Malaria Detection using Deep Learning (ResNet Model)

Table of Content

Malaria . Detection . using . Deep . Learning . (ResNet . Model) .

Mr nnam y Date
jan .jan @td .a .in

Abstract

This paper examines image analysis studies aimed at automating the detection or screening of malaria infection in thin blood film smears under a microscope. Malaria is now a leading cause of death in tropical and sub-tropical countries, killing over one million people per year worldwide, with African children responsible for 90% of fatalities. Despite the fact that successful malaria management methods now exist, the number of malaria cases continues to rise due to a variety of factors. As a result, the project's aim is to implement a solution for rapid and accurate malaria diagnosis. Our aim is to develop a fully automated image classification system to positively identify malaria parasites present in thin blood smears, This project examines image analysis studies aimed at automating the detection or screening of malaria infection in thin blood film smears under a microscope. Malaria is now a leading cause of death in tropical and sub-tropical countries, killing over one million people per year worldwide, with African children responsible for 90% of fatalities. Despite the fact that successful malaria management methods now exist, the number of malaria cases continues to rise due to a variety of factors. As a result, the project's aim is to implement a solution for rapid and accurate malaria diagnosis.

I. Introduction

- **Origination of problem**

Malaria control is a difficult problem all over the world, but particularly in Asia and Africa. People in the European region remain at risk from diseases borne by vectors both throughout the region and while traveling abroad, also 110 years after Ronald Ross was awarded the Nobel Prize for his work on malaria. Although treating malaria is a difficult problem in and of itself, early detection is also a significant issue. Plasmodium falciparum, Plasmodium vivax, Plasmodium oval, and Plasmodium malaria are the four major species of malaria parasites that infect humans. Plasmodium vivax is a parasitic infection that is often present in tropical and subtropical areas and has a severe clinical manifestation. The mainstay of disease control is quick identification of the parasite in human blood and early administration of antimalarial drugs. Before starting care, the WHO recommends that all cases of suspected malaria be confirmed using parasite-based diagnostic testing (either microscopy or rapid diagnostic test). Microscopy-based diagnosis is important in the malaria detection test for species differentiation, parasite quantification, and extreme disease control. Furthermore, because of its scalability and low operating costs, the method may be ideal for a wider range of people. From the blood of patients who are clinically suspected of having malaria, two forms of blood smears are prepared: thick and thin. The thick smear is better for parasite detection, while the thin smear is best for malaria species identification. Malaria can be detected 20 times faster in thick smear than in thin smear when the parasite load is light.

- **Existing solutions and its issues**

Most of the existing methods to this solution is carried out by Convolutional Neural Networks. But every consecutive winning design employs more layers in a deep learning to minimise the error rate after the very first CNN-based design (AlexNet) won the ImageNet 2012 competition. This works for a small number of layers, but as the number of hidden layers grows, a typical problem in neural networks called Vanishing/Exploding gradient emerges. As a result, the gradient becomes 0 or too huge. As a result, as the number of hidden layers increases, so does the training and test error rate.

This model presents the idea of the Residual Network to overcome the issue of the vanishing/exploding gradient. We employ a method called skip connections in this network. The skip connection bypasses a few stages of training and links directly to the output.

II. Literature Survey

Title	Author	Summary
Deep Learning Based Automatic Malaria Parasite Detection from Blood Smear and Its Smartphone Based Application	K. M. Faizullah Fuhad, Jannat Ferdousey Tuba, Md. Rabiul Ali Sarker, Sifat Momen, Nabeel Mohammed, and Tanzilur Rahman*	In this research paper, they proposed a completely automated Convolutional Neural Network (CNN) model for detecting malaria from small blood smear images. Various methods including refinement of filters were filtered, data augmentation, Autoencoder, feature extraction with a CNN model and separated by Support Vector Machine (SVM) or KNearest Neighbors (KNN) were performed under three training procedures called standard training, -distillation and autoencoder training to increase and improve the accuracy of the model and the performance of the consideration. Our Deep learning-based model can detect malaria
Leveraging Deep Learning Techniques for Malaria Parasite Detection Using Mobile Application	Mehedi Masud,Hesham Alhumyani,Sultan S. Alshamrani,Omar Cheikhrouhou,Saleh Ibrahim,Ghulam Muhammad,M. Shamim Hossain,Mohammad Shorfuzzaman	The purpose of this paper was to demonstrate that deep learning structures such as convolutional neural network (CNN) can be useful in detecting real-time malaria effectively and accurately from input images and reducing manual labor through a mobile app. We are currently testing the performance of a bespoke CNN model utilising a cyclical stochastic gradient descent (SGD) optimizer with automatic reading rate, and we have achieved 97.30 percent accuracy in distinguishing healthy and viral cell pictures with a high level of accuracy and sensitivity. This paper result will help diagnose malaria microscopy in a mobile app to

Malaria Parasite Detection From Peripheral Blood Smear Images Using Deep Belief Networks	Dhanya Bibin Madhu S. Nair ,(Senior Member, IEEE),P. Punitha	This paper proposes a novel method of detecting the presence of malaria parasites in human images that use blood through a network of myths (DBN). This paper introduces a professional model based on DBN to classify 4100 images of peripheral blood smear into a parasite or non-parasite category. The proposed DBN has been trained in advance by installing Boltzmann's restricted equipment using a different deviation method of pre-training. To train DBN, it removes the features from the images and start the DBN visual effects. The combined color and texture element is used as the element vector in this paper. Finally, DBN is well-organized in a discriminatory manner using a retrospective algorithm that incorporates possible class labels. The maximum DBNstructure used in this paper is 484-600-600-600-600-2, where the visible layer has 484 layers and the output layer has two four-layer layers containing 600 hidden nodes throughout the layer. The proposed method worked much better
Malaria Detection using Image Processing and Machine Learning	Cynara Gomes , Abhishek Kanojiya , Abhishek Yadav, Kirti Motwani	The most common and most common way to detect malaria is to visualize blood smears with a microscope of red blood cells infected with the virus under a microscope by qualified specialists. This method is ineffective and time consuming and the diagnosis is based on the experience and knowledge of the person performing the test. Detection of Malaria from small blood smear images requires the separation of single blood cells into small blood slide images, which can be taken by a pathologist and database, will contain undistributed cell images. Therefore, the proposed classification is done using a variety of image processing methods. Edge detection techniques and separation techniques used in this system overcome the problem of cell division by removing noise

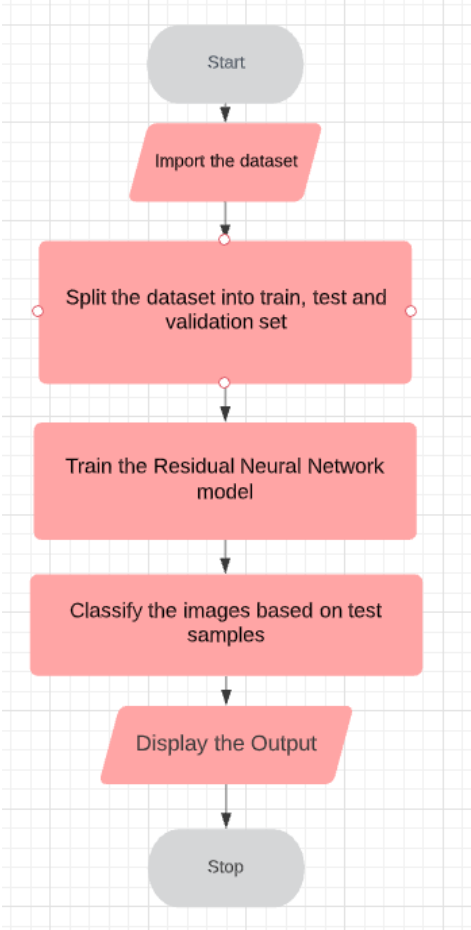
Recent Advances of Malaria Parasites Detection Systems Based on Mathematical Morphology	Andrea Loddo , Cecilia Di Ruberto and Michel Kocher	This work reviewed a number of computational microscopic imaging techniques aimed at the process of mathematical morphology, proposed in the literature to detect malaria parasites and classification of microscopic smear images. The computerized observational methods reported in the literature are based on microscopic images of the souls of the bloodstream through the discovery of computer-assisted malaria parasites and their various stages of life. Preliminary imaging, classification of erythrocytes and parasites, malaria parasites, and malaria detection strategies are discussed here. Mathematics morphology methods have been widely used for image processing purposes. Among the application fields, used fingerprint extraction, manual digit recognition, license plate detection, border extraction, de-noising extraction using morphological filters, text extraction and so on. In addition to these types of fields, mathematical morphology has been used successfully in biomedical image analysis, especially in preprocessing and classification techniques. Morphological cell analysis is used to address abnormal diagnosis and isolation, early detection of cancer. Incorporated into new biomedical methods, such as spontaneous classification and analysis of histological tumor components, boundary detection of cervical cell nuclei is considered for fragmentation and adhesion, gradeclassification and spatial distribution analysis, morphological features of specific cells, biomedical understanding of biomedical response -chemotactic and drug, or target cell morphogenesis in the continuation of a different cell cycle. Finally, it is
Detection of Malarial Parasite from Blood Smear Image	Manasvi, Sai Aarthi Ganesh , Sridhar Anjali, Kunnavakkam Vinjimoor Swetha, Krishnamoorthi Nirmala	To overcome the limitations of the standard method of calculation, the default method of testing process is presented in this paper. An automated system is provided to detect the presence of malaria parasites in blood smears using imaging techniques. The entire diagnostic procedure can be performed quickly using imaging-based tests and is more beneficial than laboratory procedures. Pre-screening and classification techniques were followed by detection of infected cells to identify the presence of malaria parasites in blood-

Detection of Malarial Parasite in Blood Using Image Processing	Pallavi T. Suradkar	The purpose of this paper is to develop an automated image classification system to better identify malaria parasites present in small blood smears, and to classify species. The algorithm produced will help in an area where an expert in small analysis is not available. An algorithm attempt to determine the presence of insects at any stage. One parasite develops in the body for 7 to 8 days without symptoms. Therefore if this algorithm is included in standard tests, the detection of malaria parasite can be detected by automatic parasite detection based on color
Comparative Study of Malaria Parasite Detection using Euclidean Distance Classifier & SVM	Ms. Snehal Suryawanshi, Prof. V. V. Dixit	This paper presents advanced techniques for detecting Malaria Parasites, in which the cell division process is divided into several steps, including image binarization using a Poisson distribution based on Minimum Error Thresholding, followed by the opening of Morphological for purification purposes. Seed drilling is done for a multi-stage LoG filter. Since the representation of frequency and filtering of Gabor is similar to that of the human viewing system, it is used to exclude the feature. The two algorithms are compared in this paper to obtain high differentiation. The results show that SVM (Support
Microscopic determination of malaria parasite load: role of image analysis	Frean J	There are two acceptable ways to express viral load: either the percentage of erythrocytes infected as calculated in a thin blood film or the number of parasites per unit of blood volume. The latter is usually tested in a thick colored film by counting insect-resistant leukocytes, and then repeated with the patient's leukocyte count if available, or a standard count. When the number of insects is too high or too low, a thick film and a small film count, respectively, is not true. Ideally, therefore, both methods should be in the best repertoire of a malaria microscopist. The

Digital analysis of changes by Plasmodium vivax malaria in erythrocytes	Edison, Maombi, Jeeva, J.B. Singh, Megha	Blood samples of malaria patients were selected based on the severity of the parasitemia, divided into low (LP), medium (MP) and high (HP) parasitemia, representing increasing levels of the severity of the disease. Healthy subjects without a history of disease were selected as the control group. Through the analysis of erythrocyte images their contours were found and local parameters were obtained, perimeter and form factor were identified. The size of the gray matter is determined by erythrocyte scanning along its maximum diameter. Comparisons of this with normal cells have shown significant changes in shape parameters. The size of the gray matter

III. Procedures for Proposed Method

Flow Diagram



Algorithm

In this project, we will use a python dependency called Pytorch which is used commonly used library in image processing and in this project we have used the concepts of Deep learning by implementing with Pytorch:

1. Firstly, we take the input images and fit it in our training model.

2. Then after training our model our model is finally trained and ready for predicting the outputs. 3. In this project, we will use Residual networks as our neural networks architecture.

3. After training, the testing of the model is done and it will give us the testing accuracy.

4. Finally, we will load our input image from dataset and we use scikit image library in our code to import our image. 5. Then our model will be ready to predict the images that the blood cell is malaria positive or not.

5. Then our model will be ready to predict the images that the blood cell is malaria positive or not.

IV Implementation

1. First we do some pre-processing of data.

```
train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                       transforms.RandomResizedCrop(224),
                                       transforms.RandomVerticalFlip(),
                                       transforms.ToTensor(),
                                       transforms.Normalize([0.485, 0.456, 0.406],
                                                            [0.229, 0.224, 0.225])])

test_transforms = transforms.Compose([transforms.Resize(256),
                                      transforms.CenterCrop(224),
                                      transforms.ToTensor(),
                                      transforms.Normalize([0.485, 0.456, 0.406],
                                                           [0.229, 0.224, 0.225])])

validation_transforms = transforms.Compose([transforms.Resize(256),
                                            transforms.CenterCrop(224),
                                            transforms.ToTensor(),
                                            transforms.Normalize([0.485, 0.456, 0.406],
                                                                 [0.229, 0.224, 0.225])])
```

2. Preparing loaders for training, test and validation

```
# define samplers for obtaining training and validation batches
train_sampler = SubsetRandomSampler(train_idx)
valid_sampler = SubsetRandomSampler(valid_idx)
test_sampler = SubsetRandomSampler(test_idx)
# prepare data loaders (combine dataset and sampler)
train_loader = torch.utils.data.DataLoader(train_data, batch_size=64,
    sampler=train_sampler, num_workers=num_workers)
valid_loader = torch.utils.data.DataLoader(train_data, batch_size=32,
    sampler=valid_sampler, num_workers=num_workers)
test_loader = torch.utils.data.DataLoader(train_data, batch_size=20,
    sampler=test_sampler, num_workers=num_workers)
```

3. Instantiate the model

```
model = models.resnet50(pretrained=True)

for param in model.parameters():
    param.requires_grad = False

model.fc = nn.Linear(2048, 2, bias=True)

fc_parameters = model.fc.parameters()

for param in fc_parameters:
    param.requires_grad = True

model
```

4. Training the model

```
train(25, model, optimizer, criterion, use_cuda, 'malaria_detection.pt')
```

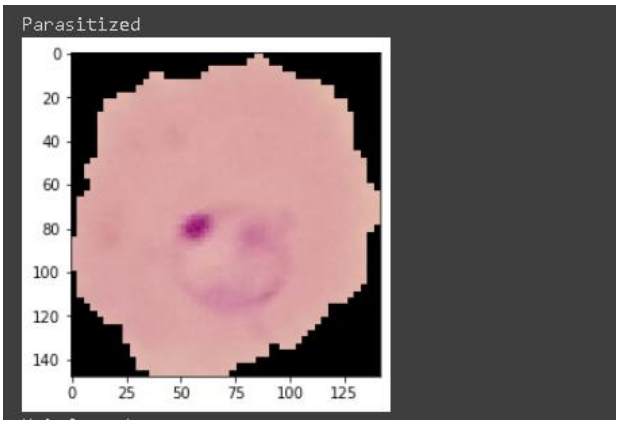
```
Epoch 1, Batch 1 loss: 0.745345
Epoch 1, Batch 101 loss: 0.529333
Epoch 1, Batch 201 loss: 0.475959
Epoch 1, Batch 301 loss: 0.448897
Epoch: 1        Training Loss: 0.448489         Validation Loss: 0.391722
Epoch 2, Batch 1 loss: 0.346056
Epoch 2, Batch 101 loss: 0.389994
Epoch 2, Batch 201 loss: 0.379996
Epoch 2, Batch 301 loss: 0.378201
Epoch: 2        Training Loss: 0.377935         Validation Loss: 0.382184
Epoch 3, Batch 1 loss: 0.393731
Epoch 3, Batch 101 loss: 0.378880
Epoch 3, Batch 201 loss: 0.377859
Epoch 3, Batch 301 loss: 0.374876
Epoch: 3        Training Loss: 0.374550         Validation Loss: 0.373698
Epoch 4, Batch 1 loss: 0.417771
```

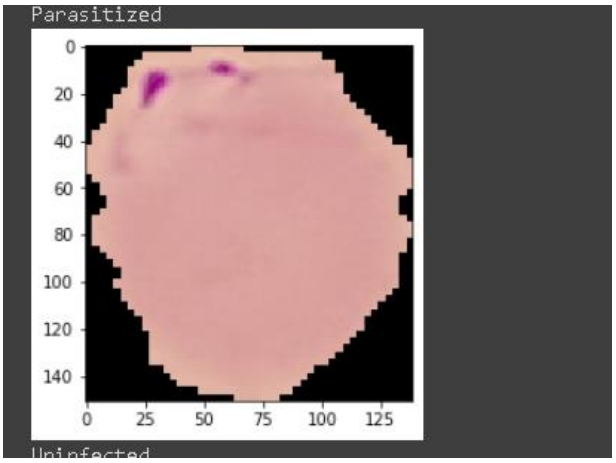
5. Testing the model

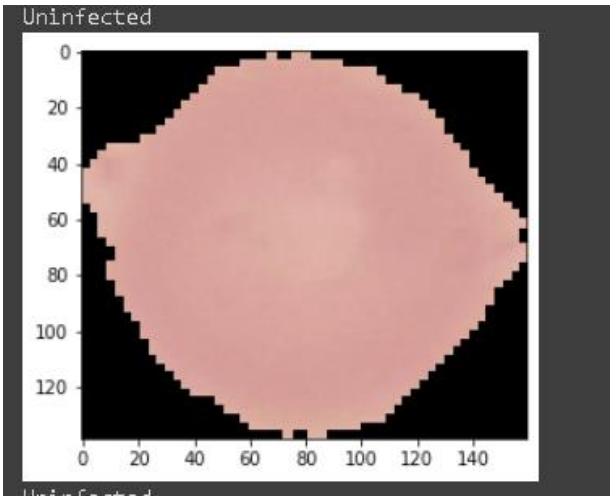
```
def test(model, criterion, use_cuda):


    test_loss = 0.
    correct = 0.
    total = 0.

    for batch_idx, (data, target) in enumerate(test_loader):

        if use_cuda:
            data, target = data.cuda(), target.cuda()
        output = model(data)
        loss = criterion(output, target)
        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data - test_loss))
        pred = output.data.max(1, keepdim=True)[1]
        correct += np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())
        total += data.size(0)

    print('Test Loss: {:.6f}\n'.format(test_loss))

    print('\nTest Accuracy: %2d%% (%2d/%2d)' % (
        100. * correct / total, correct, total))
test(model, criterion, use_cuda)
```

V Results and Discussion

We have classified the images into parasitized and uninfected.


Parasitized
0
20
40
60
80
100
120
140
0
25
50
75
100
125


Parasitized
0
20
40
60
80
100
120
140
0
25
50
75
100
125
Uninfected



Testing the model

For testing purposes, we loaded the model and obtained the output, which was then compared to the target photos, yielding the loss. The test accuracy and the test loss were also calculated.

Visualizing the output

We created functions for visualising three images each from infected to uninfected, and determining whether they were correctly categorised or not. For this, we abstracted the class names of the image that was created by the model being parasitized or uninfected Then we printed the photographs together with their class information.

VI Result and Discussion

We ran the algorithm for 5 epochs, and following are the result that is obtained:

For the five epochs, we were able to get a test loss of 37.8073% and a Test Accuracy of 83%. Hence it was observed that as the number of epochs increased, the test accuracy didn't significantly increase and it was not much better than with smaller epochs. Also as you can see from the second image of the first row of figure= there seems to be a misclassified output, where an obviously infected cell was classified as "Uninfected". This can be due to the fact the number of epochs is too low for us to get a more evenly

correct set of results. But as the number of epochs increases, the time of computation also increases, hence we were not able to run for extremely high values of epochs.

Hence, by performing the following experiments, we were able to completely learn about the ResNet50 model architecture and its functions. Also, in addition to it, we were able to learn the basics of PyTorch and its various deep learning features, and was also able to learn how to implement them in a situation involving actual real life data representation.

VII Conclusion

Malaria parasites are manually detected using microscopes by pathologists. As a result, the odds of a false positive owing to human mistake are considerable, which can lead to a catastrophic situation. Using image processing and automation, this session reduces human error while detecting the presence of malaria parasites in a sample of blood.

References

[1] Deep Learning Based Automatic Malaria Parasite Detection from Blood Smear and Its Smartphone Based Application, K. M. Faizullah Fuhad, Jannat Ferdousey Tuba, Md. Rabiul Ali Sarker, Sifat Momen, Nabeel Mohammed, and Tanzilur Rahman*

[2] Leveraging Deep Learning Techniques for Malaria Parasite Detection Using Mobile Application, Mehedi Masud,Hesham Alhumyani,Sultan S. Alshamrani,Omar Cheikhrouhou,Saleh Ibrahim,Ghulam Muhammad,M. Shamim Hossain,Mohammad Shorfuzzaman

[3] DHANYA BIBIN,MADHU S. NAIR, Malaria Parasite Detection From Peripheral Blood Smear Images Using Deep Belief Networks.

[4] Cynara Gomes , Abhishek Kanojiya , Abhishek Yadav, Kirti Motwani, " Malaria Detection using Image Processing and Machine Learning"

[5] Andrea Loddo , Cecilia Di Ruberto and Michel Kocher, "Recent Advances of Malaria Parasites Detection Systems Based on Mathematical Morphology"

[6] Manasvi, Sai Aarthi Ganesh , Sridhar Anjali, Kunnavakkam Vinjimoor Swetha, Krishnamoorthi Nirmala, "Detection of Malarial Parasite from Blood Smear Image"

[7] Pallavi T. Suradkar, "Detection of Malarial Parasite in Blood Using Image Processing" [8] Ms. Snehal Suryawanshi, Prof. V. V. Dixit, "Comparative Study of Malaria Parasite Detection using Euclidean Distance Classifier & SVM"

[9] Frean J, "Microscopic determination of malaria parasite load: role of image analysis"

[10] Edison, Maombi, Jeeva, J.B. Singh, Megha, “Digital analysis of changes by Plasmodium vivax malaria in erythrocytes”

Appendix

Source Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import torch
from torch import nn, optim
from torchvision import transforms, datasets, models
from torch.utils.data.sampler import SubsetRandomSampler
import os
print(os.listdir("../Downloads/cell_images/cell_images/"))
train_transforms = transforms.Compose([transforms.RandomRotation(30),
transforms.RandomResizedCrop(224),
transforms.RandomVerticalFlip(),
transforms.ToTensor(),
transforms.Normalize([0.485, 0.456, 0.406],
                     [0.229, 0.224, 0.225])])
test_transforms = transforms.Compose([transforms.Resize(256),
transforms.CenterCrop(224),
transforms.ToTensor(),
transforms.Normalize([0.485, 0.456, 0.406],
                     [0.229, 0.224, 0.225])])
validation_transforms = transforms.Compose([transforms.Resize(256),
transforms.CenterCrop(224),
transforms.ToTensor(),
```

```
transforms.Normalize([0.485, 0.456, 0.406],
                     [0.229, 0.224, 0.225])])

img_dir='../Downloads/cell_images/cell_images/'
train_data = datasets.ImageFolder(img_dir,transform=train_transforms)

num_workers = 0

# percentage of training set to use as validation
valid_size = 0.2


test_size = 0.1

transform = transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
        ])

num_train = len(train_data)

indices = list(range(num_train))

np.random.shuffle(indices)

valid_split = int(np.floor((valid_size) * num_train))

test_split = int(np.floor((valid_size+test_size) * num_train))

valid_idx, test_idx, train_idx = indices[:valid_split], indices[valid_split:test_split],
indices[test_split:]

print(len(valid_idx), len(test_idx), len(train_idx))


train_sampler = SubsetRandomSampler(train_idx)

valid_sampler = SubsetRandomSampler(valid_idx)

test_sampler = SubsetRandomSampler(test_idx)
```

```
train_loader = torch.utils.data.DataLoader(train_data, batch_size=64,
        sampler=train_sampler, num_workers=num_workers)
valid_loader = torch.utils.data.DataLoader(train_data, batch_size=32,
        sampler=valid_sampler, num_workers=num_workers)
test_loader = torch.utils.data.DataLoader(train_data, batch_size=20,
        sampler=test_sampler, num_workers=num_workers)

model = models.resnet50(pretrained=True)
for param in model.parameters():
        param.requires_grad = False
model.fc = nn.Linear(2048, 2, bias=True)
fc_parameters = model.fc.parameters()

for param in fc_parameters:
        param.requires_grad = True
model
use_cuda = torch.cuda.is_available()
if use_cuda:
        model = model.cuda()
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.fc.parameters(), lr=0.001 , momentum=0.9)
def train(n_epochs, model, optimizer, criterion, use_cuda,     save_path):
        """returns trained model"""

        valid_loss_min = np.Inf
```

```
for epoch in range(1, n_epochs+1):
    train_loss = 0.0
    valid_loss = 0.0
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        # move to GPU
        if use_cuda:
            data, target = data.cuda(), target.cuda()

        optimizer.zero_grad()

        output = model(data)

        loss = criterion(output, target)
        loss.backward()

        optimizer.step()
    train_loss = train_loss + ((1 / (batch_idx + 1)) * (loss.data - train_loss))
        if batch_idx % 100 == 0:
            print('Epoch %d, Batch %d loss: %.6f' %
                (epoch, batch_idx + 1, train_loss))

    model.eval()
    for batch_idx, (data, target) in enumerate(valid_loader):
        if use_cuda:
```

```
                    data, target = data.cuda(), target.cuda()

                output = model(data)

                loss = criterion(output, target)

                valid_loss = valid_loss + ((1 / (batch_idx + 1)) *
(loss.data - valid_loss))

            print('Epoch: {} \tTraining Loss: {:.6f} \tValidation Loss:
{:.6f}'.format(

                epoch,

                train_loss,

                valid_loss

                ))

            if valid_loss < valid_loss_min:

                torch.save(model.state_dict(), save_path)

    return model


train(25, model, optimizer, criterion, use_cuda, 'malaria_detection.pt')
model.load_state_dict(torch.load('malaria_detection.pt'))
def test(model, criterion, use_cuda):
    test_loss = 0.

    correct = 0.

    total = 0.

    for batch_idx, (data, target) in enumerate(test_loader):
        if use_cuda:

            data, target = data.cuda(), target.cuda()

        output = model(data)

        loss = criterion(output, target)

        test_loss = test_loss + ((1 / (batch_idx + 1)) * (loss.data -
test_loss))
```

```
            pred = output.data.max(1, keepdim=True)[1]

            correct +=
np.sum(np.squeeze(pred.eq(target.data.view_as(pred))).cpu().numpy())

            total += data.size(0)


        print('Test Loss: {:.6f}\n'.format(test_loss))


        print('\nTest Accuracy: %2d%% (%2d/%2d)' % (

            100. * correct / total, correct, total))

test(model, criterion, use_cuda)

        def load_input_image(img_path):

        image = Image.open(img_path)
        prediction_transform = transforms.Compose([transforms.Resize(size=(224, 224)),

                                                   transforms.ToTensor(),


transforms.Normalize([0.485, 0.456, 0.406],


             [0.229, 0.224, 0.225])])

        image = prediction_transform(image)[:3,:,:].unsqueeze(0)

        return image
```