Deep Learning for Binary Classification of Malaria Cell Images

Mike Gasser
ZHAW School of Engineering
`gassemik@students.zhaw.ch`

Andrés Mock
ZHAW School of Engineering
`mockand1@students.zhaw.ch`

Abstract

Malaria is a life-threatening disease that continues to impact millions globally, particularly in regions with limited access to healthcare resources. Manual diagnosis through microscopic analysis of blood smear images is both time-consuming and heavily reliant on expert knowledge. In this project, we developed a convolutional neural network (CNN) to automate the classification of malaria-infected cells using a publicly available dataset. The dataset contained labeled images of parasitized and uninfected cells, which were preprocessed through resizing, normalization, and augmentation techniques to improve model generalization. Our baseline model achieved a classification accuracy of approximately 95%, comparable to human expert performance.

To enhance model reliability, we explored hyperparameter tuning and interpretability methods, including threshold adjustment and LIME (Local Interpretable Model-agnostic Explanations). While threshold tuning successfully reduced false negatives, a critical factor in clinical settings, visual explanations from LIME revealed limited visible differences between correct and incorrect predictions, highlighting the ongoing challenges in model interpretability. Despite these limitations, our findings demonstrate that lightweight CNNs can serve as effective diagnostic support tools. Future work should focus on incorporating more diverse datasets, applying transfer learning, and validating the model in real-world clinical environments.

1 Introduction

1.1 Background and Problem Statement

Malaria remains one of the most serious global public health issues, particularly in tropical and subtropical regions. According to the World Health Organization (WHO), there were an estimated 249 million malaria cases and 608,000 malaria-related deaths globally in 2022, with the vast majority occurring in sub-Saharan Africa (World Health Organization, 2023). The disease is primarily diagnosed through microscopic examination of blood smear slides, where trained professionals identify the presence of Plasmodium parasites. While effective, this method is both time-consuming and labor-intensive, often making it impractical in resource-constrained settings.

In recent years, deep learning and computer vision techniques have shown remarkable potential in automating medical image classification tasks (Litjens et al., 2017). CNNs, in particular, have achieved state-of-the-art performance in various diagnostic applications, including dermatology, radiology, and pathology. This project explores the application of CNNs to automate malaria detection in blood smear images, offering a faster and scalable diagnostic solution that could support healthcare workers in endemic regions.

1.2 Dataset Description

The dataset utilized in this project is publicly available on Kaggle: Malaria Split Dataset (Maestroalert, 2022). It consists of 27,558 color images of blood smears categorized into two classes: *Parasitized* (cells infected with Plasmodium parasites) and *Uninfected* (healthy red blood cells). The dataset is already partitioned into training, validation, and test sets as follows:

Set	**Parasitized**	**Uninfected**	**Total**
Train	11,024	11,024	22,048
Validation	1,378	1,377	2,755
Test	1,377	1,378	2,755
Total	**13,779**	**13,779**	**27,558**

Table 1: Dataset distribution by class and split.

Most images in the dataset had dimensions close to 128×128 pixels, although slight variations were present. These inconsistencies were addressed during preprocessing to ensure uniform input dimensions for model training.

2 Methods

2.1 Data Preprocessing

Prior to training the deep learning model, a series of preprocessing steps were applied to ensure consistent input dimensions, improve training efficiency, and enhance the model's ability to generalize to unseen data.

All images in the dataset were first resized to a uniform size of 128×128 pixels. This standardization was necessary because the original images varied in dimension, which would otherwise complicate the batch processing required by convolutional neural networks.

Following resizing, all images were converted from raw pixel data into normalized tensors. Normalization was performed using the dataset-wide mean and standard deviation of the RGB channels, which were computed across the combined training, validation, and test sets. This normalization step scaled the pixel values to approximately lie in the range [-1, 1], improving numerical tability and convergence during training, thereby preventing any one color channel from disproportionately influencing the model's learning.

The computed statistics were:

Channel	**Red**	**Green**	**Blue**
Mean	0.5295	0.4239	0.4530
Standard Deviation	0.3323	0.2682	0.2821

Table 2: Channel-wise mean and standard deviation used for normalization.

These values were subsequently used to normalize each image during preprocessing, following the transformation:

$$x' = \frac{x - \mu}{\sigma}$$

where x is the pixel value, μ the mean, and σ the standard deviation of the corresponding RGB channel. This step ensures a zero-centered input with unit variance, which is a common practice for optimizing CNN training convergence.

To enhance the model's ability to generalize and to prevent overfitting, data augmentation was applied to the training set. Three predefined augmentation configurations (`none`, `basic`, and `strong`) were implemented to allow for flexible experimentation:

- `none`: No augmentation; only resizing and normalization were applied.
- `basic`: Images were randomly flipped horizontally and rotated within a small angle range (±15 degrees).
- `strong`: A more aggressive augmentation setup including horizontal and vertical flipping, full-angle rotation (±180 degrees), and color jittering (adjustments to brightness, contrast, saturation, and hue).

These transformations simulate a wider variety of real-world imaging conditions and help the model learn more robust features. The augmentation level could be selected dynamically during training by specifying the desired setting.

Importantly, no augmentation was applied to the validation and test sets in order to maintain consistency and allow fair evaluation. For these sets, only resizing and normalization were performed.

The image data was structured in folders labeled according to class: *Parasitized* and *Uninfected*. Images from these folders were loaded and organized into batches suitable for training and evaluation. The labels were inferred from the folder names, ensuring a consistent and reliable classification of input data throughout the training pipeline.

2.2 Model Selection and Training Details

To classify malaria infection from blood smear images, we implemented a CNN, a proven architecture for image-based classification tasks. The task was defined as a binary classification problem, distinguishing between *Parasitized* (infected) and *Uninfected* (healthy) cells.

Initial Model – Baseline CNN We began by designing a lightweight custom CNN architecture, referred to as *MalariaNet*. This model consists of two convolutional layers with ReLU activation functions, each followed by max-pooling layers. The output is passed through a fully connected layer for binary classification. The model takes 128×128 RGB images as input and outputs two logits representing class scores. During inference, the class associated with the higher logit is selected as the predicted label.

Loss Function and Optimization Given the binary nature of the problem, we used the Cross Entropy Loss function, which is appropriate when working directly with logits. For optimization, stochastic gradient descent (SGD) with momentum was employed. Specifically, we used a learning rate of 0.001, momentum of 0.9, and a batch size of 32. Weight decay was also used as a form of regularization to mitigate overfitting.

Training Tools and Setup The model was implemented using PyTorch, with torchvision used for dataset loading and transformation. Training progress and performance metrics were logged using Weights & Biases (W&B), and Optuna was used for automated hyperparameter optimization. Evaluation metrics such as accuracy, precision, recall, and F1-score were computed using `sklearn.metrics`.

Flexible Architecture and Hyperparameter Tuning To explore potential improvements, we introduced a more flexible CNN variant (*MalariaNet-Flex*) that allowed tuning of key architectural and training hyperparameters, including:

Hyperparameter	Search Space
Learning rate (lr)	$[10^{-5}, 10^{-2}]$ (log scale)
Momentum	$[0.5, 0.99]$
Batch size	$\{16, 32, 64\}$
Optimizer	{SGD, Adam}
Data augmentation	{none, basic, strong}
Number of filters	$\{16, 32, 64\}$
Fully connected layer size	$\{64, 128, 256\}$
Dropout rate	$[0.0, 0.5]$
Weight decay	$[10^{-6}, 10^{-2}]$ (log scale)

Table 3: Hyperparameter search space defined for Optuna.

Hyperparameter tuning was performed using Optuna with five-fold cross-validation. Although this procedure enabled extensive experimentation, we observed that the more advanced configurations did not yield significant improvements in validation accuracy. Most configurations quickly reached performance levels similar to the baseline model.

Key Observations Despite testing different augmentation strategies and training parameters, the performance gains from hyperparameter optimization remained marginal. The baseline CNN achieved an accuracy of approximately 95% on the test set—a result that was already comparable to expert human performance and state-of-the-art models reported in the literature (Malhotra et al., 2020).

Figure 1 shows the effect of different data augmentation strategies (`none`, `basic`, and `strong`) on the validation accuracy. While augmentation led to faster generalization and slightly more stable learning in early training phases, all configurations converged to similar accuracy levels after several epochs.

In parallel, hyperparameter tuning using k-fold cross-validation was performed to evaluate combinations of learning rate, momentum, and optimizer settings.

Given these observations, the final model selection favored the simpler configuration due to its high efficiency, ease of deployment, and strong generalization performance without requiring extensive tuning.

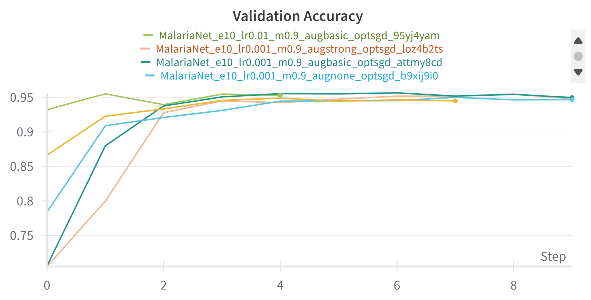

Figure 1: Validation accuracy across different augmentation strategies.

3 Results

3.1 Model Performance Summary

The final selected model, trained using the baseline CNN architecture with basic data augmentation, was evaluated on the unseen test set comprising 2,755 images. The evaluation was conducted using a classification threshold of 0.50.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Uninfected	0.9640	0.9339	0.9487	1377
Parasitized	0.9360	0.9652	0.9503	1378
Accuracy		**0.9495**		2755
Macro Avg	0.9500	0.9495	0.9495	2755
Weighted Avg	0.9500	0.9495	0.9495	2755

Table 4: Classification report on the test set (threshold = 0.50).

The model achieved an overall accuracy of 94.95% on the test set. Both classes—*Parasitized* and *Uninfected*—exhibited balanced precision and recall scores, indicating that the model is not biased toward one class and performs well in detecting true positives and true negatives alike.

3.2 Confusion Matrix Analysis

The confusion matrix further illustrates the model's performance:

- **True Positives (TP):** 1330 images of *Parasitized* cells correctly classified.
- **True Negatives (TN):** 1286 images of *Uninfected* cells correctly classified.
- **False Positives (FP):** 91 *Uninfected* cells incorrectly classified as *Parasitized*.
- **False Negatives (FN):** 48 *Parasitized* cells missed by the model.

These results suggest that the model maintains a high level of sensitivity (recall) for detecting infected cells while also keeping the false positive rate relatively low. This balance is particularly important in medical diagnostics, where both missed infections and unnecessary alerts can have serious consequences.

3.3 Threshold Adjustment and Clinical Trade-offs

To further optimize the model's clinical utility, we performed a threshold analysis with the goal of reducing false negatives—cases where malaria-infected cells are incorrectly classified as healthy. In medical diagnostics, such errors are critical, as they may result in untreated infections. Therefore, we explored adjusting the classification threshold from the default value of 0.50 to 0.35.

Class	**Precision**	**Recall**	**F1-Score**	**Support**
Uninfected	0.9746	0.9187	0.9458	1377
Parasitized	0.9231	0.9761	0.9489	1378
Accuracy		**0.9474**		2755
Macro Avg	0.9489	0.9474	0.9473	2755
Weighted Avg	0.9488	0.9474	0.9473	2755

Table 5: Classification report on the test set with adjusted threshold (0.35).

A threshold of 0.35 was selected as a balanced compromise. Compared to the default threshold of 0.50, it reduced false negatives from 48 to 33, a clinically meaningful improvement, while increasing false positives by only 21. The relative increase in false positives per avoided false negative (deltaFP / –deltaFN) was 1.5, which we considered acceptable in a diagnostic setting that prioritizes minimizing missed malaria cases over a small decrease in specificity.

At this threshold, the model achieved a slightly reduced overall accuracy of 94.74%, but a significantly improved recall of 97.61% for the *Parasitized* class. The corresponding confusion matrix is as follows:

- **True Positives (TP):** 1345
- **True Negatives (TN):** 1265
- **False Positives (FP):** 112
- **False Negatives (FN):** 33

This adjustment reflects a clinically favorable trade-off: the model becomes more sensitive to malaria infections without incurring a prohibitive number of false alarms, which aligns with diagnostic priorities in healthcare settings.

3.4 Model Interpretability with LIME

To better understand how our model makes decisions, we used LIME (Local Interpretable Model-agnostic Explanations), a method designed to explain predictions of complex models in a human-interpretable way (Ribeiro et al., 2016). LIME works by slightly modifying (or "perturbing") parts of an image and observing how the model's prediction changes. It then identifies which parts of the image contributed most to the decision.

For our analysis, we selected two examples each of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). For each case, we visualized the original input alongside the LIME-generated explanation. In the visualizations, the image is segmented into regions, and colors are used to indicate how each segment influenced the model's prediction:

- **Green areas** contributed positively to the predicted class.
- **Red areas** contributed negatively (i.e., pushed the model away from the predicted class).
- **Yellow lines** mark the boundaries between these regions.

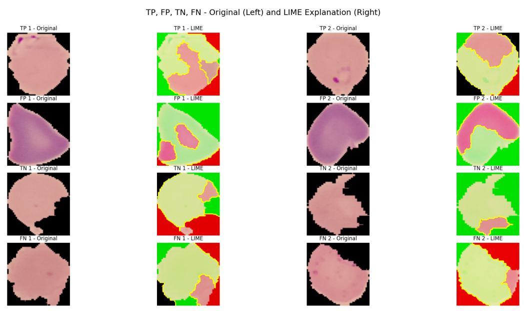

Figure 2: LIME explanations for True Positives, False Positives, True Negatives, and False Negatives. The original input images are shown on the left; corresponding LIME explanations are shown on the right.

In our case, a green region in a cell predicted as *Parasitized* indicates that the model found something resembling a parasite and used that to support its decision. In contrast, if a cell was predicted as *Uninfected*, then green regions highlight parts that convinced the model the cell was healthy. Red areas, on the other hand, point to features that contradicted the model's chosen class.

Interestingly, we were not able to identify a clear visual difference between the explanations for correct and incorrect predictions. Even in false positives and false negatives, the highlighted regions appeared similar to those in correctly classified images. This shows how subtle the model's learned features are and highlights the limitations of relying solely on visual inspection when interpreting deep learning models. LIME provides helpful insights, but interpretability remains a complex challenge, especially in sensitive applications like medical imaging.

4 Discussion

4.1 Challenges Encountered

One of the primary challenges in this project was achieving meaningful improvements beyond the baseline model. Our initial custom CNN achieved high performance—approximately 95% accuracy on the test set—with relatively few layers and a basic augmentation pipeline. Further hyperparameter tuning and architectural experiments, such as increasing filter counts or adjusting dropout and regularization, did not lead to substantial gains in model performance. This early saturation raised the risk of overfitting and made it difficult to justify additional model complexity or training time.

Another challenge lay in managing the trade-off between performance and computational efficiency. Deeper architectures, while potentially more powerful, significantly increased training time and complexity without yielding significant improvements on this specific dataset. This highlighted the importance of aligning model design with task complexity and data characteristics.

4.2 Limitations

Despite strong quantitative results, our model and pipeline are subject to several limitations. Firstly, the quality and labeling of the dataset cannot be fully verified. Since the data was collected from an open source and labeled manually, it is possible that some images may be incorrectly labeled. In medical imaging, even a small number of mislabeled cases can impact training outcomes, especially in high-stakes binary classification tasks.

4.3 On the Use of Adjusted Thresholds

We also explored the effect of adjusting the classification threshold from the default value of 0.50 to 0.35. This change was motivated by the desire to reduce false negatives, which are particularly undesirable in medical settings since they correspond to missed infections. The adjusted threshold led to a noticeable decrease in false negatives—from 48 to 33—while still maintaining a high overall accuracy of 94.74%. However, this came at the cost of an increase in false positives, which rose from 91 to 112.

In a real-world healthcare context, such a trade-off may be acceptable or even preferred. Missing an infected case can have serious health consequences, whereas a false positive simply triggers further (manual) diagnostic verification. Thus, adjusting the decision threshold to favor higher recall for the infected class could be a reasonable strategy, especially in low-resource settings where the system acts as a pre-screening tool. Nonetheless, any threshold adjustment should be validated through clinical testing and stakeholder consultation.

4.4 Future Improvements

Several directions could enhance the robustness and applicability of this work. Firstly, experimenting with deeper and more modern architectures, such as ResNet or EfficientNet, could unlock additional gains—particularly when combined with regularization techniques like batch normalization and dropout. Secondly, leveraging transfer learning from pretrained models trained on medical or microscopic imagery could improve generalization, especially when applied to more diverse datasets.

Lastly, collecting a more diverse and representative set of blood smear images from different laboratories and patient populations would be crucial for improving the model's reliability in real-world settings. Such data would not only reduce the risk of overfitting to a specific imaging style but also enhance the model's ability to generalize to new environments and edge cases.

5 Conclusion

In this project, we developed and evaluated a convolutional neural network for automated malaria detection using blood smear images. Leveraging a publicly available, pre-labeled dataset, our custom CNN achieved high performance, with an accuracy of approximately 95% on the test set. This performance is on par with expert-level diagnosis under controlled laboratory conditions and demonstrates the viability of lightweight, interpretable deep learning models in medical image classification.

Key takeaways from this work include the effectiveness of even simple CNN architectures for well-defined binary classification tasks, the importance of careful preprocessing and augmentation, and the need for thorough model interpretability—especially in healthcare applications. While the model performed well, we observed that further hyperparameter tuning yielded only marginal improvements, and LIME-based interpretability revealed the complexity of understanding model behavior in a medically meaningful way.

From a deployment perspective, adjusting the decision threshold was shown to be a viable strategy to reduce false negatives—an important consideration for clinical use, where missing an infection can have severe consequences. However, such adjustments must be validated in real-world settings with input from medical professionals.

Looking forward, this project lays the groundwork for further research involving deeper architectures, transfer learning, and more diverse datasets. Ultimately, integrating such models into diagnostic workflows could support faster and more accessible malaria screening, particularly in resource-constrained environments.

References

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AW van der Laak, Bram van Ginneken, and Clara I Sánchez. 2017. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88.

Maestroalert. 2022. Malaria split dataset. `https://www.kaggle.com/datasets/maestroalert/malaria-split-dataset`.

Rishabh Malhotra, Dhron Joshi, and Ku Young Shin. 2020. Approaching bio cellular classification for malaria infected cells using machine learning and then deep learning to compare & analyze k-nearest neighbours and deep cnns.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144.

World Health Organization. 2023. World malaria report 2023.

A Appendix

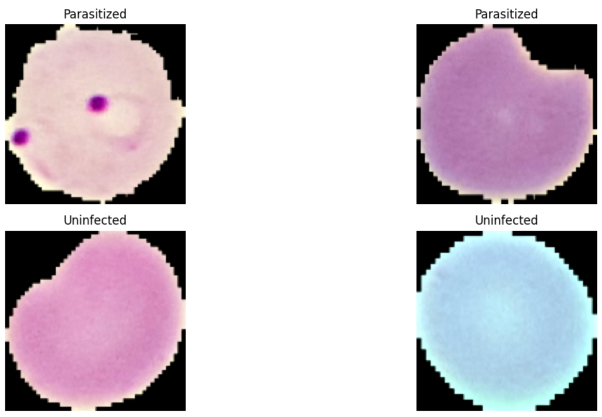

Figure 3: Sample images

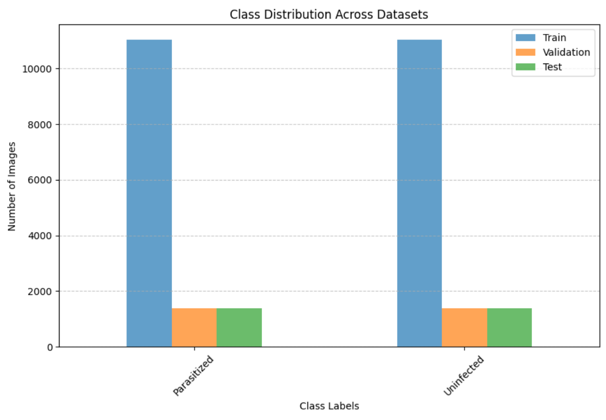

Figure 4: Class Distribution

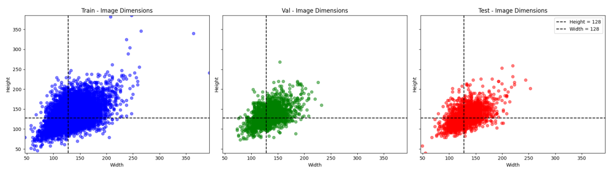

Figure 5: Image Dimensions

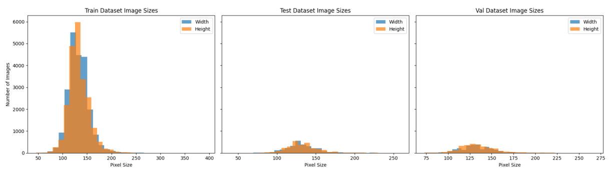

Figure 6: Image Sizes

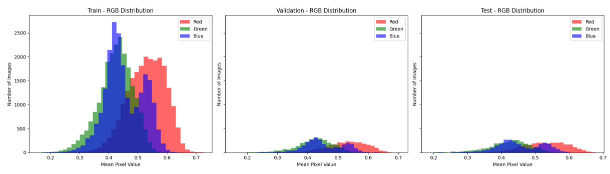

Figure 7: RGB Distribution

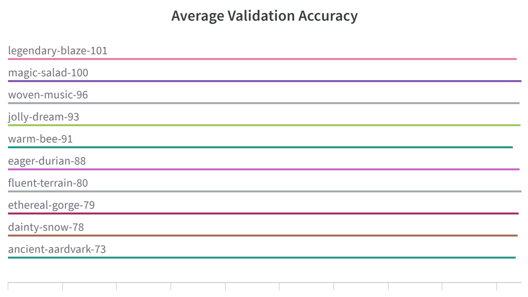

Figure 8: kfold val acc avg

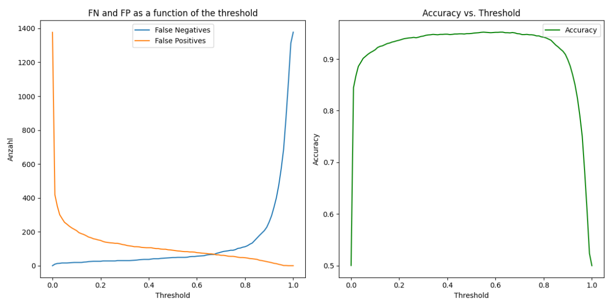

Figure 9: FN FP function threshold