In [1]:
```
import torch
import numpy as np
import os
import cv2
```

In [2]:
```
from sklearn.model_selection import train_test_split
from keras.preprocessing.image import img_to_array
```

```
Using TensorFlow backend.
```

In [3]:
```
device = torch.device('cuda')
dtype = torch.float32
```

In [4]:
```
# positive - Malaria infected
# negative - No malaria (Uninfected)
positive = os.listdir('../input/cell_images/cell_images/Parasitized/')
negative = os.listdir('../input/cell_images/cell_images/Uninfected/')
```

In [5]:
```
print(positive[:10])
```

```
['C140P101ThinF_IMG_20151005_211735_cell_159.png', 'C175P136NThinF_IMG_20
151127_142326_cell_232.png', 'C97P58ThinF_IMG_20150917_152032_cell_162.pn
g', 'C68P29N_ThinF_IMG_20150819_133236_cell_183.png', 'C129P90ThinF_IMG_2
0151004_134306_cell_140.png', 'C126P87ThinF_IMG_20151004_105100_cell_129.
png', 'C68P29N_ThinF_IMG_20150819_133350_cell_183.png', 'C84P45ThinF_IMG_
20150818_101412_cell_98.png', 'C132P93ThinF_IMG_20151004_152353_cell_158.
png', 'C80P41ThinF_IMG_20150817_111544_cell_138.png']
```

In [6]:
```
print(negative[:10])
```

```
['C214ThinF_IMG_20151106_131748_cell_148.png', 'C188P149ThinF_IMG_2015120
3_134419_cell_40.png', 'C158P119ThinF_IMG_20151115_181558_cell_45.png',
'C100P61ThinF_IMG_20150918_145042_cell_59.png', 'C114P75ThinF_IMG_2015093
0_150733_cell_19.png', 'C201ThinF_IMG_20150930_143502_cell_186.png', 'C22
4ThinF_IMG_20151112_111955_cell_121.png', 'C96P57ThinF_IMG_20150824_11244
2_cell_163.png', 'C162P123ThinF_IMG_20151116_104114_cell_27.png', 'C157P1
18ThinF_IMG_20151115_163611_cell_9.png']
```

In [7]:
```
x = []
y = []
for pos in positive:
    if pos.endswith('png'):
        img = cv2.resize(cv2.imread('../input/cell_images/cell_images/Paras
        x.append(img_to_array(img, data_format='channels_first'))
        y.append(1)
for neg in negative:
    if neg.endswith('png'):
        img = cv2.resize(cv2.imread('../input/cell_images/cell_images/Uninf
        x.append(img_to_array(img, data_format='channels_first'))
        y.append(0)

x = np.array(x)
y = np.array(y)
```

In [28]:
```
print(x.shape)
print(y.shape)
```

```
(27558, 3, 64, 64)
(27558,)
```

In [9]:
```
%reload_ext autoreload
%autoreload 2
%matplotlib inline

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from fastai import *
from fastai.vision import *
from fastai.callbacks.hooks import *

img_dir='../input/cell_images/cell_images/'
path=Path(img_dir)
path
```

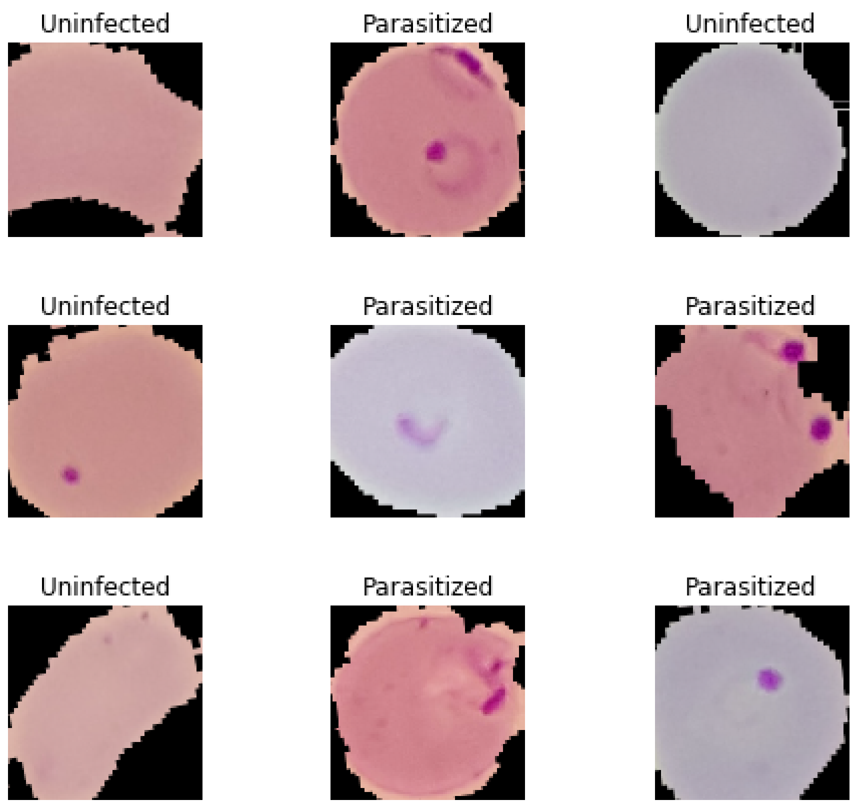
Out[9]:
```
PosixPath('../input/cell_images/cell_images')
```

In [10]:
```
data = ImageDataBunch.from_folder(path, train=".",
                                  valid_pct=0.2,
                                  ds_tfms=get_transforms(flip_vert=True, ma
                                  size=224,bs=64,
                                  num_workers=0).normalize(imagenet_stats)
```

In [11]:
```
print(f'Classes: \n {data.classes}')
```

```
Classes: 
 ['Parasitized', 'Uninfected']
```

In [12]:
```
data.show_batch(rows=3, figsize=(7,6))
```



In [13]:
```
x_train,x_validation,y_train,y_validation = train_test_split(x, y, train_si
```

```
/opt/conda/lib/python3.6/site-packages/sklearn/model_selection/_split.py:
2179: FutureWarning: From version 0.21, test_size will always complement
train_size unless both are specified.
  FutureWarning)
```

In [14]:
```
print(x_train.shape)
print(x_validation.shape)
print(y_train.shape)
print(y_validation.shape)
```

```
(27000, 3, 64, 64)
(558, 3, 64, 64)
(27000,)
(558,)
```

In [15]:
```
x_train = [i.flatten() for i in x_train]
```

In [16]:
```
print(np.array(x_train).shape)
```

```
(27000, 12288)
```

Training Model with priors - (Accuracy 63.60%)

In [18]:
```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.priors = [0.1,0.9]
model.fit(x_train,y_train)
```

Out[18]:
```
GaussianNB(priors=[0.1, 0.9], var_smoothing=1e-09)
```

In [19]:
```
print("Model accuracy: {:.2f}%".format(model.score(x_train, y_train)*100))
```

```
Model accuracy: 63.60%
```

Training Model without priors (Accuracy 63.57%)

In [26]:
```
model_without_prior = GaussianNB()
model_without_prior.fit(x_train,y_train)
```

Out[26]:
```
GaussianNB(priors=None, var_smoothing=1e-09)
```

In [27]:
```
print("Model accuracy: {:.2f}%".format(model_without_prior.score(x_train, y
```

```
Model accuracy: 63.57%
```

In [20]:
```
classes = {1: 'malaria infected cell',
           0: 'not a malaria cell'}

for i in range(10):
    ynew = model.predict(x_validation[i].reshape(1,-1))
    print(ynew)
```

```
[1]
[1]
[1]
[1]
[1]
[0]
[1]
[1]
[0]
[1]
```

In [21]:
```
predictingReshapedArray = []

for i in range(len(x_validation)):
    predictingReshapedArray.append(x_validation[i].T)
predictingReshapedArrayFinal = np.asarray(predictingReshapedArray)

print(predictingReshapedArrayFinal[0].shape)
```

```
(64, 64, 3)
```

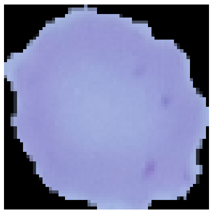
PREDICTING 2 RANDOM TEST DATA

In [22]:
```
import random
```

In [23]:
```
classes = {'1': 'malaria infected cell',
           '0': 'not a malaria cell'}
```

In [24]:
```
index = 5
predictionvalue = model.predict(x_validation[5].reshape(1,-1))
print("I think this is: " + classes[str(predictionvalue[0])])
from PIL import Image
svimg=Image.fromarray(predictingReshapedArrayFinal[5].astype('uint8'))
svimg.save(("cellImage"+str(index)+".png"))
```

```
I think this is: not a malaria cell
```



In [25]:
```
index = 8
predictionvalue = model.predict(x_validation[10].reshape(1,-1))
print("I think this is: " + classes[str(predictionvalue[0])])
from PIL import Image
svimg=Image.fromarray(predictingReshapedArrayFinal[10].astype('uint8'))
svimg.save(("cellImage"+str(index)+".png"))
```

```
I think this is: malaria infected cell
```