Data Collection and Preprocessing Phase

Date	15 August 2024
Team ID	LTVIP2024TMID24772
Project Title	Implementation of Deep Learning Techniques to Detect Malaria
Maximum Marks	6 Marks

Preprocessing Template

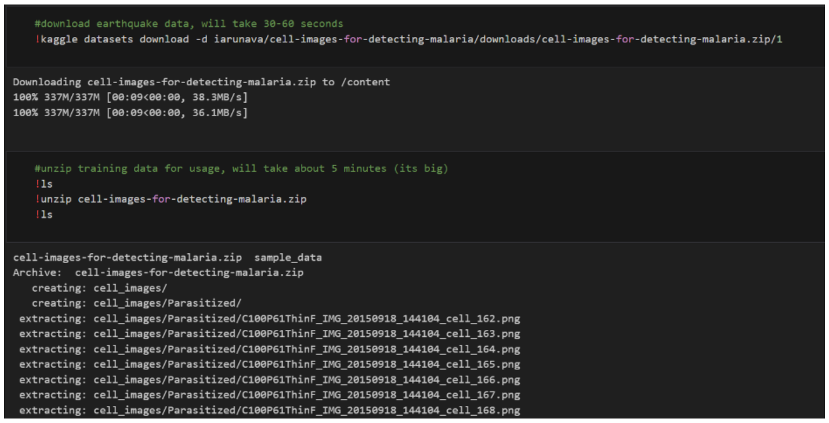
The images will be preprocessed by resizing, normalizing, augmenting, denoising, adjusting contrast, detecting edges, converting color space, cropping, batch normalizing, and whitening data. These steps will enhance data quality, promote model generalization, and improve convergence during neural network training, ensuring robust and efficient performance across various computer vision tasks.

Section	**Description**
Data Overview	The dataset will consist of blood smear images, including both malaria-infected and non-infected samples. The images will be collected from sources like Kaggle, with a total size of approximately 27,000 images in PNG/JPEG format.
Resizing	Resize images to a specified target size (e.g., 224x224 pixels) to ensure uniform input for the neural network.
Normalization	Normalize pixel values to a specific range (e.g., 0 to 1) by dividing the pixel values by 255 to improve model convergence.
Data Augmentation	Apply augmentation techniques such as flipping, rotation, shifting, zooming, or shearing.
Denoising	Apply denoising filters to reduce noise in the images.
Edge Detection	Apply edge detection algorithms to highlight prominent edges in the images.

Color Space Conversion	Convert images from RGB to grayscale or other color spaces to simplify the data and focus on relevant features for malaria detection.
Image Cropping	Crop images to focus on regions containing objects of interest, ensuring the model learns from the most relevant parts of the images.
Batch Normalization	Apply batch normalization to the input of each layer in the neural network.

Data Preprocessing Code Screenshots

Loading Data



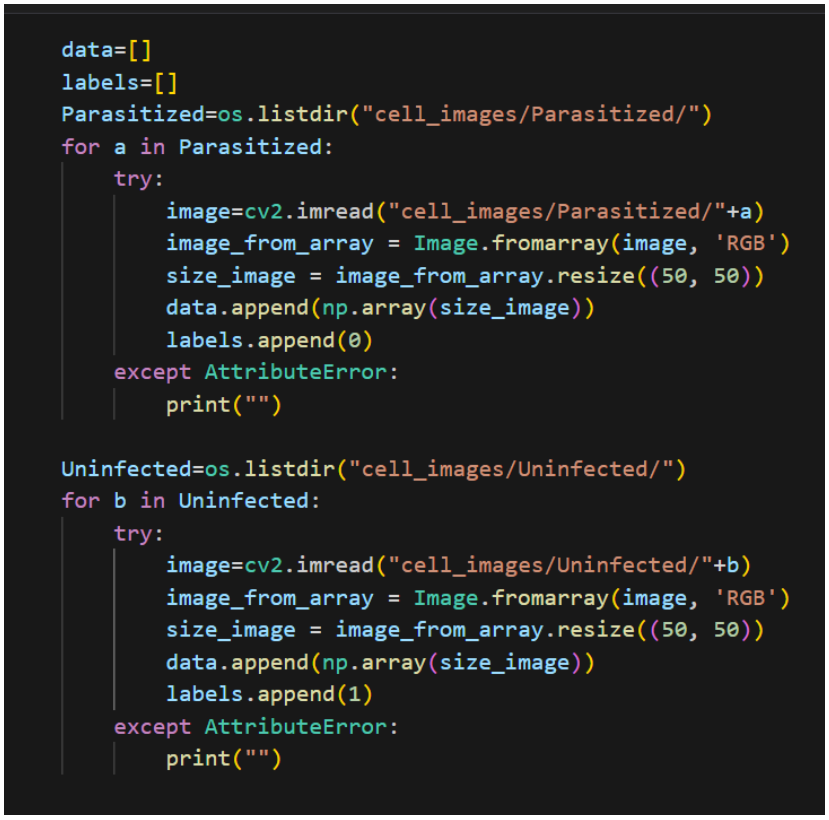
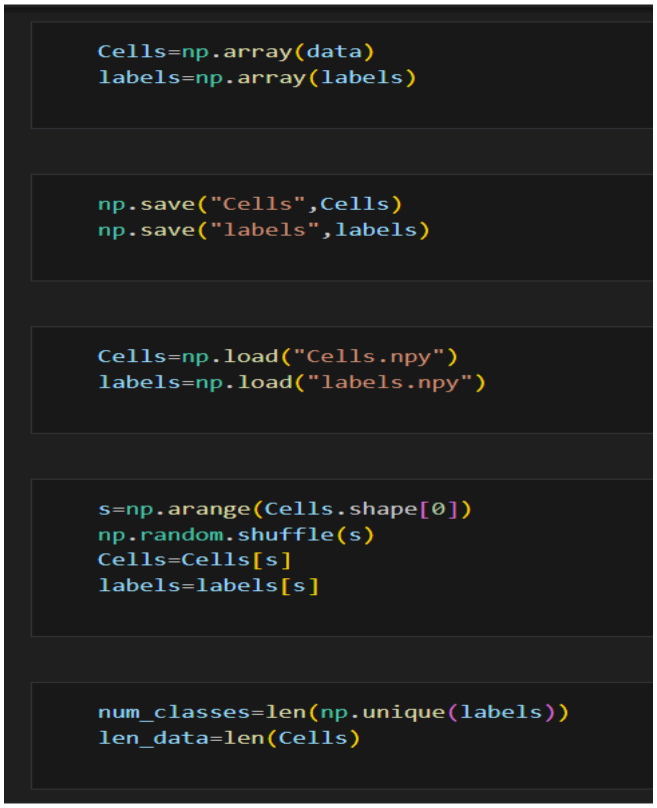
```
    #download earthquake data, will take 30-60 seconds
    !kaggle datasets download -d iarunava/cell-images-for-detecting-malaria/downloads/cell-images-for-detecting-malaria.zip/1
```

```
Downloading cell-images-for-detecting-malaria.zip to /content
100% 337M/337M [00:09<00:00, 38.3MB/s]
100% 337M/337M [00:09<00:00, 36.1MB/s]
```

```
    #unzip training data for usage, will take about 5 minutes (its big)
    !ls
    !unzip cell-images-for-detecting-malaria.zip
    !ls
```

```
cell-images-for-detecting-malaria.zip  sample_data
Archive:  cell-images-for-detecting-malaria.zip
   creating: cell_images/
   creating: cell_images/Parasitized/
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_162.png
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_163.png
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_164.png
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_165.png
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_166.png
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_167.png
 extracting: cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104_cell_168.png
```

Resizing	

	
Normalization	

```
data=[]
labels=[]
Parasitized=os.listdir("cell_images/Parasitized/")
for a in Parasitized:
    try:
        image=cv2.imread("cell_images/Parasitized/"+a)
        image_from_array = Image.fromarray(image, 'RGB')
        size_image = image_from_array.resize((50, 50))
        data.append(np.array(size_image))
        labels.append(0)
    except AttributeError:
        print("")

Uninfected=os.listdir("cell_images/Uninfected/")
for b in Uninfected:
    try:
        image=cv2.imread("cell_images/Uninfected/"+b)
        image_from_array = Image.fromarray(image, 'RGB')
        size_image = image_from_array.resize((50, 50))
        data.append(np.array(size_image))
        labels.append(1)
    except AttributeError:
        print("")
```

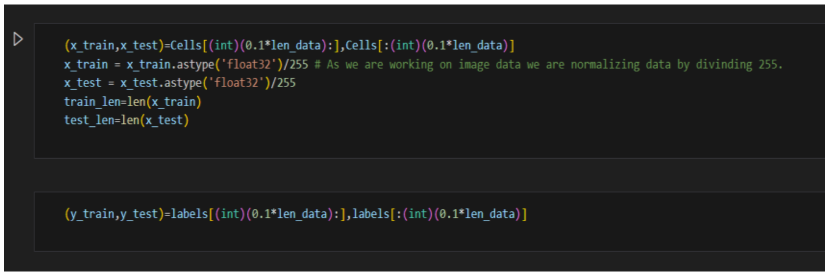
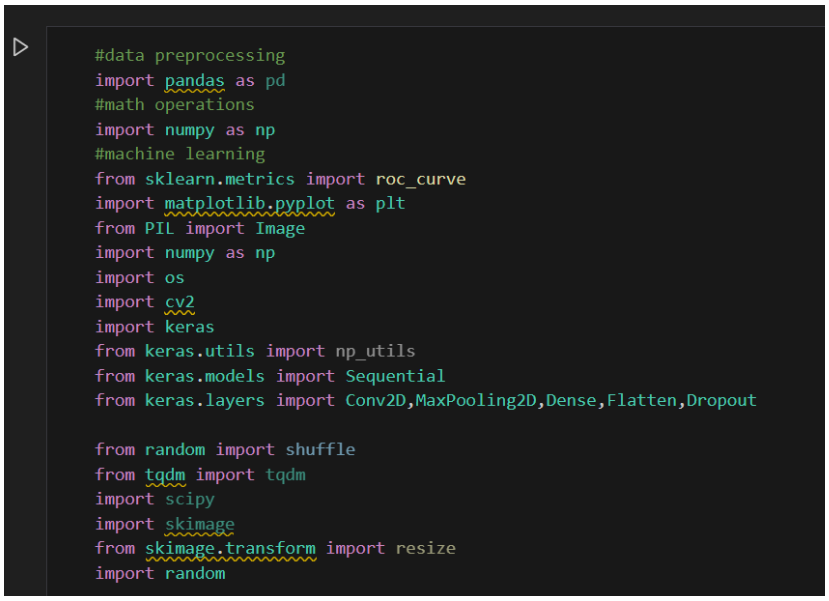
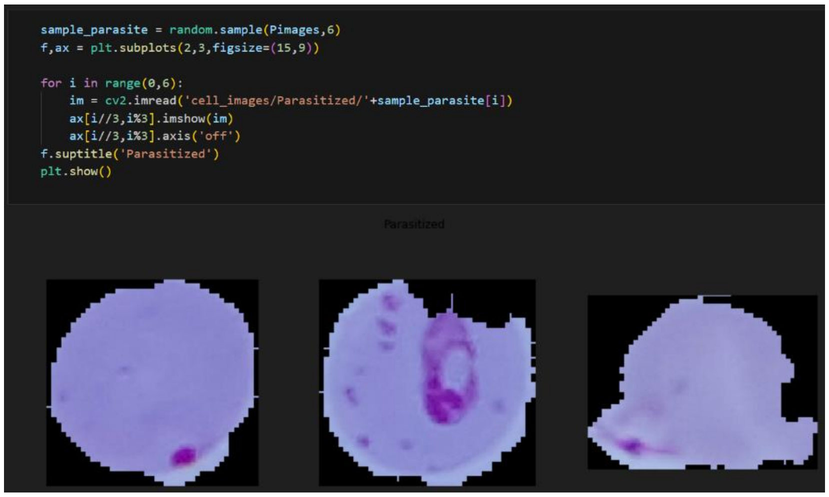
```
Cells=np.array(data)
labels=np.array(labels)
```

```
np.save("Cells",Cells)
np.save("labels",labels)
```

```
Cells=np.load("Cells.npy")
labels=np.load("labels.npy")
```

```
s=np.arange(Cells.shape[0])
np.random.shuffle(s)
Cells=Cells[s]
labels=labels[s]
```

```
num_classes=len(np.unique(labels))
len_data=len(Cells)
```

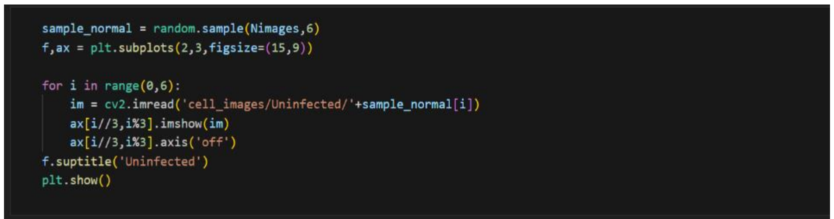
	
Data Augmentation	
Denoising	
Edge Detection	

```
(x_train,x_test)=Cells[(int)(0.1*len_data):],Cells[:(int)(0.1*len_data)]
x_train = x_train.astype('float32')/255 # As we are working on image data we are normalizing data by divinding 255.
x_test = x_test.astype('float32')/255
train_len=len(x_train)
test_len=len(x_test)
```

```
(y_train,y_test)=labels[(int)(0.1*len_data):],labels[:(int)(0.1*len_data)]
```

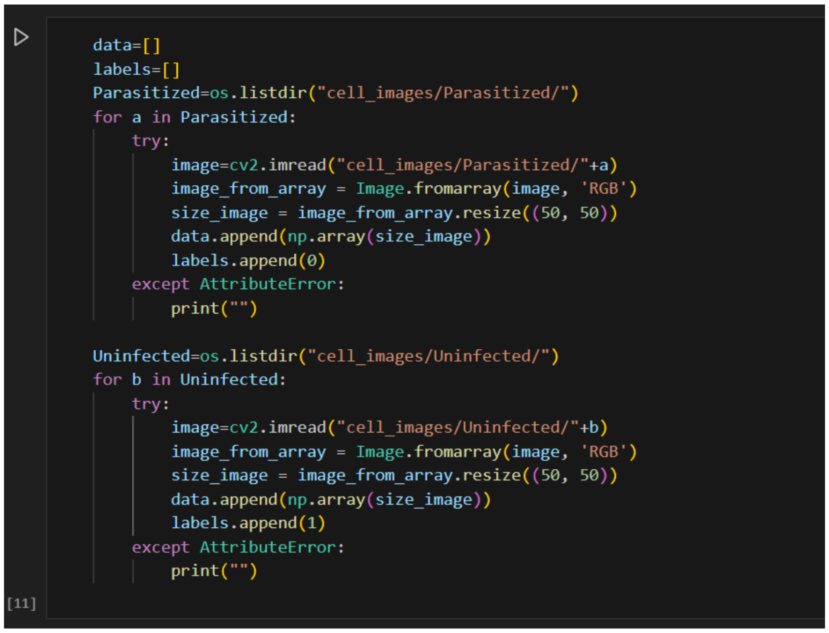
```
#data preprocessing
import pandas as pd
#math operations
import numpy as np
#machine learning
from sklearn.metrics import roc_curve
import matplotlib.pyplot as plt
from PIL import Image
import numpy as np
import os
import cv2
import keras
from keras.utils import np_utils
from keras.models import Sequential
from keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout

from random import shuffle
from tqdm import tqdm
import scipy
import skimage
from skimage.transform import resize
import random
```

```
sample_parasite = random.sample(Pimages,6)
f,ax = plt.subplots(2,3,figsize=(15,9))

for i in range(0,6):
    im = cv2.imread('cell_images/Parasitized/'+sample_parasite[i])
    ax[i//3,i%3].imshow(im)
    ax[i//3,i%3].axis('off')
f.suptitle('Parasitized')
plt.show()
```



```
sample_normal = random.sample(Nimages,6)
f,ax = plt.subplots(2,3,figsize=(15,9))

for i in range(0,6):
    im = cv2.imread('cell_images/Uninfected/'+sample_normal[i])
    ax[i//3,i%3].imshow(im)
    ax[i//3,i%3].axis('off')
f.suptitle('Uninfected')
plt.show()
```

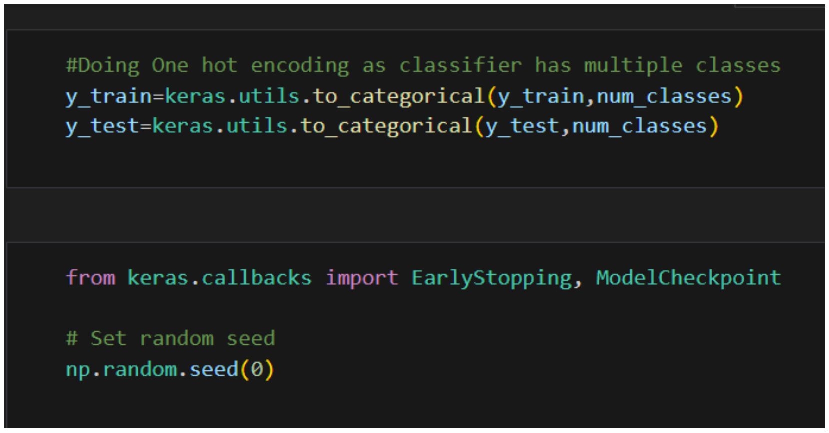
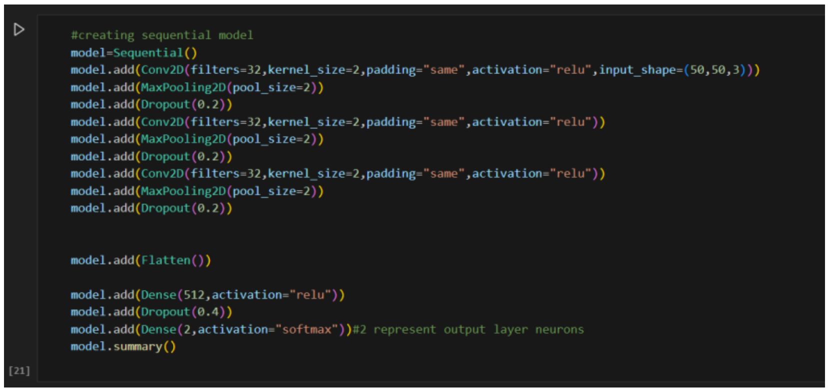
Color Space Conversion



```
data=[]
labels=[]
Parasitized=os.listdir("cell_images/Parasitized/")
for a in Parasitized:
    try:
        image=cv2.imread("cell_images/Parasitized/"+a)
        image_from_array = Image.fromarray(image, 'RGB')
        size_image = image_from_array.resize((50, 50))
        data.append(np.array(size_image))
        labels.append(0)
    except AttributeError:
        print("")

Uninfected=os.listdir("cell_images/Uninfected/")
for b in Uninfected:
    try:
        image=cv2.imread("cell_images/Uninfected/"+b)
        image_from_array = Image.fromarray(image, 'RGB')
        size_image = image_from_array.resize((50, 50))
        data.append(np.array(size_image))
        labels.append(1)
    except AttributeError:
        print("")
```

[11]

Image Cropping

	
Batch Normalization	

```
#Doing One hot encoding as classifier has multiple classes
y_train=keras.utils.to_categorical(y_train,num_classes)
y_test=keras.utils.to_categorical(y_test,num_classes)
```

```
from keras.callbacks import EarlyStopping, ModelCheckpoint

# Set random seed
np.random.seed(0)
```

```
#creating sequential model
model=Sequential()
model.add(Conv2D(filters=32,kernel_size=2,padding="same",activation="relu",input_shape=(50,50,3)))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))
model.add(Conv2D(filters=32,kernel_size=2,padding="same",activation="relu"))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))
model.add(Conv2D(filters=32,kernel_size=2,padding="same",activation="relu"))
model.add(MaxPooling2D(pool_size=2))
model.add(Dropout(0.2))


model.add(Flatten())

model.add(Dense(512,activation="relu"))
model.add(Dropout(0.4))
model.add(Dense(2,activation="softmax"))#2 represent output layer neurons
model.summary()
```
[21]