metrics	BA			PCA		
	Precision	Recall	F_1	Precision	Recall	F_1
Videometer baseline	**94.40%**	33.10%	49.01%	84.70%	87.40%	86.03%
Unet (Dice loss)	82.20%	52.40%	64.00%	90.60%	86.50%	**88.50%**
Unet (Dice loss) with augmentation	47.20%	57.70%	51.92%	**92.60%**	77.20%	84.20%
Unet (Cross entropy)	79.70%	69.20%	74.08%	86.70%	**89.80%**	88.22%
Unet (Cross entropy) with augmentation	80.40%	**77.50%**	**78.92%**	92.00%	78.30%	84.60%

Table 2: Metrics

size of the original one and the validation dataset also includes the augmented data.

Figure [7] shows an failure example of prediction. The model is not robust to segment some of the colonies in relatively small size (approximately 4 pixels). It could be because the output of the first layer (28) is not enough for extracting the feature of the images.

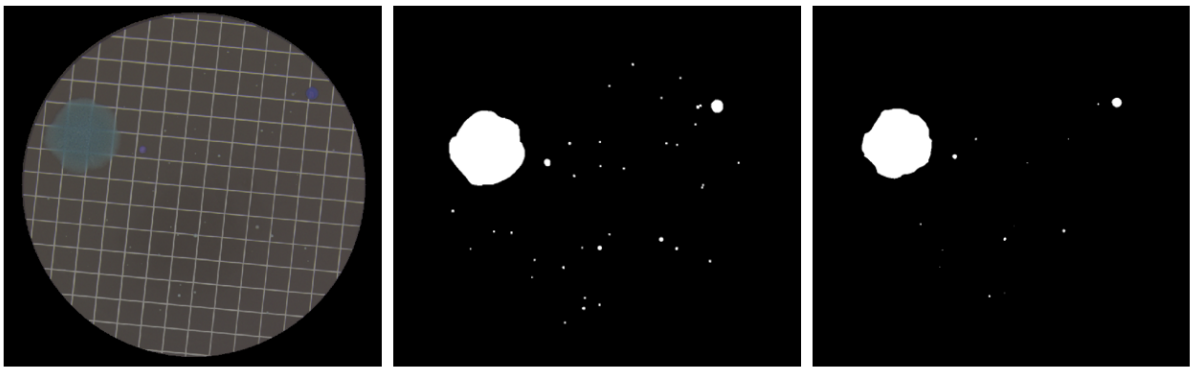

Figure 7: From left to right: Input image, ground truth, prediction

For future work, we will dramatically reduce the percentage of augmented examples (50% of original and 50% of augmented) and implement it only on training data. We may also take more sample images of each protocols and build U-net models with more layers and parameters.

6 Conclusion

We present a novel colony counting method using the deep learning network. By leveraging the state-of-art segmentation model U-net, our architecture first segments the colony from the background, then an algorithm is implemented to count the colonies. Through the experiments, we show that our work compares favorably against existing method on two new datasets.

7 Acknowledgement

This project is supported by Jens Michael Carstensen and Aske Schultz Carstensen from Videometer. The dataset is supported from Als Denmark A/S.

8 Appendix

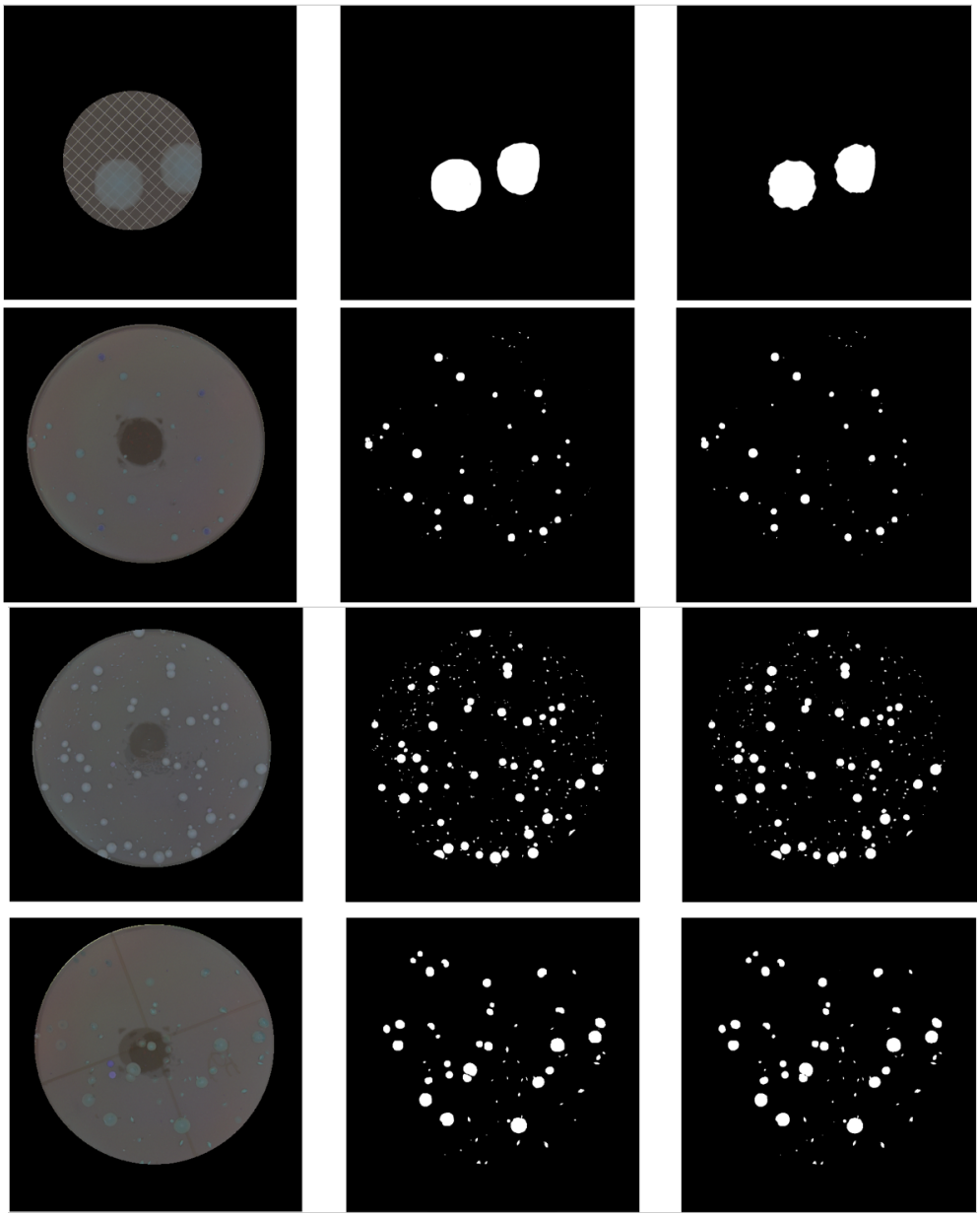

Figure 8: From left to right: Input image, ground truth, prediction

References

[1] Sarah H Carl et al. "A fully automated deep learning pipeline for high-throughput colony segmentation and classification". In: *Biology open* 9.6 (2020), bio052936.

[2] Alessandro Ferrari, Stefano Lombardi, and Alberto Signoroni. "Bacterial colony counting with convolutional neural networks in digital microbiology imaging". In: *Pattern Recognition* 61 (2017), pp. 629–640.

[3] 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*. Springer. 2015, pp. 234–241.

[4] Stojan Trajanovski et al. "Tongue Tumor Detection in Hyperspectral Images Using Deep Learning Semantic Segmentation". In: *IEEE Transactions on Biomedical Engineering* 68 (Apr. 2021), pp. 1330–1340. DOI: 10.1109/TBME.2020.3026683.

[5] reachsumit. *deep-unet-for-satellite-image-segmentation*. URL: https://github.com/reachsumit/deep-unet-for-satellite-image-segmentation (visited on 10/10/2020).

[6] Shruti Jadon. "A survey of loss functions for semantic segmentation". In: *2020 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*. IEEE. 2020, pp. 1–7.

[7] Opencv. *Opencv*. URL: https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html (visited on 10/10/2020).