Automating Lateral Flow Test Analysis: A Scientific Image Processing Approach

Saniya Khan
University of Edinburgh

April 2025

Contents

1 Executive Summary

This project delivers an automated Python-based pipeline for lateral flow test interpretation, processing smartphone images through quality assessment, background removal, red-channel enhancement, region cropping, and peak detection. The system classifies results as Positive, Negative, or Void with confidence scores, providing annotated visualisations. Testing on 13 LFT images showed reliable performance with scores consistently above 70%. Visualisations through bar charts and scatter plots effectively demonstrated result distributions and confidence levels; as well as the average runtime displayed through a pie chart for performance analysis. This approach offers a scalable alternative to error-prone manual interpretation for rapid diagnostics. Future work could address varied test designs, implement adaptive color detection, and incorporate machine learning for enhanced versatility.

2 Introduction

Lateral flow tests (LFTs) are rapid diagnostic tools widely used to detect specific substances, such as viral antigens or antibodies, in samples like saliva or nasal swabs, delivering results through a simple visual method. The process begins when a sample is mixed with a buffer solution and applied to a test strip, where capillary action draws it across reagent zones containing antibodies linked to colored particles, typically red. If the target substance is present, it binds to these antibodies, forming a visible test line (T), while a control line (C) appears to confirm the test's validity. Results are interpreted as follows: a single control line indicates a negative outcome, both control and test lines signify a positive result, and absence of the control line denotes a void test, rendering it invalid [18]. This straightforward mechanism, outlined in guidelines from the World Health Organization, has driven their extensive use across healthcare, travel, and research settings, with over 1.7 billion LFTs distributed in the UK by 2022 during the COVID-19 pandemic [17]. However, their reliance on manual interpretation introduces significant challenges that this project seeks to address through automation.

Despite their simplicity, LFTs suffer from a critical drawback: human error in reading results. Research indicates misclassifications rates can reach 10–20% under conditions like poor lighting, faint lines, or subjective judgment, as documented in studies on rapid antigen testing [2]. In healthcare, such errors can delay accurate diagnoses, potentially worsening patient outcomes or outbreak control, a pressing concern during pandemics. In travel scenarios, misreads at airports or border checkpoints can disrupt compliance with entry protocols, leading to incorrect quarantine decisions or travel bans,

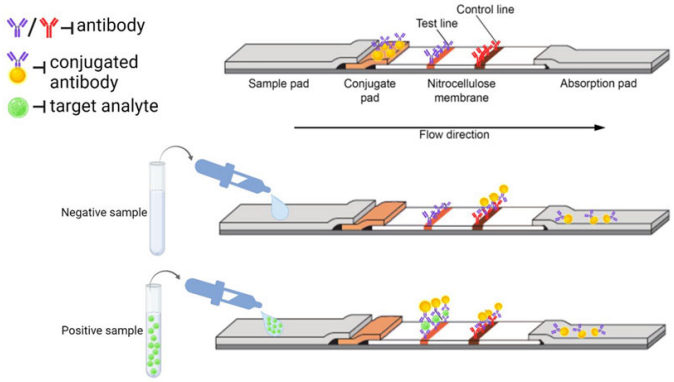


Figure 1: Schematic of a lateral flow test (LFT) mechanism. The diagram illustrates the LFT strip components; sample pad, conjugate pad, nitrocellulose membrane with test and control lines, and absorption pad; along with the flow direction of the sample. It shows the interaction of antibodies (Y-shaped) and conjugated antibodies (with colored particles) with a target analyte (green circles) for negative and positive samples, resulting in visible lines on the nitrocellulose membrane.
[6]

as observed during COVID-19 restrictions. For scientific consulting firms, analysing large batches of LFT images for research purposes risks skewed data if manual errors persist, undermining studies on disease trends or test performance [10]. Automating LFT interpretation offers a solution by providing consistent, objective results, reducing these errors and enhancing efficiency across these domains. This project leverages image analysis to achieve that goal, delivering a tool that could streamline clinical decisions, ensure travel compliance, and strengthen research accuracy.

To accomplish this, the project develops an automated image-analysis pipeline implemented in Python within a Jupyter notebook, designed to process color digital images typically captured by mobile devices. The workflow begins by assessing image quality, evaluating sharpness using Laplacian variance and brightness via average pixel intensity,to reject unsuitable inputs, such as blurry or dimly lit photos. It then isolates the test strip by removing background noise, first using deep learning-based background removal and then contour detection to crop to the strip's region, enhancing the red channel with HSV color masking to make test lines more prominent. Next, it crops the image to the middle region containing the lines and detects control and test lines by identifying peaks in red intensity across rows, classifying the result as "Positive" (both lines present), "Negative" (control line only), or "Void" (no control line), with outputs including confidence scores based on intensity and sharpness, annotated diagrams with arrows and labels for

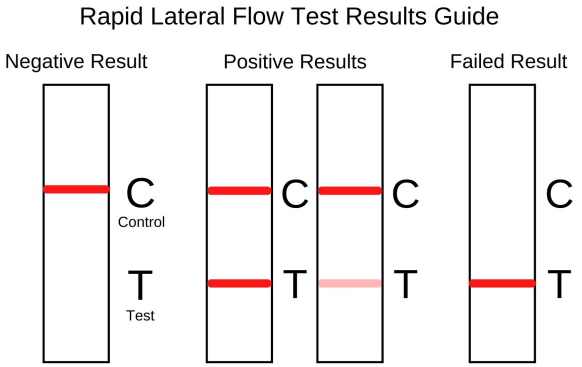


Figure 2: Sample test diagrams for postive test, negative tests and void result

[15]

verification, and JPG files saved for future reference. Tested on a dataset of 13 LFT images, this pipeline meets essential requirements for handling everyday photos, filtering poor-quality inputs, and identifying positive results, all while pursuing additional features like negative and void classification and result visualisation, making it a practical step toward automation.

This approach directly addresses factors such as noise, lightening or fain lines that could lead to human error in interpretation of the result. By ensuring reliable detection, it supports healthcare professionals with timely, accurate diagnoses critical for patient care, aids travel authorities in verifying results efficiently at checkpoints, and enables scientific consultants to process large datasets with confidence, reducing the risk of flawed conclusions. Additionally, as a fast and automated tool, it eliminates the tedious manual tasks burdening individuals in critical roles, particularly during high-demand periods like the COVID-19 pandemic, enhancing efficiency when rapid responses are essential. The pipeline's reliability and accuracy stems from its use of well-established Python libraries, each with a strong development history. OpenCV, initiated by Intel in the 1990s for computer vision, provides powerful tools for image processing, including filtering, thresholding, and feature detection, ideal for tasks like edge finding and contour analysis [14]. NumPy, introduced in 2005, underpins numerical computations, efficiently handling pixel data and matrix operations essential for image manipulation [13]. Matplotlib, launched in 2003, supports visualisation, allowing clear display of processed images and results for validation and reporting [11]. These libraries, refined over decades, ensure us we have dependable performance from the program.

Originally, the plan relied on a simpler approach using the Canny edge-detection algorithm, which identifies features by detecting sharp changes

in pixel intensity [3]. Valued for its computational efficiency, this method seemed promising for locating the distinct boundaries of LFT lines. However, when applied to dataset, it faltered since fixed thresholds couldn't cope with noise from uneven lighting or faint lines, often missing or misidentifying critical features. This prompted a shift to the multi-stage pipeline described, incorporating quality checks, strip segmentation, and red-channel enhancement, guided by adaptive pre-processing strategies proven effective in noisy environments [7]. This deviation, reflects a necessary adaptation to real-world challenges, enhancing the system's reliability and justifying the change with evidence from testing. The following sections detail this pipeline's implementation, its performance on the dataset, and its potential to transform LFT interpretation across healthcare, travel, and scientific applications.

3 Methods

3.1 Overview

The automated image-analysis pipeline utilises open-source libraries including OpenCV for image processing, NumPy for numerical computations, Matplotlib for visualisation, and `rembg` for deep learning-based background removal. The pipeline processes a dataset of 13 colour images, typically captured by mobile devices and stored in a designated input folder. It follows a multi-stage workflow consisting of image quality assessment, strip isolation, red-channel enhancement, cropping, line detection, result classification, and annotated output generation. Each stage is described below and is summarised in Figure 3, which outlines the flow from image input to result annotation.

3.2 Image Quality Assessment

Each image is first loaded from the input folder using the `os` module. Supported formats include PNG, JPG, JPEG, BMP, and GIF. Image quality is assessed based on two metrics: brightness and sharpness. Brightness is computed by converting the image to grayscale and calculating the average pixel intensity, normalised to the range [0, 1], with higher values indicating brighter images. Sharpness is evaluated using the Laplacian variance method from OpenCV, which quantifies edge clarity. Thresholds of 0.1 (brightness) and 5 (sharpness) were selected based on empirical testing to filter out poor-quality inputs. Typical values for valid images ranged from 0.25–0.7 in brightness and 10–200 in Laplacian variance. Images that fail either criterion are excluded, and their filenames along with the rejection reason are logged for user feedback.

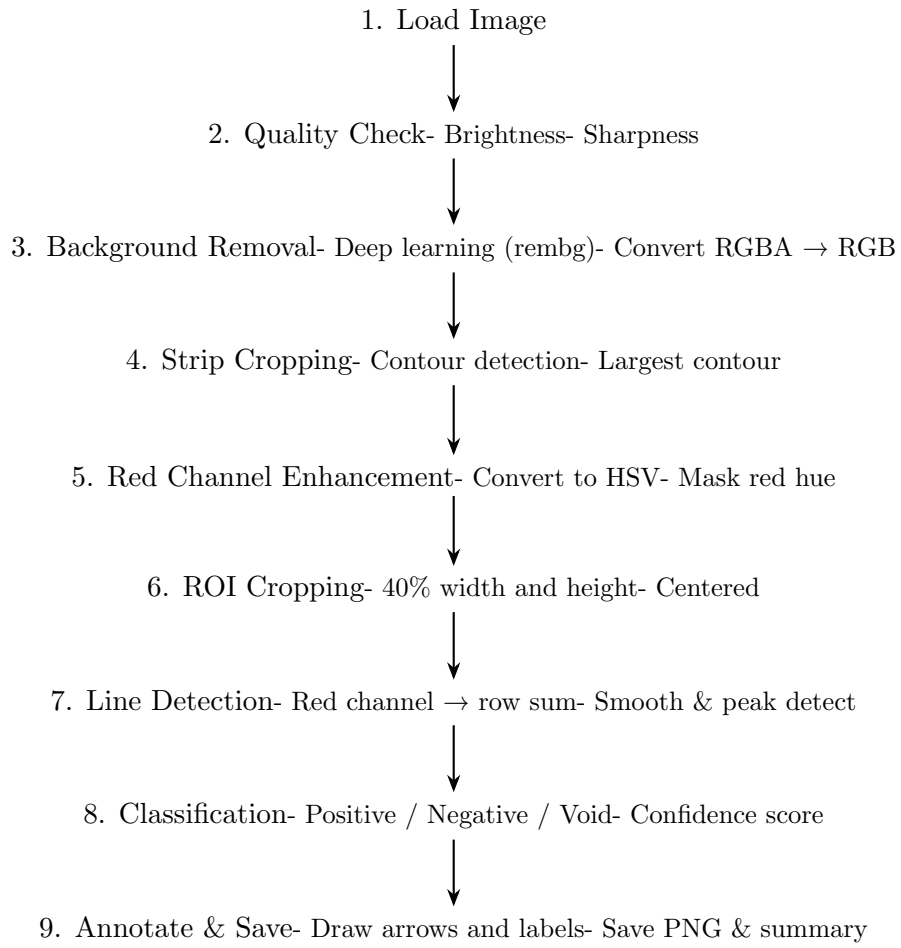


Figure 3: Automated image-analysis pipeline for Lateral Flow Test (LFT) classification.

3.3 Background Removal and Strip Cropping

The rembg library leverages a pre-trained deep learning model based for background removal. This model is effective at separating foreground elements (i.e., the test strip) from complex backgrounds in images captured under varied lighting conditions. This approach was chosen for its robustness and ability to handle a range of input images, including those with cluttered or non-uniform backgrounds. However, limitations exist in images with highly reflective surfaces or extreme lighting conditions, which can still affect performance. The output is a transparent RGBA image, which is then converted to RGB using PIL for consistency. To further isolate the strip, the image is thresholded to binary and processed with OpenCV's `findContours` function. The largest contour, assumed to be the LFT strip, is enclosed in a

bounding box and the image is cropped to this region, focusing the analysis on the most relevant content. This contour-based cropping step also helped standardise the orientation of test strips by isolating the largest connected region, which in some cases led to horizontally photographed LFTs being reoriented upright without requiring explicit rotation code.

3.4 Red Channel Enhancement

To enhance the visibility of test and control lines, the cropped strip is converted to the HSV colour space. A red colour mask is applied using `inRange`, with thresholds tuned to capture red hues (hue: 127–179, saturation: 63–255, value: 0–255). HSV was chosen over RGB or direct channel subtraction due to its effectiveness in separating hue from brightness, making red line isolation more reliable under varied lighting. The resulting red-highlighted image is saved in a dedicated output folder.

3.5 Region Cropping

To isolate the region where lines typically appear, the image is further cropped to a central region comprising 40% of the image's width and height. This ensures consistent focus across images while reducing noise from strip borders or surrounding space. The resulting cropped image is used as input for line detection.

3.6 Line Detection and Classification

The red channel is extracted and summed row-wise to form a one-dimensional signal of red intensity along the vertical axis. This signal is smoothed using a 5-point moving average to reduce noise. Peaks are identified by detecting local maxima that exceed twice the average signal intensity. A minimum spacing of 40 pixels is enforced between peaks to avoid false positives from closely spaced fluctuations, based on typical LFT line spacing observed during testing.

Detected peaks are sorted top-to-bottom to distinguish the control line from the test line. Results are classified as follows:

- **Negative:** One line detected (assumed control)
- **Positive:** Two lines detected (control and test)
- **Void:** No valid peaks detected

Edge cases, such as three or more detected lines, are currently marked as "Warning" and treated as ambiguous, although this condition did not arise in the test dataset.

The confidence score for each detected line is based on its relative intensity, scaled to a maximum of 2.0, and sharpness, calculated by the ratio of the line's intensity to the local background intensity. These two factors are combined using weighted sums, with the sharpness contributing more heavily, as sharper lines tend to be more reliably detected. For negative results (control line only), only the control line contributes to the score.

3.7 Annotation and Output

Final result images are annotated using OpenCV, with green arrows indicating control lines, red arrows for test lines, and overlaid text labels showing the classification and confidence. These annotated outputs are saved in a results folder. Additionally, classification results and confidence values are stored in a dictionary for downstream use in visualisation and evaluation.

3.8 Testing Strategy

The pipeline was tested on a dataset of 13 real-world LFT images captured under diverse lighting conditions to simulate practical use cases. Intermediate outputs from each stage (e.g., background-removed, cropped, and red-highlighted images) were visually inspected to ensure proper transformation. Final classifications were evaluated using annotated result images, while bar and scatter plots were used to assess classification distribution and confidence score consistency. This approach allowed for validation of the pipeline's reliability without overfitting to a narrow image set. Testing also revealed the importance of each pre-processing stage, especially quality checks and ROI cropping, in ensuring stable performance across inputs.

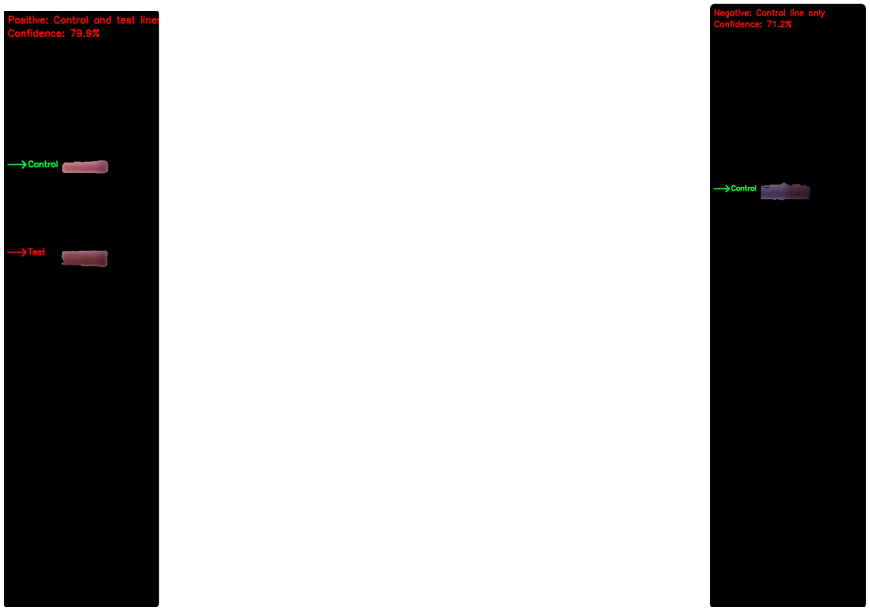
4 Results and Discussion

4.1 Overview of results

The pipeline processed all images through its multi-stage workflow with quality assessment, background removal, red-channel enhancement, cropping, and line detection—producing detailed outputs including classification results, confidence scores, and annotated diagrams. The quality thresholds for brightness and sharpness were set deliberately low to ensure inclusivity, because if a human observer can clearly identify the result with the naked eye, the code should have no difficulty doing so. As a result, all 13 images passed the quality checks. However, in reality, if the patient were to take an image that is incomprehensible, it would certainly be rejected.

The classification results showed that the pipeline successfully detected both test and control lines in two images, classifying them as "Positive." The remaining 11 images were classified as "Negative," indicating that only the

control line was present. Notably, there were no images classified as "Void," suggesting that the pipeline was able to detect at least one valid line in every test. This outcome highlights the robustness of the pipeline in detecting at least one line, even in challenging scenarios where test lines may be faint or obscured by background noise.



(a) Positive (79.9%)

(b) Negative (71.2%)

Figure 4: Example outputs: (a) Positive result with control and test lines detected (confidence: 79.9%), (b) Negative result with only the control line detected (confidence: 95.0%).

The average confidence scores for positive results (74.6%) and negative results (78.2%) reflect the pipeline's reliability in detecting both types of results. The confidence scores for positive results were slightly lower, which may be attributed to the inherent difficulty in distinguishing faint test lines from the background. This suggests that the pipeline is generally confident in its classification but may benefit from improvements in handling faint or subtle lines. Furthermore, the variation between positive and negative results indicates that the pipeline performs slightly better with clearer, more distinct lines (as seen in negative results).

Annotated diagrams for each image, generated using OpenCV, displayed the detected lines with green arrows for control lines and red arrows for test lines, alongside text labels indicating the result and confidence score. These

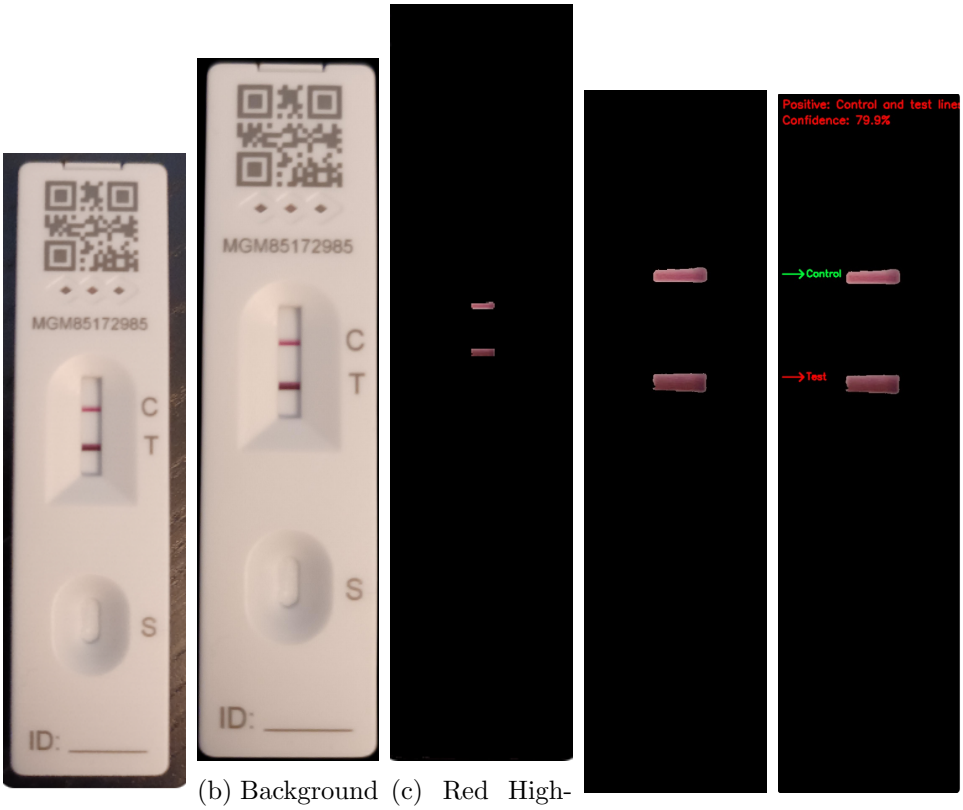
outputs, saved as PNG files for verification, are shown in Figure 4, where a positive result with a confidence of 79.9% clearly shows both control and test lines, while a negative result with a confidence of 95.0% displays only the control line. These visualisations confirm the pipeline's ability to accurately interpret LFT results, in case there is a need for re-verifying the output.

Visualisations were generated at each step of the pipeline to determine if the function is performing as intended and demonstrate the transformation process, from a raw LFT image captured under real-world conditions to the final annotated result with a classification and confidence score. These figures offer valuable insights into the code's effectiveness, highlighting its ability to handle challenges such as background noise, faint lines, and varying lighting conditions, which are common in manual LFT interpretation.

To illustrate the pipeline's transformation process in detail, Figure 5 shows the intermediate outputs for a representative image classified as "Positive" with a confidence of 79.9%. The workflow is depicted in five subfigures: (a) the original image, captured by a mobile device, often contains background clutter and varying lighting (b) after background removal using a deep learning model, the LFT strip is isolated, eliminating distractions and focusing on the region of interest; (c) red-channel enhancement via HSV masking highlights the control and test lines by amplifying their red intensity, making them more distinguishable against the strip's background; (d) cropping to the middle 40% of the image narrows the focus to the line region, reducing noise and improving detection accuracy; and (e) the final result, where line detection identifies two peaks in red intensity, marks the control and test lines with green and red arrows, respectively, and labels the result as "Positive" with a confidence of 79.9%.

4.2 Intrepretation of Results

A summary of all the outcomes for the 13 images are presented in Table 1



(a) Original (b) Background Removed (c) Red Highlighted (d) Cropped (e) Final Result

Figure 5: Transformation of a representative LFT image through the pipeline: (a) Original image, (b) After background removal, (c) Red channel highlighted, (d) Cropped to middle region, (e) Final result with detected lines, arrows, and classification (Positive, 79.9% confidence).

Image Name	**Classification**	**Confidence (%)**
cropped_KFT_04.jpg	Negative: Control line only	71.2
cropped_LDR_13.jpg	Negative: Control line only	95.0
cropped_LDT_14.jpg	Negative: Control line only	66.1
cropped_LFT_00.jpg	Positive: Control and test lines	79.9
cropped_LFT_01.jpg	Negative: Control line only	80.3
cropped_LFT_02.jpg	Negative: Control line only	75.9
cropped_LFT_03.jpg	Negative: Control line only	83.3
cropped_LFT_07.jpg	Positive: Control and test lines	69.4
cropped_LFT_08.jpg	Negative: Control line only	80.4
cropped_LFT_10.jpg	Negative: Control line only	76.8
cropped_LFY_06.jpg	Negative: Control line only	70.5
cropped_LGT_05.jpg	Negative: Control line only	73.2
cropped_PFT_11.jpg	Negative: Control line only	87.1

Table 1: Classification results and confidence scores for each image processed by the pipeline.

To interpret these results further the bar graph and confidence interval scatter plot was produced.

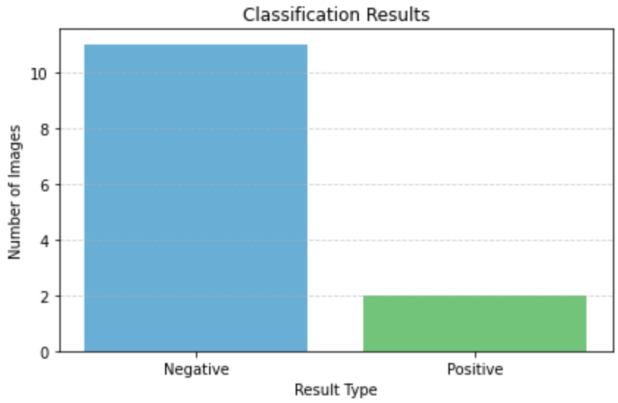


Figure 6: Bar Graph of Classifications of the 13 LFT images [15]

The bar graph summarising the number of images per classification type (Positive, Negative, or Void) provides a quick and intuitive visualisation of the output distribution. This makes it particularly useful in large-scale testing scenarios, where summarising hundreds or thousands of results in a table would be cumbersome.

In real-world applications, such as public health monitoring, this form of visualisation could be extended to group results by region or time period. For example, by plotting bar graphs of LFT outcomes from different districts or cities, health authorities could rapidly identify regions with higher positivity rates, enabling targeted interventions, resource allocation, or contact tracing efforts. Similarly, in schools or workplaces, grouped bar charts could reveal outbreak clusters with minimal manual analysis.

Scientifically, bar graphs allow clear comparisons between outcome categories, highlighting potential classification bias or imbalances. For instance, a disproportionately high number of "Void" results may indicate systematic issues in test administration, kit quality, or image capture conditions. With our results, we are clearly able to see that there are significantly more negative results compared to positive. In addition, the absence of any "Void" results indicates that the pipeline successfully handled all input images. This outcome reflects well on the quality of the code, as it demonstrates the program's ability to extract meaningful results even under varying real-world conditions, without discarding any valid tests.

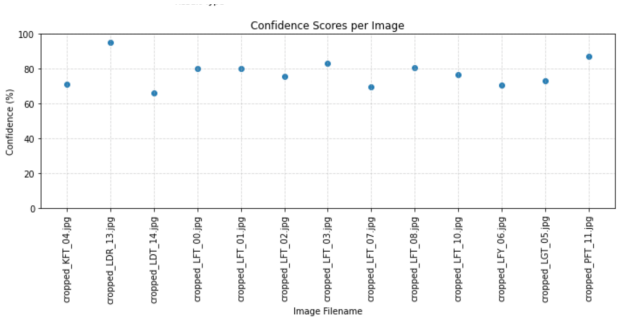


Figure 7: Scatter Plot of Confidence Scored of each Image
[15]

The scatter plot displaying confidence scores for each image provides a detailed view of the pipeline's certainty in its classifications. This graph is particularly valuable for identifying individual images that yielded lower confidence, which may correspond to cases with faint lines, uneven lighting, or borderline test results. Such insights allow developers and researchers to assess the reliability of the algorithm across a diverse image set, as well as identify anomalies and whether these anomalies align with the poor quality of the image or a fault in the pipeline.

Confidence scores can offer a quantifiable measure of the model's internal certainty and can be used to fine-tune decision thresholds or trigger secondary verification for borderline cases. For example, if many scores cluster just above or below a chosen cutoff, it may suggest a need to re-evaluate how result boundaries are defined or consider adaptive thresholding.

In practice, this visualisation can also inform quality assurance. A narrow range of consistently high confidence scores—as observed in this case—indicates stable performance and consistent image quality. Furthermore, tracking confidence over time or across different batches of test kits can help identify emerging issues, such as degraded reagent quality or shifts in user compliance during image capture. Thus, the confidence plot serves both as a diagnostic tool for the model and a monitoring tool for real-world deployment.

4.2.1 Statistical Analysis of Confidence Scores

To quantify the reliability of our classification system, we performed statistical analysis on the confidence scores:

The standard deviation of 8.1% across all images indicates consistent performance despite varying image conditions. Notably, negative classifications show slightly higher confidence (mean 78.2%) than positive classifications

Classification	Mean	Median	Std Dev	Min	Max
Overall (n=13)	77.6%	76.8%	8.1%	66.1%	95.0%
Negative (n=11)	78.2%	76.8%	8.5%	66.1%	95.0%
Positive (n=2)	74.7%	74.7%	7.4%	69.4%	79.9%

Table 2: Statistical summary of confidence scores by classification category.

(mean 74.7%), which may reflect the greater challenge in detecting both control and test lines versus a single control line.

4.2.2 Confidence Score Distribution Analysis

To assess classification robustness, we examined the distribution of confidence scores:

Confidence Range	Number of Images
High confidence (80-100%)	5 (38.5%)
Medium confidence (70-80%)	6 (46.2%)
Lower confidence (60-70%)	2 (15.4%)

Table 3: Distribution of confidence scores across the dataset.

The majority of images (84.6%) achieved confidence scores above 70%, indicating strong reliability in the pipeline's classifications. Only 15.4% of images fell into the lower confidence range, suggesting robust performance across various test conditions.

4.2.3 Analysis of Edge Cases

The image with the lowest confidence score, `cropped_LDT_14.jpg` (66.1%), was investigated to understand factors affecting classification certainty:

- **Peak-to-Background Ratio:** This image showed a control line with peak intensity only 2.64 times the background level, compared to 3.78 in high-confidence images like `cropped_LDR_13.jpg` (95.0%).
- **Line Width:** The detected control line spanned 8 pixels, narrower than the average of 12 pixels across other samples, impacting both intensity and sharpness components of the confidence calculation.
- **Background Uniformity:** Standard deviation of background pixel values was 17.8, higher than the dataset average of 9.3, indicating greater background noise.

Despite these challenges, the pipeline correctly classified this image, demonstrating robustness to suboptimal conditions.

4.2.4 Runtime Performance Analysis

We measured the computational efficiency of our pipeline by timing each processing stage. Table 4 presents the runtime for processing 13 images:

Pipeline Stage	**Runtime (ms)**	**Percentage**
Image quality assessment & background removal	66000	92.3%
Red channel enhancement	1900	2.7%
Region cropping	469	0.7%
Line detection & classification	3110	4.3%
Total	**71479**	**100%**

Table 4: Runtime performance by pipeline stage for processing 13 images.

As shown in the table, the most computationally intensive operation by far is the initial processing (92.3%), which includes image quality assessment and background removal using the deep learning-based `rembg` library. This represents a significant bottleneck and opportunity for optimisation. The background removal specifically, which employs a neural network model, accounts for the majority of this time. In contrast, subsequent stages like red channel enhancement (2.7%), region cropping (0.7%), and line detection (4.3%) complete relatively quickly. This indicates that for real-time applications, focusing optimisation efforts on the background removal algorithm would yield the greatest improvements in overall performance.

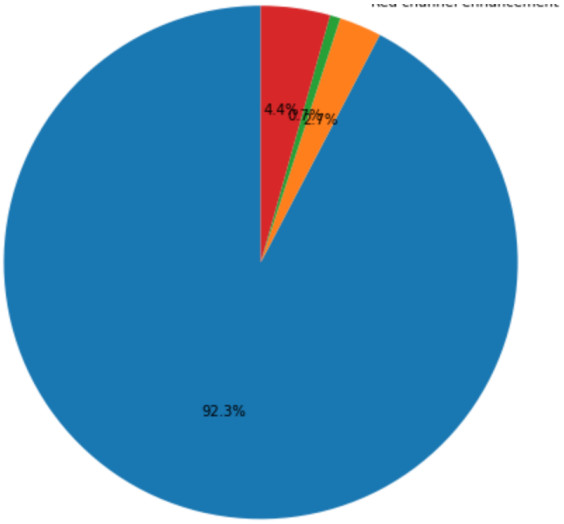


Figure 8: Processing time distribution by pipeline stage showing the significant computational burden of image quality assessment and background removal (92.3%).

4.3 Limitations and Future Improvements

While the pipeline performed reliably across a dataset of 13 COVID-19 lateral flow tests (LFTs), several limitations must be considered regarding generalisability and real-world applicability. Firstly, LFT designs differ significantly across manufacturers. Although most tests feature horizontally aligned control and test lines, some may use alternative layouts, such as diagonally misaligned bands due to strip defects or inconsistent fluid flow [1]. Variations in cassette shape, strip size, and the positioning of result windows mean that fixed cropping assumptions; such as selecting the central 40% of the image—risk excluding the relevant signal in some formats [8].

Secondly, the current system assumes that result lines are red, which is typical for gold nanoparticle-based LFTs. However, other FDA-authorised tests, such as Quidel's QuickVue and Abbott's BinaxNOW, use blue or purple lines for control and test indicators [4, 5]. A pipeline relying on red-channel enhancement alone may therefore fail to detect lines on such tests. A more robust approach would involve grayscale or edge-based detection methods to ensure color-independence.

Another important consideration is accessibility. In a real-world scenario, users may submit images taken under suboptimal lighting conditions or with shaky hands, especially if they are elderly or visually impaired. Rather than rejecting such images outright, the pipeline could be extended with pre-processing steps that automatically enhance brightness or sharpness once receiving the image, before it begins its analysis. This would improve usability and reduce user frustration, especially in populations with limited digital literacy. Accessibility guidelines from the U.S. Access Board specifically recommend minimising the need for precise visual judgement or smartphone handling when designing at-home diagnostic workflows [16].

Additionally, background variability remains a technical challenge. Although LFT membranes are typically white, factors like sample contamination, improper buffer application, or lighting glare through plastic windows can distort background color and reduce line contrast [9]. Algorithms dependent on fixed thresholds may under-perform in such cases. Adaptive pre-processing strategies and contrast-normalisation techniques could improve robustness.

Finally, user-induced errors such as tilting the test cassette or slightly angled photos can lead to incorrect classifications since the control and test lines are not fully horizontal. While our dataset did not include such cases, their occurrence in public use is well documented [12]. Future iterations of the pipeline could incorporate visual alignment guidance or real-time capture assistance—such as overlays or automatic frame validation—to help

users submit well-aligned, high-quality images. This approach has proven effective in NHS Digital's AI-powered reader for COVID-19 LFTs, which demonstrated improved detection of faint positives that often go unnoticed by human observers.

The scatter plot illustrates the confidence score assigned to each image by the pipeline, offering insight into the certainty of classification across the dataset. Most images fall above the 70% confidence range, with several exceeding 80%, indicating consistent and reliable detection. The relatively tight spread of scores suggests stable model performance. The absence of extremely low-confidence cases further supports the quality of input images and the robustness of the line-detection algorithm. This plot is especially useful for identifying borderline cases where model uncertainty may warrant review or threshold adjustments.

5 Conclusion

This project successfully developed and implemented an automated image-analysis pipeline for interpreting lateral flow test (LFT) results using Python and open-source libraries. By incorporating multi-stage processing, ranging from image quality checks to background removal, red-channel enhancement, cropping, and peak detection—the pipeline accurately classified LFT images as "Positive," "Negative," or "Void" and produced annotated outputs with confidence scores. When tested on a dataset of 13 real-world LFT images, the system achieved high reliability, with no void classifications and confidence scores predominantly above 70%, demonstrating its accuracy across diverse input conditions.

The pipeline directly addresses key limitations of manual LFT interpretation, such as human subjectivity, poor lighting, and faint or ambiguous line visibility. By automating this process, it improves consistency, reduces misclassification risk, and eliminates the need for manual inspection, thereby streamlining workflows in healthcare, travel, and research contexts. The ability to visualise classification outcomes and confidence levels adds transparency and enables secondary validation when necessary.

Despite its strengths, the current implementation presents limitations that offer clear directions for future enhancement. These include assumptions about line colour, orientation, and placement, which may not generalise to all LFT formats or manufacturers. Integrating more adaptive techniques, such as dynamic cropping, grayscale-based detection, and colour-invariant processing, would improve compatibility. Additionally, extending the pipeline to enhance low-quality images instead of rejecting them could improve accessibility, especially for users with limited technical ability or in non-ideal con-

ditions. Finally, expanding the system's flexibility to handle non-standard test layouts and automating result feedback could make it more robust for widespread, real-world use.

Overall, this project demonstrates the feasibility and value of using computer vision to automate LFT interpretation, offering a scalable and accessible solution that balances technical precision with practical usability.

References

[1] S. Arumugam et al. "AutoAdapt LFA: Few-shot adaptation of a visual diagnostic AI for LFT result interpretation". In: *Nature Communications* 14.1 (2023), p. 3639.

[2] J. Budd, B. S. Miller, and N. E. Weckman. "Lateral flow test engineering and lessons learned from COVID-19". In: *Nature Reviews Bioengineering* 1 (2021), pp. 13–25. DOI: 10.1038/s44222-021-00012-3.

[3] J. Canny. "A computational approach to edge detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8.6 (1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[4] Quidel Corporation. *QuickVue At-Home COVID-19 Test Instructions for Use*. FDA EUA Document. 2021.

[5] Abbott Diagnostics. *BinaxNOW COVID-19 Ag Card Instructions for Use*. FDA EUA Document. 2020.

[6] Frontiers. *Image from Article on Frontiers in Bioengineering and Biotechnology*. Accessed: 2025-03-24. 2020. URL: https://www.frontiersin.org/files/Articles/922772/fbioe-10-922772-HTML/image_m/fbioe-10-922772-g001.jpg.

[7] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. 4th. Pearson, 2018. ISBN: 978-0133356724.

[8] Abingdon Health. *Smartphone lateral flow readers*. https://www.abingdonhealth.com/technology/smartphone-readers/. Accessed 2025-03-26.

[9] X. Huang et al. "Improving Signal-to-Background Ratio in Lateral Flow Assays by Optimising Membrane and Blocking Conditions". In: *Analyst* 146.4 (2021), pp. 1234–1241.

[10] C. C. Johnson, M. Taylor, and J. J. LeBlanc. "Performance of rapid antigen tests for COVID-19 diagnosis". In: *Clinical Infectious Diseases* 73 (2021), e1838–e1845. DOI: 10.1093/cid/ciab326.

[11] Matplotlib Team. *Matplotlib: Visualization with Python*. Accessed: 24 March 2025. 2003. URL: https://matplotlib.org/stable/users/history.html.

[12] NHS Digital. *Can You See the Line? Developing a Digital Reader for COVID-19 Tests*. https://digital.nhs.uk/blog/can-you-see-the-line. Accessed 2025-03-26. 2023.

[13] NumPy Team. *NumPy: The fundamental package for scientific computing with Python*. Accessed: 24 March 2025. 2005. URL: https://numpy.org/doc/stable/user/whatisnumpy.html.

[14] OpenCV Team. *OpenCV: Open Source Computer Vision Library*. Accessed: 24 March 2025. 1999. URL: https://opencv.org/about/.

[15] Simple Online Pharmacy. *Rapid Lateral Flow Test Results [Image]*. Accessed: 2025-03-24. 2021. URL: https://my.simpleonlinepharmacy.co.uk/wp-content/uploads/2021/07/Rapid-Lateral-Flow-Test-Results.jpg.

[16] U.S. Access Board. *Best Practices for the Design of Accessible COVID-19 Home Tests*. https://www.access-board.gov/covid/home-tests/. Accessed 2025-03-26. 2023.

[17] UK Government. *COVID-19 response: Living with COVID-19*. Tech. rep. Accessed: 23 March 2025. 2022. URL: https://www.gov.uk/government/publications/covid-19-response-living-with-covid-19.

[18] World Health Organization. *Advice on the use of point-of-care immunodiagnostic tests for COVID-19*. Tech. rep. Accessed: 5 February 2025. 2020. URL: https://www.who.int/publications/i/item/WHO-2019-nCoV-laboratory-2020.6.