CSE 445 Section 5
Group 2

Nahid Sarwar Ratul 2212790042
Muhammad Bin Mamun 2222953042
Ekfat Jahan Ashrafy 2132236642111

Nuclei Segmentation in Whole Slide Images Using U-Net Architecture

Abstract—**In this project, a deep learning model was developed to segment cell nuclei in whole slide images (WSIs). The model uses the U-Net architecture, known for its effectiveness in image segmentation tasks, especially in medical imaging. The dataset used was from the 2018 Data Science Bowl: Nuclei Segmentation Challenge. The model was trained using a variety of techniques including data preprocessing, augmentation, and optimization. The final results showed good performance in accurately segmenting the nuclei, demonstrating the practical application of deep learning in the field of medical image analysis.**

Index Terms—**U-Net, nuclei segmentation, whole slide images, deep learning, medical image segmentation**

I. Introduction

The segmentation of cell nuclei in whole slide images (WSIs) plays a critical role in the analysis of histopathological tissue samples. Accurate segmentation aids pathologists in detecting abnormalities, such as cancer, by distinguishing cell boundaries in the images. The goal of this project was to build a deep learning model using the U-Net architecture for accurate nuclei segmentation. This task is challenging due to the complexity of tissue structures and varying staining patterns in the images. The model was trained on the 2018 Data Science Bowl dataset [1], which provides annotated images and masks of cell nuclei. Online tutorials [2] and videos [3] were also valuable in learning the U-Net architecture and implementing the model.

II. Methodology

A. Dataset Preparation

The dataset used for training and evaluation was sourced from the 2018 Data Science Bowl: Nuclei Segmentation Challenge. The dataset consists of WSI images and their corresponding mask images, where each mask corresponds to a single nucleus in the image.

1) Sample Loading: Each image is paired with several mask images, which represent individual nuclei. A function, load_sample, was used to load both the images and the corresponding masks which were then converted to RGB format. Each mask image (found in the mask folder) was read and resized to match the input size of the model (256x256 pixels). These masks were then combined into a single mask using a max function to ensure that all individual nuclei were marked in the final binary mask. The resulting mask was then expanded to have a channel dimension and binarized (values of 0 and 1 for background and nuclei, respectively).

2) Data Splitting: The dataset was divided into training and validation sets using an 80-20 split. This was achieved using the train_test_split function from Scikit-learn.

3) TensorFlow Data Pipeline: The training and validation datasets were then converted into TensorFlow tf.data.Dataset objects. This allowed for efficient batching, shuffling, and prefetching, ensuring smooth data flow during model training.

B. Model Architecture

The segmentation model used for this project was based on the U-Net architecture, which is widely used for biomedical image segmentation tasks due to its ability to perform pixel-wise classification.

1) Encoder: The encoder consists of several convolutional blocks with 2 convolutional layers followed by a max-pooling operation. The convolutions extract features from the input image, while max-pooling reduces the spatial dimensions to capture more abstract features.

2) Bottleneck: The bottleneck layer is the deepest part of the model, where the most abstract features are learned. This layer has the highest number of filters (1024), which allows the network to learn complex representations.

3) Decoder: The decoder part upsamples the feature maps using transpose convolutions (also known as deconvolutions), gradually reconstructing the image to its original size. Skip connections are used to merge the corresponding encoder layer's output with the decoder's feature map. These skip connections help the model retain fine-grained spatial information, especially important for accurate segmentation of small structures like nuclei.

4) Output Layer: The output layer consists of a 1x1 convolution with a sigmoid activation function to produce a binary mask. The sigmoid activation ensures that the output is between 0 and 1, representing the background and nucleus.

C. Training Process

1) Compilation: The model was compiled using the Adam optimizer and binary cross-entropy loss function. Since this was a binary segmentation task (nuclei vs. background), binary cross-entropy was suitable for training. The Dice coefficient was also used as an evaluation metric due to its relevance in segmentation tasks.

2) Callbacks: Early stopping was used to halt training if the validation loss did not improve for four consecutive epochs, thus preventing overfitting. Learning rate reduction was applied when the validation loss plateaued, reducing the learning rate by a factor of 0.5.

3) Training: The model was trained for 30 epochs, using the training dataset and evaluated on the validation dataset. The training process was optimized with the callbacks mentioned earlier to avoid overfitting and improve convergence.

D. Evaluation

After training, the model's performance was evaluated using the Dice coefficient, which is a measure of the overlap between the predicted segmentation and the ground truth. This metric is widely used for evaluating segmentation models, as it handles class imbalance well.

The predictions on the validation dataset were visualized alongside the ground truth masks to qualitatively assess the model's segmentation ability. The predicted masks were binarized using a threshold of 0.5 to distinguish the nuclei from the background.

III. Results

The model was trained for 30 epochs, with performance monitored across multiple metrics: accuracy, Dice coefficient, and loss. These metrics were evaluated for both the training and validation datasets, providing insight into the model's segmentation capabilities.

The model showed substantial improvement in the Dice coefficient during the early epochs. Initially, the Dice coefficient increased from around 0.2 to 0.9 by Epoch 10, reflecting the model's effective learning. However, after Epoch 10, the improvements became minimal, suggesting that the model reached a plateau in its learning, as shown in Fig. 1.

Since Epoch 28 demonstrated the best performance (with 0.9130 Dice coefficient on the validation set and 0.0621 loss), the model's weights were restored from Epoch 28, ensuring the best performance was retained. This was done using the early stopping callback, which monitors validation loss and restores the model to the best epoch when validation performance starts to degrade.

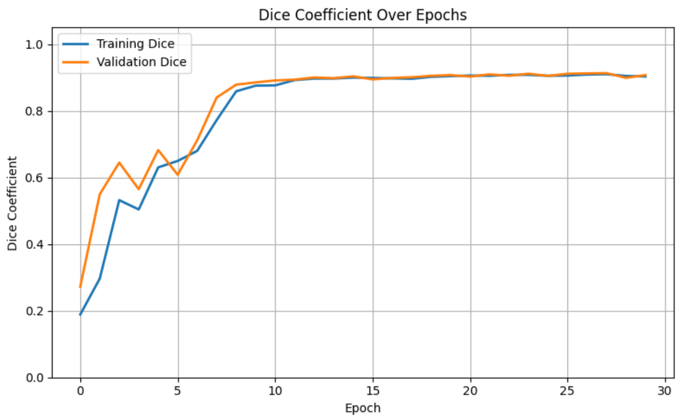

Fig. 1. Dice Coefficient over Epochs.

IV. Conclusion

The U-Net model successfully segmented cell nuclei in whole slide images from the 2018 Data Science Bowl dataset. The model's performance, measured using the Dice coefficient, showed that it could accurately detect and segment nuclei in a variety of tissue samples. Although the model performed well, there is still room for improvement, especially with respect to handling more complex images. Future work will focus on experimenting with advanced architectures, additional data augmentation techniques, and using different datasets with increased heterogeneity in backgrounds.

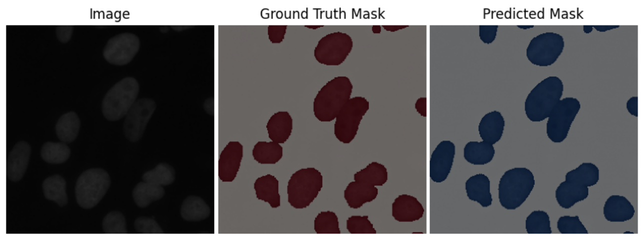

Fig. 2. Model Prediction of a WSI compared to the ground truth.

Acknowledgment

We would like to thank the creators of the 2018 Data Science Bowl dataset for providing valuable resources for this project. We would also like to acknowledge the support and resources available through Google Colab, which facilitated the execution of the training process on a cloud-based platform.

References

[1] Allen Goodman, Anne Carpenter, Elizabeth Park, jlefman-nvidia, Josette_BoozAllen, Kyle, Maggie, Nilofer, Peter Sedivec, and Will Cukierski. 2018 Data Science Bowl. https://kaggle.com/competitions/data-science-bowl-2018, 2018. Kaggle.

[2] Bharath K, "U-Net Architecture For Image Segmentation," digitalocean.com. https://www.digitalocean.com/community/tutorials/unet-architecture-image-segmentation (accessed Jul. 6, 2025)

[3] "Image Segmentation Tutorial — UNet — Oxford Pet Data — Keras Tensorflow," youtube.com. https://youtu.be/ceUvzxgyop0 (accessed Jul. 6, 2025)