In [3]:
```
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
# to share the GPU resources for multiple sessions
from tensorflow.compat.v1.keras.backend import set_session
config = tf.compat.v1.ConfigProto()
config.gpu_options.allow_growth = True # dynamically grow the memory used on the GP
config.log_device_placement = True # to log device placement (on which device the o
sess = tf.compat.v1.Session(config=config)
set_session(sess)

%matplotlib inline
```

```
Device mapping:
/job:localhost/replica:0/task:0/device:GPU:0 -> device: 0, name: GRID A100D-16C, p
ci bus id: 0000:02:00.0, compute capability: 8.0
```

```
2022-10-03 09:33:31.161661: I tensorflow/stream_executor/cuda/cuda_gpu_executor.c
c:975] successful NUMA node read from SysFS had negative value (-1), but there mus
t be at least one NUMA node, so returning NUMA node zero
2022-10-03 09:33:31.161958: I tensorflow/stream_executor/cuda/cuda_gpu_executor.c
c:975] successful NUMA node read from SysFS had negative value (-1), but there mus
t be at least one NUMA node, so returning NUMA node zero
2022-10-03 09:33:31.162114: I tensorflow/stream_executor/cuda/cuda_gpu_executor.c
c:975] successful NUMA node read from SysFS had negative value (-1), but there mus
t be at least one NUMA node, so returning NUMA node zero
2022-10-03 09:33:31.162298: I tensorflow/stream_executor/cuda/cuda_gpu_executor.c
c:975] successful NUMA node read from SysFS had negative value (-1), but there mus
t be at least one NUMA node, so returning NUMA node zero
2022-10-03 09:33:31.162409: I tensorflow/stream_executor/cuda/cuda_gpu_executor.c
c:975] successful NUMA node read from SysFS had negative value (-1), but there mus
t be at least one NUMA node, so returning NUMA node zero
2022-10-03 09:33:31.162495: I tensorflow/core/common_runtime/gpu/gpu_device.cc:153
2] Created device /job:localhost/replica:0/task:0/device:GPU:0 with 12898 MB memor
y:  -> device: 0, name: GRID A100D-16C, pci bus id: 0000:02:00.0, compute capabili
ty: 8.0
```

Dataset Download Link

https://drive.google.com/file/d/1ULKLK_R0qsLho6PzfUIOwTXOStk3w5jd/view?usp=sharing

In [ ]:
```
# Upload this file to your google drive
```

In [ ]:
```
# for Google Colab
drive.mount('/content/drive')
```

In [ ]:
```
# for Google Colab
!tar --skip-old-files -xvf '/content/drive/MyDrive/Dataset/cell_images.tar.xz' -C
```

In [ ]:
```
# for Google Colab
my_data_dir = '/content/drive/MyDrive/Dataset/cell_images'
```

In [4]:
```
# for college server
my_data_dir = '/home/ailab/hdd/dataset/cell_images'
```

In [5]:
```
os.listdir(my_data_dir)
```

Out[5]:
```
['train', 'test', '.ipynb_checkpoints']
```

In [6]:
```
test_path = my_data_dir+'/test/'
train_path = my_data_dir+'/train/'
```

In [7]:
```
os.listdir(train_path)
```

Out[7]:
```
['parasitized', 'uninfected']
```

In [8]:
```
len(os.listdir(train_path+'/uninfected/'))
```

Out[8]:
```
12479
```

In [9]:
```
len(os.listdir(train_path+'/parasitized/'))
```

Out[9]:
```
12480
```

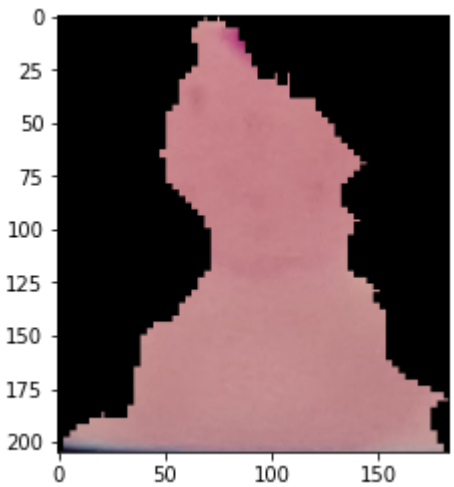
In [10]:
```
os.listdir(train_path+'/parasitized')[0]
```

Out[10]:
```
'C101P62ThinF_IMG_20150923_165215_cell_7.png'
```

In [11]:
```
para_img= imread(train_path+
                 '/parasitized/'+
                 os.listdir(train_path+'/parasitized')[0])
```

In [12]:
```
plt.imshow(para_img)
```

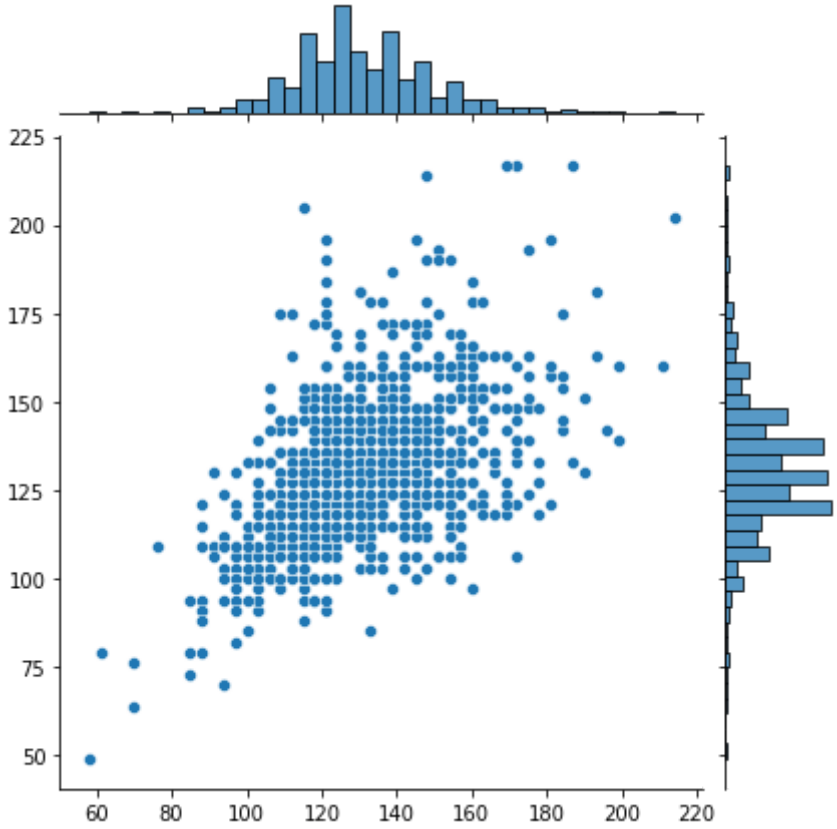
Out[12]:
```
<matplotlib.image.AxesImage at 0x7f777004cac0>
```



In [13]:
```
# Checking the image dimensions
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)
```

In [14]:
```
sns.jointplot(x=dim1,y=dim2)
```

Out[14]:
```
<seaborn.axisgrid.JointGrid at 0x7f7758792580>
```



In [15]:
```
image_shape = (130,130,3)
```

In [ ]:
```
help(ImageDataGenerator)
```

In [16]:
```
image_gen = ImageDataGenerator(rotation_range=20, # rotate the image 20 degrees
                               width_shift_range=0.10, # Shift the pic width by a r
                               height_shift_range=0.10, # Shift the pic height by 
                               rescale=1/255, # Rescale the image by normalzing it
                               shear_range=0.1, # Shear means cutting away part of
                               zoom_range=0.1, # Zoom in by 10% max
                               horizontal_flip=True, # Allo horizontal flipping
                               fill_mode='nearest' # Fill in missing pixels with t
                              )
```

In [17]:
```
image_gen.flow_from_directory(train_path)
```

Out[17]:
```
Found 24961 images belonging to 2 classes.
<keras.preprocessing.image.DirectoryIterator at 0x7f7770086b80>
```

In [18]:
```
image_gen.flow_from_directory(test_path)
```

Out[18]:
```
Found 2606 images belonging to 2 classes.
<keras.preprocessing.image.DirectoryIterator at 0x7f7770075070>
```

In [20]:
```
model = models.Sequential([
    layers.Input((130,130,3)),
    layers.Conv2D(32,kernel_size=3,activation="relu",padding="same"),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(32,kernel_size=3,activation="relu"),
    layers.MaxPool2D((2,2)),
    layers.Conv2D(32,kernel_size=3,activation="relu"),
```

```
    layers.MaxPool2D((2,2)),
    layers.Flatten(),
    layers.Dense(32,activation="relu"),
    layers.Dense(1,activation="sigmoid")])

model.compile(loss="binary_crossentropy", metrics='accuracy',optimizer="adam")
```

In [21]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 130, 130, 32)      896

 max_pooling2d_3 (MaxPooling  (None, 65, 65, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 63, 63, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 31, 31, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_5 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 6272)              0

 dense_2 (Dense)             (None, 32)                200736

 dense_3 (Dense)             (None, 1)                 33

=================================================================
Total params: 220,161
Trainable params: 220,161
Non-trainable params: 0
_________________________________________________________________
```

In [22]:
```
batch_size = 16
```

In [ ]:
```
help(image_gen.flow_from_directory)
```

In [23]:
```
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')
```

```
Found 24961 images belonging to 2 classes.
```

In [24]:
```
train_image_gen.batch_size
```

Out[24]:
```
16
```

In [25]:
```
len(train_image_gen.classes)
```

Out[25]:
```
24961
```

In [26]:
```
train_image_gen.total_batches_seen
```

Out[26]:
```
0
```

In [27]:
```
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)
```

```
Found 2606 images belonging to 2 classes.
```

In [28]:
```
train_image_gen.class_indices
```

Out[28]:
```
{'parasitized': 0, 'uninfected': 1}
```

In [29]:
```
results = model.fit(train_image_gen,epochs=4,validation_data=test_image_gen)
```

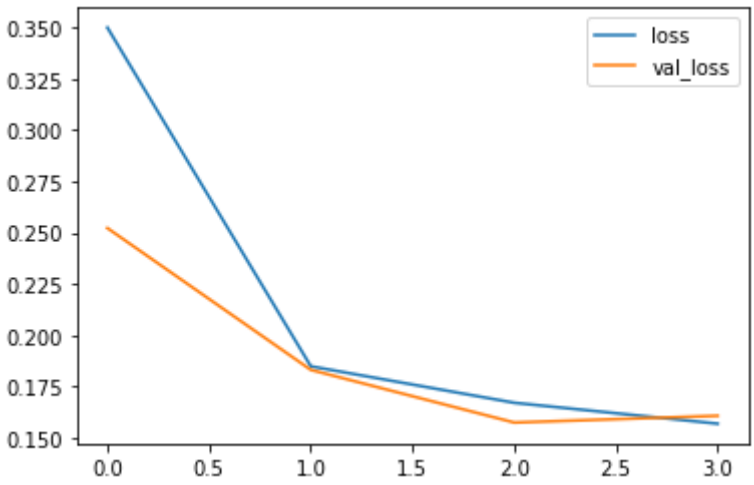
```
Epoch 1/4
2022-10-03 09:34:32.567376: I tensorflow/stream_executor/cuda/cuda_dnn.cc:384] Loaded cuDNN version 8201
2022-10-03 09:34:34.130315: I tensorflow/stream_executor/cuda/cuda_blas.cc:1786] TensorFloat-32 will be used for the matrix multiplication. This will only be logged once.
1561/1561 [==============================] - 102s 63ms/step - loss: 0.3500 - accuracy: 0.8463 - val_loss: 0.2522 - val_accuracy: 0.9144
Epoch 2/4
1561/1561 [==============================] - 101s 65ms/step - loss: 0.1847 - accuracy: 0.9405 - val_loss: 0.1830 - val_accuracy: 0.9428
Epoch 3/4
1561/1561 [==============================] - 100s 64ms/step - loss: 0.1669 - accuracy: 0.9477 - val_loss: 0.1573 - val_accuracy: 0.9459
Epoch 4/4
1561/1561 [==============================] - 99s 64ms/step - loss: 0.1568 - accuracy: 0.9494 - val_loss: 0.1606 - val_accuracy: 0.9486
```

In [30]:
```
model.save('cell_model.h5')
```

In [31]:
```
losses = pd.DataFrame(model.history.history)
```

In [32]:
```
losses[['loss','val_loss']].plot()
```

Out[32]:
```
<AxesSubplot:>
```



In [33]:
```
model.metrics_names
```

Out[33]:
```
['loss', 'accuracy']
```

In [34]:
```
model.evaluate(test_image_gen)
```

```
163/163 [==============================] - 9s 58ms/step - loss: 0.1567 - accuracy:
0.9486
```

Out[34]:
```
[0.1566585898399353, 0.9485802054405212]
```

In [35]:
```
pred_probabilities = model.predict(test_image_gen)
```

```
163/163 [==============================] - 9s 56ms/step
```

In [36]:
```
test_image_gen.classes
```

Out[36]:
```
array([0, 0, 0, ..., 1, 1, 1], dtype=int32)
```

In [37]:
```
predictions = pred_probabilities > 0.5
```

In [38]:
```
print(classification_report(test_image_gen.classes,predictions))
```

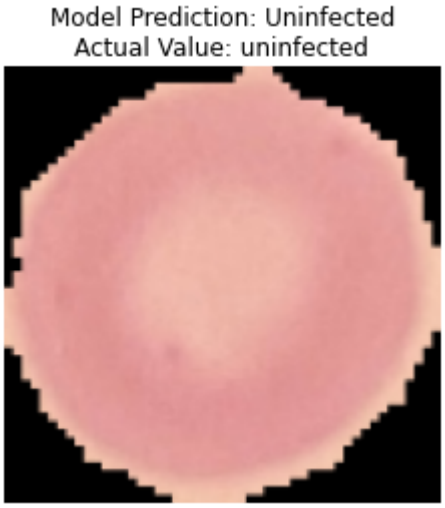
```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1306
           1       0.94      0.96      0.95      1300

    accuracy                           0.95      2606
   macro avg       0.95      0.95      0.95      2606
weighted avg       0.95      0.95      0.95      2606
```

In [39]:
```
confusion_matrix(test_image_gen.classes,predictions)
```

Out[39]:
```
array([[1225,   81],
       [  52, 1248]])
```

In [40]:
```
import random
list_dir=["uninfected","parasitized"]
dire=(random.choice(list_dir))
para_img= imread(train_path+
                 '/'+dire+'/'+
                 os.listdir(train_path+'/'+dire)[random.randint(0,10000)])
img = tf.convert_to_tensor(np.asarray(para_img))
img = tf.image.resize(img,(130,130))
img = img.numpy()
pred=bool(model.predict(img.reshape(1,130,130,3))<0.5)
plt.title("Model Prediction: "+("Parasitized" if pred else "Uninfected")+"\nActual
plt.axis("off")
plt.imshow(img)
plt.show()
```

```
1/1 [==============================] - 0s 56ms/step
```



In [ ]: