

# MalariaSpecNet – Technical Documentation

Version 1.0

## 1. Introduction

The **MalariaSpecNet** is a computer-vision platform designed to automatically detect malaria species from microscopic blood-smear images. The system integrates:

- A custom-designed 7-channel image preprocessing pipeline
- A CNN-based classifier optimized for MP-IDB microscopy images
- PyTorch Lightning training and inference modules
- A user-friendly GUI built with CustomTkinter
- Batch inference and CSV export capabilities

The system supports four classes:

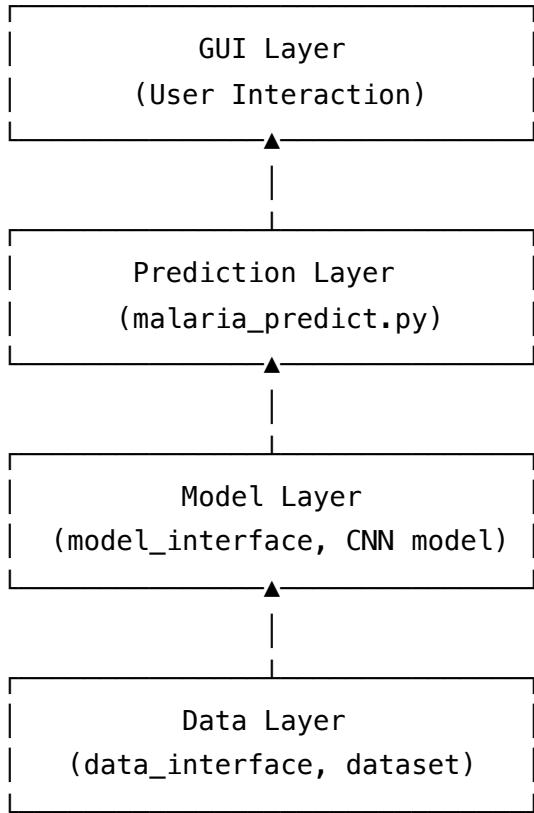
1. *Plasmodium falciparum*
2. *Plasmodium vivax*
3. *Plasmodium ovale*
4. *Uninfected (negative)*

It is designed for laboratory use, educational purposes, AI research, and automated diagnostic assistance.

## 2. System Architecture

The software adopts a modular architecture separating **data**, **model**, **inference**, and **GUI** layers. This ensures high maintainability, extensibility, and clear responsibility separation.

**Architecture Diagram:**



### 3. File Structure Overview

The project's file structure is organized as follows:

```
project/
├── data/
│   ├── data_interface.py
│   └── mpidb_dataset.py
|
├── model/
│   ├── model_interface.py
│   └── standard_net.py
|
└── malaria_predict.py
├── GUI.py
└── background/ (GUI assets)
└── requirements.txt
└── model.ckpt (trained weights)
```

# 4. Data Layer

## 4.1 Data Interface (DInterface)

A generalized data module implemented using PyTorch Lightning's DataModule design pattern.  
Responsible for dynamically loading dataset classes and preparing training/validation/testing loaders.

### Key Features:

- Dynamic dataset loading based on file name
- Automatic DataLoader creation
- Worker, batch size, and pin\_memory configuration
- Compatible with any dataset following the required structure

### Usage Example:

```
data = DInterface(  
    dataset='malaria_dataset',  
    root='data/MPIDB',  
    batch_size=64,  
    img_size=100,  
)
```

## 4.2 Malaria Dataset (MpidbDataset)

A dataset loader customized for the MP-IDB malaria dataset.

### Directory Structure:

```
root/  
|__ train/  
|__ val/  
└__ test/  
    |__ falciparum/  
    |__ vivax/  
    |__ ovale/  
    |__ negative/
```

### 7-Channel Image Representation:

Channel	Description
R, G, B	Standard RGB
L	LAB Lightness
S	HSV Saturation
Laplacian	Edge gradient
TopHat	Morphological enhancement

This enhances parasite visibility and classification accuracy.

### **Transform Pipeline:**

Albumentations is used for:

- Geometric augmentation
- Noise injection
- Brightness/contrast variation
- Training/validation resizing

## **5. Model Layer**

### **5.1 Model Interface (MInterface)**

A unified wrapper around all model architectures.

#### **Responsibilities:**

- Dynamically load model architectures from the `model/` directory
- Configure optimizers and learning-rate schedulers
- Provide LightningModule functionalities:
  - `training_step`
  - `validation_step`
  - `test_step`
  - Logging of metrics

#### **Supported Parameters:**

Parameter	Description
model	Model name (e.g., 'standard_net')
in_ch	Input channels (default = 7)
num_classes	Number of output categories
lr	Learning rate
weight_decay	Weight regularization

## 5.2 StandardNet (MPIDBCNN)

The core classifier network (custom CNN).

### Network Architecture:

```

Input (7 × 100 × 100)
↓
Block 1: Conv(7→32) → BN → LeakyReLU → MaxPool → Dropout
Block 2: Conv(32→64) → BN → LeakyReLU → MaxPool → Dropout
Block 3: Conv(64→128) → BN → LeakyReLU → MaxPool → Dropout
Block 4: Conv(128→256) → BN → LeakyReLU → MaxPool → Dropout
↓
Flatten → FC1 → LeakyReLU → FC2 (num_classes)

```

### Design Rationale:

- Lightweight and fast inference suitable for GUI deployment
- Strong discriminative ability due to multi-channel input
- Dropout used aggressively to prevent overfitting on limited datasets

## 6. Prediction Layer

Implemented in malaria\_predict.py.

### Workflow:

1. Load .ckpt model using Lightning
2. Preprocess each input image:
  - Resize
  - Convert to 7-channel representation
  - Convert to tensor
3. Perform forward inference
4. Apply softmax to compute probabilities
5. Output a pandas DataFrame with:

```
predicted_class | confidence
-----
falciparum      | 0.9823
ovale           | 0.6211
...

```

### **Example Usage:**

```
from malaria_predict import malaria_predict
df = malaria_predict('negative.ckpt', [Path('sample.jpg')])
print(df)
```

## **7. Graphical User Interface (GUI)**

The GUI system is implemented in [GUI.py](#) using CustomTkinter.

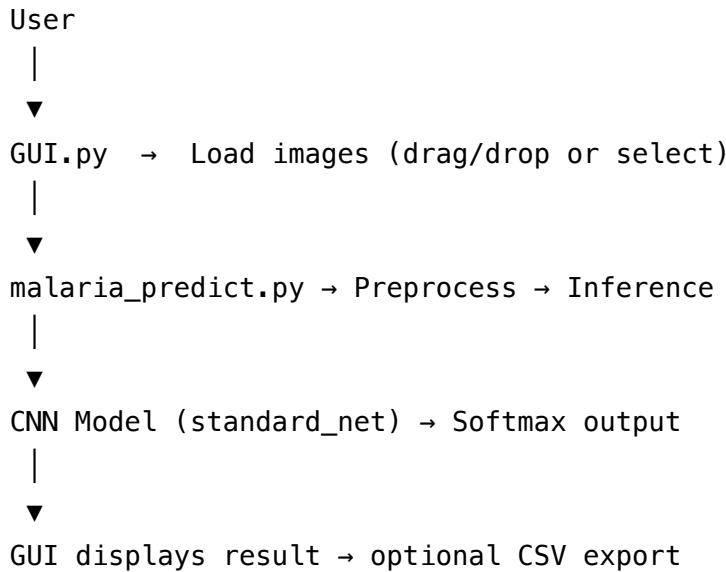
### **7.1 Core Features**

<b>Feature</b>	<b>Description</b>
<b>Drag-and-Drop Support</b>	Upload multiple images at once
<b>Image Gallery</b>	Dynamic grid preview with scaling
<b>Model Inference</b>	Runs batch classification
<b>Status Indicators</b>	Color-coded progress updates
<b>CSV Export</b>	One-click result file generation

## 7.2 User Workflow

1. **Upload** images
2. Preview thumbnails in the interface
3. Click **Run** to start classification
4. Review predictions & confidence scores
5. Click **Download** to export results to CSV

## 8. End-to-End System Workflow



# 9. Installation & Execution

## 9.1 Dependencies

```
torch
pytorch-lightning
scikit-learn
tensorboard
albumentations
opencv-python
customtkinter
pillow
pandas
tqdm
tkinterdnd2
```

Install via:

```
pip install -r requirements.txt
```

## 9.2 Launching the Application

### Option 1 · Terminal (Command Line)

```
# General usage:
python predict.py [-c CHECKPOINT] -i INPUT_FOLDER [-o OUTPUT_CSV]

# Example:
# python predict.py -i ../Test -o ../Test_predictions_result.csv
```

### Option 2 · GUI (Graphical User Interface)

```
python GUI.py
```

## 9.3 Training the Model (Optional)

The project includes a dedicated training entrypoint `train.py` to train and evaluate the 7-channel CNN using **PyTorch Lightning**.

### Example Usage:

From the project root:

```
python train.py --data_dir data/MPIDB --batch_size 32 --epochs 50
```

### Arguments

- `--data_dir` (default: `data/MPIDB`): dataset root containing `train/val/test`
- `--batch_size` (default: 32): training batch size
- `--epochs` (default: 50): maximum training epochs

## 10. Conclusion

This software provides a modular, extensible, and production-ready pipeline for malaria species classification. It combines modern deep-learning practices with a practical GUI, making it suitable for both research and operational use.