M.S. CAPSTONE PROJECT REPORT

Integrating Active Contour Models with Attention Mechanisms in Deep Learning-Based Medical Image Segmentation

Shaira Alam*
M.S. Student
University of California, Los Angeles
Department of Computer Science
shairaalam@g.ucla.edu

Vaishnavi Manthena*
M.S. Student
University of California, Los Angeles
Department of Computer Science
vm504@g.ucla.edu

Abstract

Medical image segmentation is essential for accurate diagnosis and treatment planning, particularly in identifying anomalies in CT and other imaging modalities. Our objective is to develop a deep learning-based segmentation framework that combines Active Contour Models (ACMs) and attention mechanisms, thus advancing automated medical image segmentation for improved clinical decision-making. By integrating ACMs with Convolutional Neural Networks (CNNs) and edge detection techniques, our approach enhances boundary precision and segmentation accuracy in medical images. Moreover, we investigate the novel idea of fine-tuning ACM hyperparameters by learning them within the CNN and backpropagating loss through the ACM. Our methodology addresses challenges such as weak boundaries, noise, and variability in anatomical structures, contributing to more robust and interpretable medical image analysis. This paper develops a prototype hybrid model and thoroughly evaluates it on multiple medical images.

1. Introduction

Medical image segmentation is a critical task in medical diagnostics, enabling the precise delineation of anatomical structures and abnormalities in imaging modalities such as CT, MR, and ultrasound (US). However, segmentation remains challenging due to the complexity of medical images, which often exhibit irregular shapes, subtle boundaries, and significant noise. Additionally, effective segmentation often requires models trained on domain-specific datasets, making generalization difficult. Various approaches exist for medical image segmentation, each offering distinct strengths and limitations.

Deep learning-based segmentation models have gained prominence due to their ability to learn complex patterns from medical images. Notable architectures include U-Net and SegNet, which are widely used in medical image segmentation due to their encoder-decoder structures that help retain spatial information. Attention mechanisms, when integrated with CNNs, can further enhance the model's ability to focus on fine details, leading to more precise segmentation. The choice of loss function plays a crucial role in optimizing performance, with common functions such as Dice Loss, Intersection over Union (IoU), and Binary Cross-Entropy (BCE) tailored to handle class imbalance and segmentation accuracy.

Edge-based segmentation relies on detecting strong gradients to define object boundaries, but it may struggle with weak edges and noisy images. Recent vision-language models, such as the Segment Anything Model (SAM), leverage pre-trained transformers for zero-shot segmentation, though they may require fine-tuning for domain-specific tasks. Hybrid techniques combine traditional segmentation approaches with deep learning, offering improved accuracy, especially when dealing with weak boundaries or noisy data.

Active Contour Models (ACMs) offer an alternative approach to medical image segmentation by iteratively refining contours to fit object boundaries. ACMs provide several advantages, including adaptive boundary detection that dynamically adjusts contour points to capture complex and irregular shapes. They also allow for the integration of prior knowledge by fine-tuning energy terms based on domain-specific constraints, enhancing segmentation robustness. Unlike purely pixel-intensity-based meth-

*Equal contribution

ods, ACMs incorporate internal energy constraints, making them more resilient to image noise. Variants, such as region-based ACMs, further enhance flexibility. ACMs have proven versatile across various imaging modalities, including CT, MRI, and US, and are valuable for tasks such as organ boundary detection and tumor segmentation.

Our work introduces a novel approach to CNN-based segmentation for medical image datasets by integrating Active Contour Models (ACMs) to enhance boundary precision. A key innovation is the hybrid ACM model, which not only leverages pretrained edge attention mechanisms but also incorporates ACM logic directly into the CNN architecture. This allows the model to predict hyperparameters via an ACM hyperparameter generator as part of the training process, effectively setting these parameters dynamically rather than relying on manual tuning. By combining the strengths of deep learning and incorporating the DALS Level Set ACM [6], our approach aims to improve segmentation accuracy, robustness, and generalization across diverse medical images, marking a significant advancement in the field.

2. Related Work

Neural networks, particularly Convolutional Neural Networks (CNNs), have revolutionized the field of image processing and computer vision due to their ability to automatically learn spatial hierarchies of features from raw image data. CNNs are particularly effective in the domain of medical image segmentation because of their capacity to capture complex patterns in highly dimensional data, such as CT scans, MRI, and X-ray images.

U-Net, one of the most widely used architectures for medical image segmentation, is a CNN-based architecture specifically designed for biomedical image segmentation tasks. Ronneberger et al. [11] introduced U-Net as a fully convolutional network that is particularly effective for segmenting small and irregular structures in medical images.

Several variations of U-Net have been proposed in the literature to improve performance in specific medical imaging tasks, such as 3D U-Net for volumetric data developed by Cicek et al. [5], which extends the original U-Net for 3D image segmentation, making it well-suited for MRI and CT scans. The attention U-Net of Oktay et al. [9] introduces attention mechanisms to the U-Net architecture, allowing the model to focus on the most relevant regions of an image while suppressing irrelevant information, improving the segmentation quality in complex cases.

Active Contour Models (ACMs), introduced by Kass et al. [7] are deformable curves that evolve within an image to align with object boundaries by minimizing an energy functional. The model balances internal forces, which enforce smoothness, and external forces from image gradients that attract the curve to edges. ACMs have since been extended to handle topology changes, region-based energies, and deep learning integration.

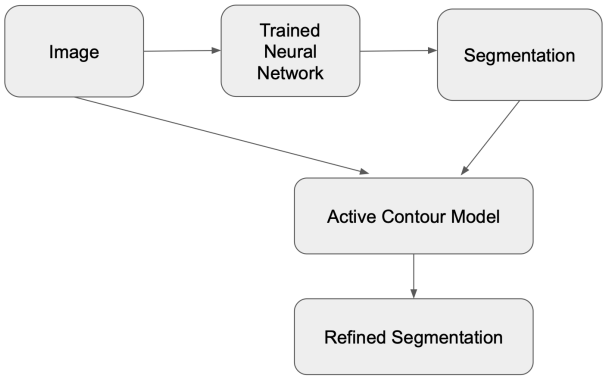

Figure 1. ACM appended after CNN

Chan and Vese [3] proposed a region-based ACM that segments objects based on intensity homogeneity rather than edges. This model is more robust to weak or blurred edges and can segment multiple objects simultaneously, making it particularly useful in medical imaging and texture segmentation.

Certain popular hybrid frameworks utilize deep learning frameworks to get initial segmentation masks for images and later refine these segmentation masks using active contour models outside the learning framework. An outline of this procedure is depicted in Figure 1. Hatamizadeh et al. [6] utilize a multiscale encoder-decoder CNN architecture whereas Nakhaei et al. [8] use the Segment Anything (SAM) architecture for the 'trained neural network' component in the image.

Vepa et al. [12] proposed a weakly supervised framework for cerebral vessel segmentation in DSA images. They use an active contour model to generate weak annotations, refined through human-in-the-loop strategies. These labels are fed into a CNN for further refinement. CLAHE is applied as a pre-processing step to enhance vessel visibility. The method reduces reliance on manual annotations and achieves state-of-the-art performance, surpassing human annotators.

3. Methodology

3.1. Edge Segmentation

The edge segmentation module is designed to extract and refine edge features from an input tensor, playing a crucial role in enhancing segmentation accuracy, particularly around object boundaries. The process consists of four main steps: edge detection, edge thresholding, filling interior regions, and final processing.

The edge detection step applies the Roberts Operator

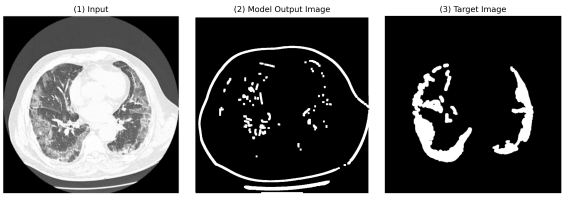

Figure 2. The Roberts operator applied on image bjorke_86.png (left). Edge-based segmentation using the Roberts operator with the thresholding and enhancements to obtain strong binary results (middle). (right) The target segmentation.

[10], a simple edge filter that computes horizontal and vertical gradients using a 2×2 convolutional kernel. It calculates the gradient magnitude to highlight regions of significant intensity changes, which correspond to edges. Each input channel is processed separately, and the results are combined into a final edge map.

Next, edge thresholding dynamically determines an adaptive threshold based on the mean and standard deviation of edge intensities. This step ensures that only the most prominent edges are retained, producing a binary edge map.

The third step fills interior regions by employing morphological operations to refine the detected edges. Dilation expands edge regions to close small gaps, while erosion shrinks edges to retain only meaningful structures. These operations help fill interior regions within strong edges, ensuring that enclosed areas are properly marked and improving segmentation consistency.

Finally, final processing enhances the detected edges for better contrast and interpolates the edge map to match the original input dimensions. The result is a refined edge-aware feature map that is later utilized in the Edge Attention mechanism.

This approach is crucial for segmentation models, as it effectively captures boundary details, leading to sharper object delineation and improved accuracy. However, while it captures major edges, it struggles to find specific abnormalities in the target, as the Roberts operator focuses on general edges rather than finer details. Despite this, it provides a solid foundation for training a neural network to leverage edge-based segmentation. An example result from this approach is shown in Figure 2.

3.2. Convolutional Neural Network

3.2.1. Architecture

The network follows an encoder-bottleneck-decoder structure. The encoder extracts hierarchical features, incorporating attention mechanisms to suppress irrelevant information and refine feature selection. The bottleneck enhances feature representation. The decoder reconstructs the segmentation mask using transpose convolutions and attention layers, refining spatial details before outputting the final mask via a Sigmoid activation. Edge-based post-processing can further improve boundary accuracy.

3.2.2. Data Preprocessing

The data preprocessing pipeline for the edge segmentation model standardizes the dataset by processing grayscale images and their corresponding segmentation masks. Images are resized to 1024×1024 pixels, normalized to the $[0, 1]$ range, and converted into PyTorch tensors. A custom EdgeSegmentationDataset class is implemented to load and preprocess image-mask pairs, ensuring that only matching files are considered. The pipeline also supports optional transformations like data augmentation (random flips, rotations) and sample size limitations. This setup ensures consistency in input data, enhancing the model's stability and performance during training.

3.2.3. Attention Mechanisms

We compare two approaches: Base Attention and Edge Attention, highlighting their differences in feature refinement and edge awareness.

Our model supports Base and Edge Attention mechanisms. Base Attention refines feature importance, while Edge Attention enhances segmentation by incorporating edge detection using the Roberts Operator.

Base Attention learns an attention map by processing the input feature map through a series of 1×1 convolutions, followed by a sigmoid activation to normalize attention values. The model scales the input features using these learned weights, refining feature importance. The process involves four key steps: feature transformation through a 1×1 convolution, refinement using another 1×1 convolution, normalization via sigmoid activation, and feature modulation, where the input features are multiplied by the attention map to selectively enhance relevant regions. However, Base Attention does not explicitly consider edge information, which can limit segmentation accuracy near boundaries. Since it applies attention uniformly across the feature space, it lacks specialized treatment of high-gradient regions.

Edge Attention extends Base Attention by explicitly incorporating edge information into the attention computation. It applies a separate edge-detection convolution, ensuring that attention focuses on boundary details to improve segmentation performance. This process starts with feature extraction using a 3×3 convolution, followed by edge awareness, where edges are extracted from the input. Then, a convolutional layer transforms edge features into attention weights, which are normalized via sigmoid activation. Finally, these computed attention weights are multiplied with the extracted features, enhancing segmentation precision around boundaries.

3.3. Active Contour Models

ACMs rely on several key components that collectively govern the contour evolution process. The intensity image serves as the primary input, representing the image on which the contour evolves. The initial contour defines the starting boundary, playing a critical role in the model's convergence and final segmentation outcome. The algorithm and energy functional dictate how the contour evolves over time, balancing internal forces for smoothness and external forces derived from the image. Additionally, hyperparameters control various aspects of the model's behavior, such as regularization strength and step size, significantly impacting the performance and stability of the segmentation.

First we describe several preprocessing techniques we implemented to utilize for ACM experiments later in this section. Next we describe our motivation and approach for exploring and evaluating different ACMs with various preprocessing techniques, initial contours, and hyperparameters. Then, we closely look into two ACMs: Level-Set ACM in DALS [6] and Chan-Vese ACM [3]. Finally, we select an ACM and motivate its integration into the CNN architecture, establishing the foundation for a unified segmentation framework.

3.3.1. Preprocessing Techniques

To enhance image quality, preprocessing techniques like grayscale conversion, additive bias correction (ABC), and Contrast Limited Adaptive Histogram Equalization (CLAHE) are applied. These methods normalize intensity, correct low-frequency distortions, and improve contrast, making object boundaries clearer. We utilize these algorithms in our experiments to see how refining the input images affects ACM performance and properties like ACM robustness to contour initialization.

Grayscale Conversion This is a simple grayscale conversion of the original image such that the final image intensity values are between 0 and 255.

Additive Bias Correction (ABC) Using the grayscale intensity image, the method estimates a smooth bias field — representing background illumination variations — with a Gaussian filter ($\sigma = 50$). The original image is then divided by this bias field to normalize intensity variations, enhancing boundary clarity. The corrected image is further normalized to the intensity range (0, 255) for consistent input in segmentation. This technique is inspired by findings from Chen et al. [4], who showed that the ABC model yields the most stable segmentation results for medical images.

Contrast Limited Adaptive Histogram Equalization (CLAHE) The CLAHE preprocessing pipeline enhances image quality for ACM-based segmentation. The grayscale image is first normalized between 0 and 1, then smoothed with a Gaussian blur ($\sigma = 1$) to reduce noise. The image is converted back to 8-bit format and processed by CLAHE,

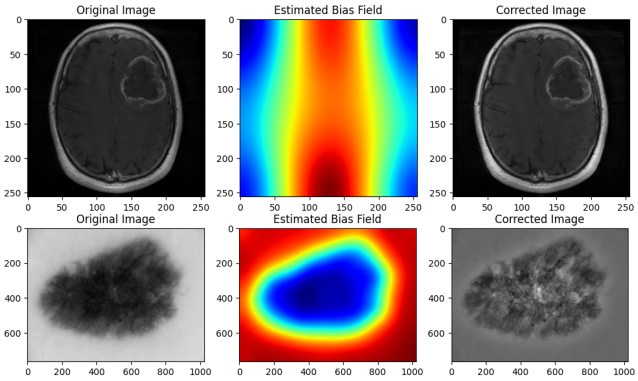

Figure 3. ABC

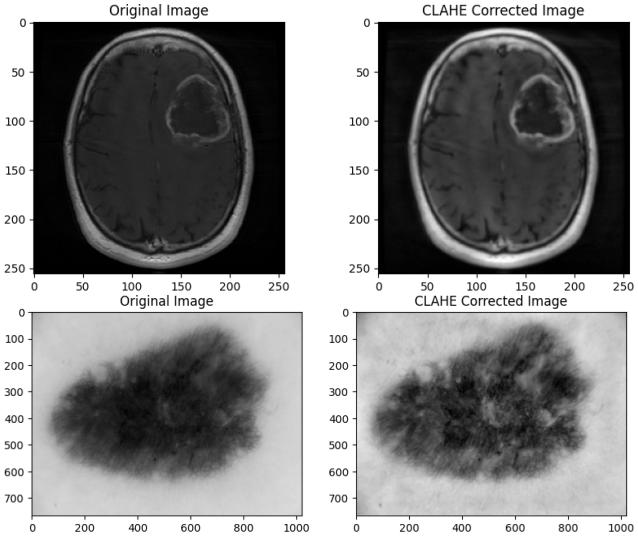

Figure 4. CLAHE

which enhances local contrast within small tiles while limiting noise amplification (clip limit $= 2$). Inspired by Vepa et al. [12], this approach improves boundary clarity for more accurate ACM-based segmentation.

3.3.2. Exploring Active Contour Models

From here we move onto implementing and exploring ACMs. The motivation behind this is to familiarize with underlying logic, experiment with nuances of different ACMs, and use this to brainstorm a promising methodology for integrating ACMs into CNNs. It is important to note that though different ACMs have the same energy minimization approach, the energy formulation and hence hyperparameters differ. However, ideas like sensitivity to hyperparameters including the initial contour carry on although the nature and extent of sensitivity vary again. We focus on two ACMs: Level-Set ACM used in the "Deep Active Lesion Segmentation" paper (DALS LSA) [6], Chan Vese ACM implementation (CV) [3].

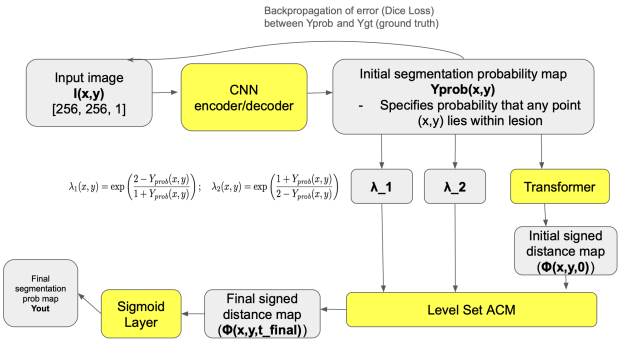

Figure 5. DALS Architecture

Our main code-base for implementing these algorithms, supportive functions, and scripts for experiments conducted in the next two sections are available in the following github link: Github Link for ACM Experiments. The "Level-Set-ACM" subfolder contains key components for Active Contour Model (ACM) implementations. The chan_vese.py script handles the main CV implementation, while level_set_acm.py contains the primary DALS LSA implementation. The lsa_helpers.py module provides visualization functions, image property checks, preprocessing techniques (e.g., ABC and CLAHE), initial contour and probability mask creation, and evaluation metrics like DICE and IoU. The lsa_run_helpers.py script consolidates the ACM pipeline, handling image processing, segmentation, logging, and result saving. The "scripts" directory contains short scripts for executing these pipelines with different parameters and medical images, while the "Results" folder stores output generated from these runs.

3.3.3. Level Set ACM (LSA) Implemented in DALS

This is a well-defined implementation by Hatamizadeh et al. [6] related to the segmentation of medical images (particularly skin lesions).

DALS Architecture An encoder-decoder CNN model is trained to produce good probability masks for segmenting images. The LSA utilizes the CNN output probability mask and attempts to further perfect it to obtain a final probability mask. Figure 5 illustrates the overall architecture in DALS.

Overview of the Level-Set ACM (LSA) The LSA begins with a 2D intial segmentation probability mask. First, $\lambda(x, y)$ maps are constructed based on predefined mathematical expressions. These maps assign weights to the contour energy terms capturing intensity differences between pixels and the average intensity inside or outside the contour, depending on the pixel's location. This weighting helps guide the contour evolution process.

Next, an initial signed distance map is generated using the distance_transform_edt function from the scipy.ndimage library. The signed distance function $\phi(x, y)$ represents the shortest distance from each point to the closest point on the contour, with positive values inside the contour, negative values outside, and zero values on the contour. The DALS LSA algorithm iteratively updates this map, evolving the contour until convergence. Finally, the resulting signed distance map is converted into a probability mask using the sigmoid function, producing the final segmentation result.

Level Set ACM Iteration Steps The main procedure for each energy minimization step is as follows:

1. Narrow Band Identification: Pixels within a narrow band around the zero level set are selected as the region where computations are performed, improving computational efficiency.
2. Intensity Estimation: The mean intensity values are computed separately for regions inside and outside the level set contour using local intensity information.
3. Energy Terms Calculation: Two main energy terms are calculated based on intensity differences between the image and the estimated mean intensities inside and outside the contour. These terms encourage the contour to align with object boundaries where the intensity difference is maximized.
4. Regularization Term: A curvature-based regularization term is computed to ensure smoothness of the evolving contour and prevent noisy or irregular boundaries. The regularization strength is controlled by the hyperparameter μ.
5. Balloon Force: An additional constant force, controlled by the hyperparameter ν, is applied to expand or shrink the contour, helping the contour evolve towards object boundaries.
6. Force Computation: The total force guiding the contour evolution is calculated by combining the intensity-based energy terms, the curvature regularization term, and the balloon force.
7. Level Set Update: The level set function is updated iteratively by applying the computed force, ensuring that the contour evolves toward object boundaries.
8. Reinitialization: The level set function is periodically reinitialized to maintain numerical stability and preserve the signed distance function property.

The main hyperparameters for the Level-Set ACM are ν, μ, and the number of iterations.

Our Implementation Details for DALS LSA To implement the required logic, we first analyzed the DALS code and extracted the Level-Set ACM components. Additionally, we set up a compatible TensorFlow environment and updated the code by replacing deprecated TensorFlow func-

(a) Ground Truth - Brain (b) Ground Truth - Skin

Figure 6. Reference Ground Truth for ACM Experiments

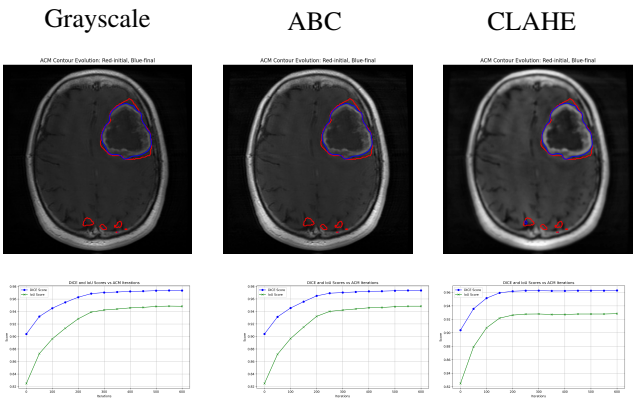

Figure 7. Experiment 1: DALS LSA Demo on brain image using a pretrained-CNN generated initial contour, $\nu = 5$, $\mu = 0.2$, number of iterations $= 600$

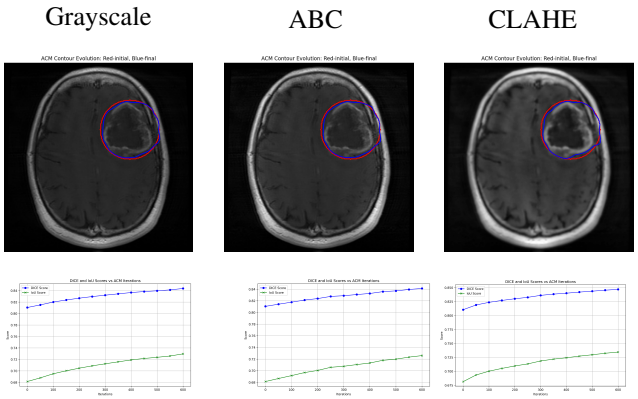

Figure 8. Experiment 2: DALS LSA Demo on brain image using a trivial circular initial contour, $\nu = 5$, $\mu = 0.2$, number of iterations $= 600$

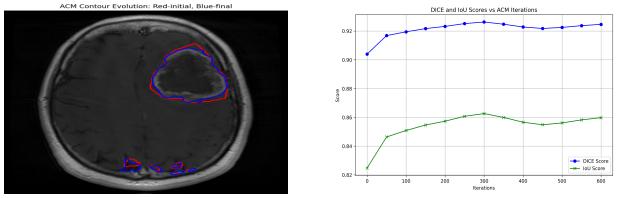

Figure 9. Experiment 3: DALS LSA Demo on brain image using a pretrained-CNN generated initial contour, $\nu = 0$, $\mu = 0$, number of iterations $= 600$

tions with their Keras equivalents. Modifications were also made to support non-square images, which was crucial for later DALS LSA experiments on various medical images. Now we will discuss applying DALS LSA on medical brain and skin images. Figure 6 shows the ground truth segmentations for the brain and skin images which were also illustrated in Figure 3 and Figure 4.

Experiments of DALS LSA Once we extracted and updated the DALS LSA, in experiment 1, we were able to reproduce the DALS brain demo as shown in Figure 7. Here, we used the same initial contour (the one generated by pretrained CNN used in DALS) and hyperparameters ($\nu = 5$, $\mu = 0.2$, number of iterations $= 600$) specified in their demo. From the DICE/IOU score plots, one can infer that a larger number of iterations does not mean better results. Sometimes the improvement halts with more iterations or becomes worse. Hence, it is crucial to choose a good value for number of iterations. In terms of the preprocessing technique, based on the final DICE and IOU scores, CLAHE underperformed compared to the ABC and grayscale versions which performed similarly.

In experiment 2 as shown in Figure 8, we used a little more trivial initial contour: a circle enclosing the area of interest. Now we see that the improvements are not nearly as significant as that in the previous experiment. A good initial segmentation allows the ACM to work well. The initial segmentation generated by a trained CNN is reasonably accurate with good probability mask values. This in turn leads to more accurate values for lambda and initial signed distance map and hence better contour evolution. This experiment shows that DALS LSA is highly sensitive to initial contour. In terms of the preprocessing technique, based on the final DICE and IOU scores, CLAHE performed the best while ABC performed the least well.

In experiment 3 we adhere to the original pretrained CNN generated initial contour, 600 iterations, and grayscale preprocessing. The goal here is to initialize ν and μ to different values and see how this affects the DALS LSA performance. Setting both $\nu = \mu = 0$ leads to the results shown in Figure 9. This experiment shows how critical the hyperparameter setting is for an image. The wrong setting can lead to the ACM performance being very poor, while the correct setting leads to accurate results. Moreover, these settings depend on particular dataset and image characteristics. The current approach to setting these hyperparameters is manual trial and error which is infeasible.

In experiment 4 as shown in Figure 10, we attempt to apply DALS LSA to skin lesion images with a very trivial initial circular contour. In terms of preprocessing techniques, all contour evolutions have very low performance

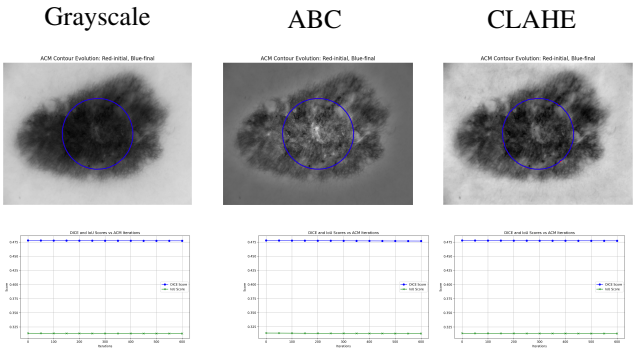

Figure 10. Experiment 4: DALS LSA applied to skin lesion image using a trivial circular initial contour, $\nu = 5$, $\mu = 0.2$, number of iterations $= 600$

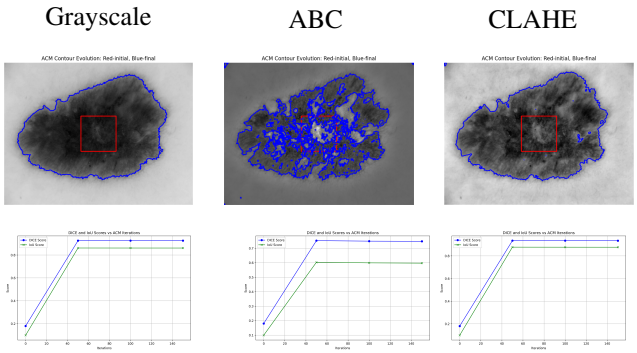

Figure 11. Experiment 1: Chan Vese applied to skin lesion image using a trivial circular initial contour

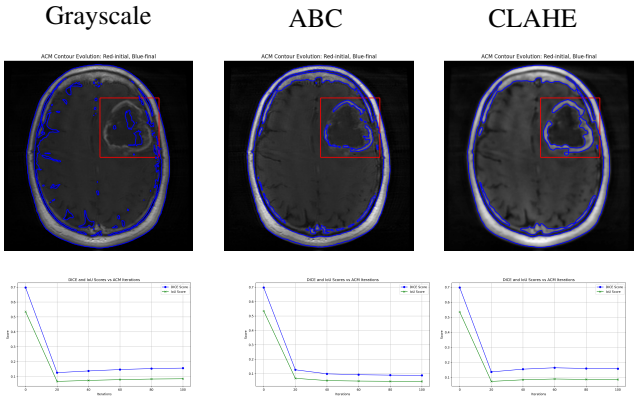

Figure 12. Experiment 2: Chan Vese applied to brain image using a trivial circular initial contour

and hardly update to go to the skin lesion boundary. The poor performance of LSA in this scenario can likely be attributed to the initial contour being deeply enclosed within the skin lesion. In DALS LSA, contour updates are primarily triggered when there is intensity variation within a small neighborhood around each pixel in the narrow band surrounding the contour. However, in this case, the lack of intensity variation within and around this narrow band results in minimal contour updates. This limitation highlights the need to explore alternative region-based ACM approaches, such as the Chan-Vese model, for comparison.

3.3.4. Chan-Vese (CV)

In order to better understand ACMs, we decided to implement another ACM for comparison purposes. We chose the Chan-Vese implementation because its region-based property differs from the localized intensity variation technique in DALS which we identified as a shortcoming in DALS making it less robust to contour initialization.

We slightly update the following kaggle implementation of Chan-Vese model for our experiments and use default hyperparameters there: Active Contour Model on Kaggle

In experiment 1 shown in Figure 11, we apply the CV ACM to a skin lesion using a trivial initial contour. Since the CV approach considers the entire image during contour updates and tries to split the image into two regions with roughly homogenous intensities, it can consider global information and is less dependent on contour initialization. Hence, it is able to properly segment the skin lesion. As can be seen, Grayscale and CLAHE preprocessing techniques have similar performance while ABC underperforms. Sometimes preprocessing steps may lead to important information being lost (parts of actual segmentation having similar intensity to parts that are not part of the segmentation). Hence when CV is applied to ABC corrceted image, there exist false negatives in the result that actually belong to the skin lesion.

We conduct a final experiment of applying CV ACM on the brain image as shown in Figure 12. Although CV approach is region-based and robust to contour initialization, this may not actually be the right approach in some cases of medical image analysis. We can see that in all three preprocessing cases the DICE/IOU scores reduce over ACM iterations because it has many false positives. We might actually want the ACM to mostly focus close to the initial contour area instead of focussing a lot on global information.

3.3.5. Summary

In this section, we examined both the DALS Level-Set ACM (LSA) and Chan-Vese (CV) ACM methods on various medical images, different preprocessing techniques, initial contours, and hyperparameter settings. Our findings indicate that DALS LSA is highly sensitive to the initial contour, whereas the CV model is more robust to contour initialization. However, the region-based approach of CV may be less effective for certain medical images. ACMs, in general, exhibit significant sensitivity to hyperparameters, making it challenging and labor-intensive to identify the optimal settings for segmenting the object of interest. While preprocessing techniques can sometimes improve perfor-

mance, their effectiveness is limited when the initial contour is poorly chosen, and in some cases, these techniques may lead to a loss of crucial information.

Given that we have already developed a CNN architecture with attention to identify suitable initial contours, we chose to integrate DALS LSA into the CNN. This decision was driven by DALS LSA's ability to refine well-initialized contours rather than relying on intensity variations across the entire image. Additionally, compared to the CV implementation used in our experiments, the DALS LSA implementation is well-defined, extensively tested, and widely accepted. For image preprocessing within the hybrid architecture, we opted for grayscale conversion, ensuring that final image intensity values remain within the range of 0 to 255.

The TensorFlow implementation of DALS LSA proved to be computationally expensive, making it difficult to systematically explore all preprocessing techniques, initial contours, and hyperparameter settings. This challenge underscores the necessity of allowing hyperparameter tuning and contour initialization to be naturally optimized within the neural network framework.

3.4. Overall Architecture

3.4.1. System Diagram

Figure 13 illustrates our hybrid ACM system. Enabling the ACM functionality in our architecture will include the 'ACM Hyperparameter Generator,' its outputs, and 'Level Set ACM' which will otherwise be disabled.

ACM Hyperparameter Generator The ACM Hyperparameter Generator uses adaptive average pooling as a flattening mechanism, followed by a fully connected layer with a ReLU activation and another fully connected layer with a Sigmoid activation. It produces three outputs: num_iters, ν, and μ, which are scaled to the correct ranges: 0–100 for num_iters, 0-10 for ν, and 0-1 for μ. The main novelty is to leverage the properties learned through the previous layers (encoder and bottleneck) and further use fully connected layers to learn image properties relevant to determining an optimal set of ACM hyperparameters that will result in effective ACM contour evolution.

Incorporating ACM into the Neural Network Incorporating the DALS Level Set ACM into the neural network required converting the original TensorFlow implementation to PyTorch, as our CNN baseline was in PyTorch. For the brain demo in Figure 7, the torch version of DALS LSA takes 1.770149 seconds, while the tensorflow version takes 92.741031 seconds. This is a speedup by a factor of 52.4. Hence, in addition to being compatible with existing torch architecture, a torch version of DALS significantly reduces the training time.

Additionally, the DALS LSA uses non-differentiable scipy functions to calculate the initial signed distance map from the probability mask. Figure 14 illustrates how this issue can cause ineffective learning. The initial signed distance map is what gets updated through the ACM iterations to get the final probability mask. Leaving this out of the gradient computational graph will hinder backpropagation along this important path and effective learning. To address this, we aimed to replicate this logic using differentiable operations, ensuring compatibility with backpropagation. Making the DALS LSA fully differentiable should theoretically allow for much more accurate gradient updates, improving the overall learning process.

Figure 15 illustrates the differences in ACM contour evolution and demonstrates the success of our approach through the DALS brain demo results, showing that the converted DALS LSA logic remains intact. Furthermore, by working with a few images and epochs and setting "torch.autograd.set_detect_anomaly(True)" we closely refined the code to eliminate backpropagation errors.

3.4.2. Training

The training procedure for the edge segmentation model follows a structured approach for efficient learning and generalization. The model is initialized, and its configuration is saved for reproducibility. Training is conducted in mini-batches, where predictions are made, and the loss is computed against ground truth masks. A learning rate scheduler adjusts the learning rate based on training progress, and gradient clipping prevents instability by controlling large gradients. Early stopping halts training if no improvement occurs after several epochs. Loss is logged and visualized to monitor progress, ensuring stable, efficient training and avoiding overfitting.

Note about training hybrid ACM model: Since Active Contour Models (ACMs) perform poorly when initialized with suboptimal or random probability masks, the training process for the hybrid ACM architecture incorporates a weight initialization strategy. Specifically, before training begins, the weights of a reasonably trained edge attention model are loaded into the non-ACM-related layers of the neural network. This serves as a pretraining step, providing a more stable starting point for training the hybrid ACM model.

3.4.3. Testing

The testing process evaluates the trained model's performance on a test dataset. It begins by organizing the output folder based on the model's timestamp and epoch. The model is set to evaluation mode to ensure consistent behavior. Loss tracking and metric computation are initialized, and the model processes each batch in the test set to generate predictions. Loss is calculated, and predictions are converted to probabilities for metric evaluation. Visual results,

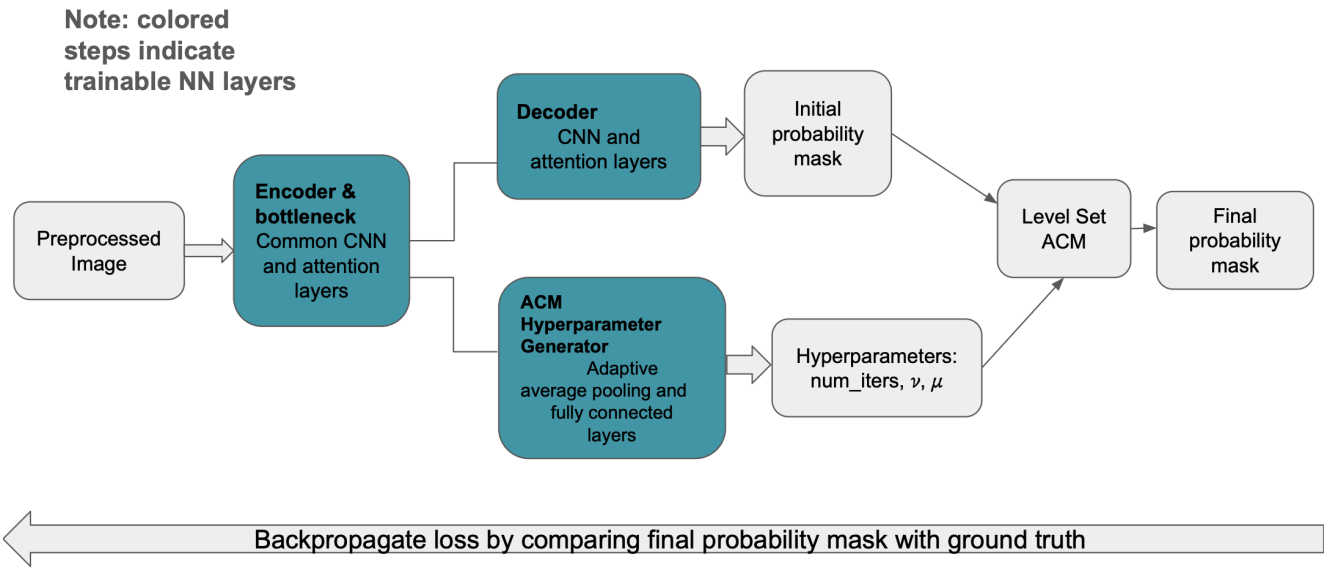

Figure 13. Hybrid ACM system diagram

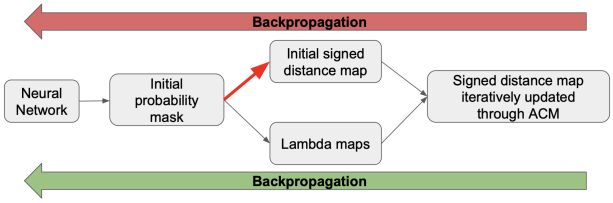

Figure 14. Differentiability and Backpropagation

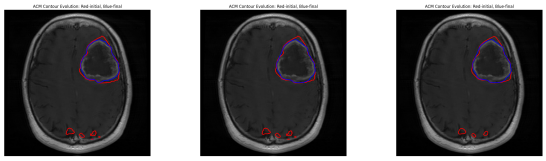

Figure 15. Demo of Tensorflow, Torch, and fully differentiable Torch versions of DALS Level Set ACM

including input-output-target comparisons, are saved. After processing all batches, performance metrics like accuracy and IoU are computed and stored. The results are saved for further analysis, providing insights into the model's effectiveness.

4. Results and Discussion

4.1. Considerations

Our model is trained with an initial learning rate of 0.001, which reduces by a factor of 0.5 if there is no improvement greater than 1e-4 after three epochs. The dataset used is the COVID-19 CT scan lesion segmentation dataset [1].

The paper discusses the use of different training configurations for various attention modules. The base attention module utilizes a model trained for 50 epochs on the entire COVID-19 CT scan dataset with an 80/20 train-test split ratio. The edge attention module also uses a model trained for 50 epochs on the full dataset with the same train-test split. The hybrid ACM module, due to time complexity constraints, leverages the pretrained edge attention model (50 epochs) and is fine-tuned on a smaller subset of 8 training images and 2 testing images from the full dataset for a total of 10 epochs. The model implementation can be found in this GitHub repository [2].

4.2. Overall Results

In testing, the results are better and are produced at a faster rate for the Edge Attention model than the Base Attention model. It can be seen from Table 1 that the edge attention model has lower average test loss and higher values for other scores like IoU and DICE compared to the base attention model. While the Hybrid ACM model does have slightly greater loss compared to the Base and Edge Attention CNNs, its ability to have hyperparameter tuning for the ACM provide novel results and a foundation for future adaptation and training.

4.3. CNN with Base Attention

For a clearer comparison of the overall performance for base attention CNN module, please refer to Figure 16. While the average Dice Score of 0.4570 is respectable for a model with only a basic attention module, there is a significant disparity between the best and worst-case scenarios.

Metric	Base Attention Model	Edge Attention Model	Hybrid ACM Model
Average Test Loss	0.0257	0.0229	0.0389
AUROC	0.9882	0.9921	0.8758
AUC	0.9882	0.9921	0.8758
Precision	0.7624	0.7570	0.8175
Recall (Sensitivity)	0.5360	0.6477	0.4023
F1 Score	0.6294	0.6981	0.5393
IoU	0.3412	0.4285	0.3820
Dice Score	0.4570	0.5519	0.4427

Table 1. Performance Metrics for Base Attention Model (50 epochs trained), Edge Attention Model (50 epochs trained), and Hybrid ACM Model (10 epochs trained with 8 training images) Evaluation

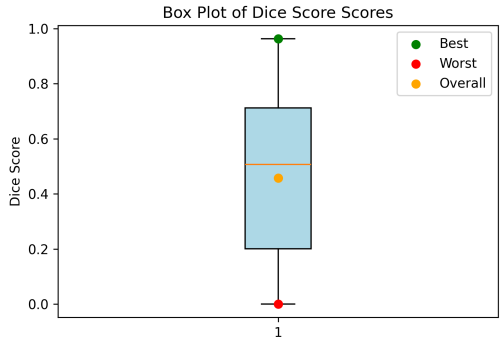

Figure 16. Dice Score Box Plot comparing results in the Base Attention Model

4.3.1. Results: Best vs Worst

The image that stands out as the best performer is shown in Figure 17. It achieved the highest precision among all the test images from the base attention model, with a Dice Score of 0.9628, which is an excellent result. While this is a strong performance, there is still potential for further improvement. Overall, these results highlight the model's strong capability and suggest opportunities for additional refinement.

The image that demonstrates the worst-case scenario is shown in Figure 18. This image exhibited the lowest precision among all the test images from the base attention model, achieving a Dice Score of 0, indicating complete prediction failure in this instance. As illustrated in Figure 18, the final target image lacks sufficient detail, which likely confuses the model and impairs its ability to correctly identify and highlight the relevant features.

4.4. CNN with Edge Attention

For a clearer comparison of the overall performance for edge attention CNN module, please refer to Figure 19. This

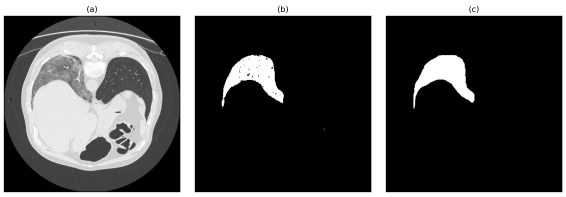

Figure 17. Best Base Attention Model result with a precision score of 0.9630 and DICE score of 0.9628 for the Jun_radiopaedia_7_85703_0_case20_39.png. (a) Input image in the dataset. (b) Predicted segmentation output from the trained model. (c) Target segmentation.

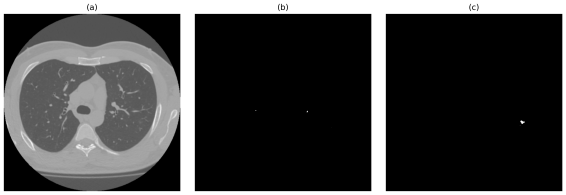

Figure 18. Worst Base Attention Model result with a Dice Score of 0.0 for the Morozov_study_0295_28.png. (a) Input image in the dataset. (b) Predicted segmentation output from the trained model. (c) Target segmentation.

has an average Dice Score of 0.5519 which is greater than the average Dice score of the base attention model.

4.4.1. Results: Best vs Worst

An image to highlight is Figure 20. This image had the best precision out of all the images tested from the edge attention model with a Dice Score of 0.9561. While this max result is not better than the Base Attention model, the average test DICE score for edge attention is still better as discussed before. Moreover, comparing the dice score box plots in Figure 16 and Figure 19 you can see that in edge CNN most of the test data dice scores fall in higher range compared to the base CNN case.

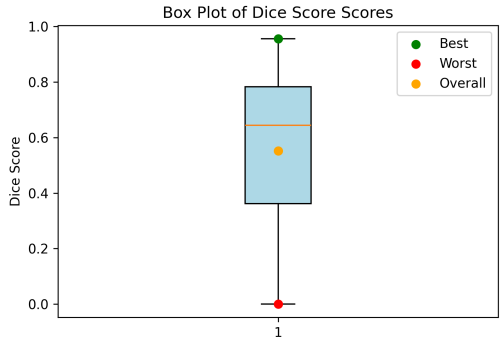

Figure 19. Dice Score Box Plot comparing results in the Edge Attention Model

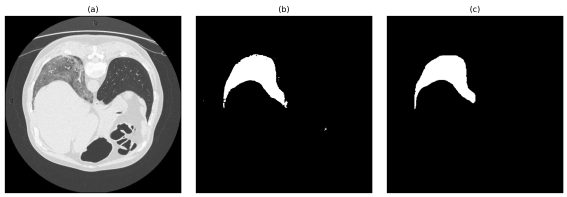

Figure 20. Best Edge Attention Model result with a Dice Score of 0.9561 for the Jun_radiopaedia_7_85703_0_case20_39.png. (a) Input image in the dataset. (b) Predicted segmentation output from the trained model. (c) Target segmentation.

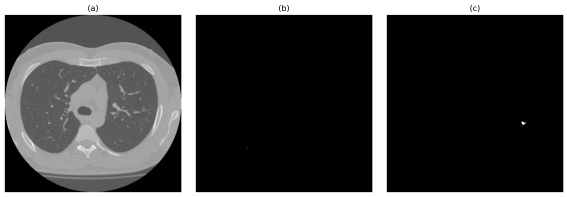

Figure 21. Worst Edge Attention Model result with a Dice Score of 0.0 for the Morozov_study_0295_28.png. (a) Input image in the dataset. (b) Predicted segmentation output from the trained model. (c) Target segmentation.

Another image to highlight is Figure 21. This image had the worst precision out of all the images tested from the edge attention model, with a Dice Score of 0.0. Similar to Figure 18, Figure 21 struggled to find the features to emphasize in segmentation, which is clear in its target result which is a very small segmentation.

4.5. Pretrained CNN with ACM

The main innovation of this approach is the ability to determine optimal ACM hyperparameters using fully connected layers, which significantly enhance the effectiveness of ACM contour evolution. Despite being trained on a small set of images atop the pretrained Edge Attention model, the results show a notable improvement in the performance of

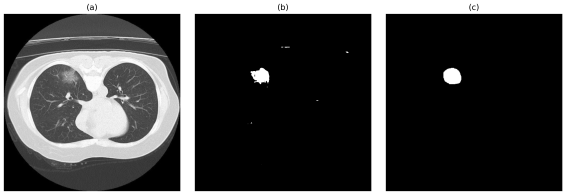

Figure 22. Hybrid ACM Model result with a Dice Score of 0.8569 for the Jun_radiopaedia_29_86491_1_case16_20.png. (a) Input image in the dataset. (b) Predicted segmentation output from the trained model. (c) Target segmentation.

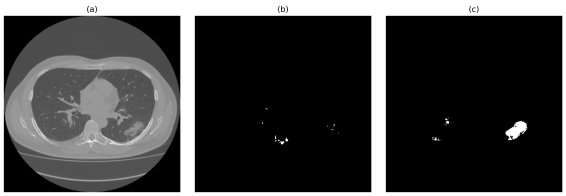

Figure 23. Hybrid ACM Model result with a Dice Score of 0.0284 for the Morozov_study_0258_22.png. (a) Input image in the dataset. (b) Predicted segmentation output from the trained model. (c) Target segmentation.

the ACM hyperparameter generator and the Hybrid ACM model. As shown in Table 1, the Hybrid ACM model achieves the best precision and comparable scores for all other metrics even though it was only trained for 10 epochs.

4.5.1. Results: Highlights

An image to highlight is Figure 22. This image had the best precision out of all the images tested from the Hybrid ACM model. The Dice Score is 0.8569. Despite this DICE score being lower than the best results for base and edge attention, this training occurred on fewer images and with only 10 epochs.

Another test result is Figure 23 and it had a non-zero Dice Score of 0.0284.

4.6. Discussion

The overall results from the Hybrid ACM model is novel in its approach to utilizing the ACM hyperparameter generator and provides a profound innovation in its possibility to learn future contours with a combined neural network and level-set ACM. The Active Contour Model (ACM) with hyperparameter tuning demonstrates good performance when trained and tested on small datasets. However, its application to larger datasets presents significant computational challenges due to the high time and memory consumption associated with gradient tracking over hundreds of ACM iterations. Since each iteration contributes to refining the segmentation mask, the backpropagation process becomes increasingly expensive, leading to inefficiencies in training. This limitation makes large-scale training impractical with-

out substantial computational resources. Despite this, the model's strong results on small datasets highlight its potential effectiveness, suggesting that it is well-suited for applications where high-quality segmentation is required on limited data samples. Future optimizations could focus on reducing computational overhead while preserving the model's ability to learn optimal ACM hyperparameters.

5. Conclusion

We have proposed a deep learning-based segmentation framework that combines Active Contour Models (ACMs) and attention mechanisms. The novelty of our approach lies in integrating edge segmentation with a CNN, enhancing feature extraction by emphasizing structural boundaries. Incorporating the ACM as an additional component refines the segmentation process by leveraging its ability to capture object contours dynamically. Moreover, our main novelty is introducing an "ACM Hyperparameter Generator" into the image segmentation CNN that can be trained to generate image-dependent optimal ACM hyperparameters for effective segmentation refinement. While the loss remains high for a period during training, this hybrid approach balances data-driven feature learning with contour-based refinement, potentially improving segmentation accuracy over time.

5.1. Future Work

Future work will focus on several key areas for improvement. This includes exploring additional image transformations and pre-processing techniques to enhance model performance. Further training will be conducted on a broader range of variations to increase the model's robustness. Additionally, model improvements such as incorporating batch normalization will be explored, along with optimizing the ACM to enhance speed and memory efficiency. A sliding window approach will also be investigated for improving the handling of large images during training. Finally, further training will be conducted with the optimized ACM to refine its performance.

6. Individual Contributions

Shaira Alam Shaira contributed to the research and development of various different edge-based image segmentation methods in order to have the most efficient and accurate segmentation when integrated with a neural network. After numerous testing with image segmentation methodologies, she designed and implemented the convolutional neural network (CNN) that became the foundation for this project. This CNN included modules for data preprocessing, an edge segmentation module with specific preprocessing, a basic attention mechanism, and an edge attention mechanism that integrated the edge segmentation module. She conducted extensive training, testing, and evaluations for the Base Attention CNN Model, Edge Attention CNN Model, and the final Hybrid Active Contour Model (ACM) CNN Model to evaluate performance across different configurations.

Vaishnavi Manthena Vaishnavi contributed to the research and development of various ACMs including the Chan Vese model and DALS Level Set ACM (LSA). She experimented these methods with various medical images and wrote code for image processing (grayscale, ABC, CLAHE). She developed and implemented a prototype for integrating the DALS LSA into the CNN such that contour initialization, ACM hyperparameter tuning, and ACM occur within the neural network. For this she also converted tensorflow DALS LSA to a fully differentiable torch version that is compatible with backpropagation. She performed local training and testing of ACM hybrid architecture with small training data sizes to ensure correct learning.

Acknowledgements

Sincere gratitude is extended to faculty advisor Professor Demetri Terzopoulos for his invaluable guidance and support throughout the project. Appreciation is also given to Noor Nakhaei, Ph.D. candidate and project mentor, for providing consistent and frequent guidance throughout the project. Thanks are due to the UCLA Computer Graphics & Vision Group, led by Dr. Terzopoulos, and Zhi Li for enabling access to the group's server, which was instrumental for training and testing the models.

References

[1] Covid-19. http://medicalsegmentation.com/covid19/, 2020. Accessed: 23 December, 2020. 9

[2] Shairaalam Alam. Medical image segmentation, 2025. Accessed: 2025-03-07. 9

[3] T. F. Chan and L. A. Vese. An active contour model without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001. 2, 4

[4] Y. Chen, Z. Wang, and G. Hu. An overview of intelligent image segmentation using active contour models. *Artificial Intelligence Review*, 46(3):315–343, 2016. 4

[5] O. Cicek, A. Abdulkadir, S. S. Lienkamp, and et al. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 424–432, 2016. 2

[6] A. Hatamizadeh, A. Hoogi, D. Sengupta, W. Lu, B. Wilcox, D. Rubin, and D. Terzopoulos. Deep active lesion segmentation. In *Machine Learning in Medical Imaging*, pages 98–106, 2019. 2, 4, 5

[7] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, 1(4):321–331, 1988. 2

[8] N. Nakhaei, T. Zhang, D. Terzopoulos, and W. Hsu. Refining boundaries of the segment anything model in medical

images using an active contour model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2024. 2

[9] O. Oktay, J. Schlemper, L. L. Folgoc, and et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint*, 2018. 2

[10] L. G. Roberts. *Machine Perception of Three-Dimensional Solids*. PhD thesis, Massachusetts Institute of Technology (MIT), 1963. 3

[11] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. 2

[12] A. Vepa, A. Choi, N. Nakhaei, W. Lee, N. Stier, A. Vu, G. Jenkins, X. Yang, M. Shergill, M. Desphy, K. Delao, M. Levy, C. Garduno, L. Nelson, W. Liu, F. Hung, and F. Scalzo. Weakly-supervised convolutional neural networks for vessel segmentation in cerebral angiography. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 585–594, 2022. 2, 4