Automated Severity Grading of Infectious Diseases through Deep Learning Utilizing LFA Images

S.M. Faiaz Mursalin
Department of Computer Science
Missouri State University
Springfield, MO, USA
sm943s@missouristate.edu

Khalid Hasan
Department of Computer Science
Missouri State University
Springfield, MO, USA
kh597s@missouristate.edu

***Abstract*—There is a desire for an intelligent system with the least error and immediate medical diagnosis results so that it can be utilized with people with less medical training and save clinical time. The test line beneath the control line in the Lateral Flow Assay (LFA) strip is more vivid the more contagious the disease is. This would result in medical image analysis to classify serious patients based on how infectious their disease is and quickly move them to further checkups, making them wait less behind less infectious patients. This proposed approach deals with how we can utilize a CNN-based architecture famously known as UNET to extract the Region Of Interest (ROI) and later go into post-processing to find out the curve of intensity among the control and test line.**

***Index Terms*—UNET, Lateral Flow Assay, Binary segmentation, ROI extraction, Intensity measurement**

I. Introduction

The search for intelligent, error-minimized medical diagnostic systems that deliver rapid results promises a revolution in healthcare. Particularly in the field of infectious diseases, rapid identification and classification of severity have a significant impact on patient triage, reducing clinical time and enabling efficient resource allocation. Lateral flow assay (LFA) strips are important tools in point-of-care diagnostics, providing a visual indication of the strength of the test line below the control line, an indicator of the transmissibility of the disease. Exploiting this unique property in medical image analysis provides new opportunities to rapidly stratify patients according to the level of infectiousness of their disease. This study focuses on applying convolutional neural network (CNN)-based architectures, specifically UNET, to accurately extract regions of interest (ROIs) from LFA images. The main goal is to plot the intensity of the control and test lines, which is a direct correlation with the level of transmissibility of the disease. By leveraging the capabilities of UNET, which is known for its effectiveness in semantic segmentation tasks, the proposed methodology detects spatial distributions and intensity gradients within LFA strips, allowing for a robust assessment of disease severity.

II. Related Work

The existing works involve training a huge data set through a deep learning model and later classifying them on what kind of Assays they are and later moving on to give the colorimetric results [1]. On the other hand, recurrent neural network and CMOS sensors were used to analyze the reactive protein from the LFA strips where networks like Long Short-Term Memory (LSTM) were utilized, and some classification techniques like Dynamic Time Warping (DTW) and histogram bin counts (HBC) were explored [2].

This project emphasizes the usage of Deep Neural Network (DNN) techniques to analyze the intensity of infectious disease from LFA images. In our proposed approach, at first, we will be creating our data set by applying the data augmentation technique on LFA images. Later we are going to divide the data set and go into pre-processing where we convert the image into Gray-scale and then normalize it. Then we do some standardization and go into segmentation. Some segmentation methods like edge detection and Mask R-CNN are used to get our Region Of Interest (ROI) feed into our neural network and get the intensity of the test lines. To get the continuous values of intensity, we will use a linear activation function at the output layer.

III. Methodology

An overview of the architecture and the flow we will follow is described in this section.

A. Data Preparation

We were supplied with 138 LFA strips from the Computer Science Department of Missouri State University. We then clicked images with different backgrounds and angles to produce 685 LFA images. To feed into our network we later implemented a masking tool such that for every image by drawing contours with the help of OpenCV we can get the subsequent masked images. The masked image being a binary mask has a white color for the ROI and the rest part is black. In this manner, we have manually created our data set to feed into our model such that the image can be trained with its mask and give us a predicted binary mask.

B. Model Definition

Among various types of semantic segmentation architecture like Mask R CNN, DeepLab, and UNET we implemented the UNET model from scratch as it works better with small-sized datasets. The network architecture is shown in Figure 1. It consists of a shrinking path (on the left) and a growing path (on the right). The reduction pass follows the typical architecture of convolutional networks. It consists of the iterative application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with step 2 for downsampling. Each down-sampling step doubles the number of feature channels. Each step in the expansion pass consists of up-sampling the feature map, followed by a 2x2 convolution (“up-convolution”) that halves the number of feature channels, and concatenation with the appropriately pruned feature map. The skip connections from the contracting path help to preserve the spatial information lost in the contracting path which helps the decoder layer to locate features more accurately. Cropping is required because edge pixels are lost at each crease. The final level uses 1x1 convolution to map each feature vector with 64 components to the required number of classes. In total, the network has 23 convolutional layers [4].

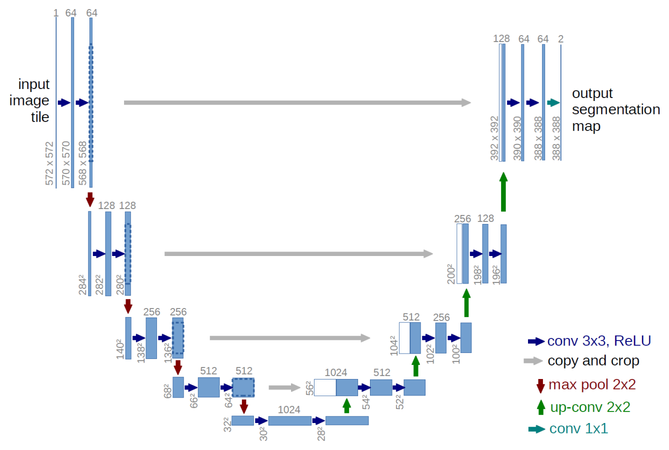


Fig. 1: UNET Architecture

C. Model Training

This module demonstrates the training process of our predefined U-Net model. Before commencing our model training, we went through several model-tuning features. This process enables the model to be trained with the best hyperparameters. Our selection of model-tuning aspects are described as follows:

- **Data Augmentation:** We artificially augmented our data set utilizing several transformation strategies like resizing, horizontal and vertical flipping, random cropping, normalizing, etc. The primary purpose of this data augmentation is to reduce over-fitting and improve the model’s ability to generalize to unseen data by exposing it to a more diverse range of examples [5]. We have employed PyTorch’s data augmentation library to perform this action.
- **Batch Size:** We applied a mini-batch of size 8 in our training process so that our computation system does not run out the memory. Also, we resized our original image to 512 X 512 so that the model does not lose enough information from the image.
- **Weight Initialization:** We have initialized the model’s weight parameters according to He initialization. He initialization is a weight initialization technique designed to work well with ReLU activation functions, addressing challenges related to vanishing gradients in deep neural networks [6].
- **Optimizer & Schedular:** We have employed PyTorch’s Adam optimizer with a learning rate of 0.0001. Adam adapts the learning rates for each parameter individually. It maintains separate adaptive learning rates for each parameter based on both the first-order momentum (like momentum optimization) and the second-order gradient information [7]. On top of this, we have utilized PyTorch’s learning rate schedular library, ReduceLROnPlateau, which incorporates our predefined Adam optimizer. This decreases the learning rate when a specified metric plateaus, in our case the validation loss. This adaptability helps the model to achieve faster convergence and better generalization, making it an essential component in the training pipeline.
- **Loss Function:** The most important part of any model training is defining the loss function. For our model, we have chosen the binary cross-entropy loss function. It is particularly suitable for problems where each instance belongs to one of two classes (e.g. in our case black or white). We have applied PyTorch’s BCEWithLogitsLoss function to compute the error. It is a variant of Binary Cross Entropy (BCE) loss that combines the sigmoid activation function and binary cross-entropy in a numerically stable way [7].

D. Post Processing

In the post-processing part of our project, we have undertaken a series of tasks that begin with cropping an original image with the help of a predicted black-white binary mask and end with plotting our expected intensity graph of an LFA strip. Figure 2 summarizes our whole tasks in a flow diagram.

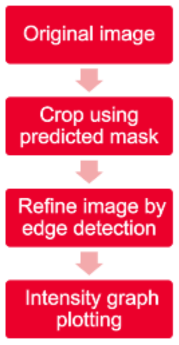


Fig. 2: Post Processing

After finishing training, the model predicts a binary mask of a specific input image. Then we proceed the post-processes as follows:

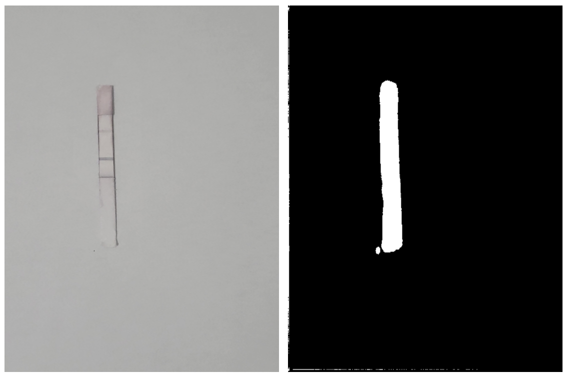
1) In the first stage, we merge the input image with the predicted mask image. Thus we get only that region from the original image which is white in the binary mask. This generates a cropped original image consisting of the exact LFA strip and moderately noisy backgrounds.
2) After generating the noisy LFA strip, we intend to refine the strip. In this case, we use an edge-detection technique and tool that assists in removing the noisy surroundings. This step ultimately returns a refined LFA strip with a transparent background.
3) This stage takes an exact LFA strip as an input and returns a line intensity plotting of the input strip. Line intensity measurement refers to the quantification of pixel intensities along a specified line or path within an image. This process is commonly used to analyze variations in intensity along a profile or to extract information from a specific region of interest. We generate an intensity profile of the LFA strip starting from the top to bottom based on the height. Afterward, we plot these line intensities in the range of the strip's height.

IV. Results

We organize our results in three steps to visualize the whole scenario starting from (1) Model prediction, (2) Region of interest (ROI) extraction, and (3) Intensity measurement.

A. Prediction

Our trained model predicts a binary mask for a given image of an LFA strip. The model finds the LFA region from the original image that makes the target area white and leaves the remaining area black.



(a) Original image (b) Predicted mask

Fig. 3: Model Prediction

Figure 4 displays the predicted binary mask 3b for a sample image 3a. We see the model has figured the region of the LFA strips out from the original image with some noisy areas.

B. ROI Extraction

(a) Noisy cropped LFA strip (b) Exact LFA strip

Fig. 4: ROI Extraction

After receiving the predicted binary mask, we merge this binary mask with the original image to crop only the LFA portion. However, Figure 4a shows the cropped image includes a bit of unexpected background along with the expected ROI. Utilizing the edge detection technique and tool, we remove this unnecessary background to get our expected ROI. Figure 4b illustrates what our extracted LFA strip looks like.

C. Intensity Measurement

In this phase, we measure the intensity of an exact LFA strip. The intensity measurement starts from the top of a strip to the bottom. Each iteration computes a line intensity starting from each pixel of the strip's height.

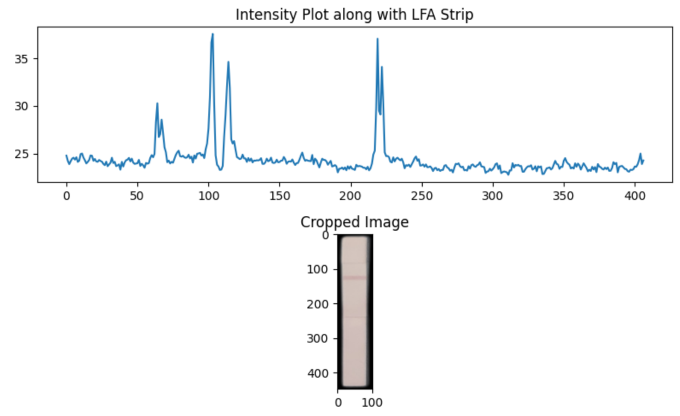


Fig. 5: Intensity measurement

Figure 5 displays an intensity plot for a given LFA strip. The graph showcases varieties of spikes starting from the height 0 to approximately 400. We can perceive three major spikes in the graph whereas other x-ticks do not have a considerable amount of spike. This behavior is commensurate with the extracted LFA strip. If we diligently observe the strip, we can find three notable lines in the image. Among them, between x-ticks 105 and 120, we see a thick line that is easily visible.

Corresponding to this thick line, we find a long and thick spike in the graph at the exact position. Moreover, we can notice the spikes approximately at x-positions 75 and 225 for their correlated lines in the image.

This experiment shows that we can identify the line intensity from any LFA strip and by doing so we can easily find out the intensity of a test line given at a certain height position.

V. Evaluation

We have evaluated our U-Net model by using pixel-level accuracy assessment. This assessment is a measure used to evaluate the performance of image processing or computer vision algorithms at the individual pixel level. It is particularly relevant in tasks such as image segmentation where the accuracy of predictions is assessed pixel by pixel. To evaluate our model, we have followed the pixel accuracy metrics which means the ratio of correctly classified pixels to the total number of pixels. In addition to that, to visualize our results' performance, we have plotted training and validation accuracy for each epoch. Figure 6 illustrates the comparison in a line graph.

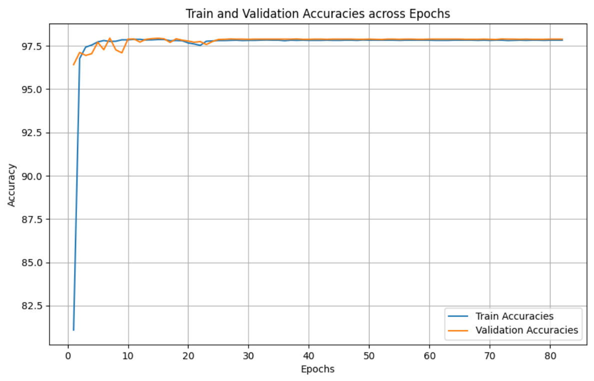


Fig. 6: Training and Validation Accuracy per Epoch

From this comparison, we notice that our model reaches above 97.5% accuracy only after 10 epochs. This holds for both training and validation data sets. After converging into that accuracy, the model maintains similar accuracy for both of these data sets.

VI. Conclusion

In this project, we have aimed to calculate the line intensities of an LFA strip from any given image. To meet this objective, we have built a U-net model that extracts an exact LFA strip from any given image with various backgrounds. Although our model can figure out the exact location of an LFA strip from any image, we get a slightly noisy background after cropping the image. We have followed the edge-detection technique and tool to remove this noisy background completely. The primary reason behind this occurrence could be the low availability of LFA image data sets. If we were able to train our model with larger data sets, then we would find a better-trained model that could extricate the LFA strip perfectly. In the future, it would be interesting to focus on this particular issue so that we do not need to depend on any other technique in the middle of the intensity measurement.

References

[1] M. H. Tania, M. S. Kaiser, K. Abu-Hassan, and M. A. Hossain, "Pathological test type and chemical detection using deep neural networks: a case study using ELISA and LFA assays."

[2] M. Jing et al., "A Novel Method for Quantitative Analysis of C-Reactive Protein Lateral Flow Immunoassays Images via CMOS Sensor and Recurrent Neural Networks."

[3] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," J. Big Data, vol. 6, no. 60, 2019.

[4] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation."

[5] Luis Perez and Jason Wang, *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*, 2017, https://arxiv.org/abs/1712.04621, `arXiv:1712.04621 [cs.CV]`.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification*, In Proceedings of the IEEE International Conference on Computer Vision, pp. 1026–1034, 2015.

[7] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization*, 2017, https://arxiv.org/abs/1412.6980, `arXiv:1412.6980 [cs.LG]`.