LDMRes-Net: A Lightweight Model for Real-Time Retinal Vessel Segmentation

Project 14: Real Time Retinal Vessels Segmenter

Team: 9900t17bkfcvivo50

Role	Full Name	zID
Member	Dingqiao Chu	z5392739@ad.unsw.edu.au
Member	Guanzhong Chen	z5512269@ad.unsw.edu.au
Scrum	Jiangye Song	z5270075@ad.unsw.edu.au
Member	Mingjun Zhao	z5433953@ad.unsw.edu.au
Owner	Shihang Yao	z5458849@ad.unsw.edu.au

Contents

1. Installation Manual... 3
1.1 Environment set up... 3
Creating and Activating a Conda Environment:..3
Installing PyTorch:..3
Installing Additional Dependencies:...4
1.2 Model Training and Prediction...4
Training the Model:.. 4
Model Prediction:... 4
1.3 Using the GUI... 4
Running the GUI:... 4
Using the GUI:... 5
2. System Architecture Diagram.. 6
2.1 Illustration of LDMRes-Net Architecture.. 6
2.2 Illustration of Dual Multiscale Residual Block..6
2.3 Optimized LDMRes-Net Architecture and System Architecture........................... 7
2.4 Optimized Dual Multiscale Residual Block Structure.. 7
3. Design Justifications..8
3.1 Evolved from Proposal to Final Design... 8
3.2 Changes in Project Requirements...8
3.3 Iterations on New Solution LDMRes-Net..8
3.4 Justification for LDMRes-Net..8
3.5 Training for LDMRes-Net...9
4. Explanations and Implementation Details of LDMRes-Net................................10
5. User-Driven Evaluation of Solution...11
5.1 Main Features of the LDMRes-Net.. 11
5.2 Evaluation Criteria... 13
5.3 Effectiveness of the Solution.. 13
6. Limitations and Future Improvements.. 15
6.1 Limitations... 15
6.2 Future Improvements... 16

1. Installation Manual

1.1 Environment set up

To ensure the correct installation of the project dependencies, follow these steps to set up the environment:

Creating and Activating a Conda Environment:

- Open terminal.
- Create a Conda environment named ldmresnet with Python 3.8 installed by running:
  ```
  conda create --name ldmresnet python=3.8
  ```
- Activate the newly created environment by executing:
  ```
  conda activate ldmresnet
  ```

Installing PyTorch:

- For Linux or Windows:
  ```
  pip install torch==2.0.0 torchvision==0.15.1
  torchaudio==2.0.1 --index-url
  https://download.pytorch.org/whl/cu118
  ```
- For macOS:

  If Homebrew is not installed, run:
  ```
  /bin/bash -c "$(curl -fsSL
  https://raw.githubusercontent.com/Homebrew/install/HEAD/ins
  tall.sh)"
  ```

  Install ffmpeg@4 and link it:
  ```
  brew install ffmpeg@4

  ln-s/opt/homebrew/opt/ffmpeg@4/opt/homebrew/opt/ffmpeg
  ```

  Install PyTorch and related packages:
  ```
  pip install torch==2.0.0 torchvision==0.15.1
  torchaudio==2.0.1
  ```

If facing a numpy conflict, resolve it by:
```
conda install -c conda-forge numpy opencv
```

Installing Additional Dependencies:

- Install all additional dependencies listed in the requirements.txt file:
  ```
  pip install -r requirements.txt
  ```

1.2 Model Training and Prediction

To use the model for training and prediction, follow these steps:

Training the Model:

- Prepare your dataset by placing the DRIVE dataset in the ./data/DRIVE/ directory, separated into training and testing sets.

- Run the training script with the following command:
  ```
  python train.py --model LDMResNet --num-classes 2
  --batch-size 10 --epochs 200
  ```

- Alternatively, change the --model parameter to unet if you want to use the UNet model.

Model Prediction:

- Ensure the paths in the predict.py file are correctly set to point to the model weights and test images.

- Run the prediction script:
  ```
  python predict.py
  ```

1.3 Using the GUI

For interactive retinal vessel segmentation using the provided graphical user interface:

Running the GUI:

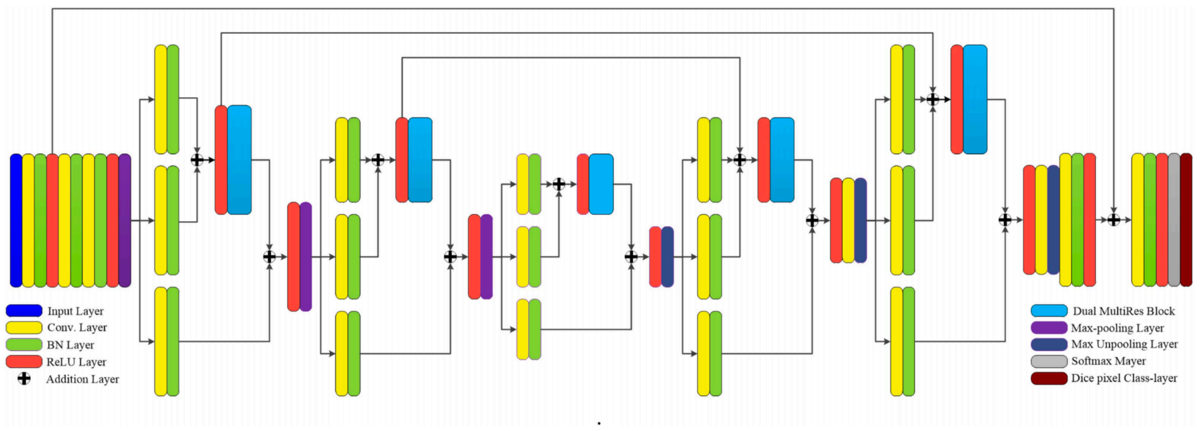
- Start the GUI by executing:
  ```
  python gui.py
  ```

Using the GUI:

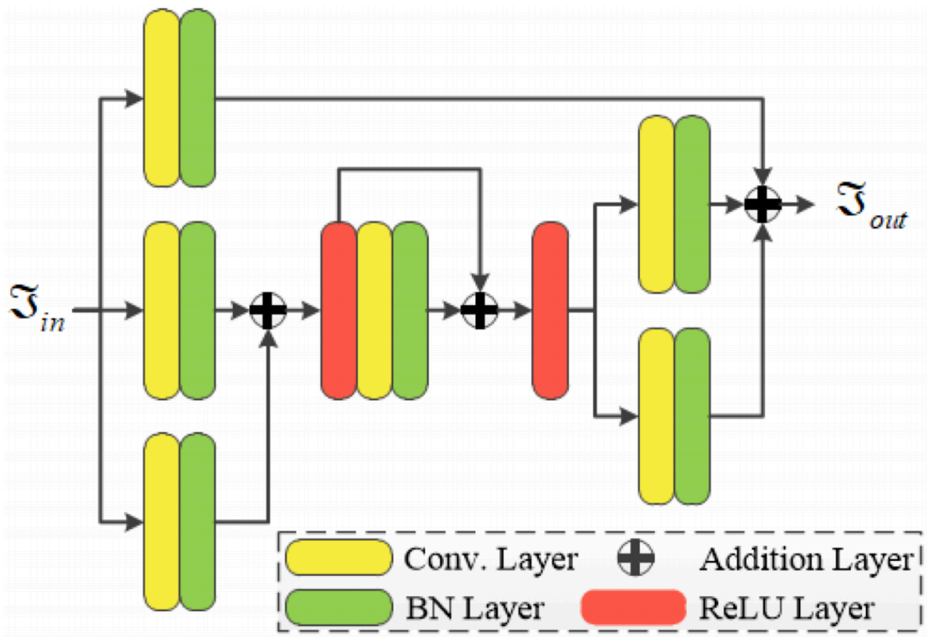
- Click the "Add Content" button to select an input, either a folder of images or a single image file.
- The selected image will be displayed on the left side of the interface; the segmentation result will appear on the right after processing.
- Processed images will be saved in the predictions/{model_name} directory.

2. System Architecture Diagram

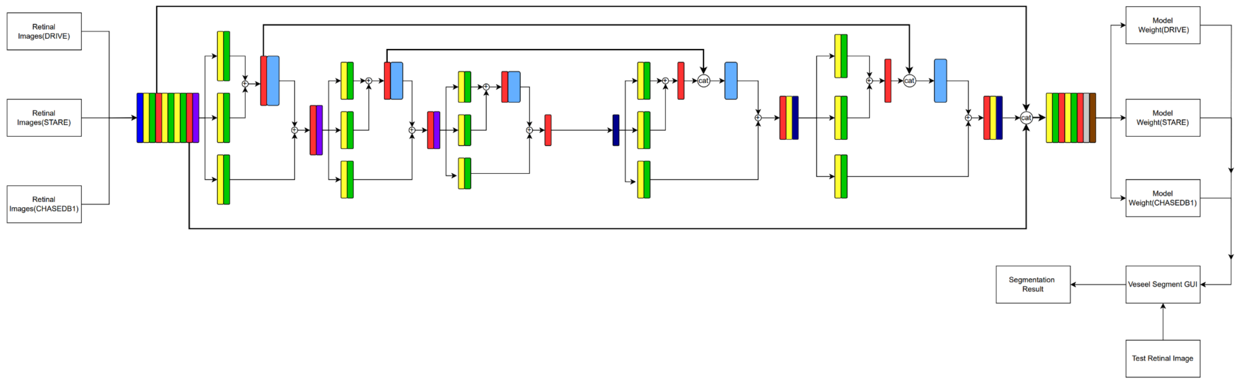
2.1 Illustration of LDMRes-Net Architecture



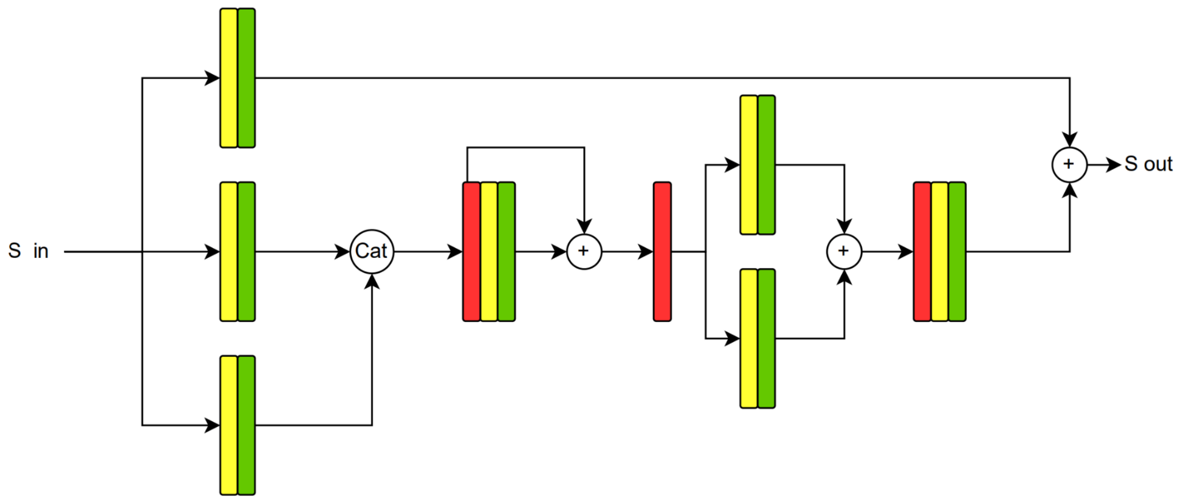
2.2 Illustration of Dual Multiscale Residual Block



2.3 Optimized LDMRes-Net Architecture and System Architecture



2.4 Optimized Dual Multiscale Residual Block Structure



dual multiscale residual block

3. Design Justifications

3.1 Evolved from Proposal to Final Design

Initial Design: Language meets Vision Transformer (LViT) model

In our initial proposal, we planned to implement the LViT (Language meets Vision Transformer) model for retinal vessel segmentation. The decision was driven by LViT’s high performance in integrating text annotations with medical images and we believed this would enhance segmentation accuracy. LViT's hybrid architecture, combining Convolutional Neural Networks (CNNs) and Transformer layers, can effectively capture both local and global features of retinal images.

3.2 Changes in Project Requirements

As the project progressed through first sprint, several key requirements changed:

1. Resource Constraints: We identified a critical need for deploying the segmentation model on resource-constrained environments. The initial focus on merging text annotations became hard to satisfy due to the increased computational overhead associated with processing both image and text data.
2. Real-Time Processing: The requirement for real-time diagnostics for under 1 second makes LViT difficult to achieve fast inference speeds without compromising accuracy.

3.3 Iterations on New Solution LDMRes-Net

To get over these evolving requirements, we iterated on our design by shifting from the LViT model to the LDMRes-Net (Lightweight Dual Multiscale Residual Network). This transition was driven by the following considerations:

- Efficiency: LDMRes-Net is specifically optimized for low computational demand, the parameter of the model is under 1 million, making it suitable for deployment on IoT devices with limited processing power.
- Balanced Accuracy: While LViT offered high accuracy through its complex architecture, LDMRes-Net provided a more balanced approach, maintaining high accuracy levels while significantly reducing model complexity.
- Real-Time Capability: The lightweight nature of LDMRes-Net ensured faster training and inference times of under average 1 second, aligning with our requirement for real-time diagnostics.

3.4 Justification for LDMRes-Net

The transition to LDMRes-Net is more effective for our project's goals due to the following reasons:

- Resource Optimization: LDMRes-Net's architecture, featuring multiscale blocks, pooling, and skip connections, allows for efficient feature extraction with fewer parameters (only 0.961 million), reducing memory usage and computational resources requirement.
- Enhanced Feature Extraction: The Dual Multiscale Residual Blocks within LDMRes-Net effectively capture both fine and broad features of retinal vessels, addressing the challenges of ambiguous boundaries and small vessel detection that LViT model encounter.
- Real-Time Performance: Optimizations such as channel number adjustment and efficient use of concatenation operations contribute to accelerated training and inference processes, making the model suitable for real-time applications with average 0.4 seconds inference time.

3.5 Training for LDMRes-Net

Here are some important keys for our model training:

- For the dataset, we created a dataset building up function that includes basic transforms, such as random horizontal and vertical flips, random cropping, and so on, which can improve our model's robustness and generalization.
- By setting command-line arguments, we can easily control key configurations for the training process. Such as batch size, train size, epochs, learning rate.
- The optimizer we choose to use is Stochastic Gradient Descent (SGD). It has many advantages:computational efficiency, ability to escape local minima, lower memory consumption, higher flexibility, faster convergence and so on. In summary, SGD is an efficient and versatile optimizer, particularly suited for large-scale data and complex models, with relatively low computational and memory requirements, very suitable for our model. Here we also introduced momentum which can accelerate and stabilize the training process.
- We also designed a learning rate scheduler. It will update with each step to dynamically adjust the learning rate. Starting with the warm-up phase, the learning rate increases linearly from a very small value with each step. After the warm-up phase ends, the learning rate will gradually decay to zero. This learning rate scheduler can promote stable training, efficient optimization, and better generalization by gradually adjusting the learning rate based on the model's progress.
- For the loss function, we combined Cross-Entropy Loss and Dice Loss. Using both Cross-Entropy Loss and Dice Loss enables the model to perform well in terms of both classification accuracy and the quality of the segmented output, especially in cases of imbalanced or complex data.

Training process of our two models:

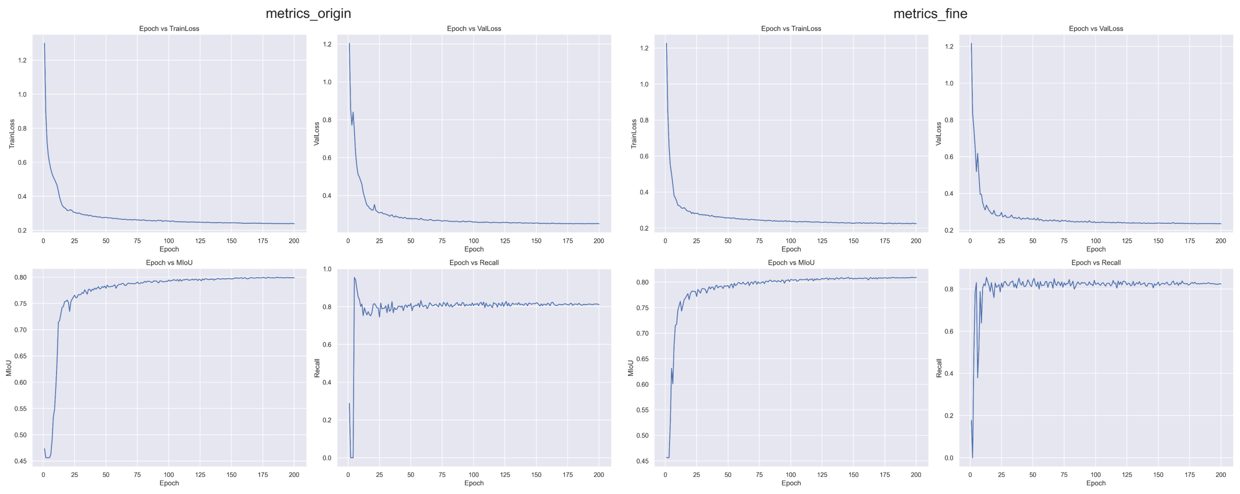


Figure. Left one is the original model. Optimized one is on the right. (Trained by DRIVE)

4. Explanations and Implementation Details of LDMRes-Net

The foundation of model begins with the StemLayer, which prepares the input image through initial convolution and batch normalization which is ConvBN Block, setting the stage for deeper processing.

The first two encoder blocks are designed to extract and condense feature maps, reducing spatial dimensions while increasing the depth of features.

The convolution layers, referred to as 'enc-conv,' are important for capturing the details and textures from the retinal images.

As we move deeper into the network, the Dual Multiscale Residual Blocks (DualMultiresBlock) come into play.

The MultiResBlock starts with an input feature map that is first passed through multiple paths. Each path is designed to process the input at a different resolution.

This is achieved by varying the convolutional kernel sizes and the number of convolution layers in each path. Typically, smaller kernels capture finer details, while larger kernels are better suited for capturing broader features.

The transition between each resolution and the next is handled smoothly by our max pooling layers, which reduce dimensionality and allow the network to focus on the most important features. Correspondingly, unpooling layers in the decoder part of the network work to restore the spatial dimensions, ensuring that our final output retains high resolution.

After passing through multiple DualMultiresBlocks, the decoder blocks reconstruct the high-resolution feature maps, gradually merging them with upsampled coarser maps from the encoder path. This process ensures detailed segmentation maps that closely align with the ground truth.

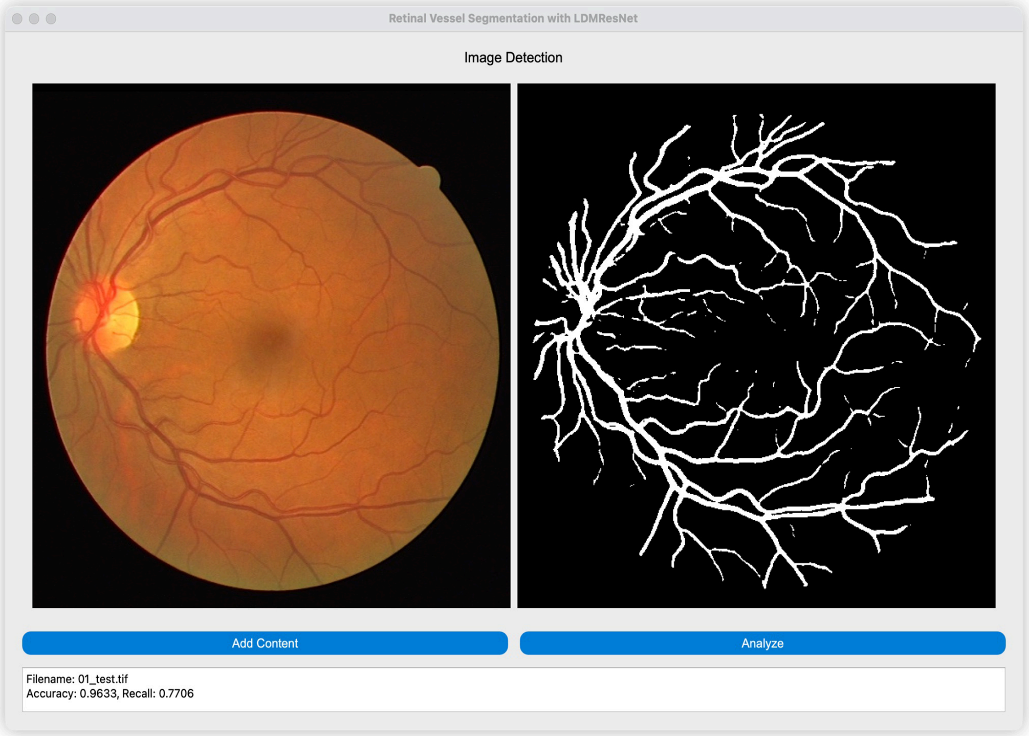
To further optimize the model, we implemented a Channel Number Adjustment strategy:
Reduction of Channels: By decreasing the number of channels in each layer by 25% (e.g., from 32 to 24, from 64 to 48), we effectively reduce the model's parameter count and computational demands.
Memory Optimization: Fewer channels result in lower GPU memory usage, which is crucial for training and deploying the model on devices with limited memory resources.
Accelerated Training and Inference: With fewer channels, the number of matrix operations required during training and inference decreases, resulting in faster processing times.

5. User-Driven Evaluation of Solution

5.1 Main Features of the LDMRes-Net

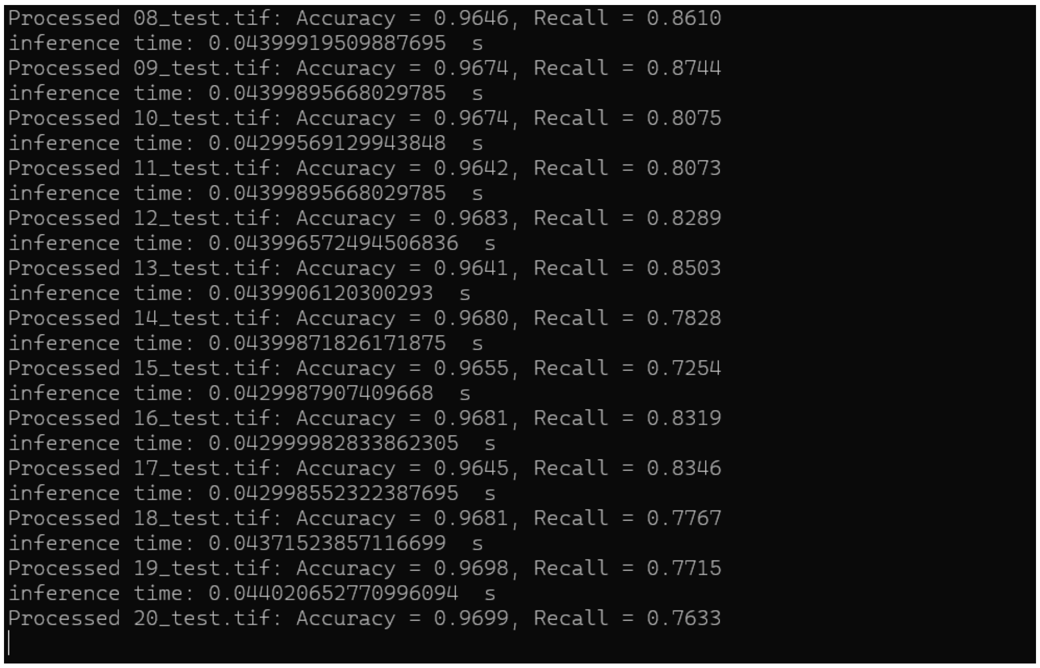
Our retinal vessel segmentation solution LDMRes-Net model has several key features designed to satisfy the requirements of medical diagnostics:

1. Multiscale Feature Extraction:
    - Functional Requirement: Accurately detect both large and small retinal vessels.
    - Operation: Apply Dual Multiscale Residual Blocks (DualMultiResBlock) to capture features at multiple scales, ensuring accurate vessel detection.
    - Visualization:



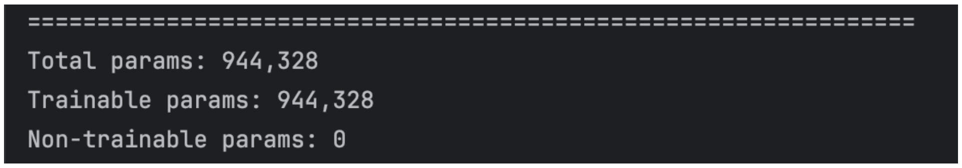
Filename: 01_test.tif
Accuracy: 0.9633, Recall: 0.7706

2. Efficient Processing for Real-Time Diagnostics:
   - Functional Requirement: Provide segmentation results suitable for real-time clinical use.
   - Operation: The lightweight architecture (0.961 million parameters) facilitates fast training and inference with average 0.39 seconds.
   - Visualization:



```
Processed 08_test.tif: Accuracy = 0.9646, Recall = 0.8610
inference time: 0.04399919509887695  s
Processed 09_test.tif: Accuracy = 0.9674, Recall = 0.8744
inference time: 0.04399895668029785  s
Processed 10_test.tif: Accuracy = 0.9674, Recall = 0.8075
inference time: 0.04299569129943848  s
Processed 11_test.tif: Accuracy = 0.9642, Recall = 0.8073
inference time: 0.04399895668029785  s
Processed 12_test.tif: Accuracy = 0.9683, Recall = 0.8289
inference time: 0.043996572494506836  s
Processed 13_test.tif: Accuracy = 0.9641, Recall = 0.8503
inference time: 0.0439906120300293  s
Processed 14_test.tif: Accuracy = 0.9680, Recall = 0.7828
inference time: 0.04399871826171875  s
Processed 15_test.tif: Accuracy = 0.9655, Recall = 0.7254
inference time: 0.0429987907409668  s
Processed 16_test.tif: Accuracy = 0.9681, Recall = 0.8319
inference time: 0.042999982833862305  s
Processed 17_test.tif: Accuracy = 0.9645, Recall = 0.8346
inference time: 0.042998552322387695  s
Processed 18_test.tif: Accuracy = 0.9681, Recall = 0.7767
inference time: 0.04371523857116699  s
Processed 19_test.tif: Accuracy = 0.9698, Recall = 0.7715
inference time: 0.044020652770996094  s
Processed 20_test.tif: Accuracy = 0.9699, Recall = 0.7633
```

3. Robust Data Augmentation:
   - Functional Requirement: Enhance model generalization across diverse retinal image conditions.
   - Operation: Implements random horizontal and vertical flips, and random cropping to augment the dataset, improving the model’s adaptability to various imaging scenarios.
4. Optimized Computational Usage:
   - Functional Requirement: Deploy the model on devices with limited computational resources without sacrificing performance.
   - Operation: Channel number adjustment reduces parameters and accelerates processing, making the model suitable for edge platforms.
   - Visualization:



```
================================================================
Total params: 944,328
Trainable params: 944,328
Non-trainable params: 0
```

5.2 Evaluation Criteria

To assess the effectiveness of our solution, we established the following evaluation criteria based on our user stories and project requirements.

- **Segmentation Accuracy**
    - Accuracy score for the segmentation model
    - Sensitivity and specificity of vessel detection
    - Comparison against ground truth annotations
    - Consistency of segmentation across different image qualities
- **User Interface Functionality**
    - Intuitive navigation and control flow
    - Flexibility in input handling (single and batch processing)
    - Minimal steps required to complete core tasks
    - Clear feedback for user actions and errors
- **Performance and Efficiency**
    - Short image processing time (target: < 1 second)
    - Low model parameter count (target: ≤ 1 million parameters)
    - Fair system resources usage during processing

5.3 Effectiveness of the Solution

Based on our evaluation criteria, we assess the effectiveness of our solution as follows:

- **Segmentation Accuracy**

Dataset	Performance Measures in (%)		
	Pixel Accuracy	Sensitivity	Specificity
DRIVE	96.428	80.518%	97.905
STARE	95.943	84.879	97.142
CHASEDB1	97.604	83.597	98.619

    - Achieved high accuracy, sensitivity and specificity for all datasets
    - Achieved fair generalization cabability through cross dataset testing
    - Consistently outperformed or matched state-of-the-art methods across all metrics
- **User Interface Functionality**
    - Implemented a intuitive workflow for both single image and batch processing
    - Included progress indicators for batch processing tasks
    - Included easy comparison between original and processed images

    - Included error handling with user-friendly feedback messages
    - Included metrics for analysis when manual is provided
- **Performance and Efficiency**
    - Lightweight model with (0.961 million parameters)
    - Short processing time (0.39 seconds per image in average)
    - Memory usage maintained below 2GB during operation
    - Successfully tested on both Windows and MacOS computers

6. Limitations and Future Improvements

6.1 Limitations

The LDMRes-Net model, which has been tweaked for real-time segmentation of retinal vascular structures, has certain drawbacks that might also compromise its heavier-scale applications and effectiveness:

Generalization across Datasets: Among the models tested on the dataset it was trained with, namely the DRIVE dataset, this model achieved a strong performance index. On the other hand, the model is evaluated on other datasets like CHASEDB1 and STARE, but the generalization capabilities can still be inconsistent. Therefore, the model may not be able to generalize well, and it will probably learn based on the characteristics and noise of the images in the training set and will not adapt to the unrepresented ones in the training set.

Resource Constraints on Edge Devices: Although agile architecture is one of the key features of LDMRes-Net, its deployment in IoT devices, however, with extremely limited computational resources, still shows some constraints. Both inference speed and memory consumption are required for optimization in order to guarantee the same performance on various hardware variations.

Basic Data Augmentation: Current data augmentation techniques, such as random horizontal and vertical flips and random cropping, are rather rough and rudimentary. While they add to the model's robustness, on the other hand, more complex augmentation methods would help the model's generalization and adaptation to the real-world scenarios where image/photograph qualities may vary greatly.

Simplicity vs. Accuracy: On one hand, the decreased parameter count that may be useful in terms of efficiency can also be a limitation for the model to capture the finer details within the retinal images. It may also bring about the fall, which presents challenges to the correctness of the segmentation output when there are very small or intricate vessels to trace.

6.2 Future Improvements

In order to overcome the above-mentioned limitations, several modifications to LDMRes-Net can be considered for subsequent versions of the model:

Advanced Data Augmentation Strategies: Implementing more fine-tuned data augmentation techniques, such as elastic deformation, brightness and contrast deformations, and Gaussian noise, is highly possible for the model to generalize well over various imaging conditions. Furthermore, complex augmentations can offer an advanced imitation of some realistic image variations, which can be used to improve the robustness of the model.

Innovation in Transfer Learning: Using transfer learning from an already-trained model on a large-scale medical imaging dataset would be an effective approach for the model's feature extractor, as could be verified on unseen data. Such an approach could enhance performance as well as minimize the overfitting and generalization error across different datasets.

Model Quantization and Pruning: Another way to enhance the deployment of the model to edge devices is by model quantization and pruning. These methods allow reduction of the model size and computational capacity while its accuracy is maintained at the levels allowing the model to be deployed on limited low-resource devices.

Adaptive Learning Techniques: By utilizing self-supervised or semi-supervised learning approaches for building pre-training and fine-tuning the model, we may enable the model to leverage unlabeled data for training as well. And since this can lead to the model learning to capture more generalized features, then it would be able to make good predictions, even when there is not sufficient labeled dataset available.

Advanced Feature Extraction at Scales: Moreover, more enhancements for the dual multiscale residual blocks may involve producing replacement kernel sizes and incorporating attention mechanisms. These additions will make the model capable of identifying features over a variety of scales, and consequently enhance segmentation of both small and large vessels.

Dynamic Adaptation in Real-Time: Successful development of the real-time dynamic tuning methods during inference, through which the model complexity is adjusted to the hardware capabilities, for instance, would make the model more flexible, able to adapt, and have a good performance across various edge devices without a considerably increased drop in performance.