Cell Nuclei Segmentation

Aayush Dubey, Anish Borkar, Godavarthi Sai Nikhil

Computer Science Department, BML Munjal University, Gurgaon Haryana 122413, India

Abstract. Background and Objectives: Biomedical research, especially in the realms of diseases like lung cancer, heart disease, Alzheimer's, and diabetes, faces the formidable challenge of time-consuming manual analysis in the quest for novel treatments. This project addresses the critical need for accelerating this process by automating nucleus detection, a foundational step in understanding cellular dynamics and advancing drug discovery. Diseases affecting millions could benefit from expedited cures, making the automation of nucleus identification paramount. Through meticulous image analysis, researchers gain insights into cellular behavior, laying the groundwork for breakthroughs in treatment modalities.The primary objective of this project is to leverage the potent UNet architecture for the development of an efficient algorithm capable of precise cell nucleus segmentation. By automating this fundamental aspect of biomedical image analysis, the goal is to significantly reduce the time required for research, ultimately expediting the path from discovery to cure. The project envisions transforming the current paradigm, where drug development spans a decade, by providing a powerful tool that can accelerate this timeline. **Material & Methods:** Employing the UNet architecture, renowned for its success in segmentation, our approach involves a meticulous training process on the diverse dataset from the 2018 Data Science Bowl competition. This dataset presents complex challenges in nucleus detection due to variations in cell types and staining. The UNet model is trained on annotated datasets, allowing it to learn intricate patterns and features critical for accurate cell nucleus segmentation. The methodological emphasis is on ensuring adaptability to diverse datasets, making the model robust and applicable across various experimental conditions. **Results:** The UNet-based model exhibits promising results in automating cell nucleus segmentation, showcasing enhanced efficiency in comparison to traditional manual methods. Evaluation metrics such as precision and recall attest to the model's accuracy in identifying cell nuclei. The potential impact of this work extends beyond individual diseases, offering a transformative tool for biomedical researchers engaged in a spectrum of investigations. By expediting nucleus detection, the model contributes to streamlining drug testing processes, potentially compressing the timeline for drug development.

Keywords: UNet, Segmentation, Cell Nuclei Segmentation, Image Segmentation

1 Introduction

In contemporary biomedical research, the pivotal role of cell nucleus segmentation cannot be overstated. As a fundamental component of image analysis, precise identification of cell nuclei serves as the linchpin for unraveling intricate cellular processes. This project delves into the critical realm of automated nucleus detection, offering a transformative solution poised to expedite advancements in our understanding of diseases ranging from prevalent afflictions like lung cancer and heart disease to rare disorders.

The context of our endeavor lies in the recognition that the traditional, manual methods of nucleus identification present a bottleneck in the research pipeline. By harnessing the power of cutting-edge technology, specifically the UNet architecture, our project aims to redefine the landscape of biomedical investigation. The importance of this endeavor becomes pronounced when considering the profound impact it could have on accelerating drug discovery, potentially shortening the arduous timeline from the laboratory to the market. The significance of automating nucleus detection becomes particularly evident when contemplating the vast spectrum of diseases, including but not limited to Alzheimer's, diabetes, and chronic obstructive pulmonary disease. By streamlining the process of identifying cell nuclei, we aspire to provide researchers with a tool that expedites their workflow and opens avenues for gaining deeper insights into the underlying biological mechanisms of diverse diseases.

In essence, this project seeks to bridge the gap between conventional methodologies and the burgeoning possibilities offered by advanced image analysis. By automating nucleus detection, we endeavor to propel biomedical research into a new era, where breakthroughs in disease understanding and treatment development occur with unprecedented efficiency. The primary objectives of our project are twofold: firstly, to harness the formidable capabilities of the UNet architecture for the development of an efficient algorithm dedicated to precise cell nucleus segmentation, and secondly, to address the pressing need for expediting research and drug discovery through the automation of nucleus detection.

In the realm of biomedical research, time stands as a formidable obstacle, particularly when it comes to the labor-intensive task of manually identifying cell nuclei. The overarching goal of our project is to confront this challenge head-on by creating a sophisticated algorithm that not only ensures accurate cell nucleus segmentation but also significantly reduces the time investment required for this critical step in the research process. The imperative for accelerating research and drug discovery through automated nucleus detection becomes evident when considering the lengthy and intricate journey from laboratory discoveries to tangible treatments. The conventional timeline for drug development, often spanning a decade, poses a substantial barrier to the rapid translation of groundbreaking findings into viable therapeutic solutions. By automating nucleus detection, we seek to dismantle this barrier, ushering in a new era where the identification of cellular structures becomes a swift and precise endeavor.

Through this project, we aim to provide researchers with a potent tool that streamlines their workflow, enabling them to focus more on the interpretation of results and less on manual, time-consuming tasks. The ultimate objective is to catalyze advancements in our understanding of diseases and expedite the development of novel treatments, potentially transforming the landscape of biomedical research. The methodology employed in this project centers on the strategic utilization of the UNet architecture, renowned for its prowess in semantic segmentation tasks, to achieve precise cell nucleus segmentation. Our approach is rooted in the recognition of UNet's effectiveness in capturing intricate patterns and features crucial for accurate segmentation in biomedical images.

The cornerstone of our analysis is the dataset derived from the 2018 Data Science Bowl competition, chosen for its richness in diversity and complexity. This dataset poses a spectrum of challenges, reflecting the real-world variations encountered in biomedical imaging. The inclusion of nuanced features such as cell types and staining variations ensures that our model is not only robust but also adaptable to the heterogeneous nature of experimental conditions.

During the training phase, the UNet model is exposed to annotated datasets, enabling it to learn the nuanced correlations between pixel intensities and corresponding cell nuclei. The emphasis on learning from annotated datasets is paramount, as it ensures that the model can generalize well to unseen data, a crucial aspect in the challenging domain of biomedical image analysis.

Furthermore, our approach acknowledges the need for a solution that extends beyond a single algorithm. As such, the project places a strong emphasis on adaptability, recognizing that future datasets and experimental conditions may vary. This commitment to adaptability aligns with the dynamic nature of biomedical research, where diverse imaging techniques and protocols are employed.

In summary, our methodology rests on the robust UNet architecture, applied to a rich dataset from the 2018 Data Science Bowl. The fusion of a powerful algorithm with a diverse dataset positions our project to contribute not only to the specific task of cell nucleus segmentation but also to the broader goal of advancing automated image analysis in biomedical research.

Later we will also be discussing the Literature Review, Methodology, Results and the Conclusion will be presented with references that inspired our work.

2 Related Work

In [1], the authors built the UNet architecture. The network consisted of a contracting path (left side) and an expansive path (right side). The contracting path consisted of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. The expansive path consisted of an upsampling of the feature map followed by a 2x2 convolution, a concatenation with the correspondingly cropped feature map from the contracting path and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer, a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. This resulting network is applied to various biomedical segmentation problems. The results were shown on the segmentation of neuronal structures in EM stacks. Warping Error, Rand Error and Pixel Error were the evaluation metrics. This UNet architecture gave the lowest Warping Error (0.000353). The Rand Error was 0.0382 and Pixel Error was 0.0611. The IOU on the PhC-U373 dataset was 0.9203 and for the DIC-HeLa was 0.7756.

The authors [2] used a combination of K-means, watershed segmentation method and Difference In Strength (DIS) map for the segmentation and edge detection tasks. The paper aimed to solve the problem of undesirable over-segmentation results produced by the watershed algorithm when used directly with raw data images. The watershed technique aims to search for regions of high-intensity gradients as the technique is used on the gradient of the image rather than on the image itself. Without a clustering algorithm like K-means, the watershed algorithm can produce an over-segmented image. The clustering method helped obtain an image of different intensity regions based on minimum distance to examine each pixel in the image and then to assign it to one of the image clusters. The only disadvantage is the heavy dependence of the techniques on the clustering part, where if the clustering procedure is not implemented correctly, the results are incorrect by the other techniques used.

The paper [3] proposed an enhanced version of classical U-Net, called U-Net +, by re-designing the encoded branch with densely connected convolutional blocks. The dataset used was the 2018 Kaggle cell nuclei segmentation competition dataset. In each convolutional block, convolutional operation was performed with kernel size 3×3, stride (1,1) and zero-padding. The batch normalization layer was employed for accelerating the training speed and the activation function for each neuron in the block was set to rectified linear unit (ReLU). The down-sampling block was implemented through the convolutional operation with stride (2,2) in row and column dimensions respectively. The rest settings of the down-sampling block were the same as the convolutional block. The up-sampling block was implemented by either transposed convolution (kernel size 3×3 and stride (1,1)) or up-sampling followed by two convolutional blocks. The evaluation metric used was IOU. The model performance of the proposed U-Net+ achieves approximately 1.0% to 3.0% gain with fewer number of weights and shorter inference time compared to U-Net and U-Net++.

The paper [4] presented an approach towards Clustering Based Image Segmentation. The presented method utilizes the L*A*B* color space to convert the images and uses cosine distance with the K-Means algorithm. The clustered image is filtered with Sobel Filter. Next, marker controlled Watershed Segmentation algorithm is used to refine the segmentation results. The evaluation metrics used were MSE (Mean Squared Error) and PSNR (Peak Signal to Noise Ratio). Throughout the process, the MSE reduced from 557110.37 to 3915.66 and the PSNR value increased from -9.2946131 dB to 12.2367546 dB.

The paper [5] presented U-Net++, an improved version of the classic U-Net. The architecture is essentially a deeply supervised encoder-decoder network where the encoder and decoder sub-networks are connected through a series of nested, dense skip pathways. The re-designed skip pathways aimed at reducing the semantic gap between the feature maps of the encoder and decoder sub-networks. The feature maps undergo a dense convolution block whose number of convolution layers depends on the pyramid level. The dense convolution block brings the semantic level of the encoder feature maps closer to that of the feature maps awaiting in the decoder. The datasets used were Data Science Bowl 2018, ASU-Mayo, MICAAI 2018 LiTS Challenge and LIDC-IDRI. The evaluation metrics were the Dice coefficient and IOU. The model was compared with U-Net and wide U-Net. The IOU was highest for UNet++ without Deep Supervision for the datasets Data Science Bowl 2018 and ASU-Mayo, the values being, 92.63 and 33.45, respectively. The IOU was the highest UNet++ with Deep Supervision for the datasets MICAAI 2018 LiTS Challenge and LIDC-IDRI, the values being, 82.9 and 77.21, respectively.

The paper [9] presented a variation of the UNet, called Sharp Dense UNet for the purpose of nucleus segmentation. The downsampling path consists of dense and transition operations are used instead of max pooling and convolution to extract more informative information. In the up-sampling path, a new up-sampling layer, merging, and dense blocks reconstitute high-resolution images. Sharpening spatial filters take the place of skip connections to stop feature mismatches between the decoder and encoder paths. The proposed model had obtained accuracy of 0.6856, 0.5248, 84.49, respectively.

3 Methodology

3.1 Dataset

For this study, we utilized the 2018 Data Science Bowl - Processed cell nuclei segmentation dataset downloaded from Kaggle [8], which contains microscopy images. The dataset consists of 670 images and 670 masks. It is important to note that the dataset used has already undergone preprocessing. This dataset, specifically created for advancing medical image analysis, poses the challenge of automating the segmentation of cell nuclei. Each image may contain multiple nuclei with distinct shapes and sizes. The dataset includes labeled images, with corresponding masks serving as the ground truth for nucleus segmentation. Evaluation metrics, such as Intersection over Union (IoU) and Dice Coefficient, were employed to assess the accuracy of segmentation predictions.

3.2 Pre-Processing

The 2018 Bowl Dataset for cell nuclei segmentation comprises data collected from various sources, encompassing diverse imaging sets. To establish a standardized input size for the neural network, all images in the dataset were uniformly resized to dimensions of 256 x 256 pixels. Later, they underwent the normalization of their pixel values, where each pixel's intensity values were normalized to a range between 0 and 1. This normalization process is a common practice in deep learning, contributing to consistent input data and improving the convergence and overall performance of the neural network.

Boundary Extraction:
Highlighting the boundaries or edges of cell nuclei within the images was an important preprocessing step. This process significantly enhances the network's ability to distinguish between different regions within an image, contributing to the overall accuracy of segmentation.

Region Filling:
Following boundary extraction, region filling was used to complete the segmentation process. This step involves filling segmented regions with appropriate labels, creating clear distinctions between different cellular structures. Region filling is integral to achieving precise identification and delineation of cell nuclei.

In digital image processing for cell nuclei segmentation, using boundary extraction, enhances boundaries, enabling accurate identification. Subsequent region filling with colors mitigates noise, improves segmentation accuracy, and ensures adaptability to varied image characteristics. This preprocessing not only prevents under-segmentation but also provides a visually interpretable output, supporting downstream analysis and facilitating the creation of a high-quality dataset for effective neural network training [7].

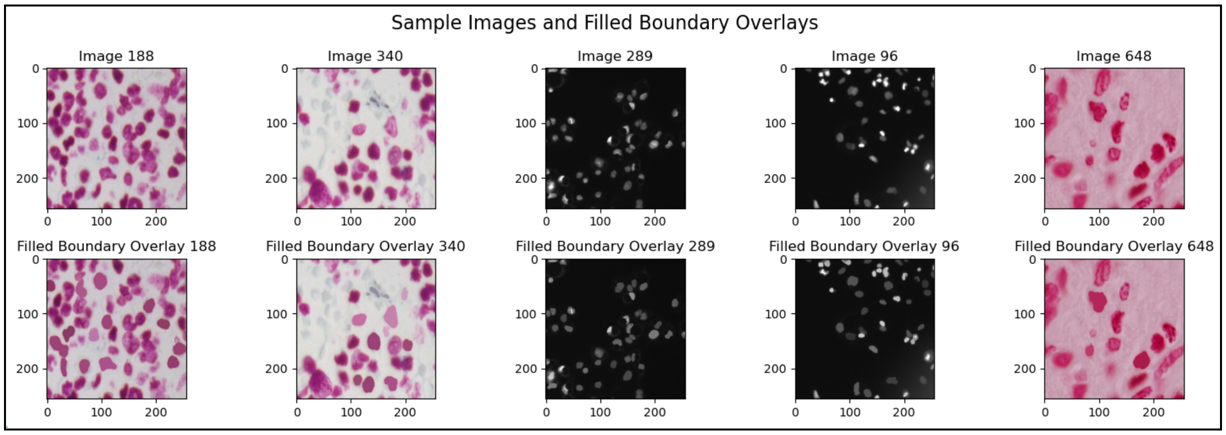

Sample Images and Filled Boundary Overlays
Image 188
Image 340
Image 289
Image 96
Image 648
Filled Boundary Overlay 188
Filled Boundary Overlay 340
Filled Boundary Overlay 289
Filled Boundary Overlay 96
Filled Boundary Overlay 648

Figure 1 - Sample Images and Filled Boundary Overlays

3.3 Model Architecture

UNet is a CNN architecture commonly used for image segmentation. It consists of an encoder and a decoder. The encoder extracts features from the input images. The decoder is responsible for reconstructing the output images from the extracted features. The encoder and decoder are connected by skip connections.

We followed this U-Net architecture, as mentioned in the figure below, as a reference to build our model [6].

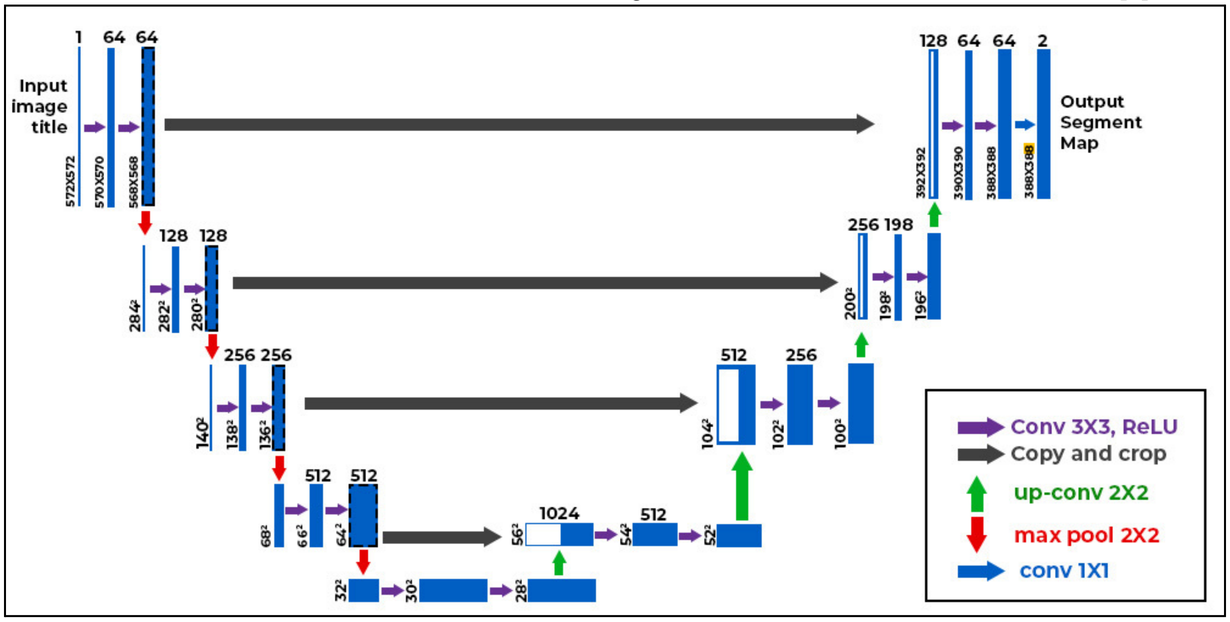
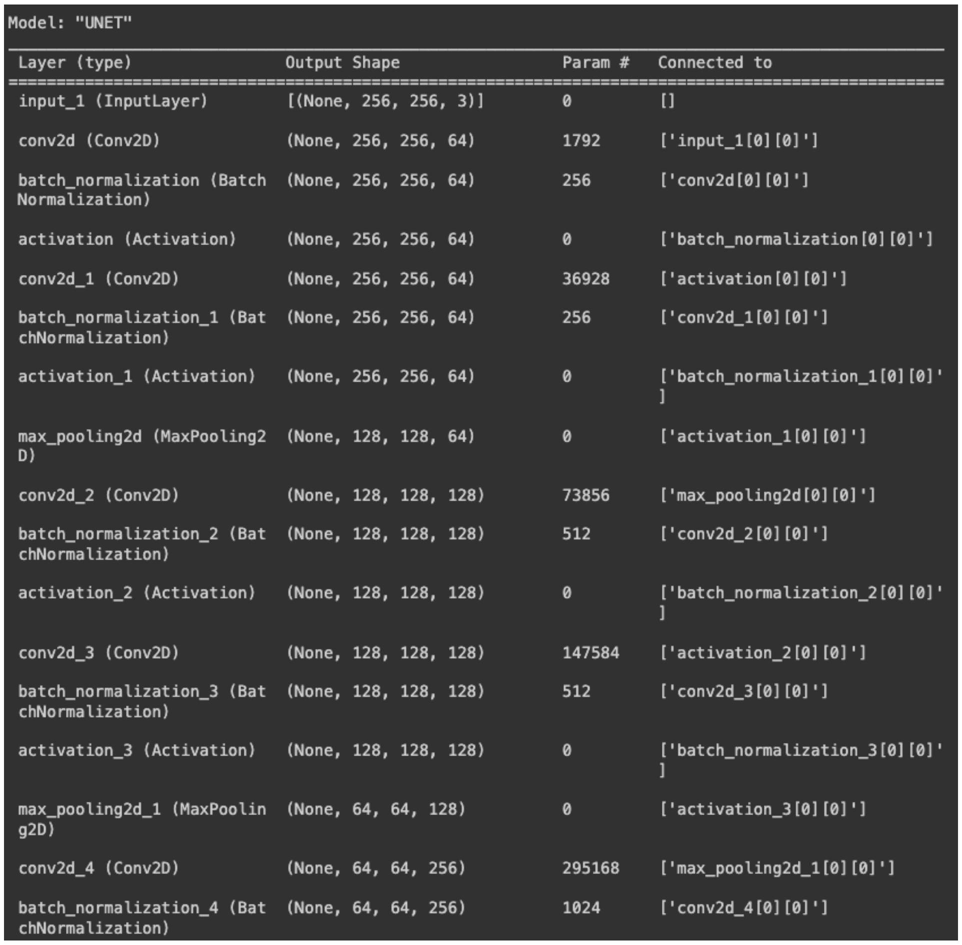
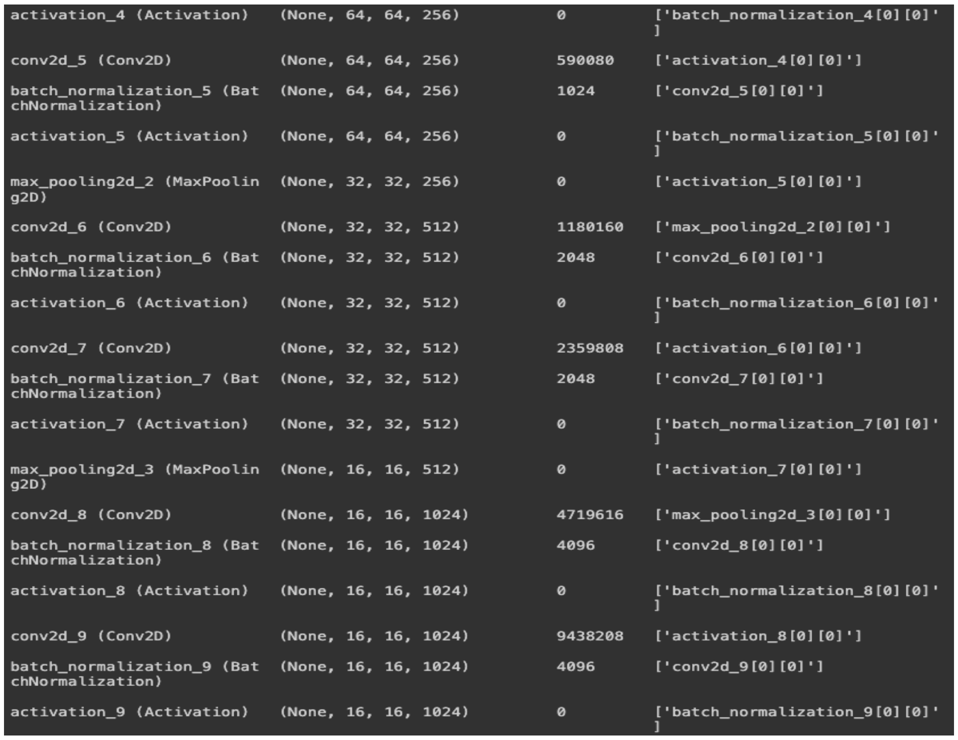


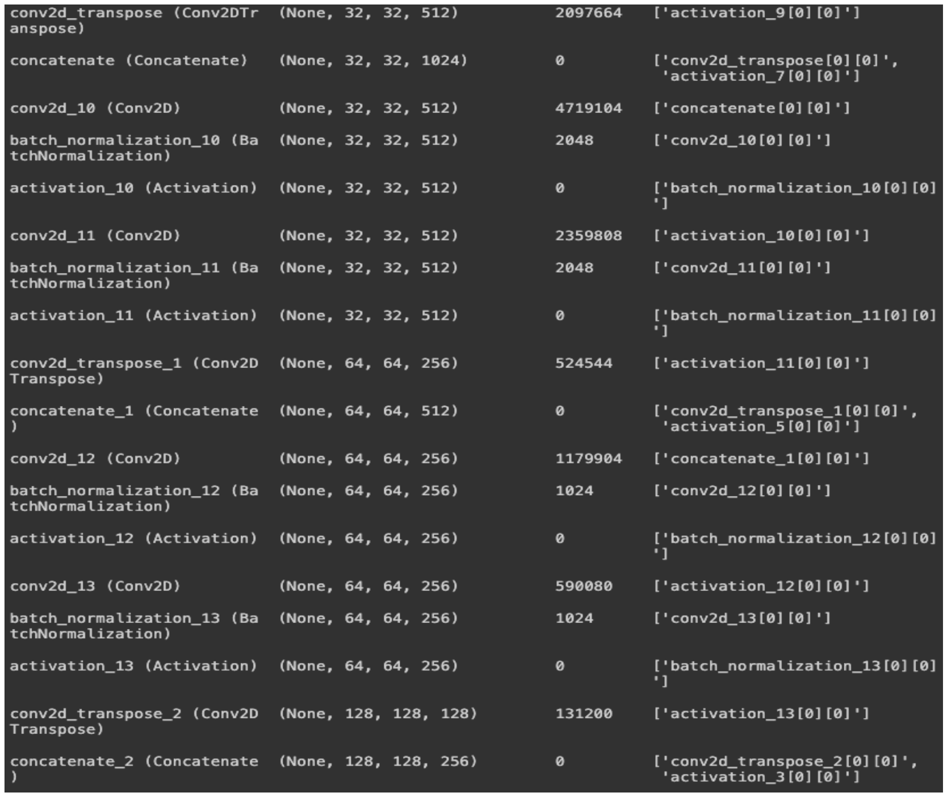
Figure 2 - UNet Architecture



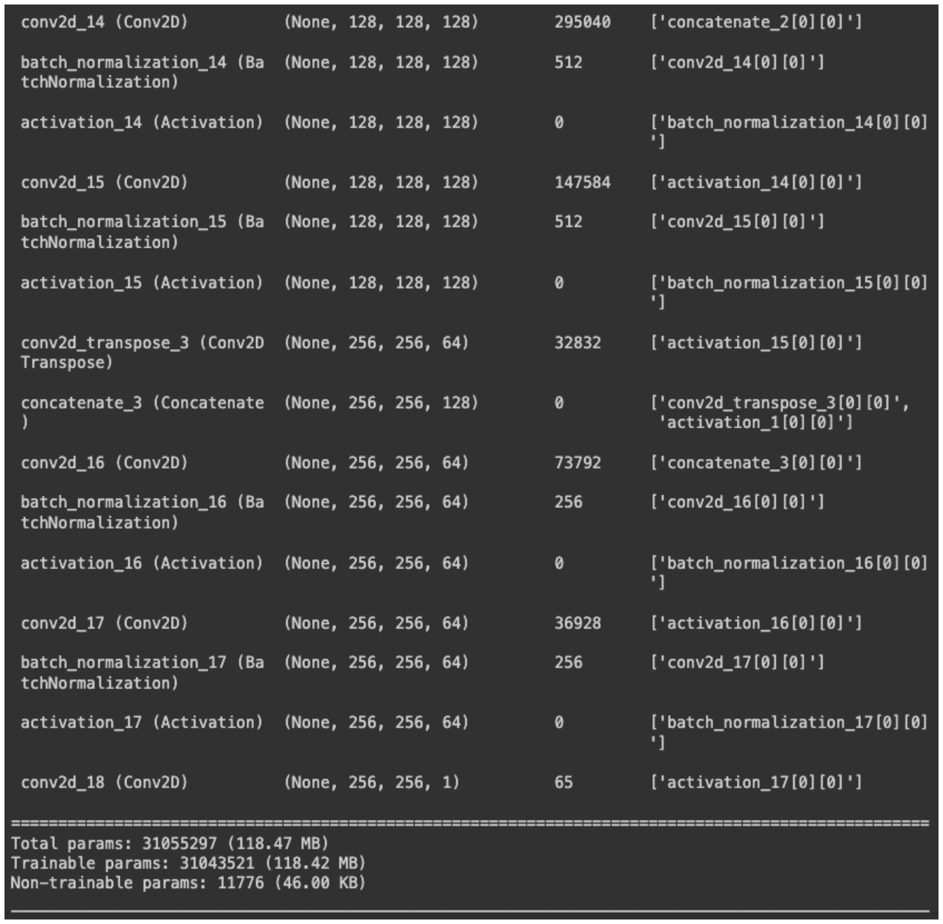
```
Model: "UNET"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 256, 256, 3)]  0         []

 conv2d (Conv2D)                (None, 256, 256, 64)   1792      ['input_1[0][0]']

 batch_normalization (Batch     (None, 256, 256, 64)   256       ['conv2d[0][0]']
 Normalization)

 activation (Activation)        (None, 256, 256, 64)   0         ['batch_normalization[0][0]']

 conv2d_1 (Conv2D)              (None, 256, 256, 64)   36928     ['activation[0][0]']

 batch_normalization_1 (Bat     (None, 256, 256, 64)   256       ['conv2d_1[0][0]']
 chNormalization)

 activation_1 (Activation)      (None, 256, 256, 64)   0         ['batch_normalization_1[0][0]'
                                                                  ]

 max_pooling2d (MaxPooling2     (None, 128, 128, 64)   0         ['activation_1[0][0]']
 D)

 conv2d_2 (Conv2D)              (None, 128, 128, 128)  73856     ['max_pooling2d[0][0]']

 batch_normalization_2 (Bat     (None, 128, 128, 128)  512       ['conv2d_2[0][0]']
 chNormalization)

 activation_2 (Activation)      (None, 128, 128, 128)  0         ['batch_normalization_2[0][0]'
                                                                  ]

 conv2d_3 (Conv2D)              (None, 128, 128, 128)  147584    ['activation_2[0][0]']

 batch_normalization_3 (Bat     (None, 128, 128, 128)  512       ['conv2d_3[0][0]']
 chNormalization)

 activation_3 (Activation)      (None, 128, 128, 128)  0         ['batch_normalization_3[0][0]'
                                                                  ]

 max_pooling2d_1 (MaxPoolin     (None, 64, 64, 128)    0         ['activation_3[0][0]']
 g2D)

 conv2d_4 (Conv2D)              (None, 64, 64, 256)    295168    ['max_pooling2d_1[0][0]']

 batch_normalization_4 (Bat     (None, 64, 64, 256)    1024      ['conv2d_4[0][0]']
 chNormalization)
```



```
activation_4 (Activation)      (None, 64, 64, 256)    0         ['batch_normalization_4[0][0]'
                                                                 ]
conv2d_5 (Conv2D)              (None, 64, 64, 256)    590080    ['activation_4[0][0]']
batch_normalization_5 (Bat     (None, 64, 64, 256)    1024      ['conv2d_5[0][0]']
chNormalization)
activation_5 (Activation)      (None, 64, 64, 256)    0         ['batch_normalization_5[0][0]'
                                                                 ]
max_pooling2d_2 (MaxPoolin     (None, 32, 32, 256)    0         ['activation_5[0][0]']
g2D)
conv2d_6 (Conv2D)              (None, 32, 32, 512)    1180160   ['max_pooling2d_2[0][0]']
batch_normalization_6 (Bat     (None, 32, 32, 512)    2048      ['conv2d_6[0][0]']
chNormalization)
activation_6 (Activation)      (None, 32, 32, 512)    0         ['batch_normalization_6[0][0]'
                                                                 ]
conv2d_7 (Conv2D)              (None, 32, 32, 512)    2359808   ['activation_6[0][0]']
batch_normalization_7 (Bat     (None, 32, 32, 512)    2048      ['conv2d_7[0][0]']
chNormalization)
activation_7 (Activation)      (None, 32, 32, 512)    0         ['batch_normalization_7[0][0]'
                                                                 ]
max_pooling2d_3 (MaxPoolin     (None, 16, 16, 512)    0         ['activation_7[0][0]']
g2D)
conv2d_8 (Conv2D)              (None, 16, 16, 1024)   4719616   ['max_pooling2d_3[0][0]']
batch_normalization_8 (Bat     (None, 16, 16, 1024)   4096      ['conv2d_8[0][0]']
chNormalization)
activation_8 (Activation)      (None, 16, 16, 1024)   0         ['batch_normalization_8[0][0]'
                                                                 ]
conv2d_9 (Conv2D)              (None, 16, 16, 1024)   9438208   ['activation_8[0][0]']
batch_normalization_9 (Bat     (None, 16, 16, 1024)   4096      ['conv2d_9[0][0]']
chNormalization)
activation_9 (Activation)      (None, 16, 16, 1024)   0         ['batch_normalization_9[0][0]'
                                                                 ]
```



```
conv2d_transpose (Conv2DTr  (None, 32, 32, 512)          2097664   ['activation_9[0][0]']
anspose)

concatenate (Concatenate)   (None, 32, 32, 1024)         0         ['conv2d_transpose[0][0]',
                                                                    'activation_7[0][0]']

conv2d_10 (Conv2D)          (None, 32, 32, 512)          4719104   ['concatenate[0][0]']

batch_normalization_10 (Ba  (None, 32, 32, 512)          2048      ['conv2d_10[0][0]']
tchNormalization)

activation_10 (Activation)  (None, 32, 32, 512)          0         ['batch_normalization_10[0][0]
                                                                   ']

conv2d_11 (Conv2D)          (None, 32, 32, 512)          2359808   ['activation_10[0][0]']

batch_normalization_11 (Ba  (None, 32, 32, 512)          2048      ['conv2d_11[0][0]']
tchNormalization)

activation_11 (Activation)  (None, 32, 32, 512)          0         ['batch_normalization_11[0][0]
                                                                   ']

conv2d_transpose_1 (Conv2D  (None, 64, 64, 256)          524544    ['activation_11[0][0]']
Transpose)

concatenate_1 (Concatenate  (None, 64, 64, 512)          0         ['conv2d_transpose_1[0][0]',
)                                                                   'activation_5[0][0]']

conv2d_12 (Conv2D)          (None, 64, 64, 256)          1179904   ['concatenate_1[0][0]']

batch_normalization_12 (Ba  (None, 64, 64, 256)          1024      ['conv2d_12[0][0]']
tchNormalization)

activation_12 (Activation)  (None, 64, 64, 256)          0         ['batch_normalization_12[0][0]
                                                                   ']

conv2d_13 (Conv2D)          (None, 64, 64, 256)          590080    ['activation_12[0][0]']

batch_normalization_13 (Ba  (None, 64, 64, 256)          1024      ['conv2d_13[0][0]']
tchNormalization)

activation_13 (Activation)  (None, 64, 64, 256)          0         ['batch_normalization_13[0][0]
                                                                   ']

conv2d_transpose_2 (Conv2D  (None, 128, 128, 128)        131200    ['activation_13[0][0]']
Transpose)

concatenate_2 (Concatenate  (None, 128, 128, 256)        0         ['conv2d_transpose_2[0][0]',
)                                                                   'activation_3[0][0]']
```



```
conv2d_14 (Conv2D)          (None, 128, 128, 128)        295040    ['concatenate_2[0][0]']

batch_normalization_14 (Ba  (None, 128, 128, 128)        512       ['conv2d_14[0][0]']
tchNormalization)

activation_14 (Activation)  (None, 128, 128, 128)        0         ['batch_normalization_14[0][0]
                                                                   ']

conv2d_15 (Conv2D)          (None, 128, 128, 128)        147584    ['activation_14[0][0]']

batch_normalization_15 (Ba  (None, 128, 128, 128)        512       ['conv2d_15[0][0]']
tchNormalization)

activation_15 (Activation)  (None, 128, 128, 128)        0         ['batch_normalization_15[0][0]
                                                                   ']

conv2d_transpose_3 (Conv2D  (None, 256, 256, 64)         32832     ['activation_15[0][0]']
Transpose)

concatenate_3 (Concatenate  (None, 256, 256, 128)        0         ['conv2d_transpose_3[0][0]',
)                                                                   'activation_1[0][0]']

conv2d_16 (Conv2D)          (None, 256, 256, 64)         73792     ['concatenate_3[0][0]']

batch_normalization_16 (Ba  (None, 256, 256, 64)         256       ['conv2d_16[0][0]']
tchNormalization)

activation_16 (Activation)  (None, 256, 256, 64)         0         ['batch_normalization_16[0][0]
                                                                   ']

conv2d_17 (Conv2D)          (None, 256, 256, 64)         36928     ['activation_16[0][0]']

batch_normalization_17 (Ba  (None, 256, 256, 64)         256       ['conv2d_17[0][0]']
tchNormalization)

activation_17 (Activation)  (None, 256, 256, 64)         0         ['batch_normalization_17[0][0]
                                                                   ']

conv2d_18 (Conv2D)          (None, 256, 256, 1)          65        ['activation_17[0][0]']

==================================================================================================
Total params: 31055297 (118.47 MB)
Trainable params: 31043521 (118.42 MB)
Non-trainable params: 11776 (46.00 KB)
__________________________________________________________________________________________________
```

The table illustrates a neural network model adopting a U-Net architecture for image segmentation. It takes a grayscale image of size (256, 256, 1) as input and outputs a segmentation mask of the same size. The model comprises three blocks: one convolution block, one encoder block, and one decoder block. The UNet architecture consists of four encoder blocks, totalling 12 layers, with each block comprising two convolutional layers followed by a max-pooling layer. The number of feature channels in the convolutional layers doubles with each block, and the max-pooling layers reduce the spatial dimension of the image half. A bottleneck, a single convolutional block between the encoder and decoder, is followed by four decoder blocks. Each decoder block has an upsampling layer followed by two convolutional layers, resulting in a total of 16 layers. The upsampling layers double the spatial dimension and halve the number of feature channels with each block. The decoder incorporates skip connections linking each block to its corresponding block in the encoder. The output layer is a single convolutional layer with one filter and a sigmoid activation function.

Total Number of Conv2D Layers:
16 (Encoder) + 2 (Bottleneck) + 16 (Decoder) + 1 (Output) = 35 Conv2D layers

Out of the 35 Conv2D layers, only 18 are directly connected to the input and output layers. The remaining layers, such as batch normalization, activation, max-pooling, and transpose convolution, are not directly connected to the input or output layers, and therefore, they are not shown in the summary. This summary offers a simplified view of the model architecture for improved readability. The model has a total of 3,10,55,297 parameters, with 3,10,43,521 trainable parameters and 11,176 non-trainable parameters, indicating a high level of model complexity.

4 Results

To assess the efficiency of our UNet-based cell nucleus segmentation model, we employed a comprehensive set of performance metrics. These metrics encompass fundamental aspects of segmentation accuracy and efficacy.

Accuracy: Accuracy represents the overall correctness of our model in correctly identifying and segmenting cell nuclei within the images. It is calculated as the ratio of correctly predicted instances to the total instances.

F1 Score: The F1 score provides a balanced measure of precision and recall. It is particularly useful in scenarios where there is an uneven class distribution. The F1 score is calculated as the harmonic mean of precision and recall.

Jaccard Score: The Jaccard score evaluates the similarity between the predicted and ground truth segmentations. It is computed as the intersection of the predicted and true positives divided by the union of predicted positives and true positives.

Precision: Precision measures the accuracy of positive predictions made by the model. It is calculated as the ratio of true positives to the sum of true positives and false positives.

Recall (Sensitivity): Recall, also known as sensitivity, gauges the model's ability to capture all positive instances. It is calculated as the ratio of true positives to the sum of true positives and false negatives.

Table 1 - Result of the model after training 35 epochs

Metric	**Value**
Accuracy	97.15%
F1 Score	89.73%
Jaccard Score	0.8269

Precision	90.7%
Recall	90.4%

Below are the segmentation metrics for a few sample images:

Table 2 - Evaluation metrics on specific images

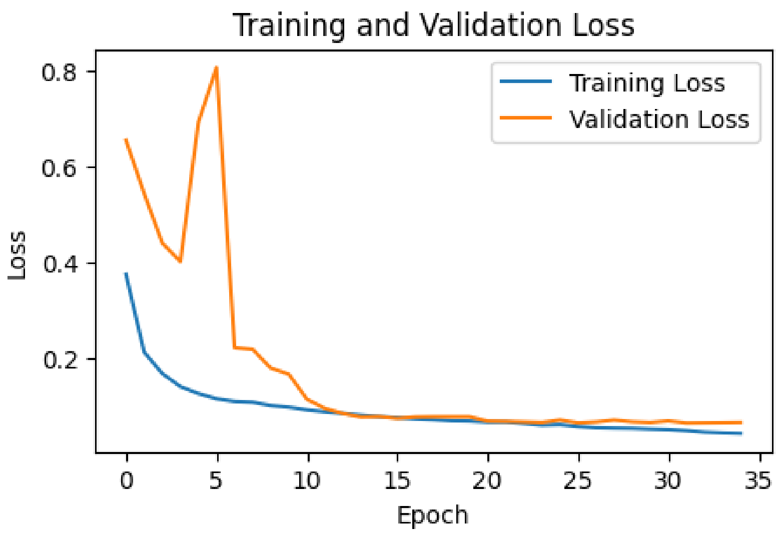
Image	Accuracy	F1	Jaccard	Recall	Precision
6034456567632f4b48dc3dfbb98534b5953c151990f4235df6c912c0a9c08397.jpg	99.65%	95.7%	0.9179	94.25%	97.22%
54cb3328e778d87f76062b0550e3bc190f46384acd8efbe58c297265d1906e84.jpg	98.8%	90.33%	0.823	86.3%	94.76%
e52960d31f8bddf85400259beb4521383f5ceface1080be3429f2f926cc9b5c2.jpg	98.03%	95.67%	0.9171	94.12%	97.2%

Analysis of Training and Validation Metrics Over Epochs:

In our U-Net Model we have taken binary cross-entropy loss, which measures the difference between the predicted mask and the ground truth mask.

Training Loss: This is the loss calculated on the training dataset during each epoch of training. It represents how well the model fits the training data. The goal during training is to minimize this loss, meaning that the model is getting better at predicting the training data.

Validation Loss: This is the loss calculated on a separate validation dataset during each epoch. The validation loss helps assess how well the model is generalizing to new, unseen data. If the training loss is decreasing, but the validation loss starts increasing, it may indicate overfitting, where the model is fitting the training data too closely and failing to generalize to new data. It may initially decrease, and when the model starts to overfit, it might increase.



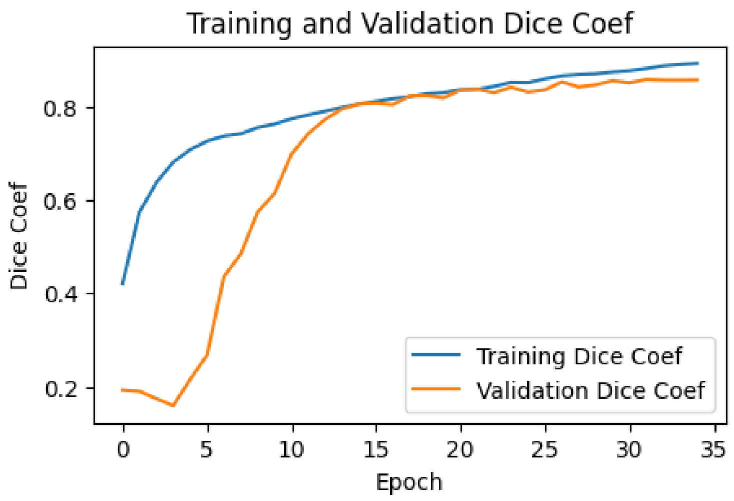
In the above figure, it can be observed that the loss decreases gradually with epochs, indicating the model's proficiency in predicting the training data. Regarding the validation loss, initially, it decreased until 3 epochs, then increased until 5, and later decreased. This implies that the model was overfitting around Epoch 5, but starting from Epoch 6, it exhibited a decreasing trend, suggesting improved generalization to the validation set.

Dice coefficient:

The Dice coefficient is commonly used to measure the agreement between the predicted segmentation mask and the ground truth mask.

Training Dice Coef: This metric measures the overlap between the predicted and true masks for the training dataset. A higher Dice coefficient indicates better overlap. During training, if the Dice coefficient increases, indicating that the model is getting better at accurately predicting the segmentation masks.

Validation Dice Coef: It is similar to the training Dice coefficient, but calculated on the validation dataset. This metric gives you an idea of how well the model is generalizing to new data. If the validation Dice coefficient is high, it means the model is performing well on data it hasn't seen during training.



In the above figure, there is an increase in the Training Dice Coef, indicating improvement in the overlap on the training set. Regarding the Validation Dice Coef, there was overfitting observed until 3 epochs, but starting from the 4th Epoch, there was an improvement in the overlap on the validation set. In summary, the initial increase in validation loss and decrease in the validation Dice coefficient indicate overfitting. However, throughout training, it improved well.

Segmentation (UNet) Results:

1. **Accuracy:**
   - The overall accuracy of the UNet model in segmenting cell nuclei is impressive, with an average accuracy of approximately 97.15%.

2. **Precision and Recall:**
   - **Precision**: The model shows high precision, correctly identifying positive cases. On average, the precision is around 90.7%, indicating a low false-positive rate.
   - **Recall**: The recall of the model is also notable, capturing a high percentage of positive cases, with an average recall of approximately 90.4%. This suggests a low false-negative rate.

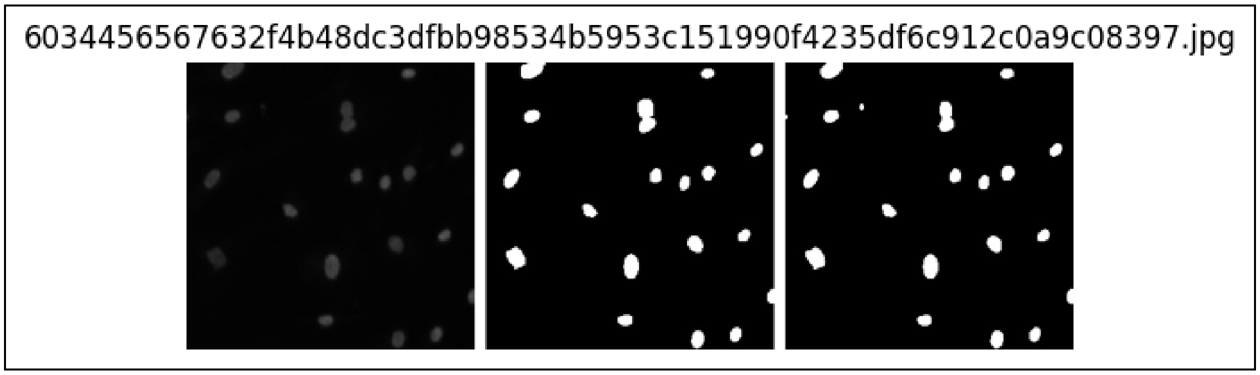

6034456567632f4b48dc3dfbb98534b5953c151990f4235df6c912c0a9c08397.jpg

Figure 3

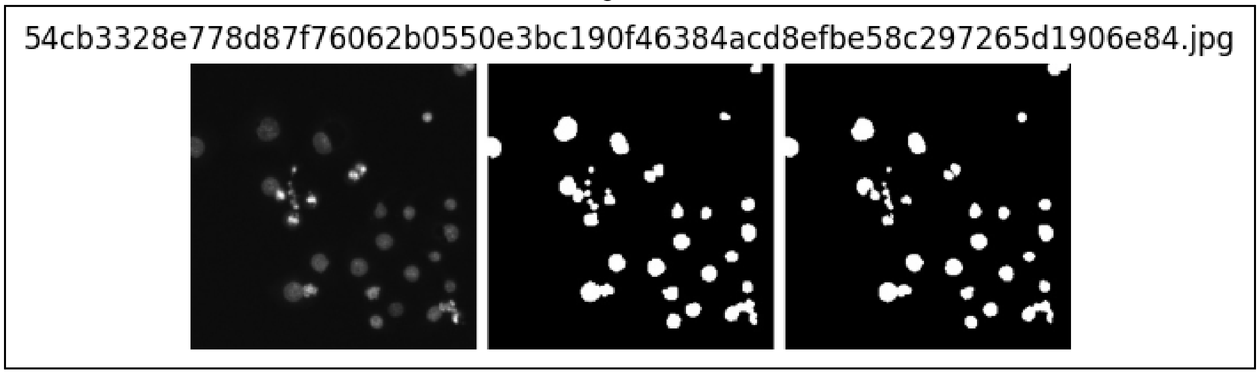

54cb3328e778d87f76062b0550e3bc190f46384acd8efbe58c297265d1906e84.jpg

Figure 4

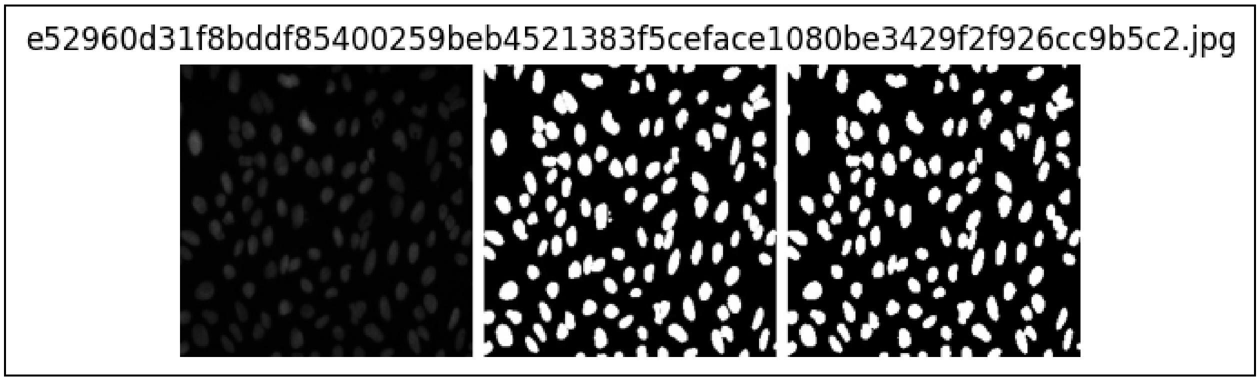

e52960d31f8bddf85400259beb4521383f5ceface1080be3429f2f926cc9b5c2.jpg

Figure 5

The segmentation for Fig 3 is exceptionally accurate, showcasing a well-balanced trade-off between precision and recall. The model adeptly captures spatial details, resulting in a high Jaccard index. Fig 4 exhibits accurate segmentation with a commendable balance between precision and recall, as echoed by the high Jaccard index. Turning to Fig 5, the segmentation is highly accurate, attaining a robust equilibrium between precision and recall, aligning with the substantial Jaccard index. These results align with the performance claims detailed in Table 2, underscoring the model's consistent and effective performance across diverse images.

5 Conclusion.

In conclusion, the segmentation model exhibits exceptional performance with a high accuracy of approximately 0.9715, showcasing its proficiency in accurately classifying pixels into foreground and background. The balanced F1 score of 0.8973 underscores the model's ability to harmonize precision and recall, crucial for tasks demanding both accurate positive predictions and comprehensive identification of true positives. Notably, the precision of 0.91019 signifies a low false positive rate, a critical aspect in applications where misclassifications carry significant consequences. The recall of 0.9040 highlights the model's sensitivity to foreground features, effectively capturing a substantial portion of true positive pixels. The Jaccard score of 0.8269 further emphasizes the model's spatial accuracy, accurately delineating the boundaries of segmented objects.

The effectiveness of the U-Net architecture, encompassing encoder, bottleneck, and decoder blocks, is evident in its successful segmentation of cell nuclei. The symmetric design, coupled with skip connections and suitable activation functions, contributes to the model's proficiency in capturing contextual information and preserving spatial details. The significance of chosen evaluation metrics, including Dice coefficient, IoU, recall, and precision, ensure a comprehensive assessment of segmentation performance, encompassing spatial accuracy, class balancing, and boundary delineation.

In practical terms, the model's high precision and recall, coupled with a balanced F1 score, render it well-suited for applications where precise cell nucleus delineation is paramount, such as in medical imaging for diagnostic purposes. However, future considerations should include a more in-depth analysis of challenges, including variations in nuclei size, shape, and instances of overlapping nuclei. Understanding these challenges will guide future refinements and fine-tuning efforts to further enhance the model's robustness and applicability.

6 References

1. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18* (pp. 234-241). Springer International Publishing.
2. Salman, N. (2006). Image segmentation based on watershed and edge detection techniques. *Int. Arab J. Inf. Technol.*, *3*(2), 104-110.
3. Long, F. (2020). Microscopy cell nuclei segmentation with enhanced U-Net. *BMC bioinformatics*, *21*, 1-12.
4. Bora, D. J., & Gupta, A. K. (2015). A novel approach towards clustering based image segmentation. *arXiv preprint arXiv:1506.01710*.
5. Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., & Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. In *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4* (pp. 3-11). Springer International Publishing.
6. https://www.geeksforgeeks.org/u-net-architecture-explained/
7. https://graphics.ics.uci.edu/CS111/Slides/woodsandgonzalez.pdf
8. https://www.kaggle.com/datasets/84613660e1f97d3b23a89deb1ae6199a0c795ec1f31e2934527a7f7aad7d8c37
9. Senapati, P., Basu, A., Deb, M., & Dhal, K. G. (2023). Sharp dense U-Net: an enhanced dense U-Net architecture for nucleus segmentation. *International Journal of Machine Learning and Cybernetics*, 1-16.