Identifying Malaria in Patient Blood Samples Using Convolutional Neural Networks

Colton B. Horton

Western Governors University

D214 - Data Analytics Graduate Capstone

Dr. Daniel Smith

June 30, 2024

Abstract

This capstone project addresses the challenge of diagnosing malaria by utilizing convolutional neural networks (CNNs) to analyze images of red blood cells from patient blood samples. The primary objective is to determine if CNNs can accurately identify malaria-infected cells with at least 90% accuracy, offering a faster and more accessible diagnosis compared to traditional methods. Data comprising healthy and malaria-infected cell images were sourced from various microbiology publications, compiled by the Lister Hill National Center for Biomedical Communications, and made available through the National Library of Medicine. This paper discusses the process of building the CNN, detailing each layer and its impact on model performance, and the capabilities of the final model. The resulting CNN demonstrates high predictive performance, accurately distinguishing between healthy and malaria-infected cells with both statistical and practical significance. The model's efficacy suggests it can be a valuable tool in clinical settings, speeding up the diagnostic process and improving patient outcomes. Finally, this paper explores future studies that could further investigate the benefits of CNNs for malaria patients.

Keywords: Infectious diseases, accessible diagnostics, deep learning, medical imaging, image classification

Research Question

Can convolutional neural networks (CNNs) be used to identify malaria-infected red blood cells in patient blood samples?

Context

Malaria is a blood-borne infection caused by *Plasmodium* parasites and carried by mosquitoes that kills hundreds of thousands of people every year worldwide. Although it is a curable infection, receiving an accurate and timely diagnosis remains a significant barrier to treatment, especially in resource-limited settings (World, 2023). The gold standard of malaria diagnosis involves visually examining a "Giemsa-stained" blood sample under a microscope by a trained microbiologist. While this method is highly effective, it is impractical in areas with limited resources and few trained professionals. Collecting, staining, and viewing blood samples are simple and inexpensive processes; however, the individual microscopic analysis of each sample is both costly and time-consuming (Centers, 2024).

In many regions where malaria is prevalent, the shortage of trained microbiologists poses a significant challenge, often leading to delayed or missed diagnoses and, consequently, higher morbidity and mortality rates. Developing a quicker, yet still accurate, diagnostic method that does not rely heavily on the availability of trained microbiologists could significantly improve access to treatment and reduce malaria-related deaths.

Justification

When choosing a tool for image classification, it is important to consider the complexity of the relationships in the image. Simple images can be handled by models such as KNN, tree-based algorithms (after applying some transformations), naive bayes classifier, etc. (V, 2024).

This project proposes using a convolutional neural network to analyze microscope images due to the complexity of how the Giemsa stain interacts with human and *Plasmodium* cells. CNNs are a type of deep learning algorithm particularly well-suited for complex image analysis (Alzubaidi et al., 2021). By automating the diagnostic process, CNNs could provide a reliable and efficient alternative to traditional microbiologist examination, making it feasible to conduct large-scale screenings even in remote and under-resourced areas. This approach holds the potential to not only expedite the diagnostic process but also to alleviate the burden on healthcare systems and professionals, ultimately improving treatment access and patient outcomes.

Discussion of Hypotheses

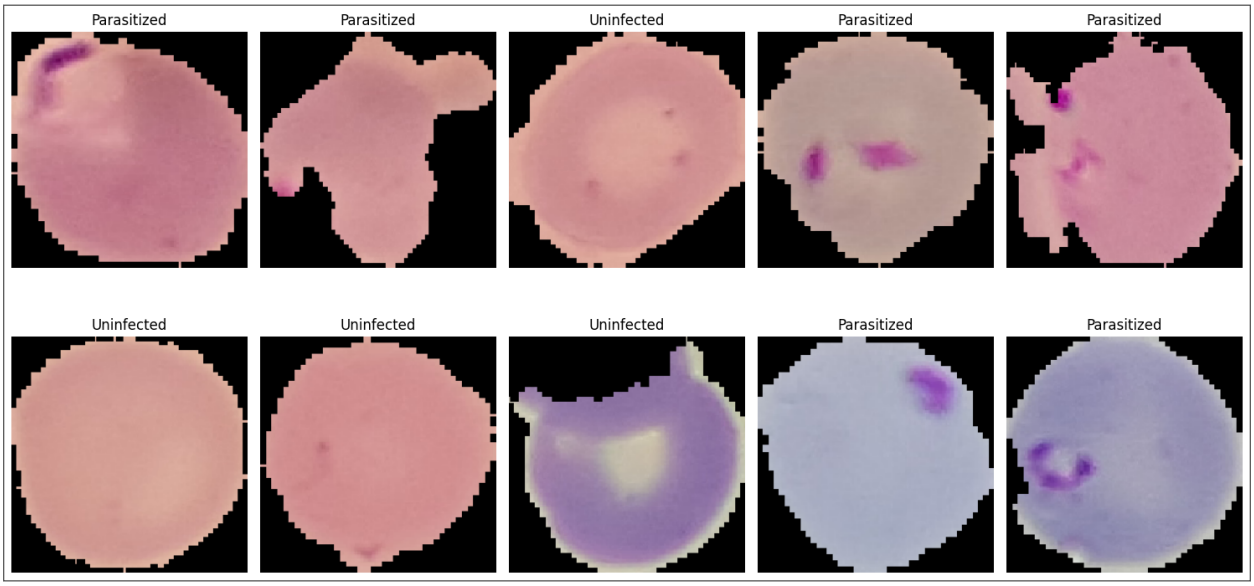
The main objectives for this project can be restated as experimental hypotheses. The null hypothesis states that a convolutional neural network cannot predict cells infected with malaria with statistical significance, achieving at least 90% accuracy. Conversely, the alternative hypothesis states that a convolutional neural network can predict cells infected with malaria with statistical significance, achieving at least 90% accuracy. These hypotheses will be formally tested using a Chi-squared test on the confusion matrix of the testing data predictions to ensure that they are statistically significant. The accuracy threshold of 90% was chosen for its practical significance in clinical settings; traditional Giemsa-stained microscopic methods conducted by trained microbiologists are approximately 90% accurate for the most common species of malaria (Stauffer et al., 2009). This will ensure that the CNN will perform at least as well as trained microbiology professionals.

Data Collection

A comprehensive and cleaned version of the relevant data for this project is available from user iarunava on Kaggle (Chakraborty, 2018). The original images were sourced from nine microbiology publications featuring images of healthy and malaria-infected blood samples. These images were specifically cropped to focus on red blood cells, as malaria targets these cells in humans. The collection was gathered together by the Lister Hill National Center for Biomedical Communications and can be accessed through the National Library of Medicine (NLM) website (Jaegar & Yang, 2021). There are 13,780 unique images of healthy red blood cells and an equal amount of images of malaria-infected red blood cells, for a total of 27,560 images in the dataset (Figure 1).

Figure 1

Sample of Healthy and Malaria-infected Red Blood Cells



During the data collection phase, one primary challenge was encountered. The NLM website organizes images by publication, complicating the process of downloading, sorting, and

storing them. Because iarunava on Kaggle had consolidated all of the images into a single, downloadable folder, simplifying access to the dataset, this repository served as the primary source for this study.

An advantage of using this dataset is its origin in published microbiology papers from peer-reviewed journals. This ensures the reliability and quality of the input data, contributing to the credibility of the analytical outcomes for this project. However, a notable disadvantage of this approach is relying on a third party for aggregating the images. While the dataset's popularity on Kaggle suggests reasonable credibility, there remains an assumption that this individual aggregated the data properly.

Data Extraction and Preparation

To extract the data, the images were downloaded from Kaggle. The dataset is organized into two folders: one for healthy cells and one for malaria-infected cells. To properly train and evaluate the neural network, these images need to be split into three groups: training, validation, and testing sets. `TensorFlow`'s `ImageDataGenerator()` function requires a specific directory structure, where each class of image is contained in separate folders within a main directory (Abadi et al., 2015).

First, Python's `os` and `shutil` packages will be used to perform a 20% split, separating the images into training and testing sets (Rossum & Drake, 2009). This will result in a main directory with training and testing subdirectories, each containing folders for the healthy and malaria-infected images. Next, `TensorFlow`'s `ImageDataGenerator()` function will be used to import both directories as training and testing sets. Additionally, the `ImageDataGenerator()` has a built-in argument to handle the 20% train-validation split

within the training sub-directory (Figure 2). By following these steps, the dataset will be properly divided for training, validating, and testing the convolutional neural network with 5,512 images for testing, 17,879 images for training, and 4,469 images for validation.

Figure 2

Data Extraction and Preparation Code

```
import os
import random
import shutil
random.seed(24)
for classification in ['1_Parasitized', '0_Uninfected']:
    source_folder = f"Data/cell_images/{classification}"
    cell_images = [f for f in os.listdir(source_folder) if f.endswith('.png')]
    random.shuffle(cell_images)
    train_test_idx = int(len(cell_images) * 0.8)
    train_files = cell_images[:train_test_idx]
    test_files = cell_images[train_test_idx:]
    for file in train_files:
        target_folder = f"Data/train_val/{classification}"
        shutil.copy(os.path.join(source_folder, file), os.path.join(target_folder, file))
    for file in test_files:
        target_folder = f"Data/test/{classification}"
        shutil.copy(os.path.join(source_folder, file), os.path.join(target_folder, file))
```

```
from keras.utils import set_random_seed
from tensorflow.keras.preprocessing.image import ImageDataGenerator
# train / validation data
set_random_seed(24)
tv_datagen = ImageDataGenerator(rescale=1/255, validation_split=0.2)
train = tv_datagen.flow_from_directory(
    'Data/train_val/',
    target_size=(192, 192),
    batch_size=32,
    class_mode='binary',
    subset='training')
val = tv_datagen.flow_from_directory(
    'Data/train_val/',
    target_size=(192, 192),
    batch_size=32,
    class_mode='binary',
    subset='validation')
# testing data
test_datagen = ImageDataGenerator(rescale=1/255)
test = test_datagen.flow_from_directory(
    'Data/test/',
    target_size=(192, 192),
    batch_size=5512,
    class_mode='binary')
```

To prepare the data for analysis, the images need to be scaled to a uniform dimension; for this project, it will be 192x192 pixels. The `ImageDataGenerator()` function is used for this purpose, as it automatically scales the images and pads the edges according to their original aspect ratios (Abadi et al., 2015). This approach is beneficial because it preserves all the information in the original photos, avoiding the loss of details that can occur with cropping. It also prevents unnecessary transformations, such as stretching the images to fit a uniform aspect ratio. One key advantage of this method is its efficiency in preparing data for use in `TensorFlow`. However, a notable disadvantage is the requirement that the original data source

be organized in local directories in order to be stored in a format compatible with `TensorFlow`, complicating the test, train, validation splitting process.

Analysis

A neural network is an artificial intelligence model inspired by the network of the human brain and its neurons. The goal of a neural network is to learn complex patterns and make predictions based on the input data. It consists of interconnected layers of nodes (or neurons), where each node contains a multiplier (or weight) and an activation function to transform the input data into an output for that particular node (Mijwel et al., 2023). Once each node of a certain layer has made its transformations, the set of outputs is then passed on to the next layer of nodes. This process repeats until a final output layer has been reached. The layers between the input layer and the output layer are called “hidden” layers and there is no fixed number or upper limit of hidden layers that a neural network must stick to, but there may exist a point of diminishing returns when increasing the number of hidden layers or nodes (Koutsoukas et al., 2017).

A convolution is a mathematical operation used in convolutional neural networks to extract features from input images. It involves sliding a filter (or kernel) across an image and computing the dot product between the filter and the local region of the image. This process produces a feature map that highlights the presence of specific patterns, such as edges or textures, within the image (Yamashita et al., 2018). It is these convolutions (or feature maps) that are passed through the neural network rather than original image itself in CNNs. Convolutions are crucial in CNNs as they enable the network to learn spatial hierarchies and detect features at

different levels of resolution and complexity, leading to effective image recognition and classification.

Creating a CNN requires careful consideration when adding each new layer to the network. This section will walk through the addition of each new layer, provide a brief explanation of its function, and discuss its impact on the model's performance. One advantage of building a neural network sequentially is that it provides a natural stopping point for model complexity. Ideally, a model should be no more complex than necessary while maintaining its predictive performance (Gupta et al., 2024). As the model's complexity increases, i.e. as more layers are added, performance will be observed through the validation set accuracy, and once it begins to plateau, the model-building process will end.

A notable benefit of neural networks is their powerful predictive performance. However, a significant disadvantage is the lack of interpretability; it's often unclear what features the model is identifying within the image as significant. Additionally, this approach requires substantial computational power and time to train each model iteration (Fan et al., 2021). Despite the lack of interpretability, increased training time, and decreased computational efficiency, the enhanced predictive performance of a well-trained neural network can justify these trade-offs.

Building the Convolutional Neural Network

To begin constructing the neural network, a foundational model needs to be established that will be applied consistently throughout the iterative building process. Firstly, each model will be a "sequential" neural network. This means that each layer passes its output exclusively to the next layer in the sequence, ensuring that no layer retrieves values from any layer other than its immediate predecessor (Montavon et al., 2018).

Regarding the compiling and training processes, the Adam optimizer has been selected for its robust generalization capabilities across a wide variety of tasks. It is also considered a reliable default choice in the industry. For the loss function, binary cross-entropy will be used due to the binary classification nature of this project (Wali, 2022). As the model optimizes the weights for each node in the neural network, it does so in iterations called "epochs." An epoch involves training the neural network on a subset of the data, gathering and storing the resulting model's parameters, then loading another batch of data to repeat the process. This epoch-based training allows for monitoring the model's performance over time and measuring overfitting. To prevent overfitting, early stopping is employed, which halts the training process when the model's performance begins to degrade in terms of generalizability (Li et al., 2024). The patience for early stopping is set to 2, meaning the neural network will cease training if the validation loss worsens for two consecutive epochs.

The initial model created was a simple neural network without any hidden layers. It consists only of an input layer and an output layer. The input layer's dimensions matched the images' height, width, and number of color channels - 192x192x3 in this case, due to the 192x192 aspect ratio and the three color channels of the images. The output layer, responsible for binary classification, was a dense layer (the standard neural network layer described in the beginning of the analysis section) with a single node and a sigmoid activation function. The sigmoid activation function ensures the output value is bound between 0 and 1. This is crucial because the output node's value will be rounded to either 0 or 1, which will then be used for the model's prediction. Additionally, a flattening layer was included just before the output layer to convert the input layer's multidimensional output into a flattened format suitable for the dense

layer. This basic model serves as a benchmark for evaluating the performance of the following models in the iterative training process. After training this neural network on the dataset, the final epoch's validation accuracy was 0.512. This low accuracy was expected, given the absence of hidden and convolutional layers, highlighting the model's limitations as a classifier in its current form.

To increase the model's complexity, an initial convolutional layer was introduced. This layer consists of 16 filters, each with a 3x3 kernel size, and uses the ReLU activation function. The ReLU (Rectified Linear Unit) activation function helps the network by introducing non-linearity, which allows it to learn and model more complex patterns. Additionally, ReLU speeds up the training process by mitigating the vanishing gradient problem, which can occur when the model's training process is slowed down by the magnitude of the weights being trained (Hu et al., 2021). By applying convolutional operations, this layer captures essential features such as edges, textures, and shapes, which significantly improved the model's performance. After training with the addition of this convolutional layer, the validation accuracy of the final epoch increased to 0.724, a notable improvement over the initial model's accuracy of 0.512. However, the model is starting to exhibit significant overfitting. By the time the early stopping ended the fitting process, the training accuracy was 0.972, much higher than the validation accuracy.

As the model's complexity increases, the impact of overfitting also becomes more pronounced. One effective method to mitigate overfitting is to incorporate dropout layers. Dropout layers work by randomly pruning node connections between layers. Although it may seem counterintuitive, this random exclusion of connections helps improve the model's generalizability to new data and reduces the gap between training and validation accuracy during

the training process (Srivastava et al., 2014). Dropout layers take a single parameter as a value between 0 and 1 that represents the proportion of connections to be pruned when connecting nodes to the next layer. For simpler models and the early hidden layers of more complex models, a lower dropout rate is recommended. As layers get deeper, they identify more complex relationships and are more prone to overfitting, thus requiring a higher dropout rate. For the current model, which includes only one hidden convolutional layer, a dropout layer with a dropout rate of 0.1 was added. This adjustment increased the validation accuracy to 0.740 and brought the training accuracy down to 0.956. While the gap between training and validation accuracy has been reduced, it is still present. This gap is expected to shrink further as the model complexity continues to increase and more dropout layers are added. Moving forward, a dropout layer will be added to the end of each additional layer.

To further build the model, a hidden dense layer with 32 neurons and the ReLU activation function was introduced. This layer increases the model's capacity to interpret the complex patterns identified by the convolutional layer. However, the addition of this dense layer has significantly increased the model's size from 2.2 MB to 70.5 MB. As the model continues to grow in complexity, its size will also expand, which could lead to challenges in storage and computation. Therefore, optimization techniques are necessary to manage this increasing size effectively.

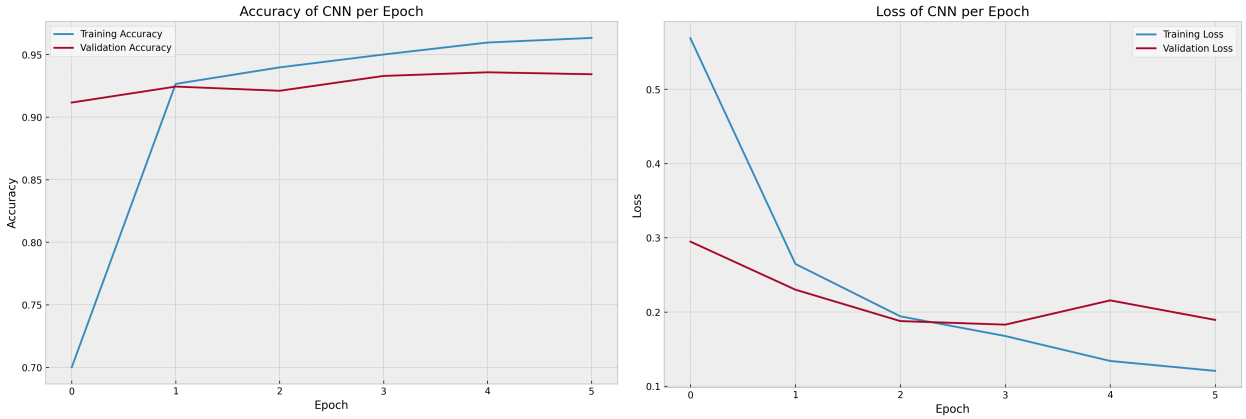
To address the issue of increasing model size and complexity, a max pooling layer was added. The max pooling layer reduces the spatial dimensions of the feature maps generated by the convolutional layers by using a window function to take the maximum pixel value for each rolling window, effectively downsampling the input. This operation decreases the number of

parameters and computational load of building the neural network, making the model more efficient. Additionally, max pooling helps to make the learned features more robust to variations and distortions in the input images (Graham, 2015). For this model, a window size of 3x3 was chosen. With the incorporation of both the dense layer and the max pooling layer, the model's validation accuracy increased to 0.897, demonstrating a significant improvement in performance while maintaining a more manageable size of 7.8 MB. Moving forward, a max pooling layer will be added to the end of each additional convolutional layer.

Continuing to refine the model architecture, an additional convolutional layer was introduced. This layer employs the ReLU activation function and utilizes 32 filters with a 3x3 kernel size. The primary objective of this convolutional layer is to further enrich the feature extraction process. By applying multiple convolutional filters, the network becomes capable of capturing increasingly complex patterns and details within the images. This addition significantly strengthened the model's performance; following training, the validation accuracy increased to 0.929 in the final epoch.

There will be one final layer added to the model because the improvements are beginning to plateau: one additional dense layer with 32 nodes and the ReLU activation function. The primary objective of this dense layer is similar to the objective of the additional convolutional layer: to further enrich the feature extraction process. By including multiple dense layers, the network increases its capability to interpret the patterns identified in the convolutional layers. The final validation accuracy of the CNN is 0.934, the final training accuracy is 0.964 (Figure 3). The final model architecture includes an input layer, two convolutional layers, two max pooling layers, a flattening layer, two dense layers, an output layer, and four dropout layers (Figure 4).

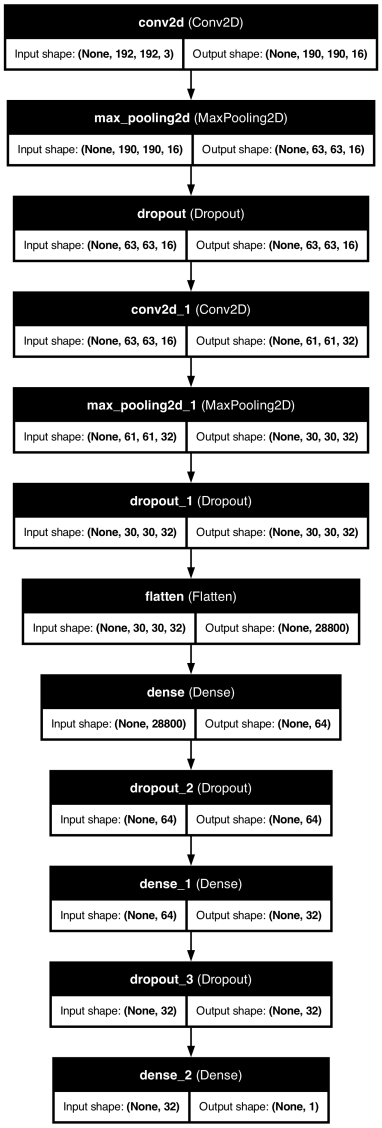
Figure 3



Note. The x-axis is zero-indexed. The model fitting ran for six epochs prior to early stopping.

Figure 4

Final CNN Model Architecture and Code




```
model = Sequential()
model.add(Input(shape=(192, 192, 3)))

model.add(Conv2D(16,(3,3), activation='relu'))
model.add(MaxPool2D(3,3))
model.add(Dropout(0.1))

model.add(Conv2D(32,(3,3), activation='relu'))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))

model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.3))

model.add(Dense(32, activation='relu'))
model.add(Dropout(0.4))

model.add(Dense(1, activation='sigmoid'))
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 190, 190, 16)	448
max_pooling2d (MaxPooling2D)	(None, 63, 63, 16)	0
dropout (Dropout)	(None, 63, 63, 16)	0
conv2d_1 (Conv2D)	(None, 61, 61, 32)	4,640
max_pooling2d_1 (MaxPooling2D)	(None, 30, 30, 32)	0
dropout_1 (Dropout)	(None, 30, 30, 32)	0
flatten (Flatten)	(None, 28800)	0
dense (Dense)	(None, 64)	1,843,264
dropout_2 (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 32)	2,080
dropout_3 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 1)	33

Total params: 1,850,465 (7.06 MB)

Trainable params: 1,850,465 (7.06 MB)

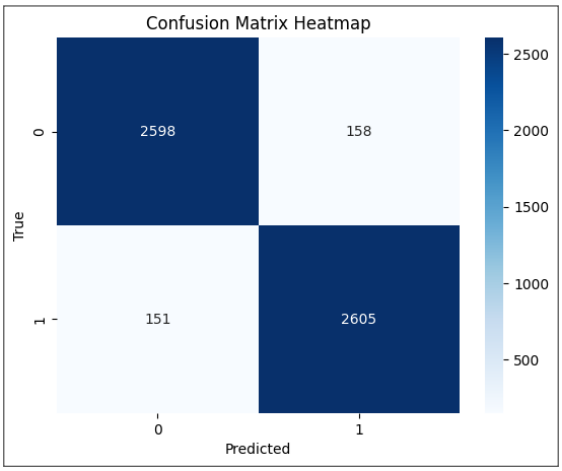
Non-trainable params: 0 (0.00 B)

Testing Data Evaluation

Now that the model has been built and trained, the testing data can be used to quantify how well the models performs on unseen data. There are two aspects to this evaluation: the statistical significance and the practical significance of the results. The goal is to have results that are both statistically significant ($\alpha = 0.05$) and practically significant (accuracy >= 90%). To assess both of these, the model will make predictions on the testing images and then the true image labels will be used together with the predictions to make a confusion matrix (Figure 5). A Chi-squared analysis will be used to determine the statistical significance of the predictions and some raw summary statistics will be used to measure the practical significance.

The results of the Chi-squared test demonstrate that the predictions are statistically significant with a test statistic of 4341.7, making the p-value approximately equal to zero. The accuracy of the model for predicting the testing images is 94.4%, meaning that the predictions are practically significant as well. Additionally, the false positive and false negative rates are approximately equal to 5.6%, indicating a good balance that does not favor one side of the predictions over the other.

Figure 5



Data Summary and Implications

Discussion of Results

The null hypothesis of this analysis states that a convolutional neural network cannot predict cells infected with malaria with statistical significance, achieving at least 90% accuracy. However, given that the p-value of the chi-square test on the confusion matrix of the predictions is approximately zero and the overall accuracy of the model on the testing images is 94.4%, The null hypothesis is rejected. This result indicates that the CNN can accurately classify blood cell images as either healthy or malaria-infected with both statistical and practical significance.

The implications of this finding are significant for regions where malaria is prevalent and resources are limited. By automating the diagnostic process, this CNN model provides a reliable and efficient alternative to traditional microscopic examination by trained microbiologists. This can accelerate the diagnostic process, alleviating the burden on the healthcare system in those areas, and potentially reduce malaria-related deaths. The success of this project also highlights the potential of machine learning in other global health initiatives.

However, a limitation of this project is the requirement for the technology needed to host the CNN at the testing sites, which may not always be feasible. Many remote and under-resourced areas may lack the necessary infrastructure, such as reliable electricity, to support the deployment of certain technological solutions like this CNN. Addressing this limitation will require careful consideration of each potential testing location and its unique challenges. Solutions may include creating portable, self-sufficient diagnostic devices that can operate independently of existing infrastructure.

Recommended Course of Action

The recommended course of action is to partner with humanitarian and medical aid programs to assess how and where this CNN can address needs and relieve strain on the medical system in the areas they serve. These collaborations can identify areas that would benefit most from the technology and devise strategies to integrate the CNN into existing healthcare frameworks. By working with these organizations, it ensures that the implementation is effective and sustainable, ultimately improving access to timely and accurate malaria diagnosis in resource-limited settings.

Future Work

Moving forward, this project could benefit from two additional analyses. Firstly, a comprehensive image processing approach should be considered. The images that were downloaded for this project had a preprocessing step applied to them prior to this analysis that isolated the individual red blood cells from the larger blood work images. This initial segmentation may have been manually conducted or done by an edge-detection artificial intelligence model due to the size of the dataset. While that model is beyond this project's scope, a subsequent analysis could focus on replicating or validating this preprocessing step.

Secondly, further investigation into minimizing the false negative rate is essential. In medical diagnostics, a false negative result can have severe consequences for patients if they delay necessary, life-saving treatment (Maxim et al., 2014). While the current model exhibits equal rates of false positives and false negatives, optimizing the model to reduce false negatives could significantly improve patient safety. This could involve adjusting the prediction threshold of the final output node from the conventional ≥ 0.5 to a slightly lower threshold like ≥ 0.45.

However, such adjustments require careful consideration, as lowering the threshold may increase false positives and reduce overall model accuracy. Therefore, any threshold modification must be justified by the potential benefits in patient outcomes.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. https://www.tensorflow.org/

Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of Deep Learning: Concepts, CNN Architectures, challenges, applications, Future Directions. *Journal of Big Data*, *8*(1). https://doi.org/10.1186/s40537-021-00444-8

Centers for Disease Control and Prevention. (2024, April 24). *Malaria diagnostic tests*. Centers for Disease Control and Prevention. https://www.cdc.gov/malaria/hcp/diagnosis-testing/malaria-diagnostic-tests.html

Chakraborty, A. (2018, December 5). *Malaria Cell Images Dataset*. Kaggle. https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria

Fan, F.-L., Xiong, J., Li, M., & Wang, G. (2021). On interpretability of Artificial Neural Networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, *5*(6), 741–760. https://doi.org/10.1109/trpms.2021.3066428

Graham, B. (2015, May 12). *Fractional max-pooling*. arXiv.org. https://arxiv.org/abs/1412.6071

Gupta, V., Li, Y., Peltekian, A., Kilic, M. N., Liao, W., Choudhary, A., & Agrawal, A. (2024). Simultaneously improving accuracy and computational cost under parametric constraints in materials property prediction tasks. *Journal of Cheminformatics*, *16*(1). https://doi.org/10.1186/s13321-024-00811-6

Hu, Z., Zhang, J., & Ge, Y. (2021). Handling vanishing gradient problem using artificial derivative. *IEEE Access*, *9*, 22371–22377. https://doi.org/10.1109/access.2021.3054915

Jaegar, S., & Yang, F. (2021, October 27). *Malaria datasheet*. U.S. National Library of Medicine. https://lhncbc.nlm.nih.gov/LHC-research/LHC-projects/image-processing/malaria-datasheet.html

Koutsoukas, A., Monaghan, K. J., Li, X., & Huan, J. (2017). Deep-learning: Investigating deep neural networks hyper-parameters and comparison of performance to shallow methods for modeling bioactivity data. *Journal of Cheminformatics*, *9*(1). https://doi.org/10.1186/s13321-017-0226-y

Li, H., Rajbahadur, G. K., Lin, D., Bezemer, C.-P., & Jiang, Z. M. (2024, January 18). *Keeping Deep Learning Models in Check: A History-Based Approach to Mitigate Overfitting*. arXiv. https://arxiv.org/html/2401.10359v1

Maxim, L. D., Niebo, R., & Utell, M. J. (2014). Screening tests: A review with examples. *Inhalation Toxicology*, *26*(13), 811–828. https://doi.org/10.3109/08958378.2014.955932

Mijwel, M. M., Esen, A., & Shamil, A. (2023). Overview of Neural Networks. *Babylonian Journal of Machine Learning*, *2023*, 42–45. https://doi.org/10.58496/bjml/2023/008

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and Understanding Deep Neural Networks. *Digital Signal Processing*, *73*, 1–15. https://doi.org/10.1016/j.dsp.2017.10.011

Rossum, G. V., & Drake, F. L. (2009). *Python 3: Reference manual*. SohoBooks.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(1).

Stauffer, W. M., Cartwright, C. P., Olson, D. A., Juni, B. A., Taylor, C. M., Bowers, S. H., Hanson, K. L., Rosenblatt, J. E., & Boulware, D. R. (2009). Diagnostic performance of rapid diagnostic tests versus blood smears for malaria in US clinical practice. *Clinical Infectious Diseases*, *49*(6), 908–913. https://doi.org/10.1086/605436

V, N. (2024, June 12). *Image classification using machine learning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2022/01/image-classification-using-machine-learning/

Wali, R. S. (2022, June). *A tunable loss function for binary classification problems*. arXiv. https://arxiv.org/pdf/2211.00176

World Health Organization. (2023, December 4). *Fact sheet about malaria*. World Health Organization. https://www.who.int/news-room/fact-sheets/detail/malaria

Yamashita, R., Nishio, M., Do, R. K., & Togashi, K. (2018). Convolutional Neural Networks: An overview and application in Radiology. *Insights into Imaging*, *9*(4), 611–629. https://doi.org/10.1007/s13244-018-0639-9