```
from google.colab import drive
drive.mount('/content/drive')
```

```
import zipfile
from google.colab import drive
z = zipfile.ZipFile("/content/drive/MyDrive/Malaria/archive.zip",'r')
z.extractall("/content/drive/MyDrive/Malaria/malaria_data")
z.close()
```

```
pip install split_folders
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: split_folders in /usr/local/lib/python3.7/dist-packages (0.5.1)
```

```
import tensorflow.compat.v2 as tf
from keras import backend as k
from keras.applications.mobilenet import MobileNet
from keras.applications import imagenet_utils
import keras.applications
from keras.engine import training
from keras.layers import VersionAwareLayers
from keras.utils import data_utils
from keras.utils import layer_utils
from tensorflow.keras.utils import Sequence
from tensorflow.python.util.tf_export import keras_export
from keras.layers import Dense,GlobalAveragePooling2D
from keras.models import Model
from keras.layers import Flatten
from keras.callbacks import CSVLogger
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import classification_report
import seaborn as sns
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_input
import numpy as np
import matplotlib.pyplot as plt
import splitfolders

from google.colab.patches import cv2_imshow
```

```
# from splitfolders.split import ratio
# input_path = "/content/drive/MyDrive/Malaria/malaria_data/cell_images"
# splitfolders.ratio(input_path,output = "/content/drive/MyDrive/Malaria/train_test_val",seed = 42,ratio =
```

```
Copying files: 22825 files [09:49, 38.70 files/s]
```

```
train_path = "/content/drive/MyDrive/Malaria/train_test_val/train"
val_path = "/content/drive/MyDrive/Malaria/train_test_val/val"
test_path = "/content/drive/MyDrive/Malaria/train_test_val/test"
pre_process_input = tf.keras.applications.resnet.preprocess_input
generator = ImageDataGenerator(preprocessing_function=pre_process_input)
size = 128
train_data = generator.flow_from_directory(train_path,
 target_size=(size,size),batch_size= 32,
 class_mode='categorical' )

val_data = generator.flow_from_directory(val_path,
 target_size=(size,size),batch_size= 32,
 class_mode='categorical' )

```

```
15
16 test_data = generator.flow_from_directory(test_path,
17  target_size=(size,size),batch_size= 32,
18  class_mode='categorical',shuffle=False )
19
```

```
Found 4564 images belonging to 3 classes.
Found 4565 images belonging to 3 classes.
Found 13695 images belonging to 3 classes.
```

```
 1
 2 image_size = [size,size]
 3 model = VGG19(input_shape= image_size + [3],weights='imagenet',include_top=False)
 4
 5 for layer in model.layers:
 6   layer.trainable = False
 7 x = Flatten()(model.output)
 8 x = Dense(3,activation='softmax')(x)
 9 classifier = Model(inputs = model.input,outputs = x)
10 classifier.compile(loss='categorical_crossentropy',optimizer = 'adam',metrics=['accuracy'])
11
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_ordering_tf_kernels_notop.
80142336/80134624 [==============================] - 0s 0us/step
80150528/80134624 [==============================] - 0s 0us/step
```

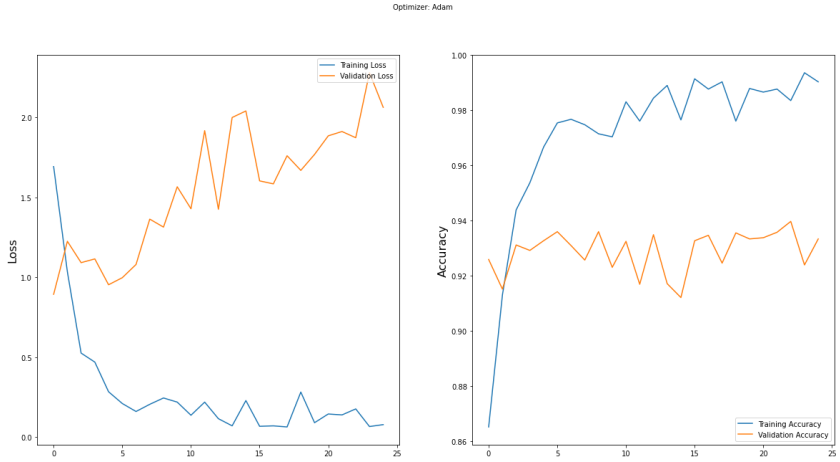
```
 1 his = classifier.fit(train_data,validation_data=val_data,epochs=25)
 2
 3 classifier.save("/content/drive/MyDrive/Malaria/VGG19_malaria.h5")
 4
 5
```

```
Epoch 1/25
143/143 [==============================] - 30s 192ms/step - loss: 1.6917 - accuracy: 0.8652 - val_loss: 0.8932 - val_accuracy: 0.9260
Epoch 2/25
143/143 [==============================] - 24s 169ms/step - loss: 1.0341 - accuracy: 0.9130 - val_loss: 1.2246 - val_accuracy: 0.9152
Epoch 3/25
143/143 [==============================] - 26s 181ms/step - loss: 0.5257 - accuracy: 0.9439 - val_loss: 1.0910 - val_accuracy: 0.9312
Epoch 4/25
143/143 [==============================] - 24s 168ms/step - loss: 0.4693 - accuracy: 0.9538 - val_loss: 1.1144 - val_accuracy: 0.9292
Epoch 5/25
143/143 [==============================] - 24s 170ms/step - loss: 0.2839 - accuracy: 0.9667 - val_loss: 0.9528 - val_accuracy: 0.9327
Epoch 6/25
143/143 [==============================] - 24s 167ms/step - loss: 0.2104 - accuracy: 0.9755 - val_loss: 0.9970 - val_accuracy: 0.9360
Epoch 7/25
143/143 [==============================] - 24s 168ms/step - loss: 0.1605 - accuracy: 0.9768 - val_loss: 1.0795 - val_accuracy: 0.9310
Epoch 8/25
143/143 [==============================] - 24s 167ms/step - loss: 0.2055 - accuracy: 0.9748 - val_loss: 1.3631 - val_accuracy: 0.9257
Epoch 9/25
143/143 [==============================] - 24s 167ms/step - loss: 0.2450 - accuracy: 0.9715 - val_loss: 1.3132 - val_accuracy: 0.9360
Epoch 10/25
143/143 [==============================] - 24s 168ms/step - loss: 0.2192 - accuracy: 0.9704 - val_loss: 1.5652 - val_accuracy: 0.9231
Epoch 11/25
143/143 [==============================] - 24s 167ms/step - loss: 0.1371 - accuracy: 0.9831 - val_loss: 1.4280 - val_accuracy: 0.9325
Epoch 12/25
143/143 [==============================] - 24s 168ms/step - loss: 0.2192 - accuracy: 0.9761 - val_loss: 1.9161 - val_accuracy: 0.9170
Epoch 13/25
143/143 [==============================] - 24s 167ms/step - loss: 0.1155 - accuracy: 0.9844 - val_loss: 1.4246 - val_accuracy: 0.9349
Epoch 14/25
143/143 [==============================] - 24s 168ms/step - loss: 0.0709 - accuracy: 0.9890 - val_loss: 1.9984 - val_accuracy: 0.9172
Epoch 15/25
143/143 [==============================] - 24s 168ms/step - loss: 0.2289 - accuracy: 0.9766 - val_loss: 2.0393 - val_accuracy: 0.9122
Epoch 16/25
143/143 [==============================] - 24s 167ms/step - loss: 0.0680 - accuracy: 0.9915 - val_loss: 1.6022 - val_accuracy: 0.9327
Epoch 17/25
143/143 [==============================] - 24s 167ms/step - loss: 0.0708 - accuracy: 0.9877 - val_loss: 1.5835 - val_accuracy: 0.9347
Epoch 18/25
143/143 [==============================] - 24s 168ms/step - loss: 0.0640 - accuracy: 0.9904 - val_loss: 1.7594 - val_accuracy: 0.9246
Epoch 19/25
143/143 [==============================] - 24s 169ms/step - loss: 0.2820 - accuracy: 0.9761 - val_loss: 1.6679 - val_accuracy: 0.9356
Epoch 20/25
143/143 [==============================] - 24s 168ms/step - loss: 0.0905 - accuracy: 0.9879 - val_loss: 1.7676 - val_accuracy: 0.9334
Epoch 21/25
143/143 [==============================] - 24s 168ms/step - loss: 0.1451 - accuracy: 0.9866 - val_loss: 1.8838 - val_accuracy: 0.9338
Epoch 22/25
143/143 [==============================] - 24s 169ms/step - loss: 0.1392 - accuracy: 0.9877 - val_loss: 1.9107 - val_accuracy: 0.9358
```

```
Epoch 23/25
143/143 [==============================] - 24s 168ms/step - loss: 0.1768 - accuracy: 0.9836 - val_loss: 1.8719 - val_accuracy: 0.9398
Epoch 24/25
143/143 [==============================] - 24s 167ms/step - loss: 0.0665 - accuracy: 0.9936 - val_loss: 2.2781 - val_accuracy: 0.9240
Epoch 25/25
143/143 [==============================] - 24s 169ms/step - loss: 0.0784 - accuracy: 0.9904 - val_loss: 2.0621 - val_accuracy: 0.9334
```

```
plt.figure(figsize = (20,10))
plt.subplot(1,2,1)
plt.suptitle("Optimizer: Adam", fontsize = 10)
plt.ylabel("Loss",fontsize = 16)

plt.plot(his.history['loss'],label = "Training Loss")
plt.plot(his.history['val_loss'],label = "Validation Loss")
plt.legend(loc = 'upper right')

plt.subplot(1,2,2)

plt.ylabel("Accuracy",fontsize = 16)

plt.plot(his.history['accuracy'],label = "Training Accuracy")
plt.plot(his.history['val_accuracy'],label = "Validation Accuracy")
plt.legend(loc = 'lower right')


```

```
<matplotlib.legend.Legend at 0x7f7190912350>
```




```
from sklearn.metrics import accuracy_score

y_true = test_data.classes
pred = classifier.predict(test_data)
pred = tf.argmax(pred,axis = 1)
print(accuracy_score(pred,y_true))

```

```
0.9323840817816721
```

```
from sklearn.metrics import confusion_matrix
import pandas as pd
cm = confusion_matrix(y_true, pred)

print(cm)
acc = str(accuracy_score(y_true, pred)*100)
acc = acc[0:5]
print(acc)
cm_df = pd.DataFrame(cm,index =
['Not Infected','Infected'], columns =
['Not Infected','Infected'])
plt.figure(figsize=(5,4))
sns.heatmap(cm_df, annot=True,cmap=plt.cm.BuPu)
plt.title('Confusion Matrix with Accuracy of: {0}%'.format(acc))
plt.ylabel('Actal Values')
plt.xlabel('Predicted Values')
plt.show()

```

```
[[7787  481]
 [ 445 4982]]
93.23
```