Scalable Malaria Cell Detection: A Comparative Analysis of Machine Learning & Deep Learning Models on Big Image Data

Brian Keller | u1364562@utah.edu | u1364562
Felipe Marin | felipe.marin.1697@gmail.com | u1442515
Wyatt Young | young.wyatt@utah.edu | u1361848

INTRODUCTION

Detecting infected cells in microscope images is a critical task in medical diagnostics, where accurate classification can support timely and effective treatment. Motivated by the challenge of automating this process, we explore several machine learning models such as Logistic Regression, Support Vector Machine (SVM), Random Forest, and Convolutional Neural Networks (CNN) to solve a binary classification problem based on colored cell images. Our goal was to compare classical methods to determine which performs best in this visual domain.

Our results will show that the CNN model outperforms the other models in both accuracy and F1-score, confirming our hypothesis that deep learning is better suited for image-based tasks due to its ability to extract and learn spatial features. While traditional models achieved moderate performance, they struggled with the high-dimensional nature of image data. This project will go over these models classification accuracy and ability to learn features automatically. Future work could involve transfer learning from larger models, more advanced architectures, and handling multi-class classification or imbalanced data to extend the system's applicability.

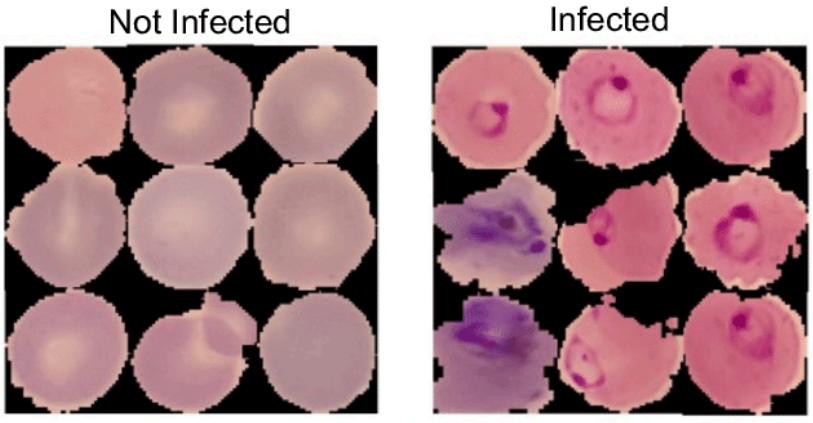
BACKGROUND

Recent advances in computer vision and machine learning have enabled automated image classification, offering promising alternatives to manual diagnosis. In particular, CNNs have shown strong performance in medical imaging tasks by learning spatial hierarchies of features directly from image data. Other machine learning methods such as Random Forests, Logistic Regression, and SVMs also provide viable classification strategies and serve as useful baselines for comparison. This project leverages these

techniques to classify cell images from blood smears, aiming to improve diagnostic accuracy and accessibility through automation.

DATA USED

For this project, we used the Malaria Cell Images Dataset available on Kaggle (https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria/data). The dataset contains 27,558 labeled images of blood infected cells, evenly split between parasitized (13,780) and uninfected (13,778) samples. Each image is a microscopic view of a red blood cell, either infected by the Plasmodium parasite or healthy. We chose this dataset due to its large size, balanced class distribution, and relevance to real-world medical diagnostics. The even class split simplifies evaluation and training, as models are less likely to be biased toward a dominant class. However, potential shortcomings include the limited diversity of the data as images come from a single source and may not fully capture variability across different populations, staining methods, or microscope types.



MOTIVATION

The core principle behind our solution is that machine learning can significantly enhance the speed, accuracy, and accessibility of medical diagnostics, specifically, malaria detection. Manual examination of infected blood cells is time-consuming, which requires trained professionals, and is prone to human error, especially in high-volume or low-resource settings. By applying machine learning to this task, we aim to automate the classification of infected and uninfected cells, enabling faster diagnosis and

reducing the manual burden on healthcare systems. Our solution explores and compares several classification approaches to identify the most effective and practical model. This approach reflects our motivation to build technology that not only solves real world problems but also has the potential for tangible impact in public health.

DESIGN

Our solution focuses on building and comparing multiple machine learning models to classify malaria infected and uninfected cell images. The overall system follows a structured pipeline encompassing data acquisition, preprocessing, model training, and evaluation.

We downloaded the raw .png image data containing an equal number of infected and uninfected red blood cell images. We used os.listdir() to load and categorize the files accordingly.

We conducted initial data exploration to inspect image dimensions and properties, which informed our preprocessing decisions. The data was then processed into two formats:

- A **greyscale dataset** (resized to 50x50) used for traditional ML classifiers (Random Forest, Logistic Regression, and SVM).
- A **color image dataset** (resized to 50x50) used exclusively for training the CNN model.

Both datasets were split into input arrays (X) and labels (y). We then visualized random samples from the dataset to ensure correct labeling and observe class balance. The greyscale data was flattened into 1D feature vectors, while the color data retained its 4D shape (height × width × channels × samples) for compatibility with convolutional layers.

Model Training 1: Random Forest Classifier

We began with the Random Forest Classifier (RFC) from sklearn.ensemble, training on the greyscale dataset. Two stages of training were conducted:

1. **Baseline Model**: A default RFC model was trained and evaluated for initial accuracy and F1-score. This provided a benchmark for future tuning.

2. **Tuned Model with Grid Search**: We performed hyperparameter tuning using GridSearchCV with 3-fold cross-validation. The parameter grid included options for:
    - Number of trees (n_estimators)
    - Maximum tree depth (max_depth)
    - Splitting and leaf thresholds
    - Feature selection strategy (max_features)

After identifying the best parameters, we retrained the RFC and observed improved performance in terms of accuracy and F1-score. A confusion matrix was plotted to visualize classification results.
This two-step approach allowed us to assess the RFC's performance under both default and optimized configurations, ensuring a fair comparison with other models.

Model Training 2: Logistic Regression
We trained a Logistic Regression classifier using the greyscale dataset. Logistic Regression offers a simple yet powerful baseline for binary classification tasks and allows us to benchmark model performance without requiring deep learning architectures.
The training process was conducted in two phases:

1. **Initial Model**:
   A Logistic Regression model with default hyperparameters (except for max_iter set to 1000 for convergence stability) was trained and evaluated. This baseline provided an early insight into the model's ability to distinguish infected vs. uninfected cells based on flattened pixel intensity values.
2. **Tuned Model with Grid Search**:
   We applied GridSearchCV for hyperparameter tuning, testing combinations of:
    - Regularization strength (C)
    - Solver algorithms (liblinear and saga)
    - Regularization types (l1 and l2 penalties)

3. This cross-validated search allowed us to find a more optimal configuration based on F1-score performance. After selecting the best parameters, we retrained the model and observed improved metrics.

As with other classifiers, a confusion matrix was plotted to evaluate the classification accuracy across both classes. This approach confirmed whether tuning provided meaningful gains in performance or if the model was limited by its linear assumptions on pixel level features.

Model Training 3: Support Vector Machine

The SVM model was trained using the greyscale dataset under the assumption that the decision boundary between infected and uninfected cells may be non-linear. For this reason, we used the radial basis function (RBF) kernel in both untuned and tuned versions of the model, as it is well-suited for complex, non-linear data distributions.
We proceeded in two stages:

1. **Initial Model**:
   The SVM was first trained with default parameters and kernel='rbf'. This baseline gave an indication of how well SVM performs on the flattened image data without prior tuning. Accuracy and F1-score were computed, and training time was recorded.
2. **Tuned Model with Grid Search**:
   Using GridSearchCV, we explored a range of hyperparameters:
   - C: Regularization parameter controlling margin vs. classification error
   - gamma: Controls the influence of individual training examples
   - kernel: Fixed to 'rbf' to maintain the non-linear mapping
3. The best parameter set was identified using 3-fold cross-validation and then applied to retrain the model. The tuned model yielded improved performance and a more optimal trade-off between margin width and misclassification rate.

As before, a confusion matrix was used to visualize prediction distribution across classes, helping validate model effectiveness beyond raw accuracy.

Model Training 4: Convolutional Neural Network

To fully utilize the rich spatial structure in the colored cell images, we trained a CNN. Unlike traditional machine learning models, CNNs can directly process image pixel grids and extract hierarchical features such as edges, textures, and shapes, making them ideal for image classification.

Two CNN architectures were evaluated:

1. **Initial CNN Model**:

   This model used a moderate-depth architecture with three convolutional layers (32, 64, and 128 filters) interleaved with max pooling to progressively reduce dimensionality. The network flattened the extracted features and passed them through a dense layer before outputting a probability via a sigmoid activation function. The model was trained using binary cross-entropy loss and the Adam optimizer for 5 epochs.

2. **Tuned CNN Model**:

   To improve performance, we increased the network's depth and width, using more filters (64, 128, and 256) and a larger dense layer. This expanded capacity allowed the model to learn more complex patterns potentially indicative of infection. Again, the model was trained for 5 epochs, with the same optimizer and loss function.

For both models, predictions were thresholded at 0.5, and performance was evaluated on the validation set using accuracy and F1-score. Training time was also recorded. As with the other models, CNNs demonstrated significant improvements when tuned, taking advantage of their ability to learn task-specific representations directly from raw image data.

EVALUATION

Each model was tested under two configurations: an initial version with default or simple settings, and a tuned version where key hyperparameters were optimized through grid search or architecture modifications. Evaluation metrics included accuracy, F1-score, and training time, all measured on a separate validation set.

Random Forest Classifier achieved solid performance out of the box, with an F1-score of 0.8180. After hyperparameter tuning, performance slightly improved to an F1 of 0.8191, showing this model is robust to parameter changes. Its training time was reasonable (around 1 minute), making it a balanced choice in terms of performance and computational cost.

Logistic Regression, as expected for a linear model, struggled with this dataset, which likely involves nonlinear and spatial patterns. The initial F1-score was 0.6401, and tuning improved it only modestly to 0.6589. Although it trained quickly, its limited capacity made it less suitable for this task.

Support Vector Machine started with a moderate F1-score of 0.7192 and actually saw a slight drop in accuracy after tuning, although its F1 improved slightly to 0.7281. Notably, **SVM** had by far the highest training time, taking over 30 minutes when tuned. This suggests diminishing returns for its computational expense, especially compared to tree-based or deep learning models.

CNN outperformed all the other models by a wide margin. Its initial accuracy was 95.0% with an F1-score of 0.9502, and tuning further improved these to 95.5%. Despite slightly higher training times than RFC or LR, the CNN was more efficient than the tuned SVM and delivered significantly better predictive performance. This highlights the CNN's strength in capturing spatial features directly from the pixel grid of colored images, a task where traditional models with handcrafted features often fall short.

Model	Initial Accuracy	Initial F1	Initial Time (s)	Tuned Accuracy	Tuned F1	Tuned Times (s)
Random Forest Classifier	0.811	0.818	58.26	0.811	0.819	60.57
Logistic Regression	0.642	0.640	21.80	0.659	0.659	36.51
Support Vector Machine	0.712	0.719	355.29	0.672	0.728	1815.73
Convolutional Neural Net	0.950	0.950	84.66	0.955	0.956	97.42

CONCLUSION

We explored multiple classification techniques for detecting infected cells where our initial hypothesis was that the CNN would outperform traditional machine learning models due to its ability to extract and learn complex spatial features directly from image data. The results clearly support this hypothesis and CNN achieved significantly higher accuracy and F1-score compared to the other models, both before and after tuning. While traditional models like Random Forest and SVM performed reasonably well, their ability to model pixel based image patterns is inherently limited. Logistic Regression, in particular, was not well suited for this specific task and underperformed. On the other hand, CNN was able to leverage convolutional layers to detect features such as edges, textures, and localized structures that are crucial in medical image analysis.

Some weaknesses and challenges include higher computational cost compared to simpler models, longer training time, especially as the model architecture becomes deeper or the dataset grows, and it requires larger datasets and more careful tuning to avoid overfitting.

Future work may explore using data augmentation to improve generalization and leverage rich feature representations, expanding the classification (like detecting

different infections) to better understand the CNN's decision making process. Overall, our findings confirm that CNNs outperform traditional methods for this image based binary classification task, validating our hypothesis and offering a strong foundation for more advanced biomedical image analysis.