# EX-04 Deep Neural Network for Malaria Infected Cell Recognition

## DATE:

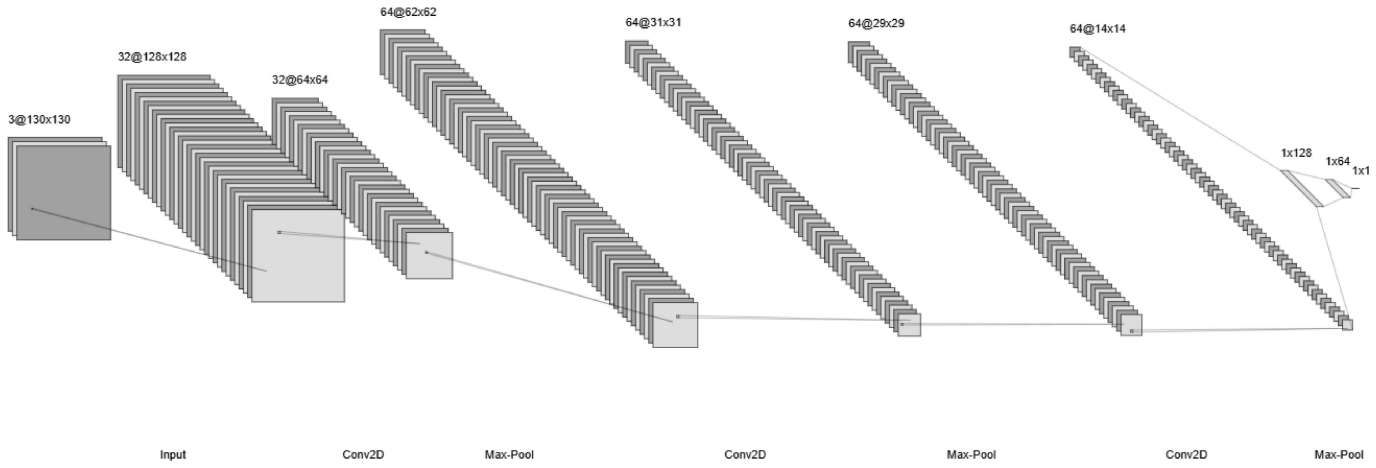
### Aim:

To develop a deep neural network for Malaria infected cell recognition and to analyze the performance.

### Problem Statement and Dataset:

The task is to automatically classify red blood cell images into two categories: parasitized (malaria-infected) and uninfected (healthy). Malaria-infected cells contain the Plasmodium parasite, while uninfected cells are healthy. The goal is to build a convolutional neural network (CNN) to accurately distinguish between these classes. Manual inspection of blood smears is time-consuming and prone to errors. By using deep learning, we can automate the process, speeding up diagnosis, reducing healthcare professionals' workload, and improving detection accuracy. The dataset consists of 27,558 annotated cell images, evenly split between parasitized and uninfected cells, providing a reliable foundation for model training and testing.

### Neural Network Model




### Design Steps

1. **Import Libraries**:Import TensorFlow, data preprocessing tools, and visualization libraries.
2. **Configure GPU**:Set up TensorFlow for GPU acceleration to speed up training.
3. **Data Augmentation**:Create an image generator for rotating, shifting, rescaling, and flipping to enhance model generalization.
4. **Build CNN Model**:Design a convolutional neural network with convolutional layers, max-pooling, and fully connected layers; compile the model.
5. **Train Model**:Split the dataset into training and testing sets, then train the model using the training data.
6. **Evaluate Performance**:Assess the model using the testing data, generating a classification report and confusion matrix.

### Program:

**Developed By: SASIDEVI V**

**Register No.: 212222230136**

```
import os
import pandas as pd
import numpy as np
```

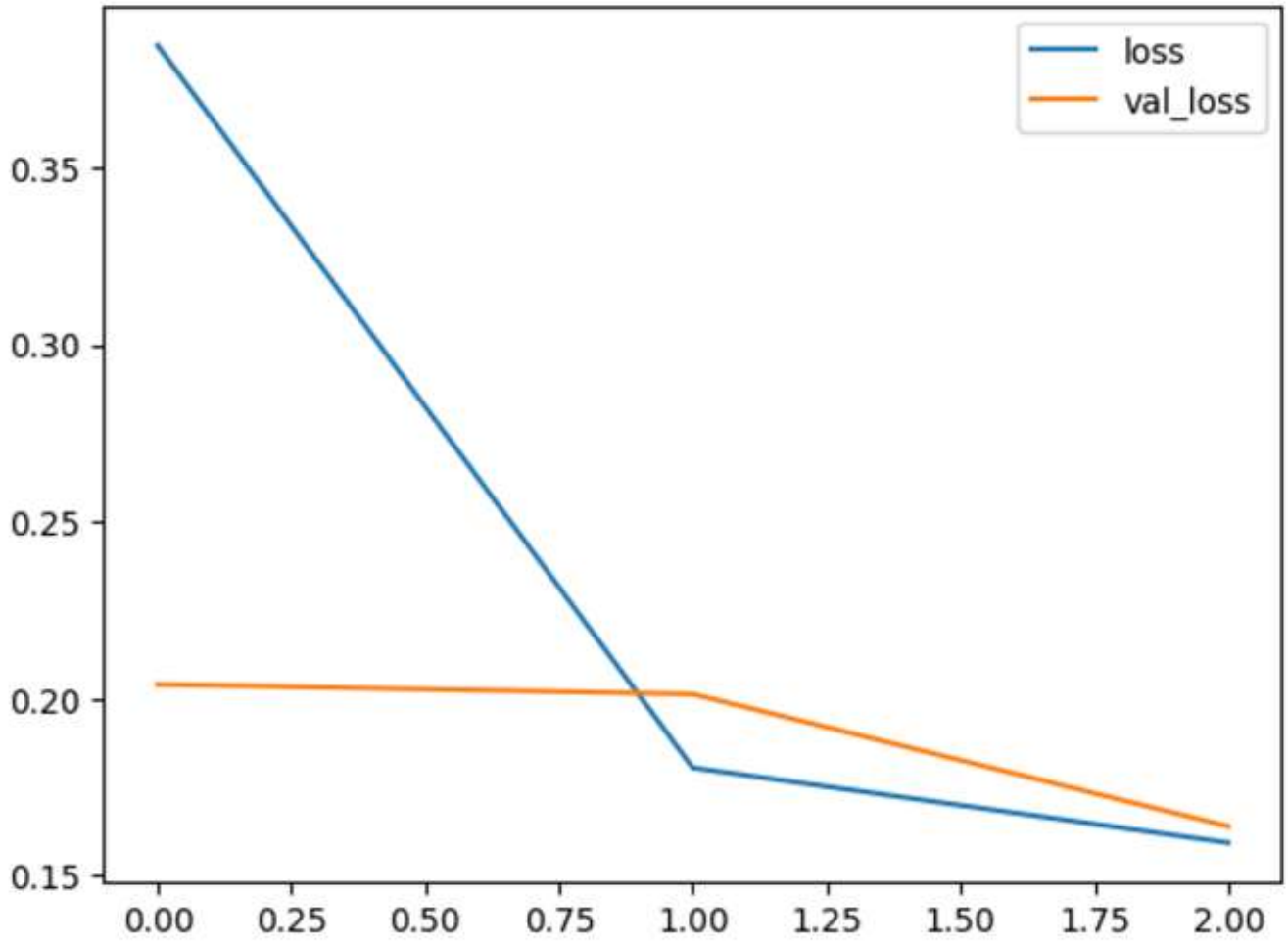
```
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import utils
from tensorflow.keras import models
from sklearn.metrics import classification_report,confusion_matrix
import tensorflow as tf
my_data_dir = 'dataset/cell_images'
os.listdir(my_data_dir)
test_path = my_data_dir+'/test/'
train_path = my_data_dir+'/train/'
os.listdir(train_path)
len(os.listdir(train_path+'/uninfected/'))
len(os.listdir(train_path+'/parasitized/'))
os.listdir(train_path+'/parasitized')[0]
para_img= imread(train_path+
                 '/parasitized/'+
                 os.listdir(train_path+'/parasitized')[0])
plt.imshow(para_img)
print('SASIDEVI - 212222230136')
dim1 = []
dim2 = []
for image_filename in os.listdir(test_path+'/uninfected'):
    img = imread(test_path+'/uninfected'+'/'+image_filename)
    d1,d2,colors = img.shape
    dim1.append(d1)
    dim2.append(d2)
sns.jointplot(x=dim1,y=dim2)
image_shape = (130,130,3)
help(ImageDataGenerator)
image_gen = ImageDataGenerator(rotation_range=20, # rotate the image 20 degrees
                               width_shift_range=0.10, # Shift the pic width by a max of
                               height_shift_range=0.10, # Shift the pic height by a max
                               rescale=1/255, # Rescale the image by normalzing it.
                               shear_range=0.1, # Shear means cutting away part of the i
                               zoom_range=0.1, # Zoom in by 10% max
                               horizontal_flip=True, # Allo horizontal flipping
                               fill_mode='nearest' # Fill in missing pixels with the nea
                              )
image_gen.flow_from_directory(train_path)
image_gen.flow_from_directory(test_path)
model = models.Sequential()
model.add(keras.Input(shape=(image_shape)))
# Add convolutional layers
model.add(layers.Conv2D(filters=32,kernel_size=(3,3),activation='relu',))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu',))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
model.add(layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu',))
model.add(layers.MaxPooling2D(pool_size=(2, 2)))
```

```
# Flatten the layer
model.add(layers.Flatten())
# Add a dense layer
model.add(layers.Dense(128, activation='relu'))
# Output layer
model.add(layers.Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
model.summary()
print("SASIDEVI V \n212222230136 ")
batch_size = 16
help(image_gen.flow_from_directory)
train_image_gen = image_gen.flow_from_directory(train_path,
                                                target_size=image_shape[:2],
                                                 color_mode='rgb',
                                                batch_size=batch_size,
                                                class_mode='binary')
train_image_gen.batch_size
len(train_image_gen.classes)
train_image_gen.total_batches_seen
test_image_gen = image_gen.flow_from_directory(test_path,
                                                target_size=image_shape[:2],
                                                color_mode='rgb',
                                                batch_size=batch_size,
                                                class_mode='binary',shuffle=False)
train_image_gen.class_indices
results = model.fit(train_image_gen,epochs=3,
                              validation_data=test_image_gen
                             )
model.save('cell_modelsasi.h5')
losses = pd.DataFrame(model.history.history)
losses[['loss','val_loss']].plot()
print("SASIDEVI \n212222230136")
model.metrics_names
model.evaluate(test_image_gen)
pred_probabilities = model.predict(test_image_gen)
test_image_gen.classes
predictions = pred_probabilities > 0.5
print(classification_report(test_image_gen.classes,predictions))
print('SASIDEVI V \n212222230136')
print(confusion_matrix(test_image_gen.classes,predictions))
print('SASIDEVI V \n212222230136')
```

## Output:

### Training Loss, Validation Loss Vs Iteration Plot

SASIDEVI
212222230136




Classification Report

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.97 | 0.91 | 0.94 | 1300 |
| 1 | 0.92 | 0.97 | 0.95 | 1300 |
| accuracy | | | 0.94 | 2600 |
| macro avg | 0.95 | 0.94 | 0.94 | 2600 |
| weighted avg | 0.95 | 0.94 | 0.94 | 2600 |

SASIDEVI V
212222230136

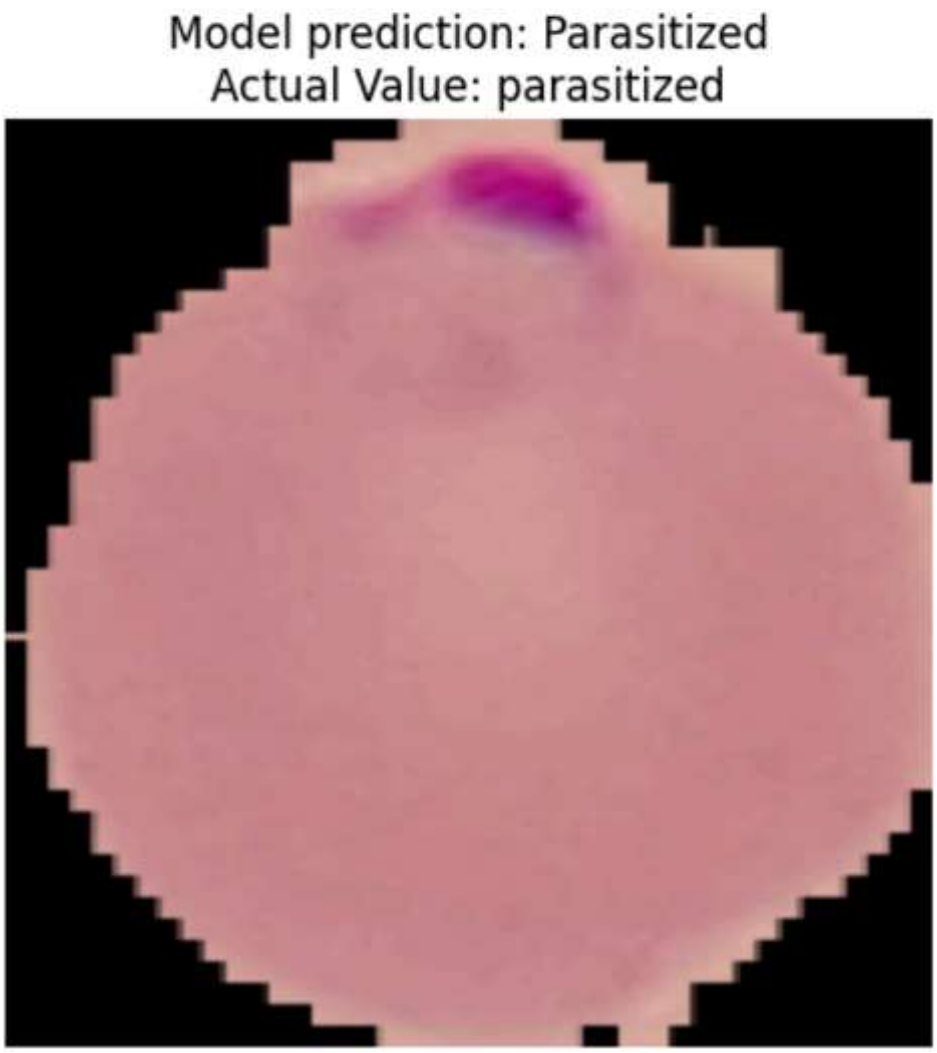
Confusion Matrix

```
[[1189  111]
 [  35 1265]]
```

SASIDEVI V
212222230136

New Sample Data Prediction

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
SASIDEVI V - 212222230136
```




## Result:

Thus a deep neural network for Malaria infected cell recognition and to analyze the performance is created using tensorflow.