ADS2002

Automated Detection of Malaria-Infected Cells from Microscopic Images using Deep Learning

Prepared by:

Tashvin Ramesh 34675280

Dayvin Rushil Athirathan 33589070

You Wei Lim 34069518

Daniel Ong 34897887

Kalubowilage Niduli Sudewna Alwis 34144021

Lee Yoong Shuen 33522200

Table of contents

1. Executive Summary …….. 4.

2. Introduction ……... 6.

- What is Malaria?..... 6.
- The Problem Statements…… 6.

3. Data Quality……..... 7.

- Dataset Overview……... 7.
- Data Quality Issues Identified……..... 7.

4. Exploratory Data Analysis (EDA) ……10.

- Visual Characteristics and Class Discrimination……....10.
- Comprehensive Image Dimension Analysis……...10.
- Preprocessing Pipeline Development……10.

5. Data Cleaning (Mislabelled Image Identification).....12.

- Scikit-learn Perceptron Model……12.
- Multi-Layered Perceptron (MLP) Model……12.
- K-Means Clustering Classifier using Pretrained……13. MobileNetV2 Model
- K-Fold Out-of-Fold (OOF) Validation using……..14. Pretrained ResNet18 Model

6. Convolutional Neural Network Model Development.....16.

- Motivation…….. 16.
- Dataset Preparation and Splitting……16.
- CNN Architecture……16.
- Training Setup and Parameters……17.
- Model Performance and Results.....…… 18.

7. Improved Convolutional Neural Network Model.....20.

- Motive of Improvement……..20.
- Improved CNN Architecture……...20.
- Improved Training Setup and Parameters……...21.
- Improved Model Performance and Results…….22.

8. Conclusion…….....23.

9. References…….....24.

10. Appendix…….....28.

Executive Summary

The Findings

Malaria remains one of the deadliest diseases and health challenges faced by many, with a death rate of hundreds of thousands of deaths annually. According to the World Health Organisation (WHO), in the African region alone, there is a devastatingly high share of the total global malaria burden, accounting for 93% of cases and 94% of deaths. In resource-limited settings, such as in Africa, where malaria is the breeding ground, there is often a shortage of skilled personnel and adequate infrastructure. In this project, a cost-effective method of automated detection of malaria-infected red blood cells needed to be implemented effectively. This would aid medical professionals in coming up with sound decision-making strategies that could help to potentially mitigate future infections. To motivate this project, four key questions were formulated to guide the analytical process and model development:

1. In what ways can red blood cells be classified as either “Uninfected” or “Parasitized”?
2. Since there were more than 10,000 images each for both datasets, is there any misclassification between them?
3. What are the possible neural network architectures that could be developed as the final model?
4. From these models, which architecture provides the best accuracy and generalisation performance when predicting unseen images?

The Data

The data that was provided was obtained through the Kaggle website in which contained two datasets named “Uninfected.jpg” and “Parasitized.jpg” contained in a zip file. Both these files had 13,779 images each, which made it a balanced dataset; however, they did contain mislabelled images. Since this was a major issue, data cleaning needed to be done to ensure that when the CNN model is trained on the dataset, it would not misinterpret and mislearn from the incorrect data, making the model perform poorer than it should.

Exploratory Data Analysis

A key observation in both datasets was that there were many images with varying image resolutions. This made some of the images clearer and some dull. The top 10 most common

images were plotted using a histogram to understand the different image sizes. It was clear that the inconsistencies would have also led to the mislabelling of the images. The most common pixel size was 130x130, however, 128x128 was the most viable option because it would fit into the RGB scale for images.

The Findings

Comprehensive data cleaning was performed to find the misclassified data within the dataset. Firstly, a simple artificial neural network called a perceptron model containing a single layer was used. This model had a testing accuracy of 54.17% which is not good for classification, as it is randomly guessing. Secondly, the Multi-layered perceptron (MLP) model was also used for the identification of misclassified images. This model layers multiple perceptrons to build a stronger classification model. This MLP model had a testing accuracy of 55.61% and thirdly, a K-Means Clustering Classifier using a pretrained model called MobileNetV2 was used to extract features into two different groups. However, this classifier did not perform well as the two clusters contained images from the parasitized and uninfected images. Lastly, a K-Fold-Out-of-Fold (OOF) Validation using a pretrained ResNet18 Model gave us successful results. A high accuracy score was obtained for predictions of misclassified images. These images were then removed from the dataset to ensure the CNN model being developed will not mislearn incorrect data.

A CNN model was then developed for the image classification section. The architecture consisted of 3 convolutional layers in order to learn patterns within the images to correctly classify unseen images. This CNN model was trained and gave a testing accuracy of 96.94%. This meant that the model performed well in generalising unseen data, and was able to make correct predictions almost 97% of the time. To further improve the base CNN model, several improvements were made to the base CNN model, including data augmentation to the training image set, increasing the depth of layers to 5, which helped to capture more features, enhancing classification ability, and adjusting the parameters for model training. This improved CNN model has obtained a testing accuracy of 98.16% which means that it performs better than the base CNN model in classifying unseen data. Therefore, the goal of the project was achieved in developing a well-trained deep learning model that was able to identify malaria-infected cells from microscopic images.

Introduction

What is Malaria?

Malaria is a life-threatening disease caused by Plasmodium protozoa, a parasite spread by the Anopheles mosquito (Bria et al., 2020). People who have this disease have symptoms that include fever, fatigue, and headaches, with some people even having seizures, coma, or death in severe cases (Bria et al., 2020). In this project, a robust and well-trained Convolutional Neural Network (CNN) for image classification was developed in order to automate the detection of malaria-infected cells. Throughout this report, the implementation of deep learning symbolised the need for support in healthcare systems in order to provide efficient, stable, and cost-effective tools for medical diagnosis.

The Problem Statements

The underlying issue faced in this project pertained to how the detection of malaria in red blood cells could be improved through the usage of neural networks in order to develop a model that would be able to generalise unseen data. The data provided had only two main classifications, called “Uninfected” and “Parasitized”. The four key questions that were discussed below laid the building blocks for the analytical approach that was used throughout this project:

- In what ways can red blood cells be classified as either “Uninfected” or “Parasitized”?
- Since there were more than 10,000 images each for both datasets, is there any misclassification between them?
- What are the possible neural network architectures that could be developed as the final model?
- From these models, which architecture provides the best accuracy and generalisation performance when predicting unseen images?

Answering these questions would provide a solid foundation for the data analysis, which would aid in decision-making strategies for medical diagnostics. Moreover, the insights obtained from this project could potentially aid healthcare professionals in developing more accurate and reliable diagnostic tools to help enhance research and reduce mortality rates.

Data Quality

Dataset Overview

The dataset used in this analysis contains malaria cell images from Kaggle. It includes two main folders, one with uninfected cell images and another with parasitized cell images. The initial setup involves importing key Python libraries such as PIL (for image processing), matplotlib (for visualization), and os (for file handling).

The first step was to check how many images were in each folder. The code loops through both folders and counts valid image files (jpg, jpeg, png) using `os.listdir()` and list comprehension. The dataset contains 27,558 images, equally split between the two categories, which are 13,779 uninfected and 13,779 parasitized. This perfect 50/50 class balance is ideal for binary classification since it removes the need for balancing techniques like oversampling or undersampling (Mujahid et al., 2024).

Data Quality Issues Identified

a) Visual Sample Analysis and Misclassification Detection

To better understand the dataset, sample images from each class were displayed using matplotlib. Five images from both uninfected and parasitized folders were shown in a grid using subplots (shown in Figure 1), with titles added and axes removed for a cleaner view.

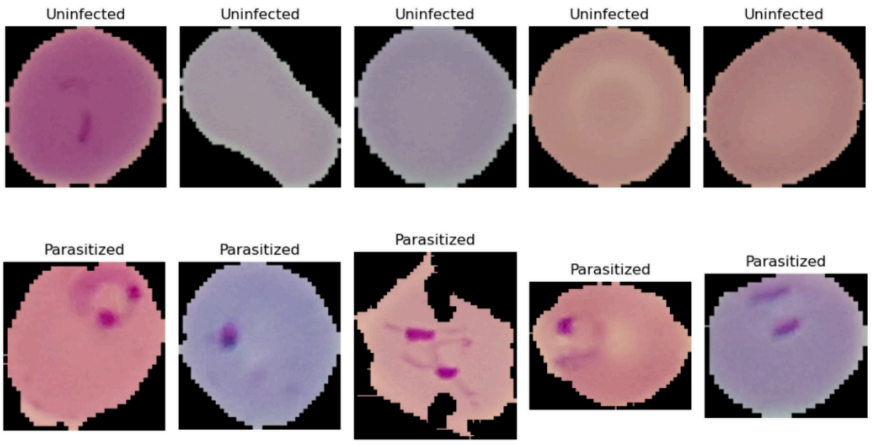


Figure 1: Five images from both uninfected and parasitized folders

When visually inspecting the samples, one uninfected image showed dark spots, which are a feature typical of infected cells. Normally, uninfected cells have smooth, light purple to pink

coloring with no dark inclusions. In contrast, parasitized cells contain dark purple or black spots representing the malaria parasite. This suggests some mislabeling in the dataset, possibly from human error or ambiguous cases. Such errors can introduce noise into the training data and slightly reduce model performance.

b) Image Size Inconsistency Analysis

Another major issue was the inconsistency of image sizes. The code checked the dimensions of every image using PIL's `.size` attribute and counted unique sizes using Python's `Counter` class. Results in Figure 2 showed 1,627 different image sizes, with the most common being 130×130 pixels, which appeared only 224 times (less than 1% of the total). Figure 3 shows the top 10 sizes ranged roughly between 121–136 pixels, showing no standard size across images.

```
1627 different size shapes present in combined dataset
Most common size: (130, 130) pixels
Number of images with this size: 224
```

Figure 2: Outputs showing the number of different-sized shapes, the most common image size, and the number of images with this size.

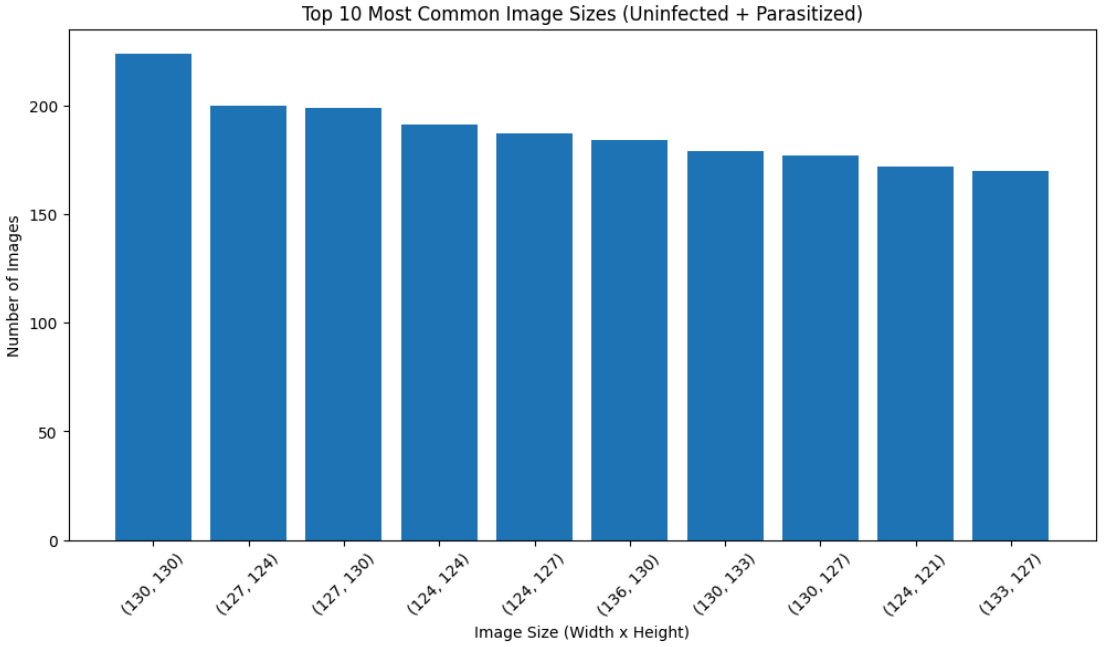


Figure 3: Bar chart. Top 10 most common image sizes for both uninfected and parasitized.

The bar chart of the top 10 sizes (Figure 3) made this inconsistency clear. Such variation suggests the images were collected from different sources or microscopes. Since machine learning models require fixed input sizes, this variability means all images must be resized

before training. Without this step, models cannot process batches or perform matrix operations effectively (Saponara & Elhanashi, 2022).

c) Image Format Assessment

All images are stored in .jpg format. This consistency is helpful because it avoids the need for file conversion. However, JPEG uses lossy compression, which can reduce image detail and introduce artifacts that might blur parasite structures (Shukla, 2023). Despite this, having one consistent format is still a positive aspect for preprocessing.

Exploratory Data Analysis (EDA)

Visual Characteristics and Class Discrimination

The EDA highlighted clear visual differences between the two classes. Uninfected cells are smooth and evenly colored in light purple or pink, while parasitized cells show dark spots (malaria parasites), varying in size and shape. These differences are visually obvious and provide a strong basis for classification. However, the presence of a few ambiguous or mislabeled images suggests that classification is not completely straightforward.

Comprehensive Image Dimension Analysis

A full scan of all 27,558 images confirmed the earlier finding of 1,627 unique sizes. The most common size (130×130) appears only 0.8% of the time, and even the top 10 sizes together make up less than 7% of all images.

Most images fall roughly between 121–136 pixels, indicating similar magnification but no fixed standard. This variation causes problems because both traditional ML models and neural networks require consistent input shapes. Neural networks, for example, process data in batches that must share the same dimensions. Therefore, image resizing becomes a necessary preprocessing step before training (Saponara & Elhanashi, 2022).

Preprocessing Pipeline Development

Based on the findings from the EDA, a comprehensive preprocessing pipeline was developed to address the identified data quality issues and prepare the dataset for model training. The implemented code applies several key transformations that standardize the data and optimize it for deep learning techniques. Each preprocessing step was designed to ensure that the dataset is both consistent and computationally efficient for subsequent modeling tasks.

a) Image Standardization Through Resizing

The first major preprocessing step involves resizing all images to a uniform dimension. The code defines `img_size = (128, 128)` as the target resolution for all images. This size was selected for several practical and strategic reasons. First, 128 pixels closely matches the most common dimensions observed in the dataset (ranging from 121 to 136 pixels), which minimizes the amount of upscaling or downscaling required and helps maintain image

quality. Second, 128 is a power of two, a property that is computationally efficient for many machine learning operations, especially in neural networks, since it optimizes memory access patterns and convolution operations (Muneeb et al., 2019).

The resizing process is implemented using the `resize()` method from the PIL library within the `load_images_from_folder()` function. Each image is opened, converted to RGB format to ensure three color channels, and resized to the target dimensions.

b) Pixel Value Normalization

The next critical preprocessing step involves normalizing pixel values from their original 0–255 range to a 0–1 scale. In digital images, each pixel's intensity is represented as an integer between 0 (no intensity) and 255 (maximum intensity) for each of the three color channels (red, green, and blue). The code performs normalization by dividing each pixel value by 255.0, converting it into a floating-point value between 0.0 and 1.0.

Normalization aligns the dataset with common practices in image-based machine learning, improving comparability with established benchmarks and facilitating potential transfer learning (Pei et al., 2023).

c) Dataset Splitting Strategy

Lastly, the processed data was split using `train_test_split()` with 80% training and 20% testing, using `stratify=y` to keep class proportions equal and `random_state=42` for reproducibility. The training set (22,046 images) trains the model, while the test set (5,512 images) evaluates its performance on unseen data. After splitting, StandardScaler is applied to standardize feature values (zero mean, unit variance). Importantly, the scaler is fitted only on the training data and then applied to the test data to avoid data leakage.

Data Cleaning (Mislabelled Image Identification)

Scikit-learn Perceptron Model

A perceptron is one of the simplest forms of an artificial neural network. It's a binary classifier that maps its input to an output value (Du et al., 2022). It takes multiple inputs, applies weights to them, sums them up, and uses an activation function to produce an output (Du et al., 2022). The Scikit-learn Perceptron model is selected as it only implements single-layer perceptrons, which is good to use for testing to identify mislabelled images in the dataset.

In our approach, the model is configured as a pure perceptron without regularization, the learning rate is 0.0001, and the tolerance is 0.001. The model is trained with 20 epochs and gives a result as shown in Figure 4 below.

```
=== Perceptron Performance ===
Accuracy: 0.5417
Confusion Matrix:
[[1352 1404]
 [1122 1634]]
Class 0 = Uninfected | Class 1 = Parasitized
Precision: 0.54
Recall: 0.59
F1 Score: 0.56
```

Figure 4: The Scikit-learn Perceptron model performance

The model has an average performance, obtaining a testing accuracy of 0.5417. The confusion matrix has no high numbers in the diagonal, showing that it doesn't divide predictions into correct categories well. The precision, recall, and F1 score are considerably poor. Thus, the Scikit-learn Perceptron model is not suitable for use.

Multi-Layered Perceptron (MLP) Model

A Multi-Layered Perceptron (MLP) is a foundational deep learning model. Generally, it consists of densely connected layers where each node connects to every node in the subsequent layer (Bikku, 2020). We use an MLP model in the PyTorch library as another method to identify mislabelled images in the dataset.

The model is defined by a feedforward neural network with two hidden layers, utilising Rectified Linear Unit (ReLU) activation, and an output layer with Sigmoid activation, which

is used for binary classification tasks, as it maps the output to a range between 0 and 1.
The model is then compiled with the Adam optimizer, binary cross-entropy loss function, and learning rate of 0.001. It is trained with 20 epochs and gives a result as shown in Figure 5.

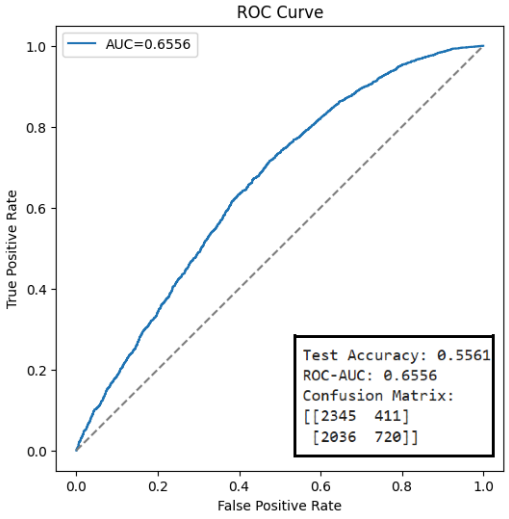


Figure 5: The performance and ROC Curve of the MLP model

Although the model slightly performs better than the Scikit-learn Perceptron model, its performance is still average. The AUC in the ROC Curve indicates a 65.56% probability that the model will rank a randomly chosen parasitized image higher than a randomly chosen uninfected image. Therefore, the MLP model is not considered to be used to identify mislabeled images.

K-Means Clustering Classifier using Pretrained MobileNetV2 Model

K-Means Clustering Classifier is an unsupervised machine learning algorithm that groups data into clusters based on their similarities (Tabianan et al., 2022). A Pretrained MobileNetV2 model with 155 layers was used for feature extraction by taking its convolutional base and using its output as a feature vector (the image's learned features).

We use the pretrained MobileNetV2 model that applies global average pooling to get compact feature vectors. After that, we use the K-Means Clustering Classifier to classify images into 2 clusters, which are Cluster 0 (parasitized) and Cluster 1(uninfected). The result is shown in Figure 6.

```
Cluster 0:
  Uninfected: 8427
  Parasitized: 6005

Cluster 1:
  Uninfected: 5352
  Parasitized: 7774
```

Figure 6: The result using K-Means Clustering Classifier with Pretrained MobileNetV2 Model

There is a mix of parasitized and uninfected images in each cluster, which is due to the MobileNetV2 features not separating the two classes clearly. Thus, the clustering fails to reflect the true infection status in each image.

K-Fold Out-of-Fold (OOF) Validation using Pretrained ResNet18 Model

The next method used to identify potentially mislabelled images in the dataset was the K-fold Out-of-Fold (OOF) validation approach, implemented using a pretrained ResNet18 model. This method was selected to ensure that each image in the dataset was evaluated by a model that had never seen that image during training. Through this approach, the risk of evaluation bias was minimised, and the influence of any mislabelled samples within a fixed training or validation split was effectively mitigated.

In this approach, the dataset was partitioned into five folds (k folds) of approximately equal size, and each fold was trained on four folds (k-1), while validation was performed using the remaining fold. This process was repeated five times to obtain out-of-fold predictions for the entire dataset (Berrar, 2019). To balance computational efficiency and performance, each fold was trained for three epochs.

The pretrained ResNet18 model architecture was fine-tuned for binary classification by replacing the final fully connected layer with a new layer comprising two output units. The model was trained using the *CrossEntropyLoss* function, the *Adam* optimiser, and a learning rate of $1e{-}4$.

The obtained validation accuracies were consistently high across all folds, with a mean accuracy of 0.9687 and an overall OOF F1-score of 0.9689, indicating the model's stable performance. Moreover, the confusion matrix and classification report (Figure 7) demonstrated strong performance across both classes, confirming the model's effectiveness in identifying mislabelled samples.

	precision	recall	f1-score	support
Parasitized	0.98	0.96	0.97	13779
Uninfected	0.96	0.98	0.97	13779
accuracy			0.97	27558
macro avg	0.97	0.97	0.97	27558
weighted avg	0.97	0.97	0.97	27558

Figure 7: Classification report summarizing the model's performance in detecting mislabeled images of Parasitized and Uninfected cells, including precision, recall, F1-score, and support for each class.

Through OOF predictions, 862 potentially mislabelled images were identified. These were the instances in which the predicted label differed from the original label. Among these, 370 images were classified as high-confidence swaps, where the predicted class probability exceeded 0.9. However, not all 862 samples were removed from the dataset for the following reasons:

1. Many of the remaining samples were associated with low prediction confidence, indicating uncertainty rather than clear evidence of mislabelling.

2. The manual verification of high-confidence images indicated some labels still appear inconsistent with the assigned label.

3. Although the removal of all 862 images (approximately 3% of the dataset) may seem minor, each image in medical imaging is considered to capture unique variation in cells. Therefore, removing uncertain samples could reduce data diversity and compromise the robustness of the model.

Therefore, 370 high-confidence mislabelled data identified were removed from the dataset. This method ensured that each image was independently evaluated by a robust model, making the identified high-confidence label swaps meaningful indicators of potential mislabelling. Following this refinement, the final dataset contained 27188 images.

Convolutional Neural Network Model Development

Motivation

A Convolutional Neural Network (CNN) is a specialized deep learning model engineered to autonomously learn spatial features from image data, making it a foundational methodology in contemporary computer vision applications (Yamashita et al., 2018). It operates by employing convolutional layers in place of fully connected layers, using filters (kernels) to derive salient spatial patterns, followed by a pooling mechanism to reduce dimensionality and get translational invariance to optimize the processing of grid-like data such as images (Ketkar & Moolayil, 2021). CNNs are effective in the medical domain due to their superior efficacy in achieving high accuracy and efficiency in analyzing complex clinical visual modalities (Kourounis et al., 2023).

Dataset Preparation and Splitting

The dataset was divided using the conventional 70/15/15 split: 70% (19,031 images) allocated for training, 15% (4,078 images) for validation, and 15% (4,079 images) for testing. The conventional split provided sufficient generalization and reserved enough data for accurate evaluation. Using DataLoader from the Pytorch library, the data was loaded using a batch size of 32.

CNN Architecture

The CNN was implemented in PyTorch, following an elementary and effective architecture optimized for binary image classification. The neural network is composed of two fundamental components: a feature extraction block and a classification block.

Three convolutional layers were used in the feature extraction layers to incrementally extract spatial and texture-based feature representations from the images. Based on Figure 8 below, the CNN was engineered with three input channels for RGB image handling. Gradual filter expansion was incorporated in the convolutional layers, with the filter counts starting at 32, then increasing to 64, and finally to 128. 3 x 3 kernels were used in each layer to recognize nuances and intricacies in the images, with a padding of 1 to preserve the spatial dimensions after convolution. ReLU activation is used throughout the network to introduce non-linearity

to help the neural network learn elaborate patterns. Batch normalization stabilizes training and improves convergence. MaxPooling with a 2 x 2 filter reduces spatial size and relieves computational load. After 3 successive stages of convolution and pooling, the extracted features are well condensed and sufficient for the classification layers.

```
self.features = nn.Sequential(
    nn.Conv2d(3, 32, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(32), nn.MaxPool2d(2),
    nn.Conv2d(32, 64, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(64), nn.MaxPool2d(2),
    nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(128), nn.MaxPool2d(2),
)
```

Figure 8: Convolutional layers implemented for the CNN model

In the classification stage, the extracted features are then flattened (converted into a 1D vector). Based on Figure 9 below, a fully connected layer with 128 units processes the features that were extracted and maps them onto a higher-level representation space. ReLU activation was then used to introduce non-linearity to the model and rectify the vanishing gradients problem. To reduce overfitting and regularize the model, a dropout layer with a rate of 0.5 was implemented. The final layer includes a sigmoid function to produce a scalar probability for binary classification.

```
self.classifier = nn.Sequential(
    nn.Flatten(),
    nn.Linear(128 * 16 * 16, 128),
    nn.ReLU(),
    nn.Dropout(0.5),
    nn.Linear(128, 1),
    nn.Sigmoid()
)
```

Figure 9: Activation functions within the CNN model

Training Setup and Parameters

The model was then trained using supervised learning. The Adam optimizer with a learning rate of 1e-4 was used to maintain steady convergence and prevent overshooting during training, while binary cross-entropy loss was chosen because the task involves binary classification between parasitized and uninfected image labels, as seen in Figure 10 below.

```
criterion = nn.BCELoss()
optimizer = optim.Adam(model.parameters(), lr=1e-4)
```

Figure 10: Loss function and optimizer used for training the CNN model

As seen in Figure 11 below, training was done over 30 epochs to provide the network with enough iterations to learn hierarchical features while not being compromised by excessive overfitting. During each epoch, forward propagation was used to generate predictive outputs, while backwards propagation computed and adjusted gradients.

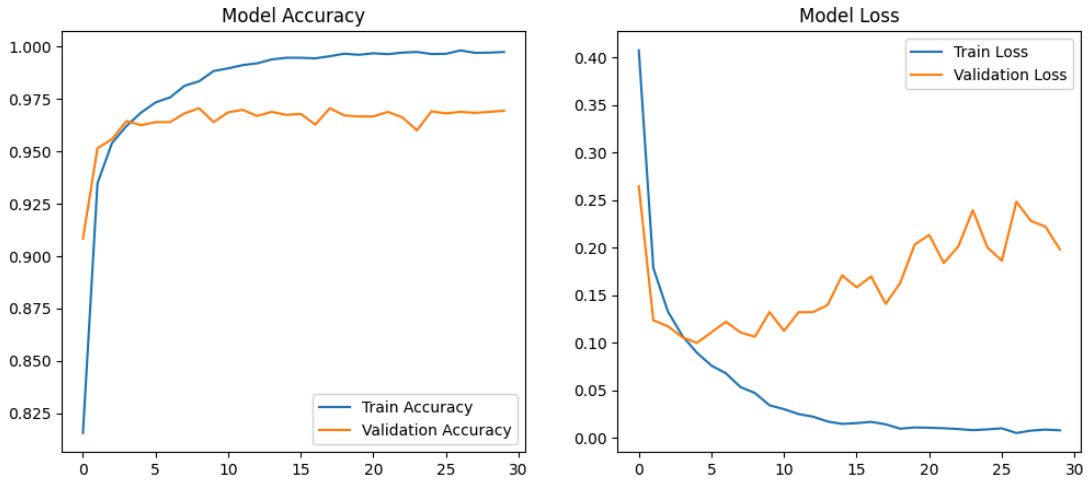
```
EPOCHS = 30

for epoch in range(EPOCHS):
    model.train()
    running_loss = 0.0
    correct, total = 0, 0

    for images, labels in train_loader:
        images, labels = images.to(device), labels.float().unsqueeze(1).to(device)

        optimizer.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        preds = (outputs > 0.5).float()
        correct += (preds == labels).sum().item()
        total += labels.size(0)

    train_loss = running_loss / len(train_loader)
    train_acc = correct / total
```

Figure 11: Code for training the CNN model for 30 epochs

Model Performance and Results

After training, the model achieved a training accuracy of 99.74%, a validation accuracy of 96.93%, a good test accuracy of 96.94 %, and a test loss of 0.1883 as seen in Figure 12 below:

```
Epoch [30/30]  Train Loss: 0.0079 | Train Acc: 0.9974  Val Loss: 0.1982 | Val Acc: 0.9693
```




```
Final Test Loss: 0.1883
Test Accuracy: 0.9694
```

Figure 12: Performance scores for the CNN model

Furthermore, the ROC results, as seen in Figure 13 below, indicated excellent model performance; the curve was in close proximity to the top-left boundary of the graph, and the AUC score of 0.9935 confirmed the model's high accuracy in binary class separation.

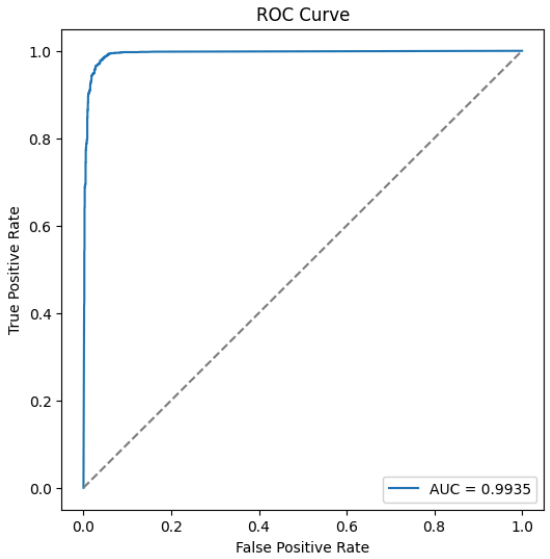


Figure 13: ROC-AUC Curve for the CNN model

The Confusion Matrix in Figure 14 below, shows the model's precision was 85.7%, which indicates that most of the predicted parasitized samples were correctly identified; the specificity was 83.9 %, which indicates that most of the uninfected samples were classified correctly, although a minority was misclassified; the F1 score of 0.92 is a reflection of the model's capability in balancing precision and recall. All of these metrics indicate that the model is clinically dependable in detecting malaria-infected cells.

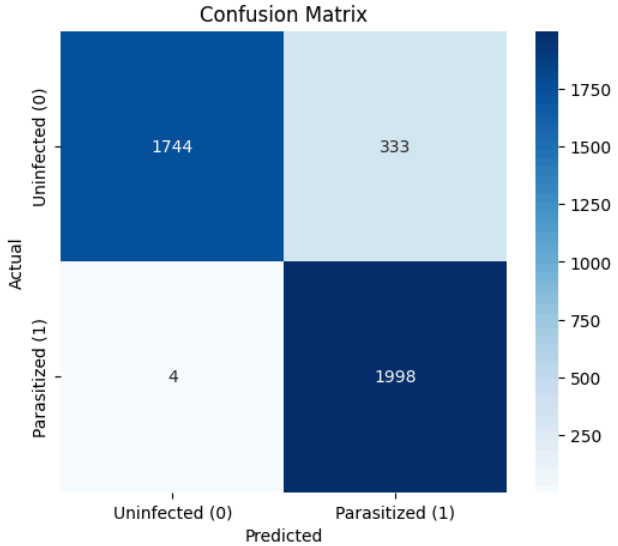


Figure 14: Confusion matrix for CNN Model

Improved Convolutional Neural Network Model

Motive of Improvement

As the model's performance in the CNN Model Development in section 4 above already achieved 96.94% accuracy, it has already reached an excellent model threshold based on the definition given by Šimundić (2009), which is still in practice by the NIH, as the diagnostic accuracy for classification of images on unseen data is already well above 90%. However, there were efforts aimed at further improving the CNN model's classification performance to achieve a higher accuracy. Hence, various methods that were explored will be discussed in the following sections.

Improved CNN Architecture

The improved CNN architecture retained mostly the same architecture as the base CNN, with improvements made to the training image set, which included data augmentation techniques. This included transformations to images in the training set, being randomly cropped, flipped, color-adjusted, and shifted vertically, as seen in Figure 15 below. These data augmentation techniques help the model be robust and to have a greater generalization of the data, which prevents overfitting (Poojary et al., 2021).

```
train_transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.RandomResizedCrop(128, scale=(0.8,1.0)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomVerticalFlip(), #Improve
    transforms.RandomRotation(15),
    transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.2),
    transforms.RandomAffine(0, translate=(0.1,0.1)),  #Improve
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])

val_transform = transforms.Compose([
    transforms.Resize((128, 128)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225])
])
```

Figure 15: Data augmentation implemented for the improved CNN model

Furthermore, two additional convolutional layers were added to increase the filter counts up to 512, compared to only 128 from the baseline CNN model, as seen in Figure 16 below.

```
class AdvancedCNN(nn.Module):
    def __init__(self):
        super().__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(32), nn.MaxPool2d(2),
            nn.Conv2d(32, 64, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(64), nn.MaxPool2d(2),
            nn.Conv2d(64, 128, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(128), nn.MaxPool2d(2),
            nn.Conv2d(128, 256, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(256), nn.MaxPool2d(2),  #added layer
            nn.Conv2d(256, 512, 3, padding=1), nn.ReLU(), nn.BatchNorm2d(512), nn.MaxPool2d(2),  #added layer
        )
```

Figure 16: Convolutional layers implemented for the improved CNN model

These additional layers enable the model to capture additional depth from visual features. This can be supported by Alzubaidi et al. (2021), that adding additional layers in a CNN model aids the model's ability to classify images correctly.

Improved Training Setup and Parameters

The training setup modifications can be seen in Figure 17 below, where changes were made to the loss function, learning rate, and the number of epochs, along with implementing a learning rate scheduler and an early stopping mechanism.

```
criterion = nn.BCEWithLogitsLoss()   #Improve
optimizer = optim.Adam(adv_model.parameters(), lr=3e-4)

scheduler = optim.lr_scheduler.ReduceLROnPlateau(
    optimizer,
    mode='min',
    factor=0.5,      # reduce LR by 0.5
    patience=4,      # wait 4 epochs before reducing
    min_lr = 1e-6,
    verbose=True
)
```

```
EPOCHS = 50
patience = 25
# Early stopping
if val_loss < best_val_loss:
    best_val_loss = val_loss
    early_stop_counter = 0
else:
    early_stop_counter += 1
    if early_stop_counter >= patience:
        print(f"Early stopping at epoch {epoch+1}")
        break
```

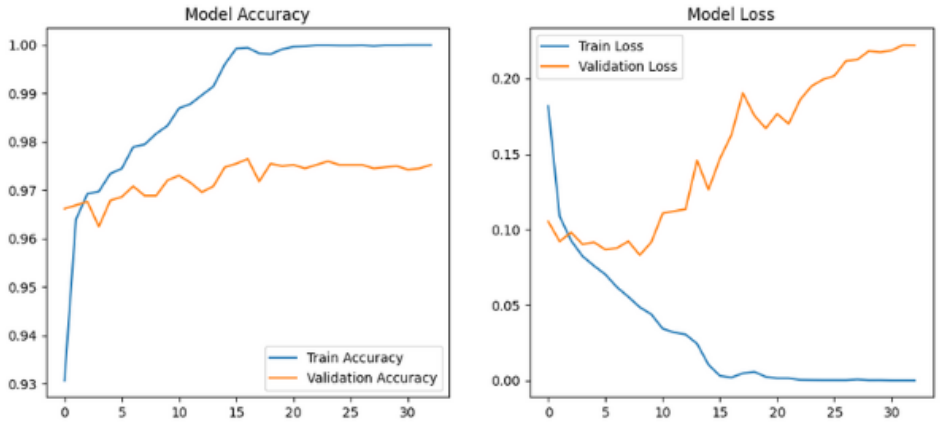
Figure 17: Code for Training setup and parameters

BCEWithLogitsLoss() was used as the loss function as it combines both the Binary Cross Entropy Loss as used in the baseline model with the Sigmoid layer, ensuring more mathematically accurate and stable outputs. A learning rate of 3e-4 was defined at the start, with a scheduler implemented to reduce the learning rate by half for every 4 epochs where there was no improvement in the accuracy of the model. This allowed the model to converge more rapidly as it avoids a plateau in learning patterns of the data (Lydia & Francis, 2019). Lastly, the number of epochs was increased to 50 to allow the model to train longer, with an early stopping mechanism being implemented, where it terminates the training if there is no significant improvement in the validation accuracy of the model after 25 epochs. Implementing the early stopping mechanism helps in preventing overfitting and also saves significant time when training the model (Anam, 2024).

Improved Model Performance and Results

Based on Figure 17 below, the model stops at 34 epochs due to the stopping mechanism. The training accuracy at the last epoch reached 100% with the validation accuracy reaching 97.52%.

```
Epoch [33/50]  Train Loss: 0.0000 | Train Acc: 1.0000  Val Loss: 0.2221 | Val Acc: 0.9752
Early stopping at epoch 34
```

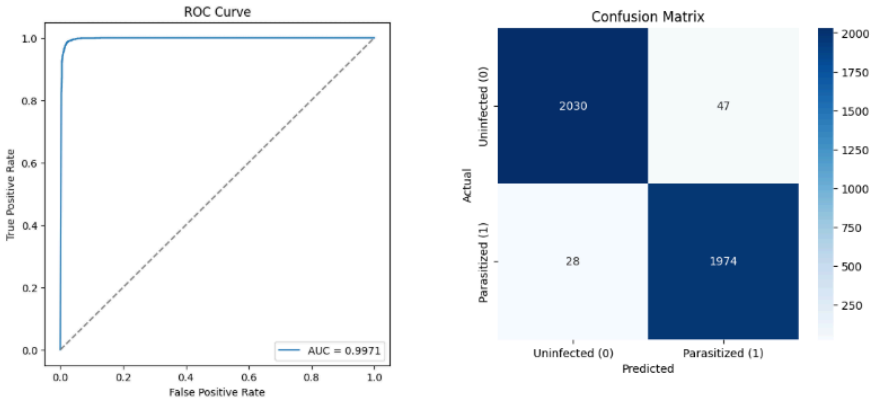



```
Final Test Loss: 0.5166
Test Accuracy: 0.9816
```

Figure 17: Performance scores for the improved CNN model

It can be seen in Figure 17 above that as the number of epochs increases, the training accuracy increases and the training loss decreases.

Furthermore, this model achieved an overall test accuracy of 98.16% which is higher than the base CNN model, which had 96.94 % accuracy. In addition to this, the ROC Curve and the Confusion Matrix can be seen in Figure 18 below:



AUC-ROC Score: 0.9971
Test Accuracy: 0.9816

Precision : 0.9767 Specificity : 0.9774
Sensitivity : 0.9860 F1-score : 0.9814

Figure 18: ROC Curve and Confusion Matrix scores for improved CNN

Figure 18 above further shows that the improved CNN model has indeed increased in most scores compared to the baseline CNN model, with the AUC-ROC score being higher, at 0.9971, and a higher precision score of 0.9767. This indicates that the improved model is better than the base CNN model in detecting and correctly classifying malaria-infected cells.

Conclusion

In conclusion, the aims of the project were answered through the various methodologies outlined at the beginning of the report. The data that was provided contained only two types of classification: “Uninfected” and “Parasitized”, with more than 13,000 images in each classification. Throughout the entirety of this project, the image datasets were successfully leveraged to gain relevant insights into how the usage of deep learning could be implemented for analysis. Upon observing the datasets, it was clear that there were mislabelled images that had to be dealt with before any further data exploration. Models such as the Single Layer Perceptron and Multi-layered Perceptron were implemented to try and combat these misclassifications. However, the testing accuracy was fairly low for both models. On top of this, K-Means Clustering using a pretrained MobileNetV2 and a K-Fold Out-of-Fold (OOF) method was also used, but the latter performed the best for image misclassification. It was then decided to proceed to the model development using a CNN after removing the misclassified images displayed by the K-Fold OOF method. Further analysis was conducted through the development of the CNN model in order to capture the image data. Although the testing accuracy was relatively high, the model could still be improved; hence, an improved CNN model was developed through data augmentation techniques, adding more layers and parameter tuning. Ultimately, at the end of the project, a robust and well-trained CNN model was created and proved to be a powerful and reliable tool for medical diagnosis, as it demonstrated the potential to assist healthcare professionals in the automation of detecting malaria-infected red blood cells.

Reference List

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

Anam, K. (2024). Early stopping on CNN-LSTM development to improve classification performance. *Journal of Applied Data Sciences*, *5*(3), 1175–1188. https://doi.org/10.47738/jads.v5i3.312

Berrar, D. (2019). *Cross-validation*. In *Reference Module in Life Sciences* (Vol. 1, pp. 542–545). Elsevier. https://doi.org/10.1016/B978-0-12-809633-8.20349-X

Bikku, T. (2020). Multi-layered deep learning perceptron approach for health risk prediction. *Journal of Big Data,* 7(50). https://doi.org/10.1186/s40537-020-00316-7

Bria, Y. P., Yeh, C., & Bedingfield, S. (2020). Significant symptoms and nonsymptom-related factors for malaria diagnosis in endemic regions of Indonesia. *International Journal of Infectious Diseases*, *103*, 194–200. https://doi.org/10.1016/j.ijid.2020.11.177

Du, K., Leung, C., Mow, W. H., & Swamy, M. N. S. (2022). Perceptron: Learning, generalization, model selection, fault tolerance, and role in the Deep learning era. *Mathematics*, *10*(24), 4730. https://doi.org/10.3390/math10244730

Ketkar, N., & Moolayil, J. (2021). Convolutional Neural Networks. In *Deep Learning with Python* (pp. 197–242). Apress. https://doi.org/10.1007/978-1-4842-5364-9_6

Kourounis, G., Elmahmudi, A. A., Thomson, B., Hunter, J., Ugail, H., & Wilson, C. (2023). Computer image analysis with artificial intelligence: a practical introduction to convolutional neural networks for medical professionals. *Postgraduate Medical Journal*, *99*(1178), 1287–1294. https://doi.org/10.1093/postmj/qgad095

Lydia, A. A., & Francis, F. S. (2019). Learning rate scheduling policies. *International Journal of Innovative Technology and Exploring Engineering*, *9*(1), 3641–3644. https://doi.org/10.35940/ijitee.a4648.119119

Mujahid, M., Kına, E., Rustam, F., Villar, M. G., Alvarado, E. S., De La Torre Diez, I., & Ashraf, I. (2024). Data oversampling and imbalanced datasets: an investigation of performance for machine learning and feature engineering. *Journal of Big Data*, *11*(1). https://doi.org/10.1186/s40537-024-00943-4

Muneeb, U., Koyuncu, E., Keshtkarjahromi, Y., Seferoglu, H., Erden, M. F., & Cetin, A. E. (2019). Robust and Computationally-Efficient Anomaly Detection using Powers-of-Two Networks. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1910.14096

Pei, X., Zhao, Y. H., Chen, L., Guo, Q., Duan, Z., Pan, Y., & Hou, H. (2023). Robustness of machine learning to color, size change, normalization, and image enhancement on micrograph datasets with large sample differences. *Materials & Design*, *232*, 112086. https://doi.org/10.1016/j.matdes.2023.112086

Poojary, R., Raina, R., & Mondal, A. K. (2021). Effect of data-augmentation on fine-tuned CNN model performance. *IAES International Journal of Artificial Intelligence*, *10*(1), 84. https://doi.org/10.11591/ijai.v10.i1.pp84-92

Saponara, S., & Elhanashi, A. (2022). Impact of image resizing on deep learning detectors for training time and model performance. In *Lecture notes in electrical engineering* (pp. 10–17). https://doi.org/10.1007/978-3-030-95498-7_2

Shukla, A. (2023). Efficient and Optimized Usage of Various Image Formats (JPEG/ JPG vs PNG) in Applications. *Journal of Mathematical & Computer Applications.* *https://doi.org/10.47363/JMCA/2023(2)131*

Šimundić, A. (2009, January 20). *Measures of Diagnostic Accuracy: Basic Definitions*. https://pmc.ncbi.nlm.nih.gov/articles/PMC4975285/#abstract1

Tabianan, K., Velu, S., & Ravi, V. (2022). K-Means clustering approach for intelligent customer segmentation using customer purchase behavior data. *Sustainability*, *14*(12), 7243. https://doi.org/10.3390/su14127243

Yamashita, R., Nishio, M., Do, R. K. G., & Togashi, K. (2018). *Convolutional neural networks: an overview and application in radiology*. Insights Imaging. https://doi.org/10.1007/s13244-018-0639-9

Appendix

Completed by: You Wei Lim

Group Project Assessment for ADS2002							
Studio: Monday							
Group: Malaria (Group 3)							
		Tashvin Ramesh (34675280)	Kalubowilage Niduli Sudewna Alwis (34144021)	Dayvin Rushil Athirathan (33589070)	Daniel Ong (34897887)	Lee Yoong Shuen (33522200)	You Wei Lim (34069518)
Coordination	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Research	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Coding	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Report Writing	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Presentation Preparation	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Attendance at Final Presentation	This entry should be 1 (present) or 0 (absent)	1	1	1	1	1	1

Completed By : Dayvin Rushil Athirathan

Group Project Assessment for ADS2002							
Studio: Monday 10am – 1pm							
Group: Malaria (Group 3)							
		Tashvin Ramesh (34675280)	Kalubowilage Niduli Sudewna Alwis (34144021)	Daniel Ong (34897887)	You Wei Lim (34069518)	Lee Yoong Shuen (33522200)	Dayvin Rushil Athirathan (33589070)
Coordination	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Research	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Coding	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Report Writing	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Presentation Preparation	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Attendance at Final Presentation	This entry should be 1 (present) or 0 (absent)	1	1	1	1	1	1

Completed By : Kalubowilage Niduli Sudewna Alwis

Group Project Assessment for ADS2002							
Studio: Monday							
Group: Malaria (Group 3)							
		Tashvin Ramesh (34675280)	Dayvin Rushil Athirathan (33589070)	You Wei Lim (34069518)	Daniel Ong (34897887)	Lee Yoong Shuen (33522200)	Kalubowilage Niduli Sudewna Alwis (34144021)
Coordination	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Research	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Coding	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Report Writing	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Presentation Preparation	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Attendance at Final Presentation	This entry should be 1 (present) or 0 (absent)	1	1	1	1	1	1

Completed By : Daniel Ong

Group Project Assessment for ADS2002							
Studio: Monday							
Group: Malaria							
		Tashvin Ramesh 34675280	Dayvin Rushil Athirathan 33589070	You Wei Lim 34069518	Daniel Ong 34897887	Kalubowilage Niduli Sudewna Alwis 34144021	Lee Yoong Shuen 33522200
Coordination	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Research	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Coding	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Report Writing	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Presentation Preparation	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Attendance at Final Presentation	This entry should be 1 (present) or 0 (absent)	1	1	1	1	1	1

Completed By : Tashvin Ramesh

Group Project Assessment for ADS2002							
Studio: Monday 10am – 1pm							
Group: Malaria (Group 3)							
		Member 1	Member 2	Member 3	Member 4	Member 5	Member 6
Coordination	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	Tashvin Ramesh (34675280)	Dayvin Rushil Athirathan (33589070)	Daniel Ong (34897887)	You Wei Lim (34069518)	Kalubowilage Niduli Sudewna Alwis (34144021)	Lee Yoong Shuen (33522200)
Research	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Coding	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Report Writing	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Presentation Preparation	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Attendance at Final Presentation	This entry should be 1 (present) or 0 (absent)	1	1	1	1	1	1

Completed By : Lee Yoong Shuen

Group Project Assessment for ADS2002							
Studio: Monday (10am-1pm)							
Group: Malaria (Group 3)							
		Tashvin Ramesh (34675280)	Dayvin Rushil Athirathan (33589070)	You Wei Lim (34069518)	Daniel Ong (34897887)	Kalubowilage Niduli Sudewna Alwis (34144021)	Lee Yoong Shuen (33522200)
Coordination	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Research	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Coding	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Report Writing	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Presentation Preparation	The sum of this row should be 100. The entry should reflect the group member's percentage contribution to this project task.	16.66	16.66	16.66	16.66	16.66	16.66
Attendance at Final Presentation	This entry should be 1 (present) or 0 (absent)	1	1	1	1	1	1