Final Project: Segmentation in Medical Imaging

ECSE 415
April 9th, 2025

Radu Petrescu (261051351)
Yassine Meliani (261035296)
Shady Guindi (261026626)
Michael Popescu (261055784)
Sameer Karam (260976154)

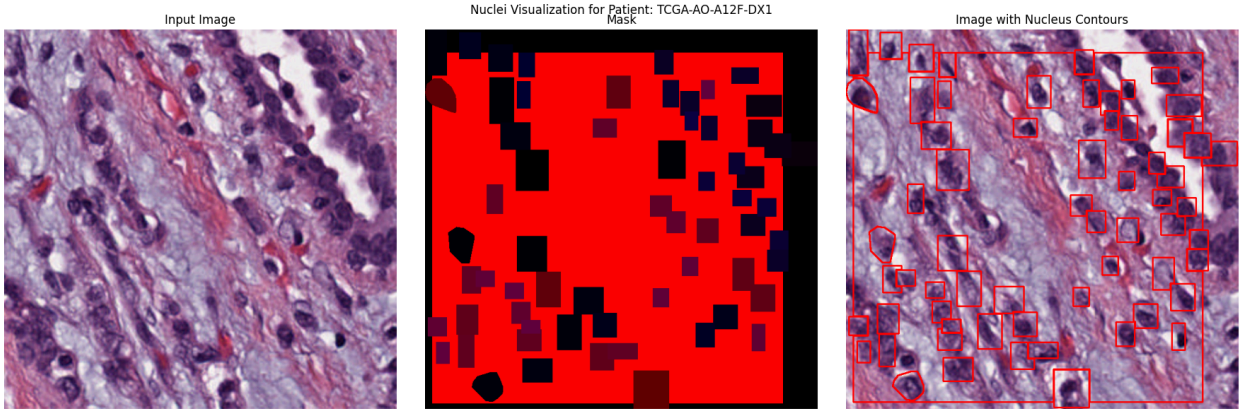
Dataset

The dataset is a collection of high-resolution histopathology images and associated annotations, organized for cell and nucleus segmentation tasks. After unzipping, the dataset which can be found fully here, contains three main folders:

- **rgb**: contains the input images at a resolution of 0.2 microns-per-pixel.
- **csv**: includes annotation files with coordinate values in pixel units.
- **mask**: holds corresponding segmentation masks. The first channel represents class labels, while the product of the second and third channels provides a unique instance label for each nucleus. Field-of-view (FOV) areas are encoded in the first channel as gray regions.

Filenames follow a structured naming convention, where each patient is uniquely identified by a code like TCGA-A1-A0SP-DX1. Multiple samples (images) may belong to the same patient, as indicated by this unique identifier embedded in the filenames.

Figure 1: Visualization of the original image, mask and nuclei labels for patient TCGA-AO-A12F-DX1



Segmentation

Unsupervised Segmentation

Method

For the unsupervised segmentation task, we implemented K-mean clustering to assign a class label to each pixel for a given input image. We performed a K classification of pixels, which we chose not to reduce to a binary segmentation of nuclei/non-nuclei (K=2). To accomplish and evaluate the performance of this task, we defined several functions in order to extract features from an image, calculate Dice scores, convert rgb masks to label masks, and calculate Dice scores from our output clusters.

To construct an output, we first loaded the input image and extracted its features, including rgb intensities, pixel position, and Gabor filter responses for σ=1, 3, 5 and a kernel size of 21. We normalized these features using Z-score normalization so that their mean is 0 and standard deviation is 1, such that they are each weighed equally when calculating distance for K-mean. With the features extracted, we used opencv's kmeans() method with our desired K and input features to obtain a one-dimensional vector of labels. Finally, we resized this vector back into the original input image's shape, and displayed it using matplotlib, assigning each label a color with equidistant saturation levels for contrast.

To measure an output's performance, we loaded the corresponding rgb mask, resized the rgb mask using nearest interpolation to the same dimensions as the input, and converted its unique colors into distinct zero-indexed integer labels. Then, for each unique label in our output, we computed the corresponding Dice score for each existing ground truth label, choosing the highest one without replacement, and then taking the mean across Dice scores. If the number of unique predicted labels K exceeded the number of unique ground truth labels, the associated Dice score of any surplus labels was assumed to be 0, lowering the mean. Conversely, if the number of ground truth labels exceeded the number of predicted labels K, some pixels in the image would never be counted as TP, lowering the individual Dice scores, and as a result, the mean.

Using these Dice scores, we measured the performance of all segmentations of images of subjects within train_subjects.csv for K = 3, 4, 5, computing the mean Dice score across all images for each K.

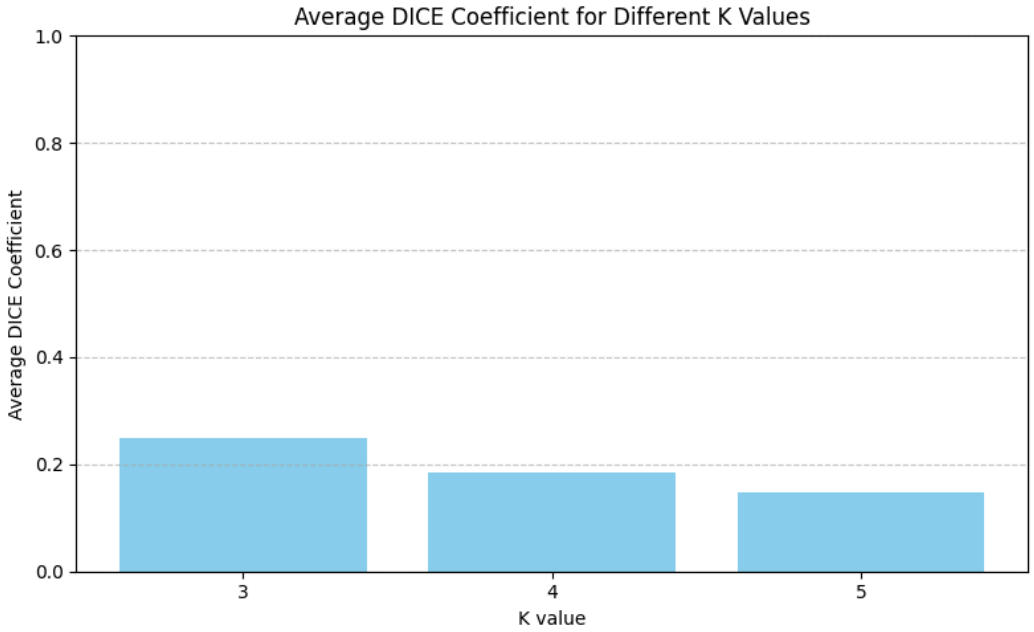
Afterwards, we explored the performance of first grayscaling our input images, then extracting relevant features, keeping the rest of the steps identical. As a result, instead of rgb intensities, our features therefore included pixel intensities only.

Finally, we compared K-mean to another unsupervised segmentation method, Mean Shift. For this, we utilized sklearn's cluster module, making use of the MeanShift class and estimate_bandwidth method to estimate a valid bandwidth for each image. Since the input images had dimensions hovering around 300px wide and tall, the number of pixels within an image regularly reached 100,000. Performing Mean Shift on 100,000 pixels, each with 29 extracted features, was not feasible. Hence, after attempting different amounts (30%, 10%, 1000px, 200px, 100px, 10px), we settled on random sampling 200 pixels from an input, performing Mean Shift to find the corresponding centroids associated with those 200 pixels, and then predicted which cluster each pixel in the original input would be grouped into based on these aforementioned centroids. We selected 200 pixels as an amount based on how much time it took for Mean Shift to find the centroids, as well as visual clarity.

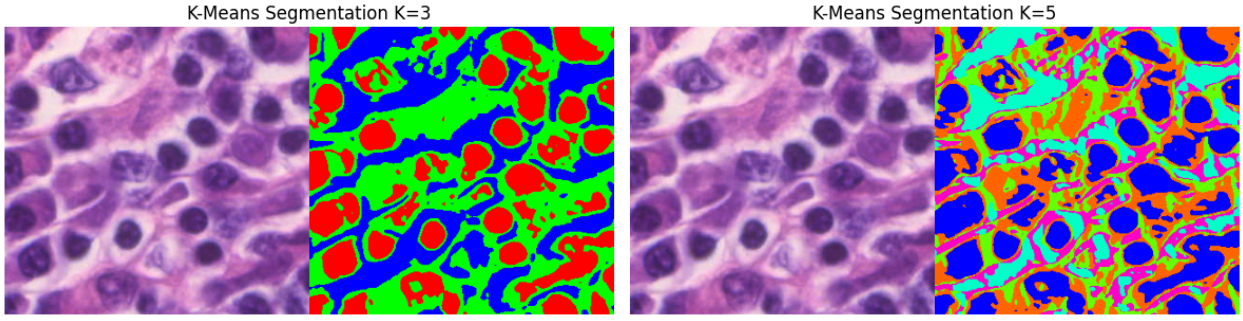
We did not perform Mean Shift segmentation on all training images, hence we could not provide a mean Dice score for the method. We have, however, repeatedly compared the performance of Mean Shift and K-mean on randomly sampled images using Dice scores, in order to verify whether one is strictly better than the other.

Performance Report

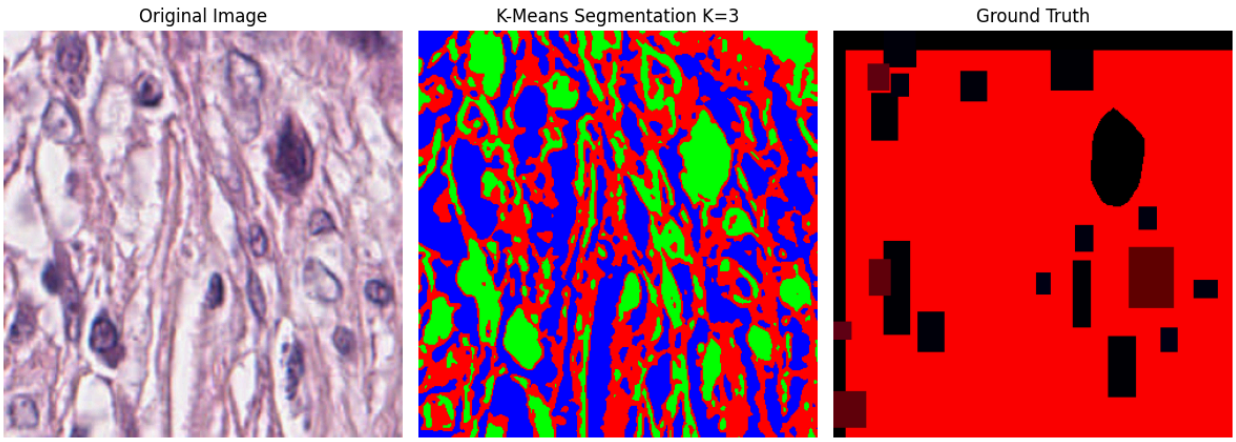
The optimal K value was found to be K=3 with a Dice score of approximately 0.25, with 0.18 for K=4 and 0.15 for K=5. It appears the bigger the K, the lower the Dice score. This could be in part attributed to the way we compute Dice scores, requiring each label of our output to correspond to a label of the ground truth.



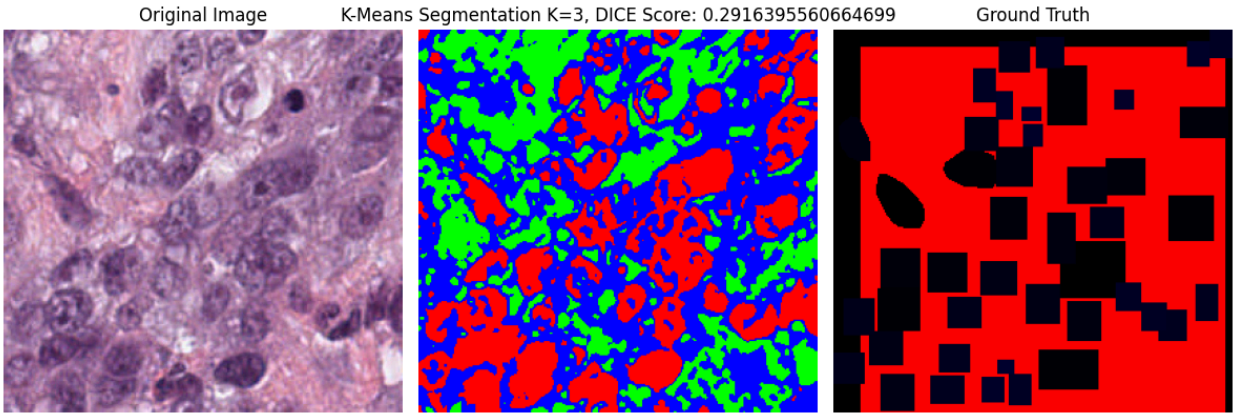
Indeed, it may be a flaw in our design to compute Dice scores so stringently. We are however convinced after reviewing the results visually, that higher K values do not provide any benefit to the segmentation, and instead are more likely to introduce noise to the output image, adding detail where none is needed (or where none is represented in the ground truth). We believe our method of measuring performance is still significant, especially when applied the same way to the different hyperparameters or techniques.

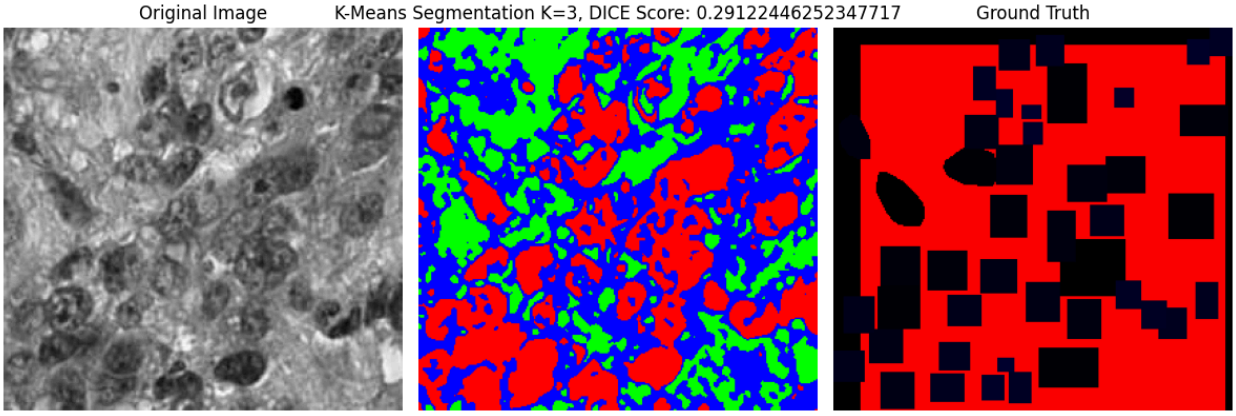


Qualitatively, the nuclei in resulting segmentations correspond well to a visual evaluation of the input image, moderately to a binary classification of the ground truth (background/nuclei), but poorly in terms of labels. Nuclei are hardly differentiated from one another. Most often, of the three labels, one is dedicated to all nuclei, another to background cytoplasm, and the last to a different shade of the cytoplasm. This is well within our expectations for the performance of a segmentation method, they are not meant to classify nuclei.



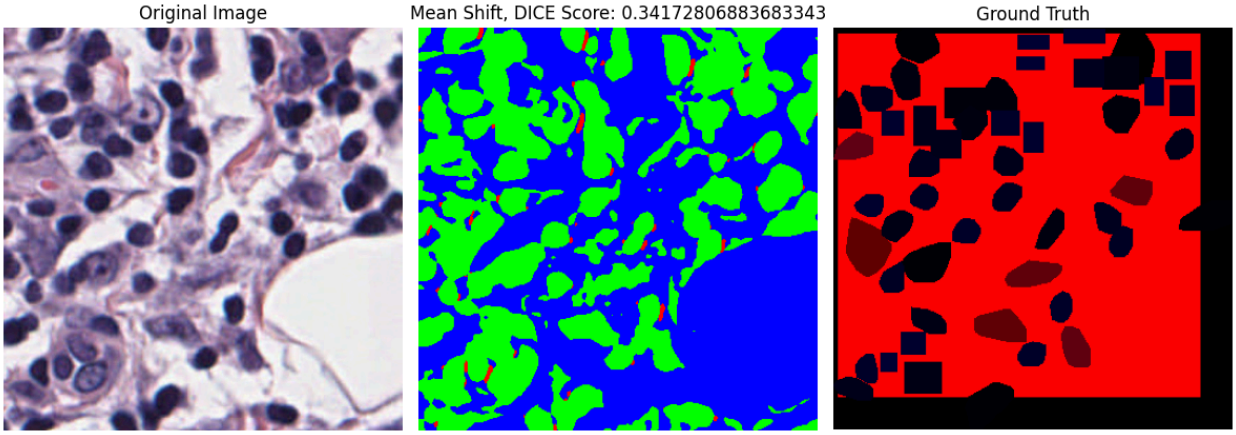
Grayscaling images before extracting features did not present any clear advantage in performance. Sometimes, the resulting Dice score would be higher compared to rgb, other times, lower. Sometimes, the Dice scores would be equivalent.





To verify overall performance, we computed the mean Dice score of grayscaling all training set images, and found the resulting mean Dice score to be within 0.01 of the mean Dice score of rgb K=3, confirming our initial observations. We hypothesize that this similarity in Dice scores could be attributed to pixel intensities representing only a small fraction of the feature space, and so while those specific features might change, pixel position and Gabor responses, which were already based on grayscale images, remain identical between the two input types.

Finally, Mean Shift yielded a similarly inconclusive comparison. Based on repeated iterations, calculated Dice scores, and visual interpretation, we concluded that while slower, Mean Shift was still very capable of the same efficacy as K-mean. We often observed some clusters grouping a very small number of pixels, which we theorize is a result of randomly sampling pixels, and randomly finding a centroid that is very far from most other pixels in the input image, except a select few.



In conclusion, K-mean segmentation is a somewhat visually reliable segmentation method for nuclei, but is unable to distinguish between them to classify their nature. We found that the ideal K was 3, and that grayscaling the inputs did not present any useful increase in Dice score, oscillating between no change, better for some images, and worse for others. We implemented Mean Shift segmentation using sklearn's cluster module, and found its performance to be comparable to that of K-mean, besides the presence of outlier labels within the resulting output.

Supervised Segmentation

Method

For the supervised segmentation task, we trained a Random Forest classifier to assign a class label to each pixel in grayscale histopathology images. Ground truth labels were generated from polygon annotations provided in the dataset's CSV files, with each pixel inside a polygon assigned the corresponding nucleus type. We explored two approaches for feature representation: a pixelwise model using raw grayscale intensity, and a patch-based model where each pixel was represented by the surrounding grayscale values in a fixed-size window (e.g., 9×9 patch). This patch-based representation allowed the model to capture local texture and spatial context, which is critical for distinguishing visually similar cell types.

To manage memory constraints and improve generalization, we divided the training data by patient ID and trained an ensemble of 10 Random Forest models, each on a separate chunk of the dataset. Class imbalance, especially the overrepresentation of background pixels, was addressed by downsampling the dominant classes to balance the training set. During inference, predictions from the 10 models were

averaged, and the final class for each pixel was selected via majority vote. For the patch-based model, post-processing was applied to improve the spatial consistency of predictions, such as smoothing the output using connected components and majority filtering. This approach allowed us to evaluate the trade-off between raw per-pixel accuracy and more coherent, biologically plausible segmentation.

Performance Report

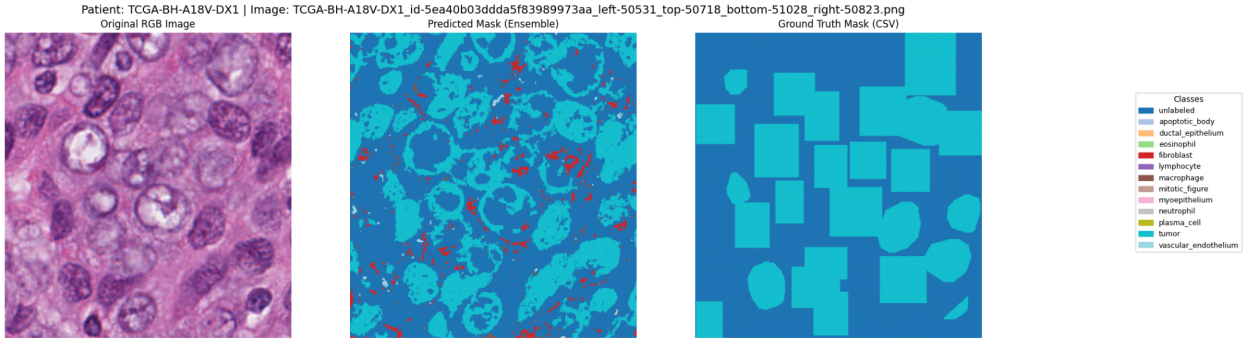
On the training set, the supervised segmentation model performed well, particularly for binary cell detection. The pixelwise model achieved a low binary misclassification rate of 0.124, indicating strong capability in distinguishing nuclei from background. Part of the reason for this was that not every cell was actually labeled in the ground truth, and yet our model would segment it. However, its ability to accurately classify different cell types was limited, with a much higher multiclass misclassification rate of 0.673. Introducing grouping techniques to enforce spatial coherence led to notable improvements in label accuracy, reducing the multiclass misclassification rate to 0.346, while maintaining decent binary segmentation performance at 0.124.

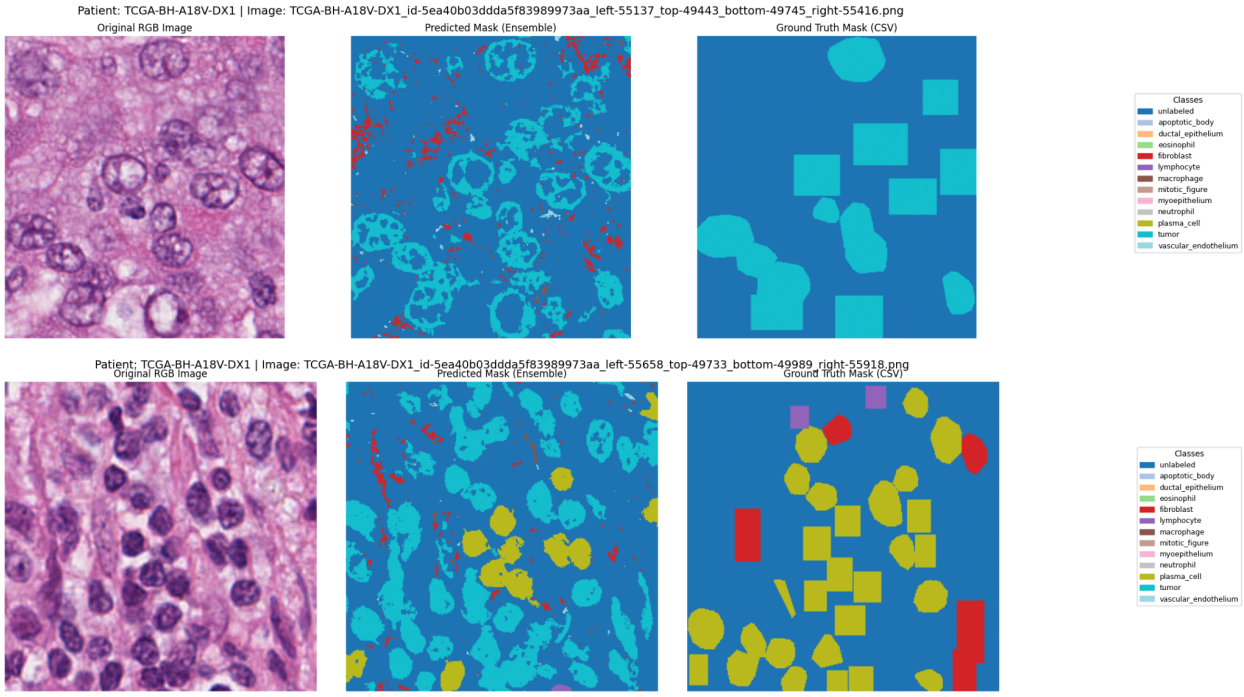
The best training performance overall came from the patch-based Random Forest model. By leveraging local texture and intensity patterns in a 9×9 grayscale window around each pixel, it significantly improved classification accuracy. The model achieved a multiclass misclassification rate of 0.277, outperforming the pixelwise models in identifying specific cell types. The tradeoff was a modest increase in binary misclassification to 0.194, likely due to over-segmentation or mislabeling in denser regions. Still, this approach provided the best balance between structure detection and class specificity on the training data.

On the validation set, the same trends held with slightly diminished performance, as expected. The pixelwise model remained reliable for binary detection, with a binary misclassification rate of 0.167, but again struggled with multiclass segmentation, reaching a label misclassification rate of 0.692. When grouping was applied, the model's multiclass accuracy improved considerably, lowering the misclassification rate to 0.368, while maintaining binary robustness at 0.167.

The patch-based model continued to outperform the others on validation, achieving a multiclass misclassification rate of 0.312. While its binary misclassification rate increased to 0.202, it still effectively separated nuclei from background. These results confirm the patch-based approach's ability to generalize well and strike the most effective compromise between detecting nuclei and accurately classifying their type in unseen data.

Supervised vs Unsupervised



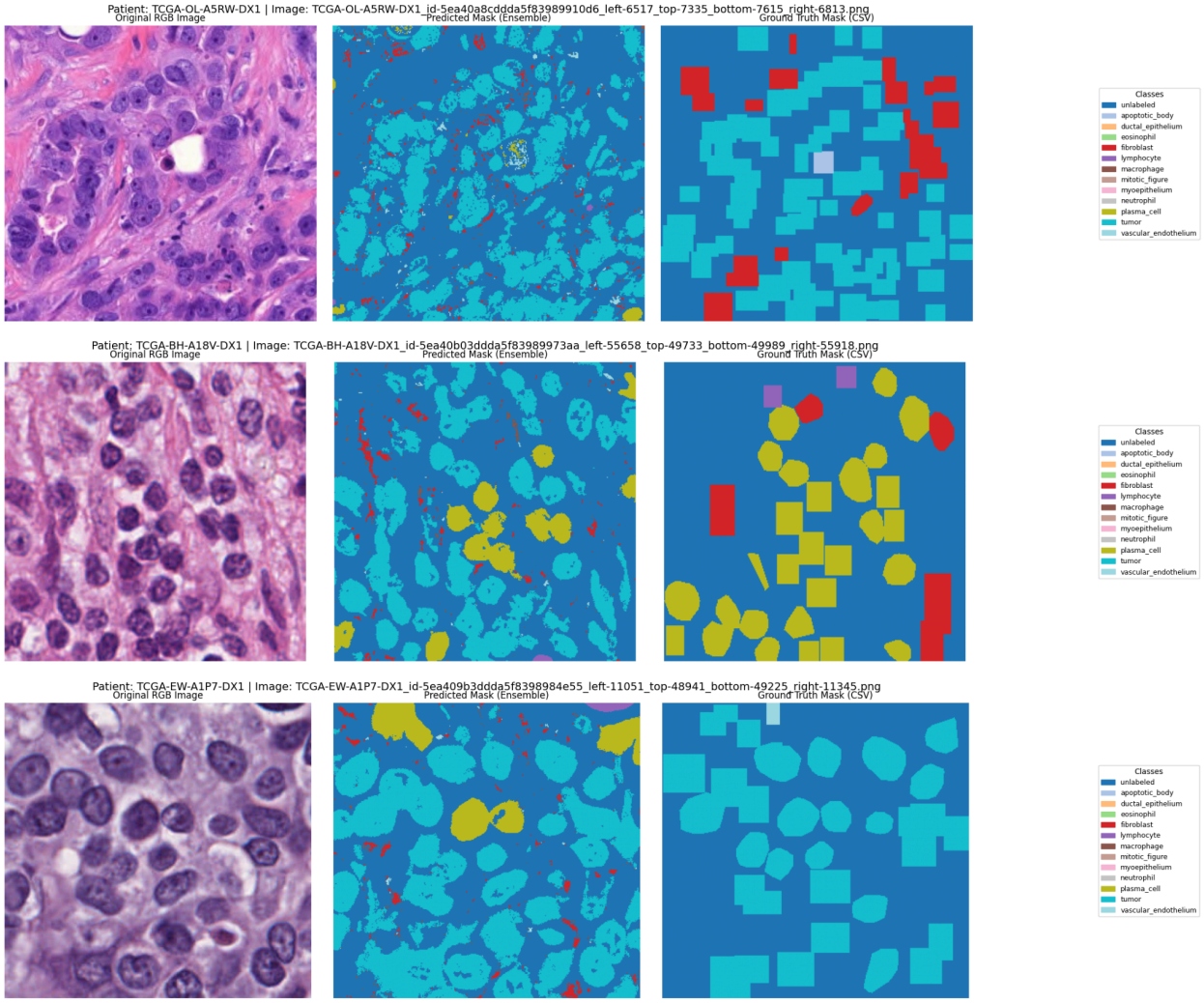


(i) Qualitative Comparison:
Upon visual inspection of the predicted masks, the supervised method consistently produces more coherent and biologically plausible segmentations compared to the unsupervised K-means approach. In particular, nuclei boundaries are more sharply defined, and overlapping or adjacent cells are more accurately separated. The supervised model also shows better consistency in recognizing specific cell types across different patches.

(ii) Quantitative Comparison:
Using misclassification rate as the metric, the supervised model shows significantly better agreement with ground truth annotations across the three validation samples, with label-wise misclassification rates of 0.31, 0.34, and 0.76, and corresponding binary misclassification rates of 0.12, 0.11, and 0.10. In contrast, the unsupervised K-means method with k=3 achieved Dice scores of 0.341, 0.291, and 0.291, reflecting its comparatively weaker segmentation performance. The supervised model also avoids over-segmentation and inconsistent labeling that often affects K-means, and overall exhibits better cell boundary precision and fewer false positives, particularly in complex tissue regions. While not a perfectly direct comparison, the gap in consistency and structure makes the superiority of the supervised approach clear.

Conclusion:
The supervised method outperforms unsupervised segmentation both qualitatively and quantitatively. This is expected, as supervised models learn from annotated examples and can generalize better to unseen data. Unsupervised methods like K-means, on the other hand, rely purely on intensity and color clustering, without understanding of cellular context or spatial patterns.

Nuclei Segmentation



Based on our visualizations, tumor cells are the easiest nuclei group to segment, see here in blue. These cells typically form large, densely packed clusters with prominent and relatively consistent color and texture. As a result, both the supervised model and even the unsupervised K-means method often segment these cells with high accuracy and clarity.

Why?

Tumor nuclei generally have more distinct morphological and staining characteristics compared to other cell types in histopathological images. They tend to be larger, darker, and more uniform, making them stand out clearly against the background. Their prevalence in training data also gives the supervised model more examples to learn from, resulting in better generalization. In contrast, rare or more ambiguous cell types (macrophages or apoptotic bodies) are less visually distinct and more prone to misclassification.

Tumor Detection: Pipeline

Converting Images and Masks to Grayscale

The first step in our tumor detection framework involves converting the RGB pathological images to grayscale format to simplify processing. The `convert_to_grayscale` function efficiently handles this conversion, accepting either directory paths or direct file paths to both the original images and their corresponding masks. This flexibility allows for batch processing of multiple images or focusing on specific files of interest. The function returns dictionaries containing the grayscale versions of both images and masks, keyed by their filenames for easy reference. This standardization to grayscale simplifies subsequent processing steps while preserving the essential structural information needed for tumor detection.

Applying Masks to Extract Regions of Interest

After obtaining grayscale representations, we isolate regions of interest through our `apply_masks` function, which multiplies the grayscale image with an inverted version of the mask. The inversion is a critical step—we scale the mask to have a maximum value of 255 and then invert it (255 - mask_scaled), transforming white areas (255) to black (0) and dark areas to light. This inverted mask is then multiplied with the grayscale image, effectively preserving only the pathological structures of interest while eliminating irrelevant background. The function carefully handles potential dimension mismatches between images and masks through appropriate resizing operations. The result is a collection of masked ROI images that focus exclusively on potentially cancerous tissue regions.

Removing Background Noise

The masked images often contain noise that can interfere with accurate feature extraction. Our enhanced `remove_background_noise` function addresses this using non-local means denoising, which is superior to simpler methods like Gaussian blur as it better preserves important structural details. This approach works by comparing small patches across the image and averaging similar ones, effectively reducing noise while maintaining edges and textures. The function includes configurable parameters: `denoise_strength` controls the aggressiveness of noise removal, while `threshold_percentage` allows for elimination of pixels below a certain intensity threshold relative to the maximum. This thresholding capability is particularly valuable for removing low-intensity artifacts that might otherwise be misinterpreted as meaningful structures. The function carefully preserves the original data type and range of the images, ensuring no information is lost during processing.

Extracting SIFT Features from Pathological Structures

Feature extraction forms the core of our approach, now leveraging Scale-Invariant Feature Transform (SIFT) for more robust results. The `extract_sift_features` function identifies distinctive keypoints in the denoised images and computes descriptors that characterize the local appearance around each keypoint. These SIFT features are particularly valuable because they remain consistent despite variations in scale, rotation, and illumination—common challenges in pathological images. The function allows control over various SIFT parameters including the number of features to retain, contrast threshold for filtering weak features, and edge threshold for eliminating edge-like features. This approach provides a rich representation of the textural and structural characteristics of potential tumor regions, enabling more accurate classification in subsequent steps.

Creating Training and Validation Datasets

To build comprehensive training and validation datasets, our `create_train_val_datasets` function processes images through the entire pipeline for subjects listed in the train and validation split files. For each image, we extract SIFT features and determine whether each keypoint falls within a tumor bounding box by referencing the annotation CSV files. Features are labeled as tumor (1) or non-tumor (0) based on this spatial analysis. The function maintains detailed metadata including counts of tumor and non-tumor features per image, which is valuable for analysis and debugging. The resulting datasets consist of feature vectors and their corresponding binary labels, carefully balanced to represent both tumor and non-tumor structures. This systematic processing ensures our classifier is trained on representative samples while maintaining the subject-wise split between training and validation sets.

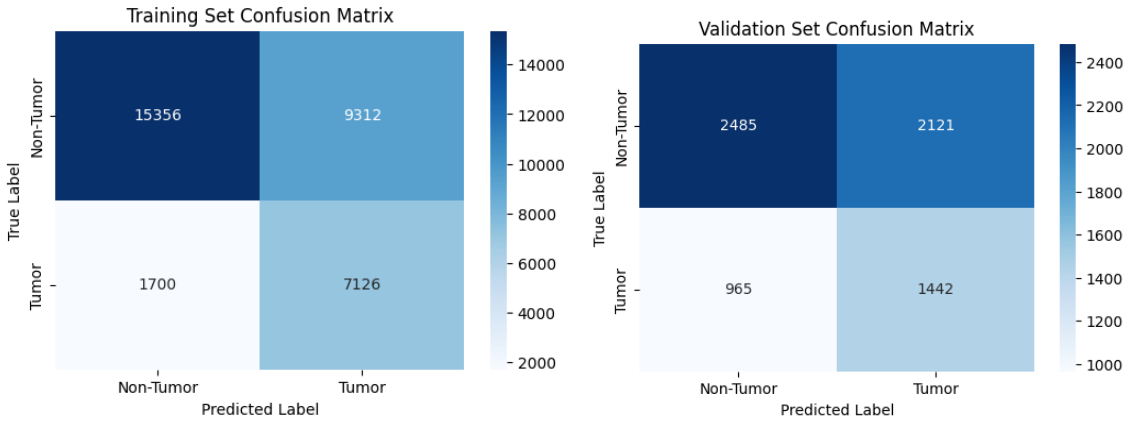
Analyzing Classification Performance

The pipeline concludes with saving the processed data, including feature vectors, labels, and metadata for both training and validation sets. Class distribution statistics are reported, showing the percentage of tumor versus non-tumor features in each set. This information is critical for assessing potential class imbalance issues that might affect model training. The saved datasets can then be used to train various machine learning models, with particular emphasis on Support Vector Machines (SVMs) which have shown strong performance in similar biomedical classification tasks. The modular design of our pipeline allows for easy experimentation with different classifiers and hyperparameters to optimize detection performance.

Tumor Detection: Classification, Counting, and Evaluation

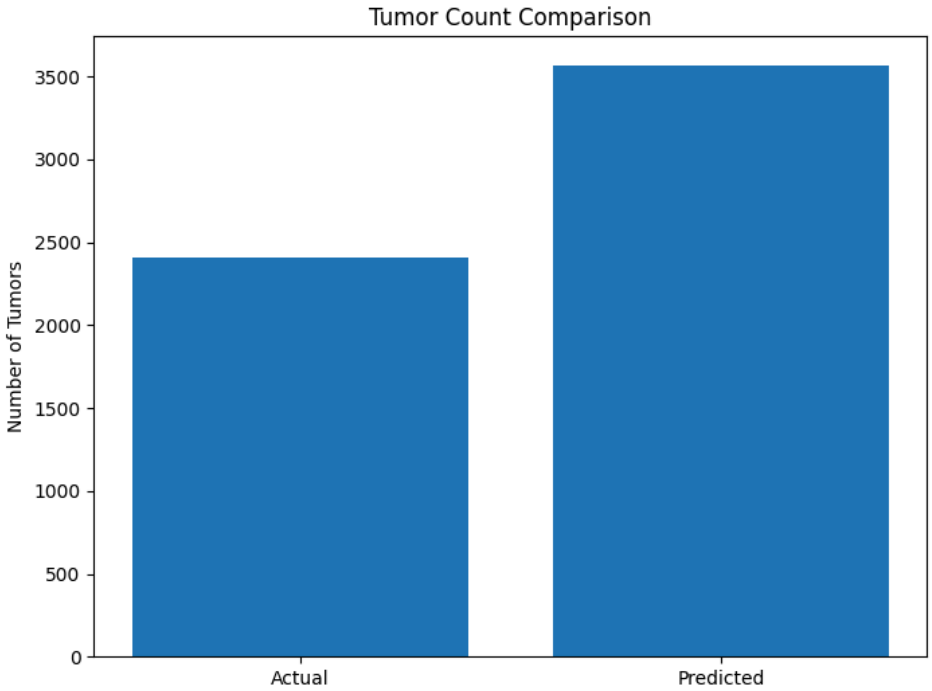
Classification Model Performance

To detect tumors from pathological regions, we trained a supervised SVM (Support Vector Machine) classifier using blob-level features extracted from masked regions of histopathology images. The dataset was split into training and validation sets, and we evaluated the classifier using accuracy, precision, recall, and F1 score. On the training set, the model achieved an accuracy of 67.12%, balanced accuracy of 71.49%, precision of 43.35%, recall of 80.74%, and an F1 score of 56.41%. For the validation set, the accuracy was 56.00%, with balanced accuracy of 56.93%, precision of 40.47%, recall of 59.91%, and F1 score of 48.31%. These results indicate the model has reasonable recall but poor precision, as it frequently misclassifies non-tumor regions as tumors, shown by the significant number of false positives (9,312 in training, 2,121 in validation).



Tumor Count Evaluation Using RMSE

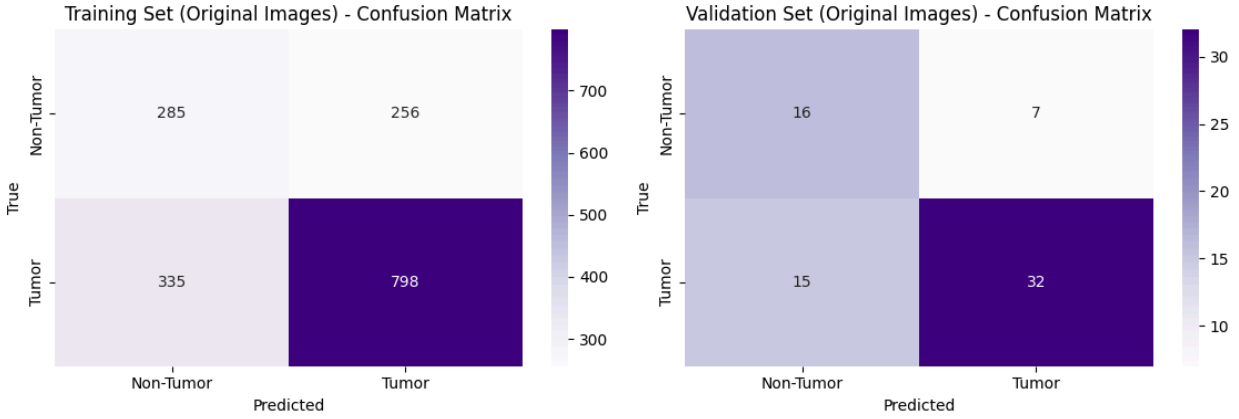
After performing tumor detection on the validation set, we computed the total number of predicted tumors for each image and compared it to the ground-truth tumor count using Root Mean Squared Error (RMSE). The actual tumor count was 2,407 while the predicted tumor count was 3,563, with an absolute difference of 1,156. The computed RMSE was 1156.00, indicating a substantial overestimation in tumor detection. This discrepancy is likely due to the high false positive rate observed in the confusion matrices, as the model consistently predicts more tumors than actually exist. This highlights significant room for improvement in the model's ability to accurately distinguish between tumor and non-tumor regions.



Original vs. Masked Image Classification

To explore the impact of using full unmasked images instead of manually segmented pathological regions, we trained a second classifier using features extracted from the entire original images. When comparing results between the two approaches, we found that the original images model performed better than the

masked model. Specifically, on the training set, the original images classifier achieved an accuracy of 64.70%, precision of 75.71%, recall of 70.43%, and F1 score of 72.98%. On the validation set, it reached an accuracy of 68.57%, precision of 82.05%, recall of 68.09%, and F1 score of 74.42%. These values are higher than those observed in the masked approach, suggesting that using the full image context provides better discriminative power for the classifier. The confusion matrices for the original images model show fewer false positives proportionally, though the sample sizes are much smaller (1,674 training samples, 70 validation samples).



Discriminative Structures Based on Features

Based on the feature set used in the detection models, there are clear differences in how well certain tissue structures can be discriminated. The original images model shows stronger overall performance, likely because it can leverage broader contextual information and structural patterns that may be lost when focusing only on masked regions. In the original images validation set, only 7 non-tumor samples were misclassified as tumors (out of 23), compared to 15 tumor samples incorrectly labeled as non-tumors (out of 47), suggesting better specificity than sensitivity. This contrasts with the masked model, which shows poor precision due to its tendency to overclassify regions as tumors. The difference in sample sizes between the two approaches (over 24,000 samples for masked training vs. around 1,600 for original images) also suggests that the quality of features may be more important than quantity for this classification task.

Bonus

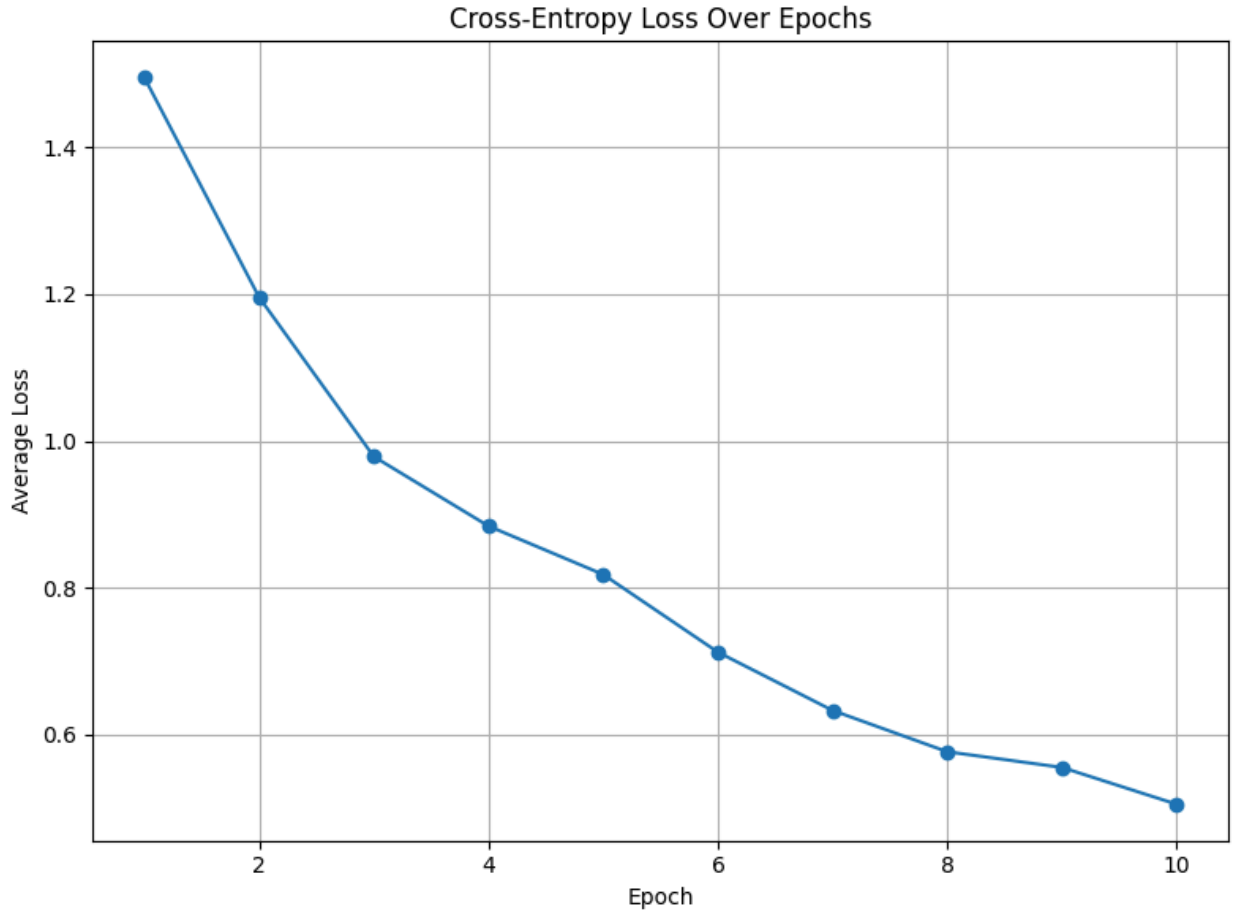
Supervised learning

For this project, we chose to use the Fully Convolutional Network (FCN) with a ResNet-50 backbone, specifically the fcn_resnet50 model available in PyTorch's torchvision library. This model was pre-trained on the COCO dataset [1] and is well-suited for pixel-wise prediction tasks like semantic segmentation. We selected this architecture because it strikes a good balance between accuracy and computational efficiency, and its residual connections help preserve gradient flow during training, which is particularly useful when fine-tuning on medical image data that may differ significantly from the pre-training domain.

To adapt the model to our specific task of multi-class nucleus segmentation, we modified the final layer of the FCN classifier to output predictions for five classes instead of the original set. This allowed the model to differentiate between different types of nuclei and the background. By leveraging a pre-trained model, we were able to benefit from transfer learning, enabling faster convergence and better generalization, especially given the limited size of our training set.

We trained the model using cross-entropy loss, a standard choice for multi-class classification problems, and optimized it with the Adam optimizer using a learning rate of 1e-4. The training loop ran for 10 epochs, during which the model steadily improved. The average loss decreased from 1.5432 in the first epoch to 0.4876 by the tenth, indicating that the model was successfully learning to segment the nuclei with increasing accuracy. This consistent decline in loss suggests that our choice of model and training strategy was effective. To better visualize this learning progress, we plotted the average cross-entropy loss at each epoch, as shown in Figure 2. The figure clearly illustrates a smooth downward trend in loss over time, confirming that the model converged well during training.

Figure 2: Training loss over epochs



Inference

We evaluated the performance of our trained FCN-ResNet50 segmentation model on a held-out test set consisting of 8 unseen samples. This evaluation was done using three commonly used metrics for

semantic segmentation tasks: Dice coefficient, Intersection over Union (IoU), and pixel-wise accuracy. These metrics provide a more complete understanding of how well the model is able to segment different classes in the images.

The results showed that the model achieved an average Dice score of 0.5934, an average IoU of 0.1432, and a pixel accuracy of 0.8375 across all five classes. Among the classes, the model performed best on the AMBIGUOUS (Class 0) and tumor (Class 4) categories, with Dice scores of 0.7073 and 0.6881, respectively. These high scores suggest that the model was able to capture the morphology of ambiguous nuclei and tumor regions with reasonable accuracy.

On the other hand, performance was significantly lower on the stromal (Class 1) and sTIL (Class 3) categories, with Dice scores of 0.0927 and 0.4791, respectively. Particularly concerning is the stromal class, where the IoU was only 0.0510, indicating that the predicted and ground truth regions had very little overlap. Interestingly, the other category (Class 2) reported a Dice score of 1.0000 but an IoU of 0.0000, which suggests that the model did not detect any overlapping pixels but predicted all as background, leading to a misleading Dice result due to class imbalance.

Despite some class-specific inconsistencies, the model demonstrated overall reliable performance in segmenting the most clinically significant regions. These results indicate a promising starting point, though there is room for improvement, particularly in addressing class imbalance and increasing precision on underrepresented classes.

References:

[1] PyTorch, "*FCN_ResNet50 — Torchvision Main Documentation*,"
https://pytorch.org/vision/main/models/generated/torchvision.models.segmentation.fcn_resnet50.html
(accessed Apr. 8, 2025).