In [1]:

```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import itertools
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Dropout, Flatten, Inpu
```

```
C:\Users\asus\Anaconda3\lib\site-packages\h5py\__init__.py:36: FutureWarning: C
onversion of the second argument of issubdtype from `float` to `np.floating` is
deprecated. In future, it will be treated as `np.float64 == np.dtype(float).typ
e`.
  from ._conv import register_converters as _register_converters
```

In [67]:

```python
import tensorflow as tf
```

In [68]:

```python
TRAIN_DIR = 'Parasitized'
TEST_DIR = 'Uninfected'
```

In [69]:

```python
IMAGE_WIDTH=128
IMAGE_HEIGHT=128
IMAGE_SIZE=(IMAGE_WIDTH, IMAGE_HEIGHT)
IMAGE_CHANNELS=3 # RGB color
```

In [127]:

```python
import os

path ='\Parasitized\*.png'

files = []
# r=root, d=directories, f = files
for r, d, f in os.walk(r'C:\Users\asus\ML\ML_Basics\Deep_learning\Malaria_Med
    for file in f:
        if '.png' in file:
            files.append((file))

```

In [128]:

```python
import os

#path ='\Uninfected\*.png'

filesu = []
# r=root, d=directories, f = files
for r, d, f in os.walk(r'C:\Users\asus\ML\ML_Basics\Deep_learning\Malaria_Med
    for file in f:
        if '.png' in file:
            filesu.append((file))

```

In [129]:

```
TargetP =['Parasitized']*len(files)

TargetU =['Uninfected']*len(filesu)

files.extend(filesu)
TargetP.extend(TargetU)
```

In [130]:

```
Mal = {
    'Location':files,
    'Target':TargetP
}
df = pd.DataFrame(Mal)
```

In [126]:

```
import os

# Function to rename multiple files
def rename():
    i = 0

    for filename in os.listdir("Uninfected"):
            dst ="Uninfected."+str(i) +".png"
            src ='Uninfected/'+ filename
            dst ="Uninfected/"+ dst
            os.rename(src,dst)

            i += 1

rename()

import os

# Function to rename multiple files
def rename():
    i = 0

    for filename in os.listdir("Parasitized"):
            dst ="Parasitized."+str(i) +".png"
            src ='Parasitized/'+ filename
            dst ="Parasitized/"+ dst
            os.rename(src,dst)

            i += 1

rename()
```

Start

In [174]:

```python
filenames = os.listdir("BT")
categories = []
for filename in filenames:
    category = filename.split('.')[0]
    if category == 'Parasitized':
        categories.append('1')
    else:
        categories.append('0')

df = pd.DataFrame({
    'filename': filenames,
    'category': categories,

})
```

In [175]:

```python
df.head()
```

Out[175]:

	category	filename
0	1	Parasitized.0.png
1	1	Parasitized.1.png
2	1	Parasitized.10.png
3	1	Parasitized.100.png
4	1	Parasitized.1000.png

In [176]:

```python
from sklearn.utils import shuffle
df = shuffle(df)
```

In [177]:

```python
df.head()
```

Out[177]:

	category	filename
25866	0	Uninfected.8477.png
8183	1	Parasitized.4962.png
22772	0	Uninfected.5692.png
25251	0	Uninfected.7923.png
21933	0	Uninfected.4937.png

In [178]:

```python
df.shape
```

Out[178]:

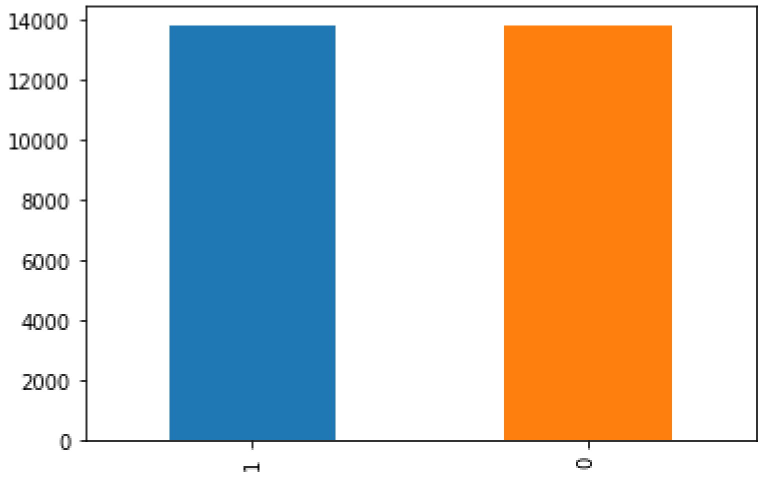
```
(27557, 2)
```

In [179]:

```
df['category'].value_counts().plot.bar()
```

Out[179]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17c4de695f8>
```



In [180]:

```
seed = 128
rng = np.random.RandomState(seed)
```

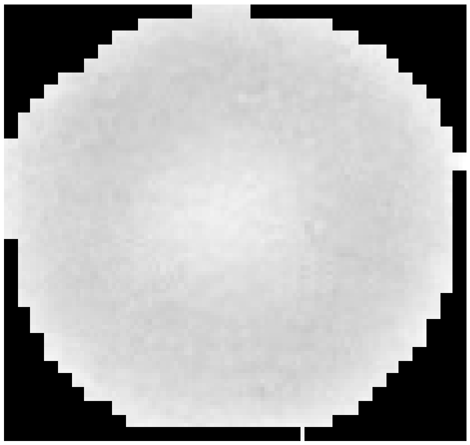
In [182]:

```
from scipy.misc import imread
import matplotlib.pyplot as pylab
img_name = rng.choice(df.loc[df['category'] == '0'].filename)
filepath = os.path.join('BT',  img_name)

img = imread(filepath, flatten=True)

pylab.imshow(img, cmap='gray')
pylab.axis('off')
pylab.show()
```

```
C:\Users\asus\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: DeprecationW
arning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```



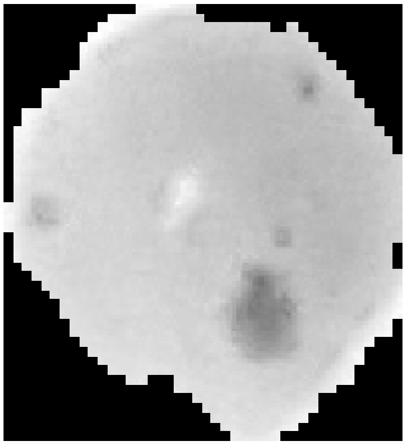
In [183]:

```python
from scipy.misc import imread
import matplotlib.pyplot as pylab
img_name = rng.choice(df.loc[df['category'] == '1'].filename)
filepath = os.path.join('BT',  img_name)

img = imread(filepath, flatten=True)

pylab.imshow(img, cmap='gray')
pylab.axis('off')
pylab.show()
```

```
C:\Users\asus\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: DeprecationW
arning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```



In [184]:

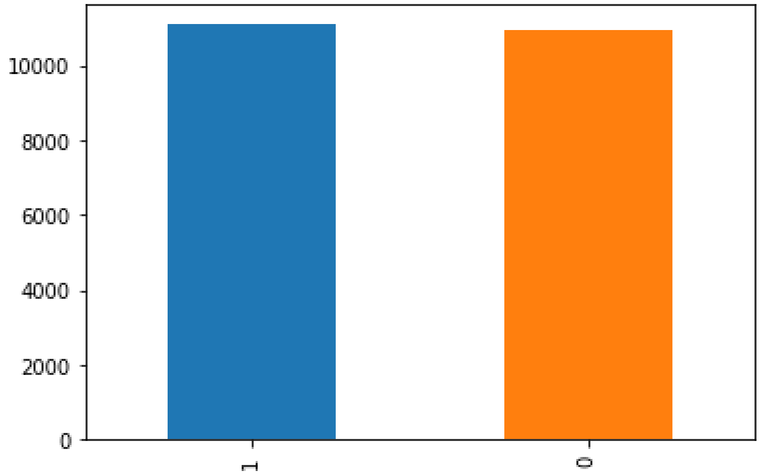
```python
from sklearn.model_selection import train_test_split
train_df, validate_df = train_test_split(df, test_size=0.20, random_state=42)
train_df = train_df.reset_index(drop=True)
validate_df = validate_df.reset_index(drop=True)
```

In [185]:

```
train_df['category'].value_counts().plot.bar()
```

Out[185]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17c4e144cc0>
```

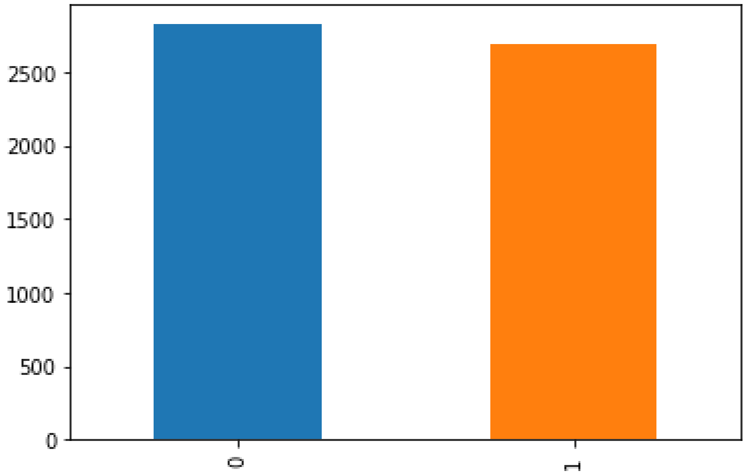


In [186]:

```
validate_df['category'].value_counts().plot.bar()
```

Out[186]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x17c4e195eb8>
```



In [187]:

```
total_train = train_df.shape[0]
total_validate = validate_df.shape[0]
batch_size=15
```

In [ ]:

```

```

In [188]:

```
from keras.preprocessing.image import ImageDataGenerator, load_img
train_datagen = ImageDataGenerator(
    rotation_range=15,
    rescale=1./255,
    shear_range=0.1,
    zoom_range=0.2,
    horizontal_flip=True,
    width_shift_range=0.1,
    height_shift_range=0.1
)

```

In [189]:

```
train_generator = train_datagen.flow_from_dataframe(
    train_df,
    "../BT/",
    x_col='filename',
    y_col='category',
    target_size=IMAGE_SIZE,
    class_mode='binary',
    batch_size=batch_size
)
```

```
Found 0 images belonging to 0 classes.
```

In [194]:

```
train_df.head()
```

Out[194]:

	category	filename
0	1	Parasitized.6465.png
1	0	Uninfected.9547.png
2	1	Parasitized.6972.png
3	0	Uninfected.8921.png
4	1	Parasitized.12634.png

In [202]:

```python

temp = []
size=(128,128)
for img_name in train_df.filename:
    image_path = os.path.join('BT', img_name)
    img = imread(image_path, flatten=True)
    img.resize(size)
    img = img.astype('float32')
    temp.append(img)

train_x = np.stack(temp)


```

```
C:\Users\asus\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: DeprecationW
arning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
```

In [203]:

```python
temp = []
for img_name in validate_df.filename:
    image_path = os.path.join('BT', img_name)
    img = imread(image_path, flatten=True)
    img.resize(size)
    img = img.astype('float32')
    temp.append(img)

test_x = np.stack(temp)


```

```
C:\Users\asus\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: DeprecationW
arning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
  after removing the cwd from sys.path.
```

In [204]:

```python
train_x.shape
```

Out[204]:

```
(22045, 128, 128)
```

In [205]:

```python
test_x.shape
```

Out[205]:

```
(5512, 128, 128)
```

In [209]:

```python
train_y = train_df.category
test_y = validate_df.category
```

In [211]:

```python
# Reshaping the array to 4-dims so that it can work with the Keras API
train_x = train_x.reshape(train_x.shape[0], 128, 128, 1)
test_x = test_x.reshape(test_x.shape[0], 128, 128, 1)
input_shape = (128, 128, 1)
# Making sure that the values are float so that we can get decimal points aft
train_x =train_x.astype('float32')
test_x = test_x.astype('float32')
# Normalizing the RGB codes by dividing it to the max RGB value.
train_x /= 255
test_x /= 255
print('x_train shape:', train_x.shape)
print('Number of images in x_train', train_x.shape[0])
print('Number of images in x_test', test_x.shape[0])

```

```
x_train shape: (22045, 128, 128, 1)
Number of images in x_train 22045
Number of images in x_test 5512
```

MEDICAL DOMAIN THESIS

In [228]:

```python
# Importing the required Keras modules containing model and layers

from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
from tensorflow.keras.models import Sequential
# Creating a Sequential Model and adding the layers
model = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(train_x, train_y,batch_size = 200,  epochs=20,shuffle=True)


```

```
Epoch 1/20
22045/22045 [==============================] - 382s 17ms/sample - loss: 1.9986
- accuracy: 0.5209
Epoch 2/20
22045/22045 [==============================] - 479s 22ms/sample - loss: 0.6511
- accuracy: 0.6154
Epoch 3/20
22045/22045 [==============================] - 379s 17ms/sample - loss: 0.5715
- accuracy: 0.7037
Epoch 4/20
22045/22045 [==============================] - 310s 14ms/sample - loss: 0.4800
- accuracy: 0.7770
Epoch 5/20
22045/22045 [==============================] - 307s 14ms/sample - loss: 0.3909
- accuracy: 0.8357
Epoch 6/20
22045/22045 [==============================] - 305s 14ms/sample - loss: 0.3138
- accuracy: 0.8750
Epoch 7/20
22045/22045 [==============================] - 301s 14ms/sample - loss: 0.2438
- accuracy: 0.9122
Epoch 8/20
22045/22045 [==============================] - 300s 14ms/sample - loss: 0.1863
- accuracy: 0.9399
Epoch 9/20
22045/22045 [==============================] - 305s 14ms/sample - loss: 0.1422
- accuracy: 0.9580
Epoch 10/20
22045/22045 [==============================] - 304s 14ms/sample - loss: 0.1052
- accuracy: 0.9737
Epoch 11/20
22045/22045 [==============================] - 304s 14ms/sample - loss: 0.0768
- accuracy: 0.9835
Epoch 12/20
22045/22045 [==============================] - 303s 14ms/sample - loss: 0.0589
```

```
- accuracy: 0.9888
Epoch 13/20
22045/22045 [==============================] - 304s 14ms/sample - loss: 0.0429
- accuracy: 0.9939
Epoch 14/20
22045/22045 [==============================] - 303s 14ms/sample - loss: 0.0338
- accuracy: 0.9949
Epoch 15/20
22045/22045 [==============================] - 310s 14ms/sample - loss: 0.0280
- accuracy: 0.9966
Epoch 16/20
22045/22045 [==============================] - 311s 14ms/sample - loss: 0.0231
- accuracy: 0.9970
Epoch 17/20
22045/22045 [==============================] - 309s 14ms/sample - loss: 0.0246
- accuracy: 0.9966
Epoch 18/20
22045/22045 [==============================] - 309s 14ms/sample - loss: 0.0205
- accuracy: 0.9971
Epoch 19/20
22045/22045 [==============================] - 303s 14ms/sample - loss: 0.0180
- accuracy: 0.9975
Epoch 20/20
22045/22045 [==============================] - 302s 14ms/sample - loss: 0.0149
- accuracy: 0.9981
```

```
Out[228]: <tensorflow.python.keras.callbacks.History at 0x17cda139438>
```

```
In [226]: model.evaluate(test_x, test_y)
```

```
5512/5512 [==============================] - 2s 348us/sample - loss: 0.7192 - a
ccuracy: 0.5666
```

```
Out[226]: [0.7191535421310558, 0.566582]
```

```
In [ ]:
```

```
In [ ]:
```

```
In [ ]:
```

Early Stop

To prevent over fitting we will stop the learning after 10 epochs and val_loss value not decreased

Learning Rate Reduction

We will reduce the learning rate when then accuracy not increase for 2 steps

In [206]:

```
from keras.callbacks import EarlyStopping, ReduceLROnPlateau
earlystop = EarlyStopping(patience=10)
learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
                                            patience=2,
                                            verbose=1,
                                            factor=0.5,
                                            min_lr=0.00001)
callbacks = [earlystop, learning_rate_reduction]
```

In [207]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, D

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(IMAGE_WIDTH, IMA
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy', optimizer='rmsprop', metrics=['accu

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 126, 126, 32)      896       
_________________________________________________________________
batch_normalization_v2_4 (Ba (None, 126, 126, 32)      128       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 63, 63, 32)        0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 63, 63, 32)        0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 61, 61, 64)        18496     
_________________________________________________________________
batch_normalization_v2_5 (Ba (None, 61, 61, 64)        256       
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 30, 30, 64)        0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 30, 30, 64)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 28, 28, 128)       73856     
_________________________________________________________________
batch_normalization_v2_6 (Ba (None, 28, 28, 128)       512       
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 14, 14, 128)       0         
_________________________________________________________________
```

```
dropout_6 (Dropout)          (None, 14, 14, 128)       0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_2 (Dense)              (None, 512)               12845568  
_________________________________________________________________
batch_normalization_v2_7 (Ba (None, 512)               2048      
_________________________________________________________________
dropout_7 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 513       
=================================================================
Total params: 12,942,273
Trainable params: 12,940,801
Non-trainable params: 1,472
_________________________________________________________________
```

In [ ]:

```
epochs=3 if FAST_RUN else 50
history = model.fit_generator(
    train_x,
    epochs=epochs,
    validation_data=test_x,
    validation_steps=total_validate//batch_size,
    steps_per_epoch=total_train//batch_size,
    callbacks=callbacks
)
```