ECS 174 Project Final Report

Jonathan Levitsky
Sambhav Agarwal
Sai Chamarty

1 Introduction

Biomedical image segmentation is crucial for identifying pixels in a microscopic image corresponding to cells and the background structures. This is foundational in biomedical engineering applications, which include cell tracking, disease progression and robotic surgical tools where pixel-level precision directly influences the decisions.

Before deep-learning based convolution methods were developed, researchers detected edges and segmented through scanning over small window over images to complete segmentation. This process was proven slow, inefficient, and repetitive because it lacked greater detail and had a small window. It struggled to capture more context, and failed where edges were blurry or the lighting conditions were poor.

The paper we study introduces the U-Net architecture [6], which processes the image holistically instead of patch-by-patch. U-Net's encoder-decoder structure with skip connections enables efficient and accurate biomedical segmentation, making it significantly more practical for research and clinical workflows. In this paper, we revisit the original U-Net model and study how modern optimization techniques and increased dataset size influence its performance on biomedical segmentation tasks.

2 Problem Statement and Motivation

Biomedical cell segmentation requires accurately identifying which pixels correspond to cells and which ones correspond to background structure. This task is difficult because annotated biomedical datasets are small, and cell boundaries can be overlapping. Microscopic images may contain noise, and variable illumination and brightness. Pixel-level segmentation remains important for applications in cell tracking, disease analysis, and biomedical workflows.

The original 2015 U-Net paper [6] demonstrated that an encoder-decoder architecture achieves strong segmentation performance with limited data. The model was trained under the constraints of the architecture available then, which was a very small dataset and basic gradient-descent training without much optimization. While newer architecture such as U-Net++ [8] build on the original design, it introduces additional architectural complexity, making it harder to follow its set-up and requires more compute to train.

The goal of the project is to revisit the U-net architecture and evaluate its performance improvement using modern training practices. We used techniques such as batch normalization, dropout, advanced optimizers (ADAM), and learning-rate scheduling and scaling to larger datasets. We kept the architecture unchanged and treated the original U-Net as our baseline. We aim to determine whether contemporary training methods and increased data can enhance segmentation accuracy without modifying the network design.

3 Dataset

We evaluated U-Net on biomedical microscopy datasets to understand how dataset size and image resolution can affect segmentation performance. We first implemented the U-Net paper in Python, as the original implementation was in Caffe [5], and ran it on the dataset used in the original 2015 U-Net paper (ISBI Cell Tracking Challenge [2]) for their second model task, cell segmentation. The dataset, known as PhC-C2DH-U373, they used was called "Glioblastoma-astrocytoma U373 cells on a polyacrylamide substrate" and only had approximately 40 labeled images. PhC-C2DH-U373 allows the model to distinguish cells from their background and distinguish separate cells from each other by assigning all pixels in each cell in one image the same non-zero ID and all background pixels a zero label. The dataset also requires data augmentation to have enough examples to train a generalizable model, due to the small sample set. Using the same dataset as the original publication allows us to replicate the conditions under which U-Net was developed and establish a baseline.

To study how U-Net performs with more data and greater variability, we used a bigger dataset such as the 2018 Data Science Bowl (DSB2018) [1]. This dataset contains more diverse cell shapes, imaging conditions, and annotation styles, making it useful for testing generalization and providing a bigger dataset compared to the original ISBI images.

All images are preprocessed to a consistent input shape required by the U-Net implementation, and augmentation is applied as was described by the U-net paper to increase variability in the training set. These augmentations are described in the next section. By controlling dataset size while keeping the architecture fixed, we isolate how data scaling and modern training techniques affect the performance of the model.

4 Method

Our method consists of three parts. We will first go over how we implemented the original model, then we will discuss augmentations that we made to the original model architecture, and lastly we will discuss other tasks/datasets that we applied the model to. First, we implemented the U-net model from

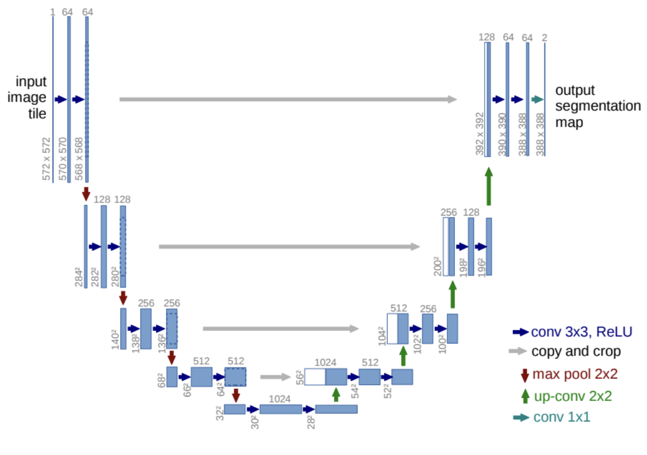

Figure 1: Layers of U-net

the 2015 paper as faithfully as possible. We chose to create a Pytorch implementation [5], to convert the architecture to a more readable version and to allow us to use the modern tools available in the Pytorch framework. Our model layers followed exactly the architecture provided, as shown in Figure 1. We implemented the model using the same momentum parameter as in the original U-Net implementation, 0.99.

Although the paper is not specific in its methods, it mentions for data augmentation that it uses elastic deformation. However, the PhC dataset comes in 696x520 resolution images, so we upsampled the second dimension to match the input size, 572, and we used elastic deformation on the horizontal dimension to create data samples of size 572x572. Then, we initialized the weights using He initialization. The original implementation used He weight initialization because it allows each feature map to start with approximately unit variance, which is ideal for ReLu-based layers.

The original paper also introduces the pixel-wise weighting scheme, described in Equation 1, to emphasize borders

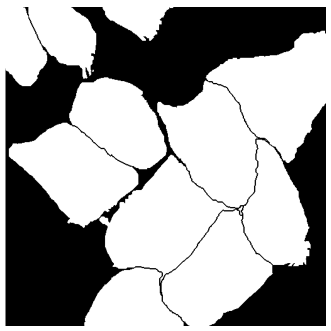

Figure 2: Segmented image

between cells. This prevents a model training on a cell segmentation dataset from merging adjacent cells into one large blob. The weighting formula increases the loss penalty in the narrow regions where two cells nearly touch, so the model learns precise separation boundaries, as shown in Figure 2.

$$w(\mathbf{x}) = w_c(\mathbf{x}) + w_0 \cdot \exp\left(-\frac{(d_1(\mathbf{x}) + d_2(\mathbf{x}))^2}{2\sigma^2}\right) \quad (1)$$

To modify the architecture that we built using Pytorch, we introduced several optimizers commonly used in modern segmentation pipelines. These changes did not alter the current architecture of U-net, but focused on improved convergence, performance and training.

Batch Normalization (BN): The 2015 U-Net architecture did not include Batch Normalization. We added BN after every convolution layer through both the encoder and decoder process. BN reduces covariant shift, stabilizes gradient, and allows the model to train with higher learning rates.

Dropout for Regularization: We added dropout layers in the bottleneck to reduce overfitting. Biomedical datasets can be small and can lead to memorization, so adding dropout helped the model generalize by preventing reliance on a particular activation.

ADAM Optimizer instead of SGD: The original U-Net used standard Stochastic-Gradient Descent with momentum. We used the ADAM optimizer, which provided adaptive learning rates and allowed it to converge faster without significant hyperparameter tuning.

Learning-Rate Scheduling (ReduceLROnPlateau): To avoid a plateau during training, we used a LR scheduler that lowered the learning rate when validation performance stops improving. This helped stabilize the dice score.

Combined Loss Function (Cross Entropy + Dice Loss): Cross-entropy may bias the model while predicting the background class. To counter this, we used a weighted combination of cross-entropy and dice loss. Dice loss is sensitive to thin boundaries and small structures, making it more optimized for cell segmentation.

After implementing the modern optimization techniques, we trained the improved U-Net on the 2018 Data Science Bowl to evaluate how these changes affect the model's performance on a larger and more heterogeneous benchmark.

Training the optimized model on this dataset allowed us to assess whether the added regularization, adaptive optimization, and loss balancing improved generalization in complex biomedical images.

5 Experimental Results

We use three key metrics to determine each model's performance. First, Dice score [3] [7] measures the overlap between prediction and ground truth. This works well for biomedical datasets because they have an extreme class imbalance, and Dice ignores the background region and focuses on the object. Dice score was not implemented in the original implementation, however we calculated it as a modern metric to compare against our other architectural implementations of U-net. Second, we used Intersection over Union or IoU score [4]. This measures the overlap as well, but is stricter than Dice, because it divides by the union of both class labels, which includes all mistakes (false positives and false negatives). Third, we use pixel accuracy, which is just a percentage of pixels correctly classified. This metric is not as good as Dice or IoU score for cell segmentation because counts background the same as the positive object, so if an image is 98% background, predicting that an image is all background would give 98% accuracy. However, it still helps us get an idea of the model's overall performance.

5.1 Original U-Net implementation

Our faithful U-net implementation does slightly worse than the original U-net paper. Our highest Dice score on our validation set occured at epoch 350 of training. It was 91.95%, and our highest IoU score was 85.21%. Our graphs can be seen in Figure 3. We believe the U-net paper does slightly better than our implementation because they used extensive cross-validation to find the train/validation split of the PhC-U373 data that showed the best results. We were unable to perform this cross validation due to a lack of compute, but we believe our results show promise that our model is an almost 1:1 reproduction of the original implementation. The training had a lot of noise, and the scores fluctuated wildly for both the training and validation between each epoch. We believe this happened because we chose a batch size of 1, and our validation set is also only 20% of the original dataset, which is approximately 5 images. Our highest Dice score epoch did not come at the very last epoch, so we implemented early stopping, through saving the earlier checkpoint with the best performance. We chose to evaluate the model's performance based on Dice score because it is the most modern score and the most reflective of the model's performance on a biomedical segmentation dataset compared to the other two scores. An example of what the model would output, in image form, compared to the original image can be found in Figure 6 in the appendix.

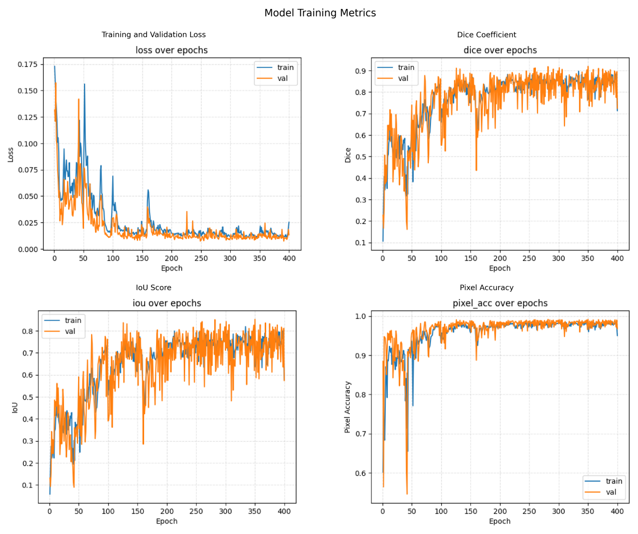

Figure 3: Evaluation of original model

5.2 U-Net With Modern Optimizers

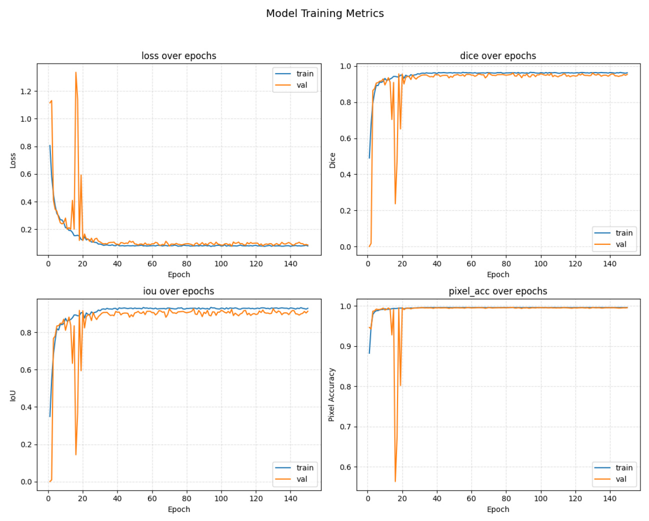

Figure 4: U-Net with Optimizers Results

We then added some of the modern optimizers such as BatchNorm, dropout, ADAM, DICE + CE Loss, and a learning-rate scheduler and trained the model over the PhC dataset. After doing so, we noticed smoother training dynamics and improved convergence. ADAM reduced the noise in the original SGD curves and reached higher DICE scores more quickly. BatchNorm stabilized gradients and allowed the model to use higher learning rates, while dropout reduced overfitting on the PhC dataset.

The combined Dice + CE loss significantly improved the boundary quality, reducing the tendency of the baseline model to merge touching cells. With these optimizations, our model achieved a Dice score of 96.2% at epoch 84, outperforming the baseline reproduction. In addition to that, the validation curves were less volatile and indicated slower overfitting and more stable learning. This shows that U-Net's historical performance had a lower ceiling due to the older training practices.

We also saw improvements in IoU scores after adding modern optimizers. The IoU is stricter than Dice because it penalizes all boundary errors, so our baseline model showed noisy IoU curves. With the optimizers, the IoU curves became smoother and reached a higher peak value, indicating better separation of touching cells and fewer false positives and false negatives.

5.3 Training over DSB2018 Dataset

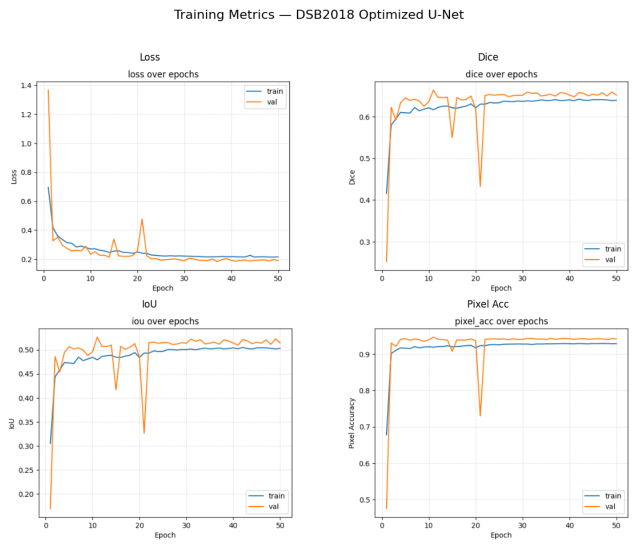

Figure 5: U-Net with Optimizers on DSB2018 Dataset Results

In addition to the PhC-U373 dataset, we trained our U-Net model on the 2018 Data Science Bowl dataset (DSB2018) to evaluate the robustness of our architecture on a larger and more diverse segmentation task. Although both datasets involve cell or nuclei segmentation, they differ in how ground-truth annotations are represented. The PhC-U373 dataset provides a single labeled mask for each image in which each cell instance is assigned a distinct integer ID (e.g., 0 for background, 1–N for individual cells). This instance-level labeling aligns with the design of the original U-Net paper, which introduced boundary-sensitive weight maps that rely on distances between separately labeled cell instances.

DSB2018 also provides instance-level annotations, but in a different format: instead of a single labeled mask, each nucleus is stored as its own binary mask file inside the image directory. To maintain consistency with our semantic-segmentation pipeline, we converted both datasets into a unified binary format by merging all non-background instance labels into a single foreground mask. This conversion allowed us to reuse the same architecture, preprocessing steps, and loss functions across datasets. However, removing instance IDs also removes explicit boundary information between individual nuclei, which makes segmentation of densely packed regions more difficult.

This difference in annotation structure—and the greater variability and complexity of the DSB2018 images—helps explain why our model achieved lower Dice and IoU scores on the DSB2018 dataset compared to the more homogeneous PhC-U373 dataset. Despite these challenges, using both datasets enabled us to evaluate how well our U-Net generalizes across different microscopy modalities and labeling formats.

6 Evaluation and Comparison

To test how well our models work outside their own training data, we evaluated them across the PhC-U373 and DSB2018 datasets. The model trained on DSB2018 performed reasonably well on the PhC validation set, reaching a Dice score of 0.7738, IoU of 0.6318, and pixel accuracy of 0.9545. This shows that training on a large, varied nuclei dataset can still produce features that transfer somewhat to phase-contrast cell images.

The opposite direction gave very low results. The optimized PhC model reached only 0.1985 Dice, 0.1231 IoU, and 0.1953 pixel accuracy on the DSB subset. The original U-Net showed similarly low scores. This happens because the two datasets come from completely different imaging styles. PhC images show whole cells with soft boundaries, while DSB2018 contains fluorescent nuclei with sharp edges and very different textures. A model trained on PhC does not learn the kind of features needed to detect nuclei in fluorescence images. A model trained on DSB2018, however, sees more variation and can adapt a little better.

Overall, U-Net generalizes well only when the datasets use similar imaging conditions. When the image appearance changes completely, the model cannot transfer what it learned. In order to get better results, our future experimentation could include U-Net ++ or Attention based CNNs or Transformers.

7 Conclusion

In this project, we revisited the original 2015 U-Net architecture and examined how modern training techniques influence its performance on biomedical image segmentation. We first reproduced the baseline model and then added optimizers such as Adam, BatchNorm, dropout, and combined Dice + Cross-Entropy loss. These updates improved training stability, faster convergence, and better boundary accuracy on the PhC dataset. The optimized model showed smoother loss curves and higher Dice and IoU scores than the baseline, revealing that many of the original paper's limitations stem from older training practices rather than the architecture itself.

When we extended our experiments to the more diverse Data Science Bowl dataset, overall performance decreased due to its higher variability. However, the optimized model still performed more robustly than the baseline, showing that modern training choices can significantly improve U-Net's resilience across challenging datasets. Overall, our results highlight the importance of optimization strategies in maximizing U-Net's effectiveness for biomedical segmentation tasks.

Contributions

Sambhav: Helped with the initial implementation of the U-Net Baseline architecture in PyTorch. Designed, experimented, and integration optimization techniques (BatchNorm, Dropout, ADAM, Dice + CE Loss, LR Scheduling, etc) and training the optimized model on the PhC Dataset. Wrote section 1, 2, and part of section 3. Wrote section 5.2 and helped with section 7. Created drafts of Slides, and helped with training and debugging.

Jonathan: Replicated the entire initial Pytorch implementation of the original U-net architecture, including the data processing, augmentation, loss function, weight map computation, actual model architecture, training loop, and result visualization. Created an inference pipeline for the original model. Wrote part of Section 3, everything in Section 4 up to where the Batch Normalization part begins, and all of the introduction for Part 5 and all of Part 5.1. I also added an example of the inference pipeline running in the Appendix.

Sai: Contributed to the reproduction of the baseline U-Net architecture and assisted in debugging the initial training pipeline. Identified, prepared, and integrated the larger Data Science Bowl 2018 dataset into the project. Implemented preprocessing for multi-mask images, trained the model on the expanded dataset, and evaluated performance differences across datasets to assess generalization and robustness.

8 GitHub and Demo Link

Our project repository is available at: `https://github.com/JonathanPLev/Resnet-ReImplementation`. Our demo is available at: `https://www.youtube.com/watch?v=fpBJSmlubyg`.

9 References

[1] 2018 data science bowl. `https://www.kaggle.com/competitions/data-science-bowl-2018/`. Accessed: 2024-12-02.

[2] Isbi cell tracking challenge dataset. `https://celltrackingchallenge.net/`. Accessed: 2024-12-02.

[3] Dice, L. R. Measures of the Amount of Ecologic Association Between Species. *Ecology 26*, 3 (July 1945), 297–302.

[4] Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. *CoRR abs/1411.4038* (2014).

[5] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019.

[6] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)* (2015).

[7] Sørensen, T., Sørensen, T., Biering-Sørensen, T., Sørensen, T., and Sorensen, J. T. A method of establishing group of equal amplitude in plant sociobiology based on similarity of species content and its application to analyses of the vegetation on danish commons.

[8] Zhou, Z., Siddiquee, M. M. R., Tajbakhsh, N., and Liang, J. Unet++: A nested u-net architecture for medical image segmentation, 2018.

10 Appendix

An example of our inference pipeline output, compared to the original image, is displayed below.

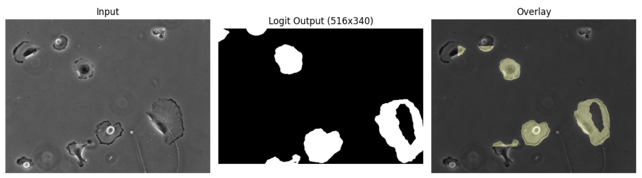

Figure 6: Inference Example