Instance segmentation of cell nuclei using convolutional neural network

Younes El Mokhtari
Université Libre de Bruxelles

***Abstract*—Segmentation of cell nuclei in microscopy images is a fundamental step in quantitatively analyzing biological and biomedical imaging data. Deep learning techniques, particularly Convolutional Neural Networks (CNNs) like FCN, U-Net, and Mask-RCNN, have emerged as effective tools for this task. In this paper, we report our results on instance segmentation of cell nuclei on the Kaggle Data Science Bowl (DSB) 2018 challenge. Our model is a modified version of U-Net with a ResNet encoder and incorporates watershed post-processing for refining instance segmentation. Our approach demonstrates compelling results on test images, achieving performance comparable to the top 5 solutions in this challenge. The code for our work and more technical details can be found at https://github.com/yelmokht/nuclei-segmentation.**

***Index Terms*—Nuclei; cells; segmentation; convolutional neural networks; neural networks; U-Net; deep learning; semantic segmentation; instance segmentation; Kaggle Data Science Bowl 2018 challenge.**

I. Introduction

Segmenting nuclei is the first step in microscopic image analysis, directly impacting outcomes. This task is highly challenging due to factors like color variation from different staining methods, artifacts, varied size, shape, and texture of cell nuclei, as well as instances of nuclei touching or overlapping, which pose obstacles for segmentation algorithms. Significant research has focused on nuclei detection, particularly for its diagnostic relevance in conditions like cancer. While traditional image processing methods have been employed, they often fall short of achieving optimal performance due to the inherent complexity of the images. However, over the past decade, there has been remarkable progress in deep learning. Techniques leveraging deep neural networks have achieved state-of-the-art performance in automatic medical image segmentation, surpassing traditional approaches and highlighting the potential of deep learning in this domain. [1]

In this paper, we focused on the Kaggle Data Science Bowl (DSB) 2018 competition, where participants were tasked to develop a computer model capable of accurately segmenting nuclei in diverse 2D light microscopy images without manual intervention and generalizing to unseen biological experiments without additional annotation or training. The competition provided a training dataset containing images with corresponding solutions (segmentation masks), and participants were required to generate segmentations for test images using a two-stage evaluation process. Especially, a holdout set consisting of 15 varied image sets from biological experiments not included in the training set was used to assess the model performance under different experimental conditions realistically[2]. Accurately identifying nuclei automatically helps speed up the process of locating and discovering cells under various settings, allowing biologists to focus their attention on solutions. This speeds up research, facilitates medication development, and ultimately improving health and quality of life. [3]

For this challenge, we adopted a bottom-up strategy, employing the U-Net architecture with a ResNet encoder to segment nuclei across a range of various images. Subsequently, we applied watershed post-processing techniques to achieve instance segmentation of cell nuclei. The obtained results showed compelling results in segmenting individual nuclei within the images.

The rest of the paper is organized as follows: Section II provides an overview of the model architecture. In Section III, we describe the experiments and the overall methodology. In Section IV, we present the results obtained followed by a detailed discussion in Section V. Lastly, Section VI concludes the paper.

II. Architecture

For this project, we have modified the standard U-Net architecture, especially focusing on improving the backbone or encoder part. U-Net is widely used in medical image segmentation, especially for segmentating cell nuclei. Despite these changes, the basic structure of U-Net stays the same, with both a contracting and expanding pathway. [2]

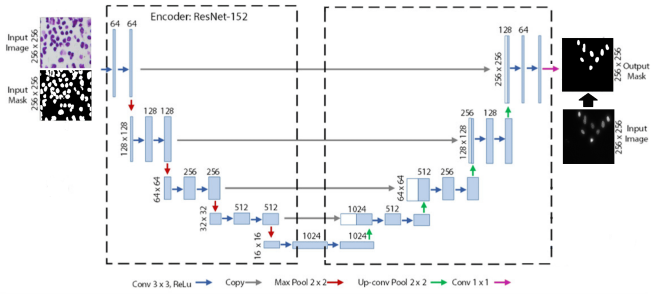


Fig. 1: The model architecture

A. *Encoder Block (ResNet 152)*

In the encoding part of the U-Net model, convolutional blocks are utilized to progressively extract features from the input image. These convolutional blocks consist of two successive convolutional layers, each followed by batch normalization and rectified linear unit (ReLU) activation. The convolutional layers serve to convolve the input tensor with learnable filters, allowing to identify various patterns and features of different sizes within the image. Batch normalization normalizes the activations of each layer, stabilizing training and accelerating convergence. The ReLU activation function introduces non-linearity into the model, enabling it to learn complex representations. After each set of convolutional operations, max-pooling layers are used to make the feature maps smaller. This helps the network focus on important details and increases its overall understanding of the image. Finally, skip connections are made between encoder and decoder blocks that have corresponding features, ensuring that fine details in the image are retained as it is downsampled and then upsampled. [4]

We integrated ResNet-152 as the encoder component in our U-Net model to extract features for segmentation. During downsampling, ResNet models incorporate a residual map after several encoding blocks of 3x3 convolutional layers. In ResNet-152, an extra 1x1 convolutional layer is included before and after the 3x3 convolutional layer, known as a 'bottleneck' building block. This encoder enhances network depth and yields improved segmentation compared to U-Net's original encoder, as observed in our experiments. [5]

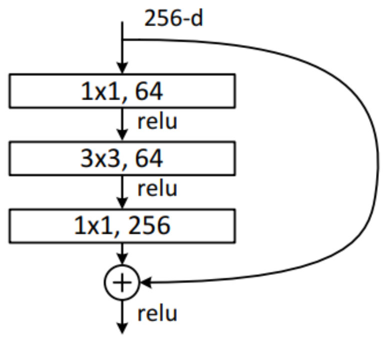


Fig. 2: A building block for ResNet-152

B. Decoder Block (Traditional U-Net)

In the decoding part of the U-Net architecture, the goal is to reconstruct the high-resolution segmentation map from the features obtained in the encoding phase. This process starts by increasing the size of the feature maps using operations like transpose convolutions, which enlarge the spatial dimensions. Then, the feature maps are combined with skip connections from the encoding phase to incorporate detailed context information. These connections help bridge the gap between the encoding and decoding phases, allowing the model to capture both local and global features effectively. Next, convolutional blocks refine the feature maps further to capture intricate details necessary for accurate segmentation. Finally, the output layer, consisting of a 1x1 convolutional layer with sigmoid activation, produces the final segmented output. [4]

The decoder block in our model follows the conventional U-Net architecture, gradually upsampling feature maps from the encoder to generate the output mask. [4]

III. Experiments

In this section, we explore the dataset, preprocessing steps, data augmentation techniques, hyperparameters, test time augmentation, post-processing methods, and evaluation metrics used to validate our proposed model.

A. Dataset

We used the dataset sourced from the DSB 2018 Kaggle competition. This dataset proved to be particularly challenging due to its wide-ranging conditions, including diverse cell types, varying illumination, and different imaging techniques such as bright-field, dark-field, and fluorescence microscopy, leading to differences in image sizes. As depicted in Figures 3a and 3c, there's significant variability in cell appearances. Moreover, uneven illumination across samples complicates nuclei detection and segmentation tasks, making it challenging to discern cell details and boundaries without contrast adjustment, as demonstrated in Figure 3b. The training set contains 670 cell images, each accompanied by its corresponding segmentation saved as image files. The stage 1 test set contains 65 images and stage 2 test contains 3016 images but only 106 images were used for the final evaluation. [3]

B. Training, validation and test set

Since the training set is relatively small, we added the stage 1 set to it, resulting to 715 images for the training set. We used a ratio 0.7:03 to construct the validation set. For the test set, we used the stage 2 test containg 106 images. [3]

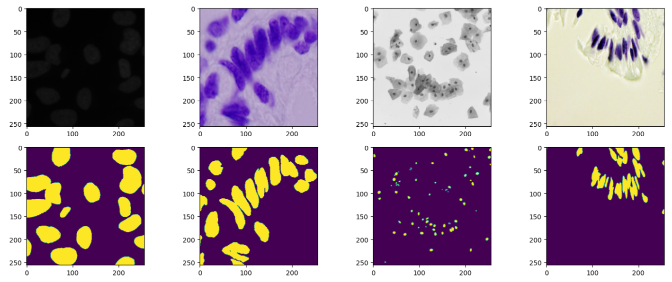


Fig. 3: Some images of training set with their coresponding masks

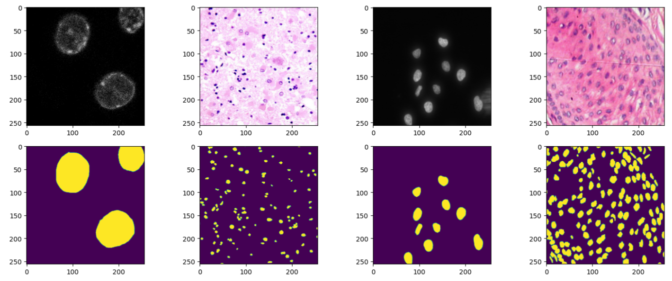


Fig. 4: Some images of test set with their coresponding masks

C. Pre-processing

We analyzed the size of the images for the training and test images and found that 256x256x4 was the most common size. We decided to resize all images to 256x256x3 and all masks to 256x256x1 and normalized them.

D. Data augmentation

Data augmentation is essential to teach the network the desired invariance and robustness properties, when only few training samples are available. In case of microscopical images, we primarily need shift and rotation invariance as well as robustness to deformations and gray value variations. Especially random elastic deformations of the training samples seem to be the key concept to train a segmentation network with very few annotated images. [4]

Since we had only 670 training images, we had to use data augmentation in order to increase the number of images and prevent our model for overfitting. One can also use external dataset with similar images but we didn't use this time [6]. We used Keras ImageDataGenerator to implement data augmentations such as rotation up to 90°, width shift, height shift, shear, zoom, and horizontal/vertical flip.

images

E. Hyperparameters

In our model training, we opted for the Jaccard loss function to handle the loss computation. The Jaccard loss function is formulated as follows:

$$J(x, y) = 1 - \frac{|x \cap y|}{|x \cup y|} \tag{1}$$

where x represents the predicted segmentation mask and y the ground truth segmentation mask. It computes the dissimilarity between the intersection and the union of the two sets. This loss function is particularly useful for tasks like nuclei segmentation, where accurately delimiting the boundaries of objects is crucial. [7]

We employed the Adam optimizer, recognized as one of the most commonly used and efficient optimization algorithms in machine learning [8]. Our model underwent training for 100 epochs with a batch size of 32. We used ModelCheckpoint to save the best model. We employed Keras/TensorFlow as the backend framework for our experiments. The experiments were conducted on Google Colab, utilizing an NVidia T4 GPU with 16 GB of RAM. Our code is available here: https://github.com/yelmokht/nuclei-segmentation.

TABLE I: Hyperparameters

No	Hyperparameters	Settings
1	Optimizer	Adam
2	Loss function	Jaccard
3	Callbacks	ModelCheckpoint
4	Batch size	32
5	Epochs	100

F. Test time augmentation

For the inference part, we employed a technique known as test time augmentation (TTA). TTA involves applying various simple augmentations to the test images, such as rotation, flipping, image rescaling etc. TTA improves the robustness of a model by averaging model predictions on the augmented test images with no additional training expense. Consequently, this augmentation strategy results in improved segmentation accuracy compared to making predictions solely on the original images. [9]

In our case, we used rotations at various angles (90°, 180°, 270°, 360°) along with horizontal and vertical flips. This approach yielded better results during the evaluation of the segmented masks.

image

G. Post processing

For the post processing part, we implemented several techniques to enhance our segmented masks and segment nuclei from it:

1) Otsu thresholding: We employed the Otsu thresholding method to automatically get the optimal threshold for generating binary masks from predicted masks.

2) Filling holes: Following thresholding, we applied a filling algorithm to address any small gaps or holes within the segmented regions. This ensure that the segmented nuclei have the correct shape and the mean size can be computed.

3) Markers using distance map and average nuclei size: To identify potential nuclei centroids, we utilized a combination of distance transform and the average nuclei size. This approach generates markers that serve as initial seeds for the watershed algorithm, allowing us to separate overlapping nuclei in the masks.

4) Watershed with markers: Finally, we applied the watershed algorithm with markers to separate isolated and overlapping nuclei on the masks. Then, we labelize the masks to get individual mask for each nuclei.

image

H. Evalutation metrics

The evaluation metric used for the DSB competition is based on the mean average precision, as defined on the competition website, at different intersection-over-union (IoU) thresholds. A successful nucleus detection was determined by an IoU test (also known as the Jaccard index):

$$IoU(x, y) = \frac{|x \cap y|}{|x \cup y|} = \frac{|x \cap y|}{|x| + |y| - |x \cap y|} \tag{2}$$

which measures the overlap between prediction pixels x and the ground truth pixels y over the intersection of the two areas. Using a threshold ranging from 0.5 to 0.95 with steps of 0.05, true positive (TP) detections, false positive (FP) detections and false negative (FN) detections were identified. For a threshold of 0.5, a predicted object is considered a “hit” if the IoU is greater than 0.5. For each threshold t, a modified version of precision was calculated for all thresholds in (0.5, 0.95). These scores were averaged for all thresholds, and then the mean of the average scores is reported over the images in the test dataset. [10]

$$DSB(t) = \frac{TP(t)}{TP(t) + FP(t) + FN(t)} \tag{3}$$

$$Score = \frac{1}{|thresholds|} \sum_t \frac{\mathrm{TP}(t)}{\mathrm{TP}(t) + \mathrm{FP}(t) + \mathrm{FN}(t)} \tag{4}$$

In addition to the DSB-score, we evaluated our model with three additional metrics based on the IoU detection test: precision, recall and F1-score. [10]

$$precision = \frac{TP}{TP + FP} \tag{5}$$

$$recall = \frac{TP}{TP + FN} \tag{6}$$

$$F1\ score = 2 \times \frac{precision \times recall}{precision + recall} \tag{7}$$

IV. Results

In this section, we present the results obtained from our model. The learning curve of our proposed model is illustrated in Figure 5. The model converges after 30 epochs, achieving a training loss of 0.1806 and an IoU score of 0.8201.

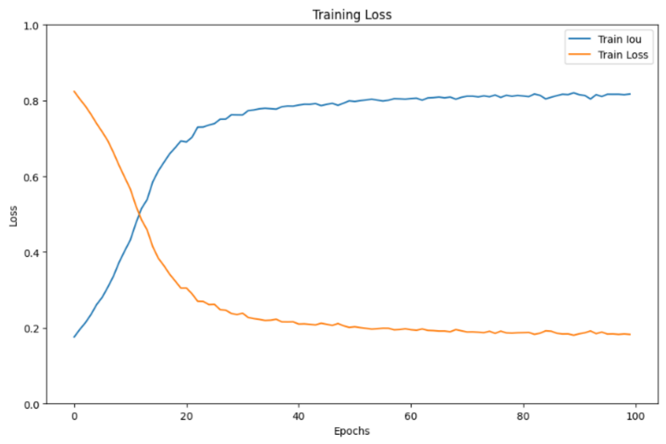


Fig. 5: Training loss

Additionally, our model demonstrates validation convergence after the same number of epochs, reaching a loss of 0.1884 and an IoU score of 0.8047 as illustrated in Figure 6.

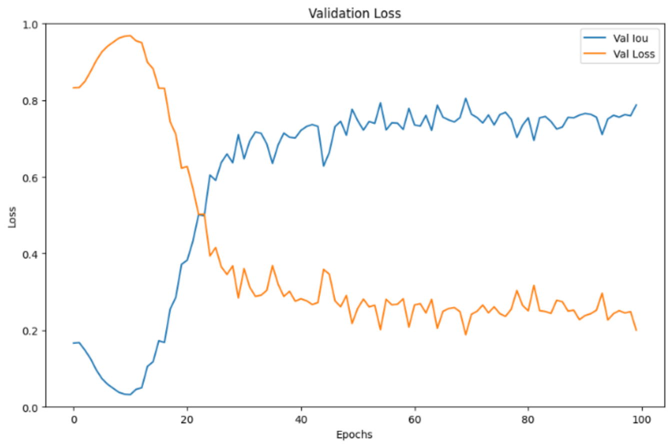


Fig. 6: Validation loss

For comprehensive evaluation, we include measures of accuracy, precision, recall, and F1-score, yielding 0.9624, 0.8873, 0.9192, and 0.8961, respectively in Figure 7

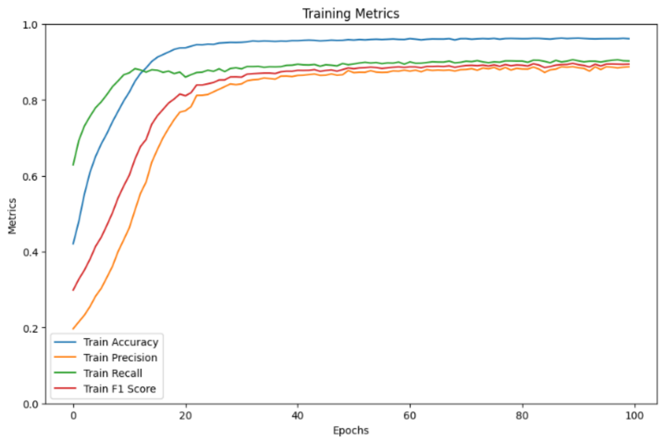


Fig. 7: Training metrics

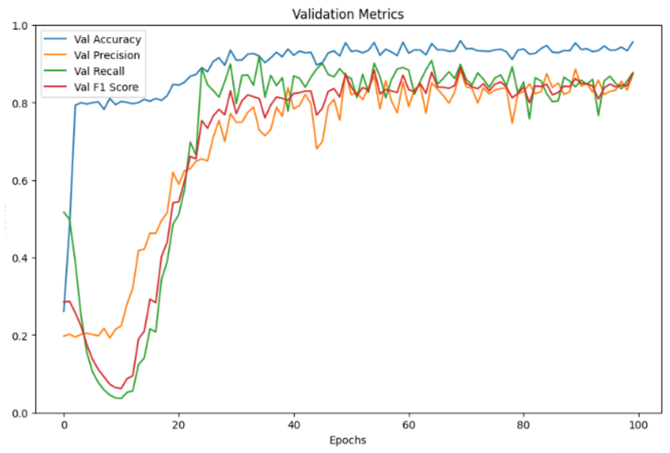


Fig. 8: Validation metrics

Here, we show the segmentation results obtained after post-processing. Our model tend to overestimate nuclei shape as it can be seen in the Differences column in blue in Figure 9 and some part are missing in red. But overall, the results appear to be compelling.

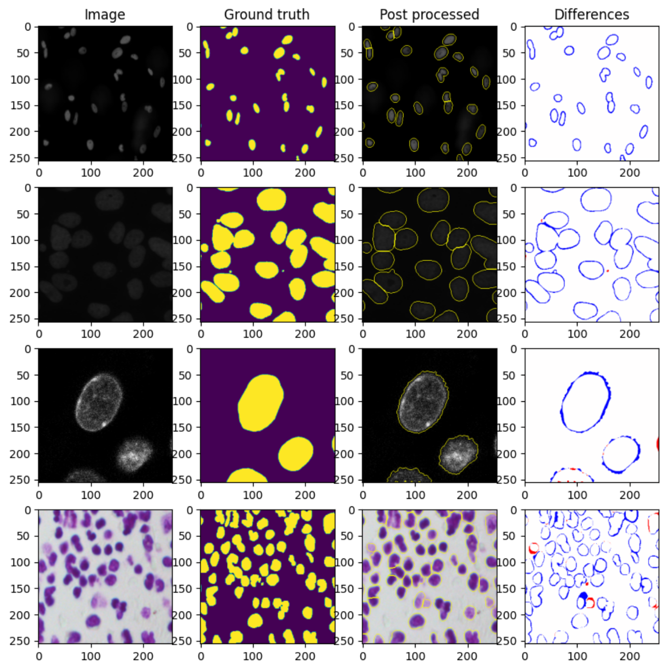
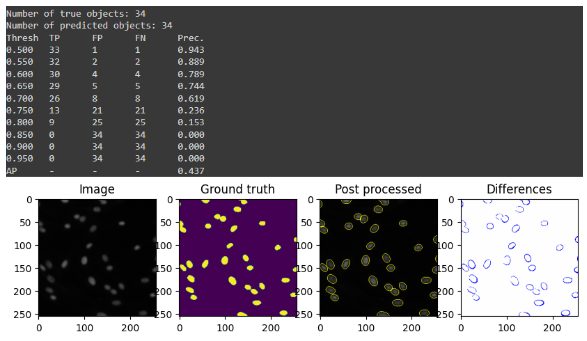


Fig. 9: Examples of some post processed images from test set. In Differences column, blue indicates extra part and red indicates missing part

Here, we detailed the computation for each threshold for one image. As the threshold increases, it becomes increasingly challenging to achieve a high score, as the segmentation requires a higher level of precision.



```
Number of true objects: 34
Number of predicted objects: 34
Thresh  TP      FP      FN      Prec.
0.500   33      1       1       0.943
0.550   32      2       2       0.889
0.600   30      4       4       0.789
0.650   29      5       5       0.744
0.700   26      8       8       0.619
0.750   13      21      21      0.236
0.800   9       25      25      0.153
0.850   0       34      34      0.000
0.900   0       34      34      0.000
0.950   0       34      34      0.000
AP      -       -       -       0.437
```

Fig. 10: Example of one post processed image with the details of evaluation metric

Finally, we summarize the values obtained:

TABLE II: Evaluation Metrics

Accuracy	Precision	Recall	F1 score	Top DSB	Our DSB
0.9624	0.8873	0.9192	0.8961	0.6313	0.5187

V. Discussions

Caicedo et al. [2] found that most participants in their study relied on deep convolutional neural networks (CNNs) for their competion. Participants used various CNN architectures for image segmentation, with creative solutions to improve their scores. Interestingly, the top three participants used different methods: an ensemble of U-Nets, a fully convolutional feature pyramid network (FPN), and a Mask-RCNN model. What distinguishes the winners was a combination of pre-processing and post-processing techniques, as well as the application of best practices during training (mostly data balancing and data augmentation). We chose U-Net for its simplicity and effectiveness over Mask-RCNN and other networks. Even though U-Net isn't designed specifically for instance segmentation, it produced impressive results with effective post-processing, as shown by the top model for the competition.

One of the biggest challenges we faced was training the model to accurately define nuclei contours across images with different colors, sizes, and brightness levels. As shown in Figure 9, the segmentation isn't perfect, with some areas being overestimated. Training was crucial, and we had to carefully adjust the parameters. Data augmentation played a important role too, especially since our dataset was small and the model tended to overfit. Some data augmentation techniques were particularly important such as color shifting color to deal with different image colors and scaling methods to prevent distortions [11]. In our case, we didn't implement color shifting as it did not work but if we had managed to use it, it might have made our results better with color images.

We have not tried all the options to make our model better yet. There's still room to explore new approaches. We could tweak things like batch size and learning rates, or try different ways to pre-process our data. The top competitors used a batch size of 16 and an adaptive learning rate as the training went on. They also used heavy data augmenation techniques to enhance their data which helped a lot to generalize their model. Making improvements in how we spot markers for our post-processing could also really make a difference in our final results. For example, the competition winner used a weighted loss to help the model to better delimit the nuclei. They also added an extra head in their U-Net architecture to better find markers for the post-processing part. We didn't have time to try all that, but it could surely have made our results even better. [11]

Finally, the competition employs a highly punitive metric that goes to thresholds ranging from 0.5 to 0.95 and computes their mean, resulting in relatively low scores compared to conventional standards. However, this metric prioritizes the need for precise segmentation and accurate detection of nuclei as it can be seen in Figure 10

VI. Conclusion

In our study, we tackled the task of segmenting cell nuclei using deep learning methods, focusing on the Kaggle Data

Science Bowl (DSB) 2018 challenge. By tweaking and improving our U-Net model with a ResNet encoder and post-processing techniques like watershed, we achieved compelling results comparable to the top performers in the competition. Despite facing challenges in accurately delimiting nuclei and dealing with different types of images, our approach showed good potential.

References

[1] F. Long, "Microscopy cell nuclei segmentation with enhanced U-Net," *BMC Bioinformatics*, vol. 21, no. 1, p. 8, Dec. 2020. [Online]. Available: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-019-3332-1

[2] J. C. Caicedo, A. Goodman, K. W. Karhohs, B. A. Cimini, J. Ackerman, M. Haghighi, C. Heng, T. Becker, M. Doan, C. McQuin, M. Rohban, S. Singh, and A. E. Carpenter, "Nucleus segmentation across imaging experiments: the 2018 Data Science Bowl," *Nature Methods*, vol. 16, no. 12, pp. 1247–1253, Dec. 2019. [Online]. Available: https://www.nature.com/articles/s41592-019-0612-7

[3] "2018 Data Science Bowl | Kaggle." [Online]. Available: https://www.kaggle.com/competitions/data-science-bowl-2018

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 2015, arXiv:1505.04597 [cs]. [Online]. Available: http://arxiv.org/abs/1505.04597

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Dec. 2015, arXiv:1512.03385 [cs] version: 1. [Online]. Available: http://arxiv.org/abs/1512.03385

[6] "What is Overfitting? | IBM." [Online]. Available: https://www.ibm.com/topics/overfitting

[7] R. Azad, M. Heidary, K. Yilmaz, M. Hüttemann, S. Karimijafarbigloo, Y. Wu, A. Schmeink, and D. Merhof, "Loss Functions in the Era of Semantic Segmentation: A Survey and Outlook," Dec. 2023, arXiv:2312.05391 [cs] version: 1. [Online]. Available: http://arxiv.org/abs/2312.05391

[8] "ADAM Optimizer | Baeldung on Computer Science," Feb. 2023. [Online]. Available: https://www.baeldung.com/cs/adam-optimizer

[9] N. Moshkov, B. Mathe, A. Kertesz-Farkas, R. Hollandi, and P. Horvath, "Test-time augmentation for deep learning-based cell segmentation on microscopy images," *Scientific Reports*, vol. 10, no. 1, p. 5068, Mar. 2020, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41598-020-61808-3

[10] R. Hollandi, A. Szkalisity, T. Toth, E. Tasnadi, C. Molnar, B. Mathe, I. Grexa, J. Molnar, A. Balind, M. Gorbe, M. Kovacs, E. Migh, A. Goodman, T. Balassa, K. Koos, W. Wang, J. C. Caicedo, N. Bara, F. Kovacs, L. Paavolainen, T. Danka, A. Kriston, A. E. Carpenter, K. Smith, and P. Horvath, "nucleAIzer: A Parameter-free Deep Learning Framework for Nucleus Segmentation Using Image Style Transfer," *Cell Systems*, vol. 10, no. 5, pp. 453–458.e6, May 2020. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2405471220301174

[11] "2018 Data Science Bowl." [Online]. Available: https://www.kaggle.com/competitions/data-science-bowl-2018/discussion/54741