

# **MALARIA CELL CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK**

**PRESENTED BY**

**DEBASMITA CHATTERJEE  
PRADYMNA KUMAR ROUT  
RHITAM MONDAL  
TOMIN JACOB**  
Group- G-9

**UNDER SUPERVISION OF  
DR. MADHUCHHANDA DASGUPTA**

# **CONTENTS**

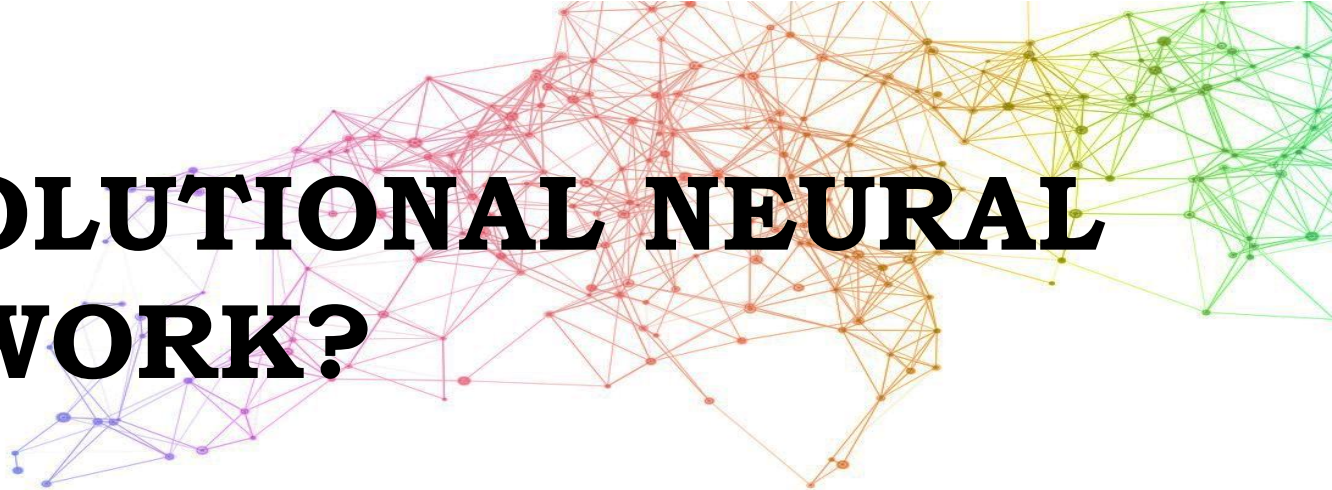
- 1. INTRODUCTION**
- 2. METHODOLOGY**
- 3. DATASET COLLECTION**
- 4. DATA PREPROCESSING**
- 5. MODEL TRAINING AND VISUALIZATION**
- 6. MODEL TESTING**
- 7. MODEL EVALUATION**
- 8. CONCLUSION**
- 9. APPENDICES**
- 10. REFERENCES**



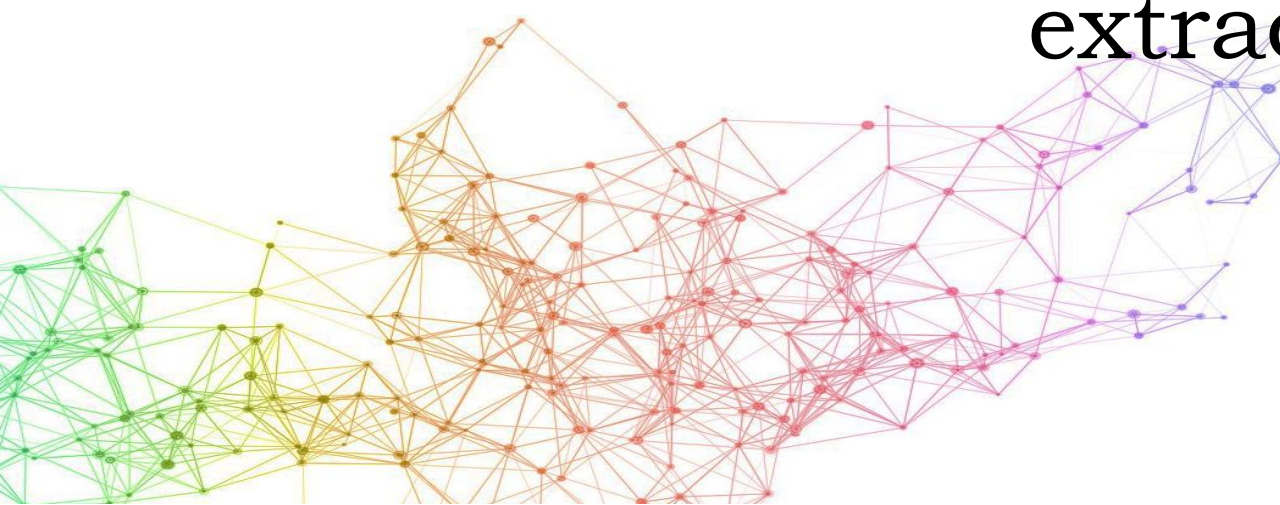
# INTRODUCTION

- Malaria is a life threatening disease spread by Anopheles mosquitoes and caused by Plasmodium.
- A CNN is proposed for malaria cell classification.

# WHAT IS A CONVOLUTIONAL NEURAL NETWORK?

An abstract graphic in the top right corner consisting of a complex network of interconnected nodes and lines. The nodes are colored in a gradient from purple to green, and the lines are thin and light-colored, creating a web-like structure.

A Convolutional Neural Network is a type of deep learning model particularly effective for image classification and feature extraction.

An abstract graphic in the bottom left corner, similar to the one in the top right, showing a network of nodes and lines. The nodes are colored in a gradient from green to purple, and the lines are thin and light-colored.

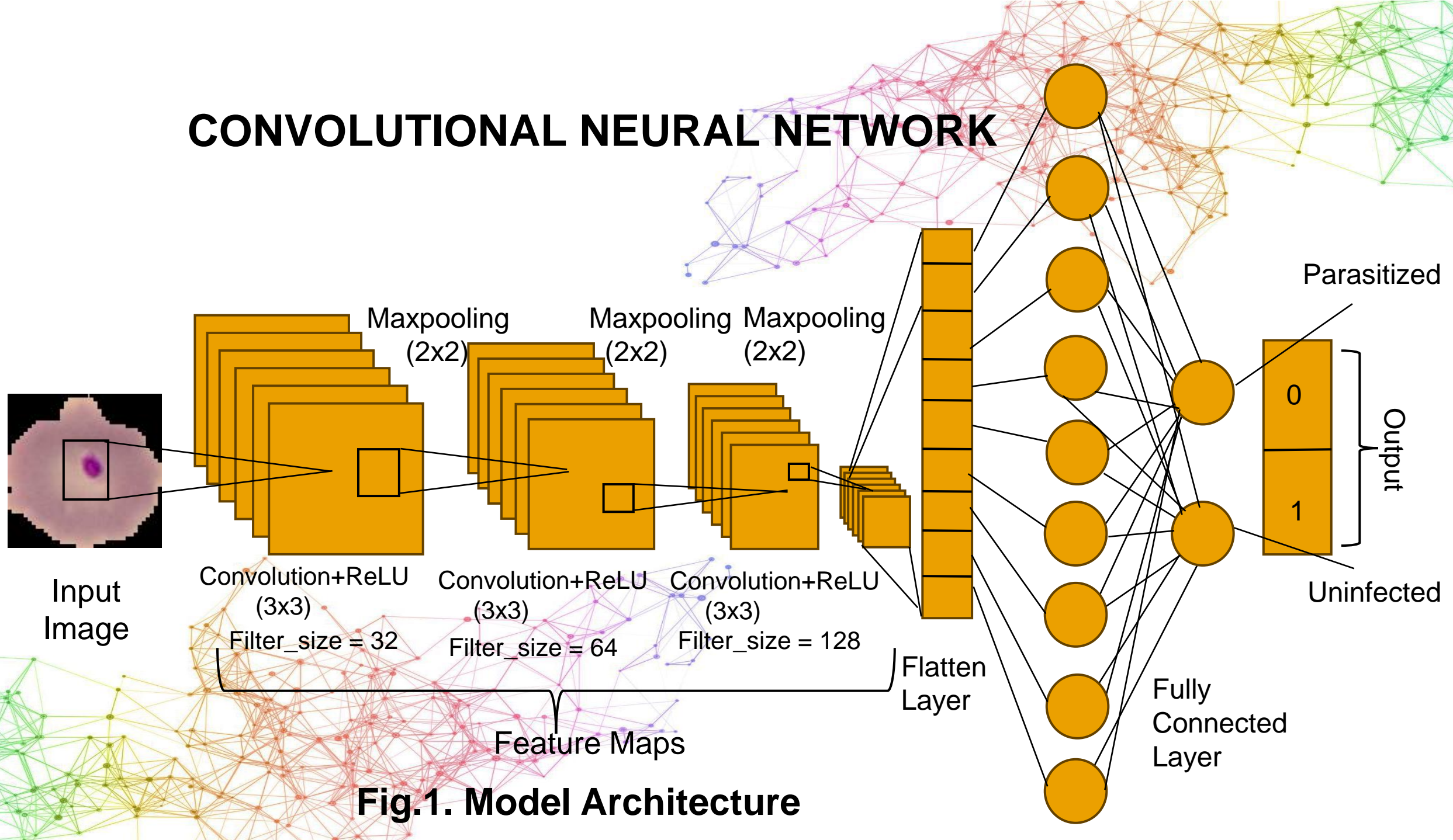


# METHODOLOGY

The background is a dark, abstract composition. It features a dense network of glowing, multi-colored lines (purple, blue, yellow, and red) that originate from several points on the left and fan out towards the right, creating a sense of dynamic movement and connectivity. On the far left, there are faint, semi-transparent snippets of code in a light blue font, including lines like 'getInstanceIdentifier', 'tagset', and 'getInstanceIdentifier'. The right side of the image is filled with a dense, textured pattern of small, glowing dots in various colors, primarily red and purple, which adds depth and complexity to the overall visual.



# CONVOLUTIONAL NEURAL NETWORK



**Fig.1. Model Architecture**



The CNN architecture contains the following features:

- Three Convolutional layers with kernel dimension (3,3) and filter size 32, 64 and 128 respectively.
- ReLU activation function is used to introduce non-linearity.
- Three maxpooling layers with pool size (2,2)
- Flatten layer after feature extraction
- Two dense fully connected layers
  - 128 fully connected neurons in hidden layer
  - 2 neurons in the output layer for the final classification outcome.
- The classification is finally calculated using softmax function.

# MODEL BASIC BLOCKS

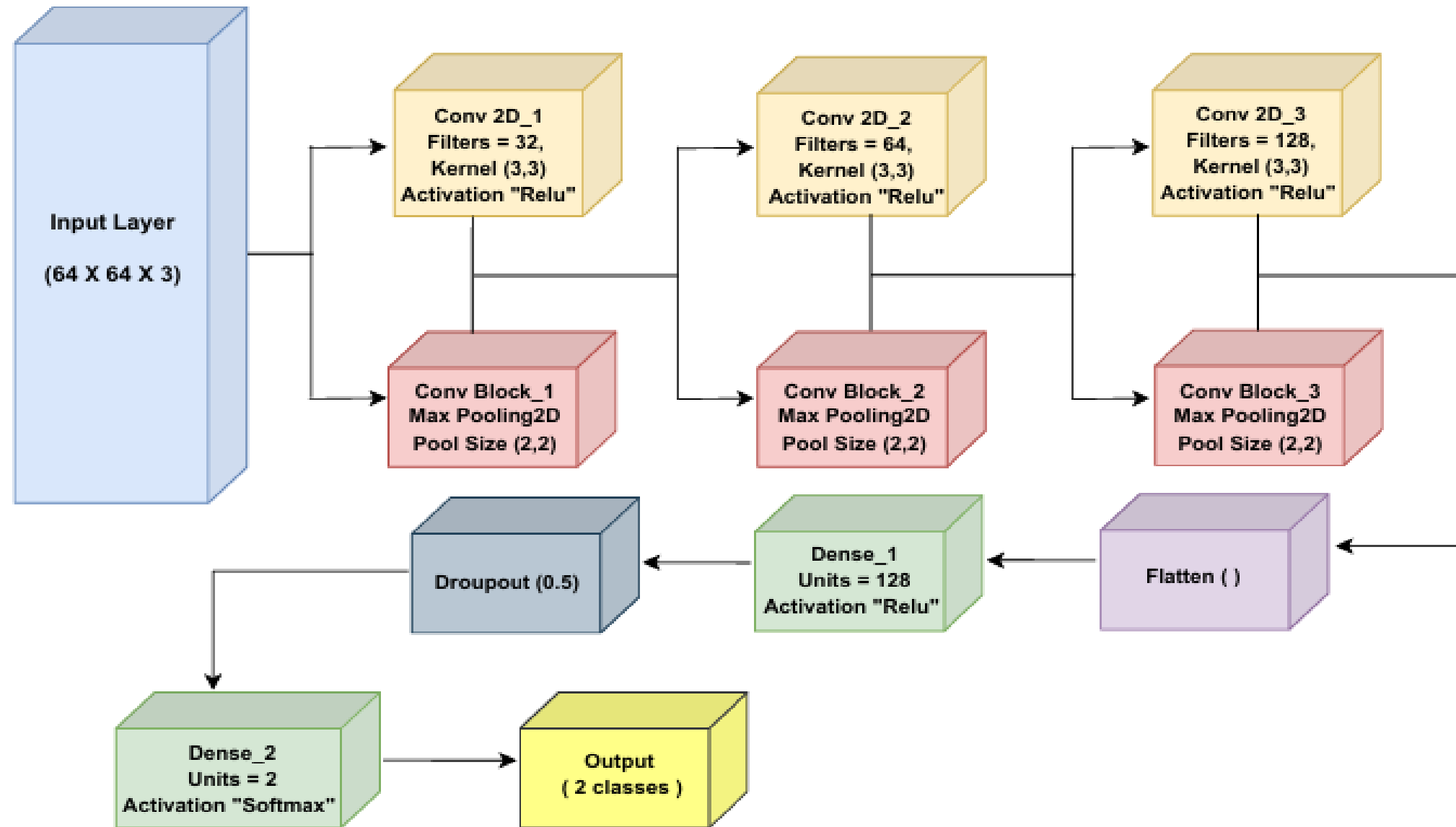


Fig.2. Model Basic Blocks



# CNN ARCHITECTURE



```
graph TD; A[CNN ARCHITECTURE] --> B[CONVOLUTIONAL BLOCKS]; A --> C[CLASSIFICATION LAYER]; B --> D[Conv2D (The Feature Detector) and MaxPooling2D (The Shrinker)]; C --> E[Flatten, Dense Layer, and Output Layer];
```

## CONVOLUTIONAL BLOCKS

- **Conv2D (The Feature Detector):** These layers use small filters to scan the image and detect features like edges, textures and patterns. We started with 32 filters, then 64 and finally 128, all employing a (3, 3) kernel size which got more complex with each layer. We used the ReLU activation function to introduce non-linearity.
- **MaxPooling2D (The Shrinker):** This layer reduces the size of the image data, keeping only the most important features. This makes the model faster and less prone to getting fixated on tiny details (overfitting).


## CLASSIFICATION LAYER

- **Flatten:** The 3D feature maps were unwrapped into a single long list of numbers.
- **Dense Layer:** A standard neural network layer with 128 neurons processed this list. We added a Dropout layer (0.5), which randomly ignores half the neurons during training. This forces the remaining neurons to be more robust and is a powerful way to prevent overfitting.
- **Output Layer:** The final layer had 2 neurons (one for each class) with a Softmax activation function for class probability prediction.

# **DATASET COLLECTION**

The background is a dark, abstract composition. It features a dense field of small, glowing dots in shades of purple, blue, and red, which appear to be data points or nodes. From these points, numerous thin, curved lines of the same colors radiate outwards, creating a sense of movement and connectivity. The lines are more concentrated in the center and left side, where they form a complex, web-like structure. The overall effect is one of a vast, interconnected digital space, possibly representing a dataset or a network.





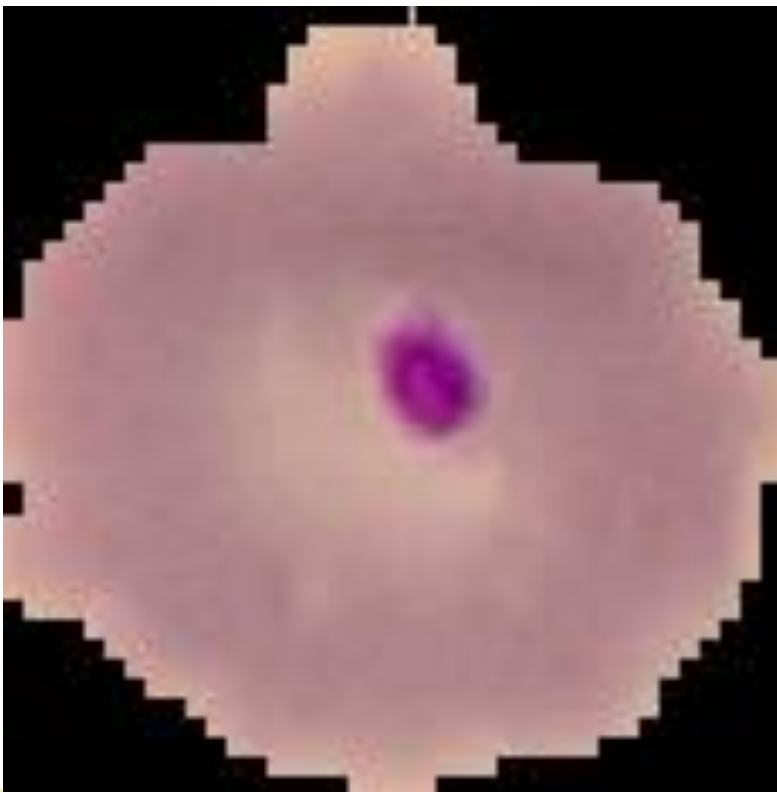
➤ The dataset consists of a large set of parasitized and uninfected cell images observed under microscope.

➤ The dataset has a total of 27,558 images.

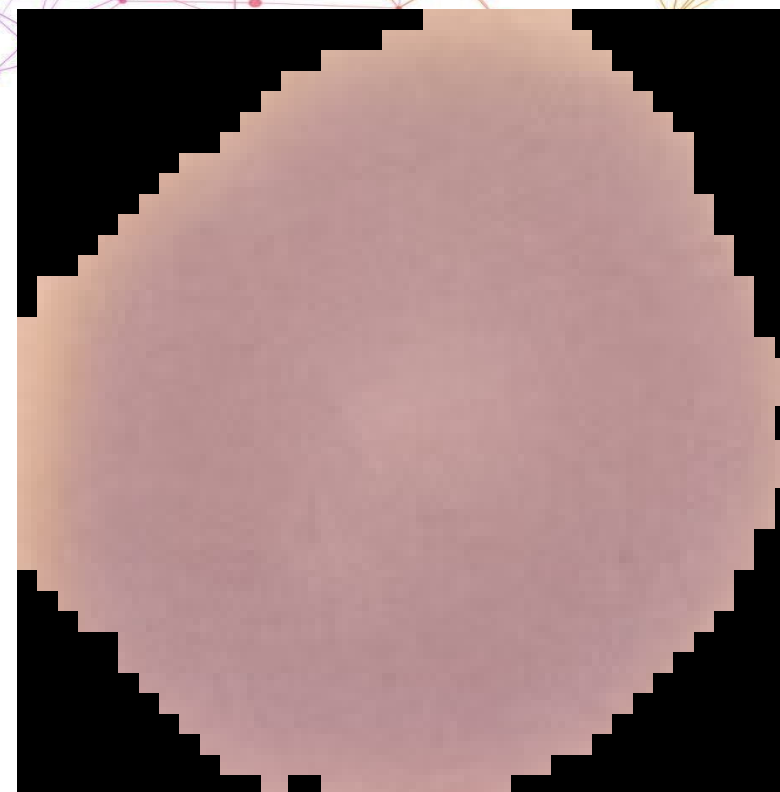
➤ The dataset has been extracted from <https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria>



## **DATASET IMAGES**



**PARASITIZED CELL IMAGE**



**UNINFECTED CELL IMAGE**

**Fig.3. Cell Images**



# DATA PREPROCESSING

The background is a dark, abstract composition. It features a dense field of small, glowing dots in shades of purple, blue, and red, particularly concentrated on the right side. On the left, there are several bright, multi-colored (yellow, orange, red) flares or nebula-like shapes. A complex network of thin, glowing lines in purple and blue weaves across the image, connecting various points and creating a sense of dynamic movement and data flow.



## DATA TRANSFORMATION

The images are resized to 64 X 64 pixels for efficient deep learning analysis.

## DATA LABELING

The parasitized cell images are assigned label '0' and uninfected cell images are assigned label '1'.

The images and their respective labels are stored in separate numpy arrays.





## DATA PREPROCESSING

**Normalization:** Image pixels are represented by numbers, typically from 0 to 255. We divided all these numbers by 255.0 to scale them down to a small range between 0 and 1.

**One-Hot Encoding:** The labels were simple: 0 for Uninfected, 1 for Parasitized. These labels are converted into a "one-hot" format (e.g., 1 becomes  $[0, 1]$ , 0 becomes  $[1, 0]$ ).

**Splitting the Data:** The entire dataset is divided into two main parts: 70% for Training (the material the model learns from) and 30% for Testing (unseen material to check how well it learned).

The background is a dark, deep blue gradient. It is filled with a complex network of glowing, thin lines in shades of purple, magenta, and yellow. These lines originate from several points on the left side and fan out towards the right, creating a sense of dynamic movement and connectivity. Scattered throughout the background are numerous small, glowing dots in similar colors, some appearing in clusters. In the bottom left corner, there is faint, semi-transparent white text that appears to be code or technical data, including terms like 'InstanceIdentifier' and 'Tag'.

# MODEL TRAINING AND VISUALIZATION



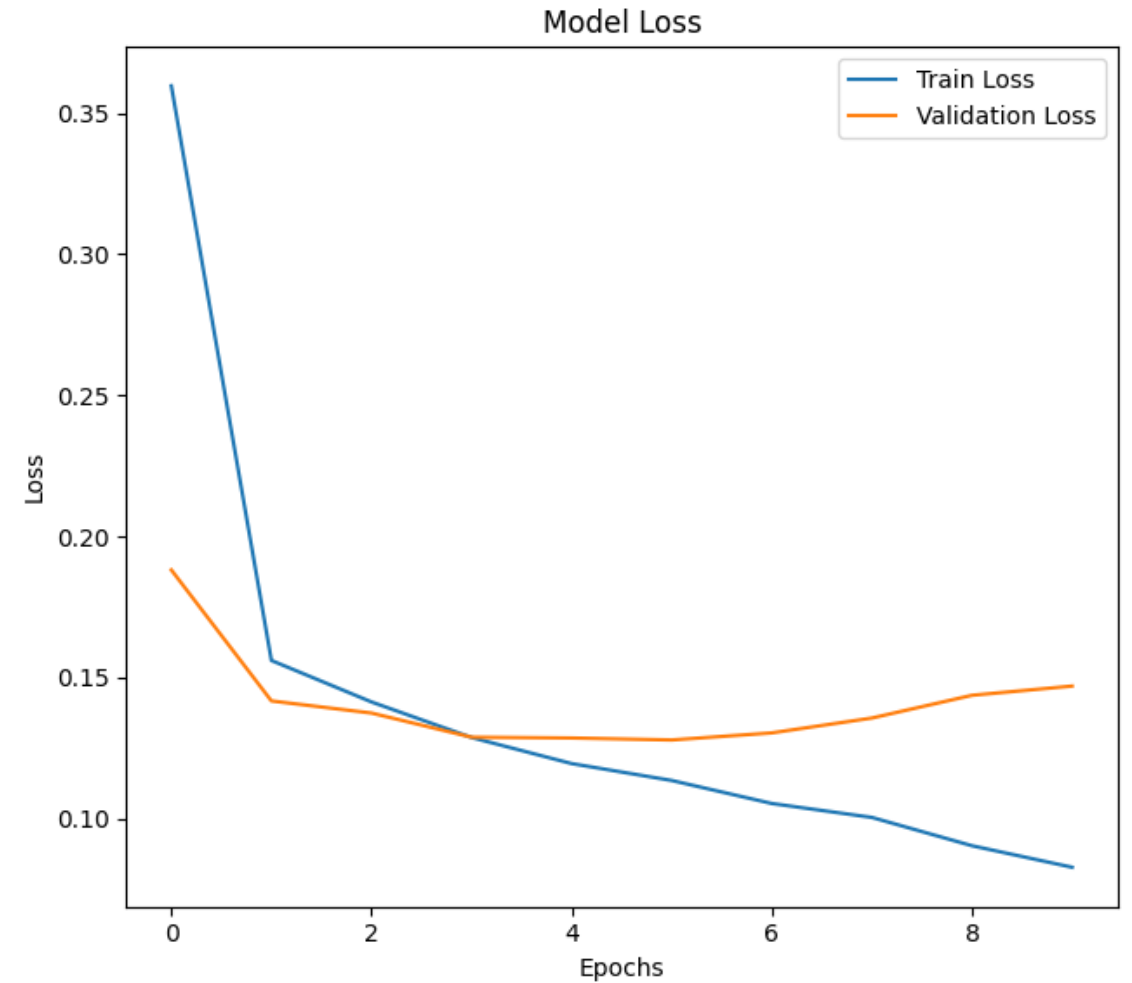
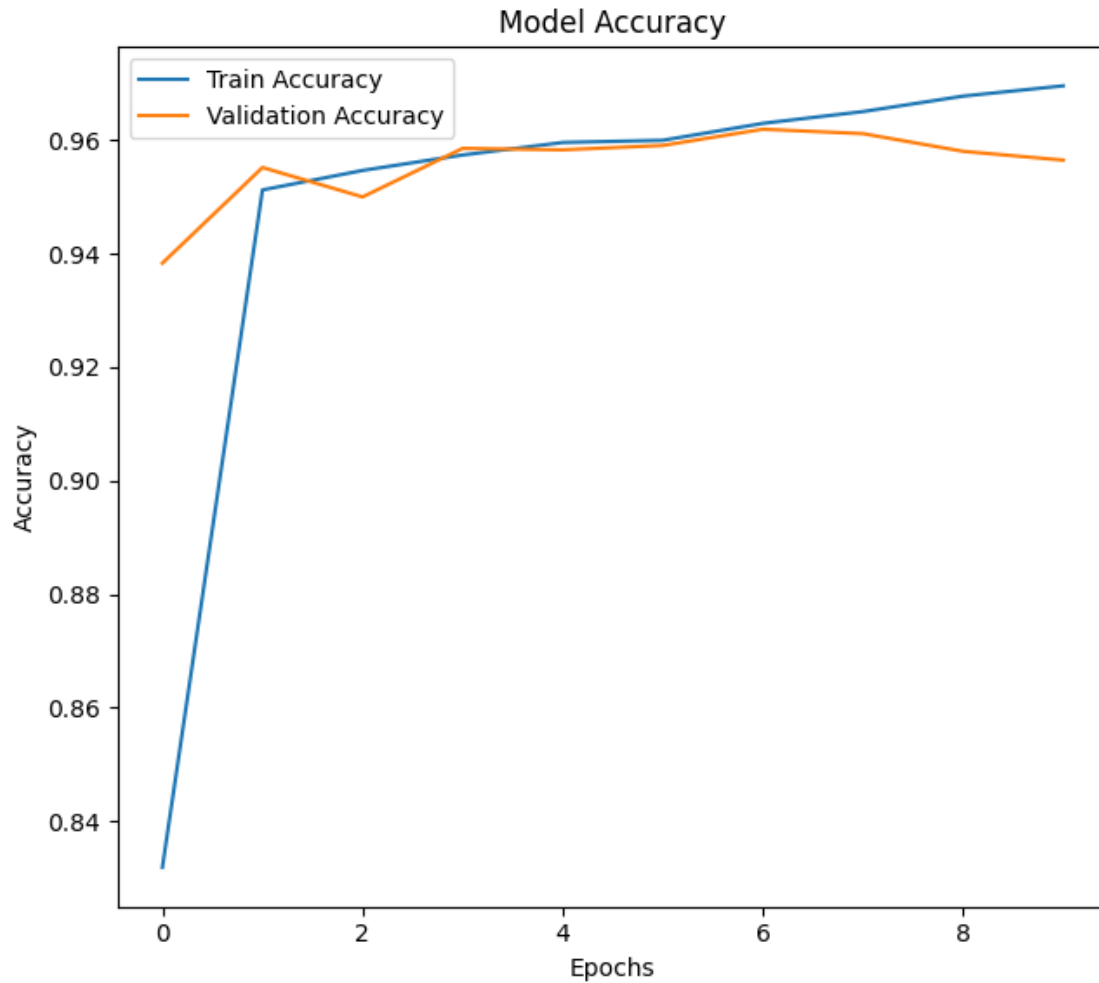
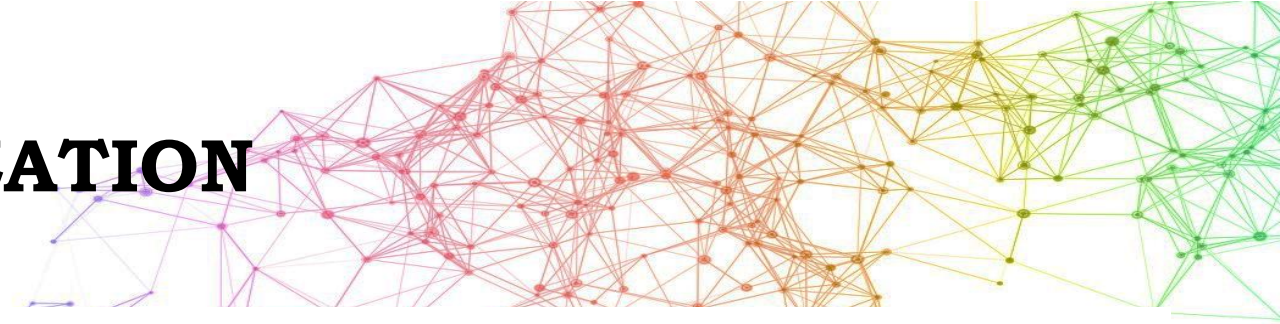
# MODEL TRAINING

- The model is compiled using the Adam optimizer
- Then, the model is trained for 10 epochs using a batch\_size of 32. (10 full passes through the training data).

```
Epoch 1/10
483/483 ————— 195s 371ms/step - accuracy: 0.8319 - loss: 0.3596 - val_accuracy: 0.9383 - val_loss: 0.1881
Epoch 2/10
483/483 ————— 203s 419ms/step - accuracy: 0.9512 - loss: 0.1560 - val_accuracy: 0.9552 - val_loss: 0.1417
Epoch 3/10
483/483 ————— 154s 320ms/step - accuracy: 0.9546 - loss: 0.1413 - val_accuracy: 0.9500 - val_loss: 0.1374
Epoch 4/10
483/483 ————— 249s 417ms/step - accuracy: 0.9574 - loss: 0.1287 - val_accuracy: 0.9585 - val_loss: 0.1288
Epoch 5/10
483/483 ————— 178s 366ms/step - accuracy: 0.9596 - loss: 0.1195 - val_accuracy: 0.9583 - val_loss: 0.1285
Epoch 6/10
483/483 ————— 175s 361ms/step - accuracy: 0.9600 - loss: 0.1135 - val_accuracy: 0.9590 - val_loss: 0.1278
Epoch 7/10
483/483 ————— 186s 386ms/step - accuracy: 0.9629 - loss: 0.1053 - val_accuracy: 0.9619 - val_loss: 0.1303
Epoch 8/10
483/483 ————— 234s 448ms/step - accuracy: 0.9650 - loss: 0.1004 - val_accuracy: 0.9611 - val_loss: 0.1356
Epoch 9/10
483/483 ————— 186s 384ms/step - accuracy: 0.9677 - loss: 0.0904 - val_accuracy: 0.9580 - val_loss: 0.1437
Epoch 10/10
483/483 ————— 170s 351ms/step - accuracy: 0.9695 - loss: 0.0827 - val_accuracy: 0.9565 - val_loss: 0.1469
```

**Fig.4. Epoch Visualization**

# VISUALIZATION



**Fig.5. Training Accuracy and Loss Graph**



# MODEL TESTING

The background is a dark, abstract composition. It features a dense network of glowing, multi-colored lines (purple, blue, yellow, and red) that flow and curve across the frame, creating a sense of dynamic movement and connectivity. In the lower-left corner, there are faint, semi-transparent snippets of code, including terms like 'getInstance', 'register', and 'Identifier'. The overall aesthetic is high-tech and digital, suggesting themes of data, networks, and artificial intelligence.



# MODEL TESTING

**Loading the model :** The pre-trained CNN model (malaria\_cnn\_model.h5) is loaded using Keras. A quick check confirmed that the model was successfully loaded and ready for testing.

**Preparing the data :** The unseen datasets are normalized and encoded into one-hot encoding.

**Testing and Visualization :** The dataset is finally loaded into the model. The dataset is processed with 10 epochs and batch\_size 32 in the model. To assess the model's performance on unseen datasets, the loss and accuracy graphs are plotted.

**Assessing the Runtime:** In this stage, the full evaluation process was timed using Python's time module.



# OUTPUT PANEL VISUALIZATION

```
✓ Model loaded successfully!
📁 Loading dataset (this may take a while)...
✓ Loaded 27558 images.
🔧 Test set size: 8268 images

⚙️ Evaluating model on test data...
259/259 ██████████ 2s 6ms/step - accuracy: 0.9520 - loss: 0.1314

✓ Test Loss: 0.1314
✓ Test Accuracy: 95.20%
259/259 ██████████ 2s 6ms/step

📄 Evaluation Summary:
      Metric  Value
Total Test Images  8268
Correct Predictions 7871
      Accuracy (%) 95.20
      Loss 0.1314

🕒 Total Runtime: 0 min 43.01 sec
🎉 Evaluation complete!
```

Fig.6. Output Panel

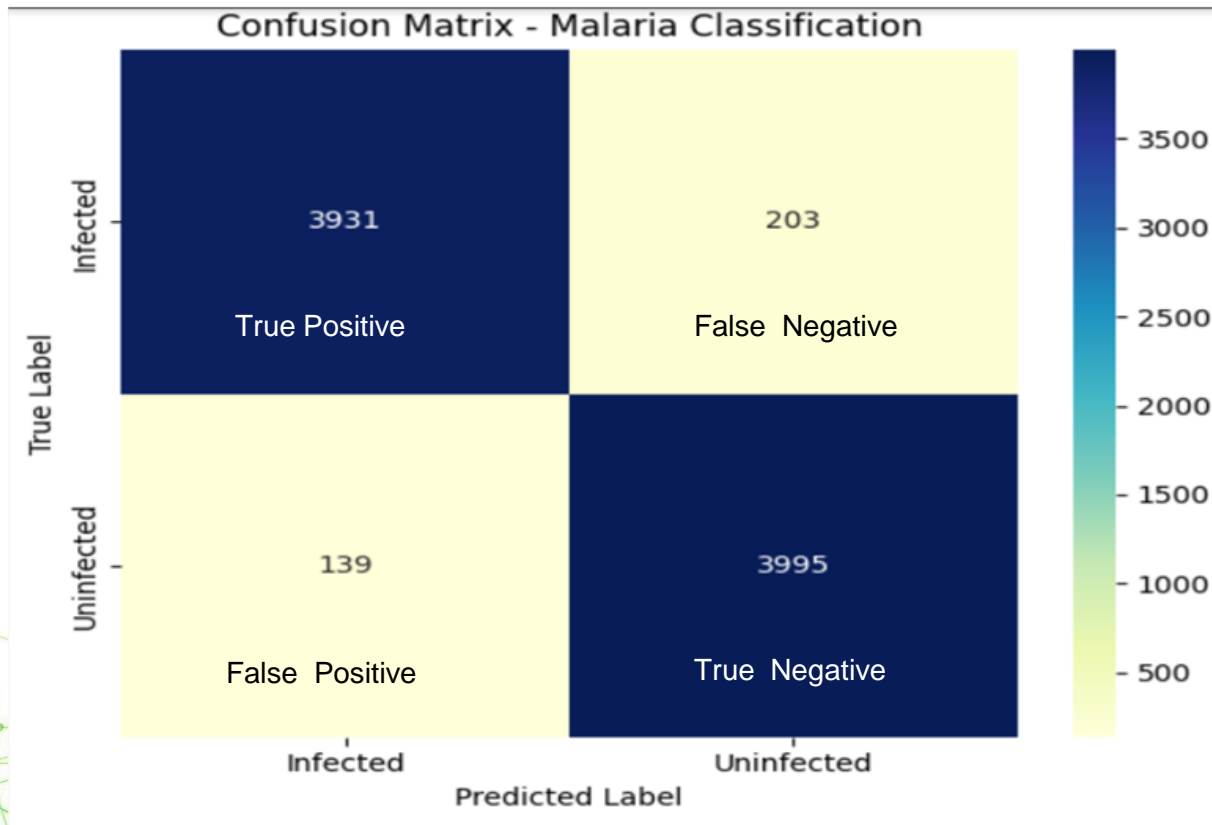
# MODEL EVALUATION

- Model Evaluation assesses a model's overall performance, efficiency, and reliability.
- They provide standardized, quantitative measures to understand the model's performance.
- Through the Evaluation metrics, we can
  - ensure that the model meets the desired accuracy and robustness requirements.



# CONFUSION MATRIX

The confusion matrix summarizes the model's performance by showing the number of correct and incorrect classifications for each category.



- The model correctly identified 3931 infected and 3995 uninfected cells, with only minor misclassifications.
- This indicates strong classification ability and high predictive reliability.

**Fig.8. Confusion Matrix**

# FORMULAS

The important formulas to calculate the model's evaluation metrics are:

Note: True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN).

$$\text{Accuracy (A)} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

$$\text{Precision (P)} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall (R)} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

$$\text{F1 Score (F)} = \frac{2 \times \text{P} \times \text{R}}{\text{P} + \text{R}}$$



# Evaluation Metrics

Table.1. Evaluation Metrics

Metric	Result (%)
Accuracy	95.86
Precision	95.16
Recall (Sensitivity)	96.64
F1 Score	95.90

# CONCLUSION

- The CNN model developed for malaria cell classification has shown strong and balanced performance across all metrics.
- The high accuracy and F1 score validate its robustness and reliability.
- With minimal misclassifications, the model proves to be efficient for automated malaria detection and could serve as a supportive diagnostic tool in healthcare environments.



# APPENDICES

[1]Github Link for the project: <https://github.com/DebasmitaChatterjee/Malaria-Cell-Classification.git>

[2]Dataset: <https://www.kaggle.com/datasets/iarunava/cell-images-for-detecting-malaria>



# REFERENCES

- [1] Chima, J. S., Shah, A., Shah, K., & Ramesh, R. (2020). Malaria cell image classification using deep learning. International Journal of Recent Technology and Engineering, 8(6), 5553-59.
- [2] Sivaramakrishnan, R., Antani, S., & Jaeger, S. (2017, October). Visualizing deep learning activations for improved malaria cell classification. In Medical informatics and healthcare (pp. 40-47). PMLR.
- [3]<https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/>
- [4]<https://eranfeit.net/how-to-classify-malaria-cells-using-convolutional-neural-network/>





**THANK YOU**