Importing Libraries & Setting up Directories

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib.image import imread
import seaborn as sns
import os
```

In [2]:
```
my_dir = 'C:\\Users\\breje\\OneDrive\\Desktop\\ML Dataset\\Deep Learning - Jo
```

In [3]:
```
os.listdir(my_dir)
```

Out[3]:
```
['test', 'train']
```

In [4]:
```
train_dir = my_dir+'\\train'
os.listdir(train_dir)
```

Out[4]:
```
['parasitized', 'uninfected']
```

In [5]:
```
test_dir = my_dir+'\\test'
os.listdir(test_dir)
```

Out[5]:
```
['parasitized', 'uninfected']
```

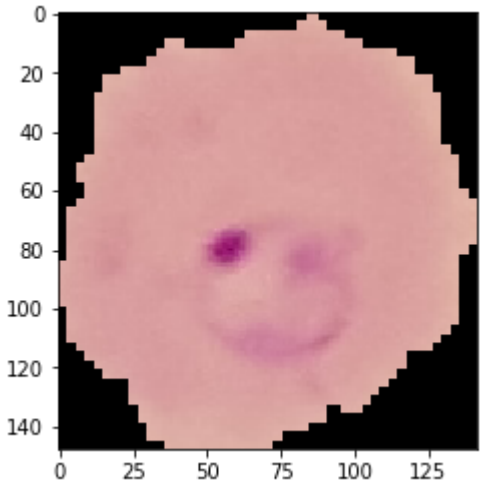
Exploring the Dataset

In [6]:
```
os.listdir(train_dir+'\\parasitized')
```

Out[6]:
```
['C100P61ThinF_IMG_20150918_144104_cell_162.png',
 'C100P61ThinF_IMG_20150918_144104_cell_163.png',
 'C100P61ThinF_IMG_20150918_144104_cell_164.png',
 'C100P61ThinF_IMG_20150918_144104_cell_165.png',
 'C100P61ThinF_IMG_20150918_144104_cell_166.png',
 'C100P61ThinF_IMG_20150918_144104_cell_167.png',
 'C100P61ThinF_IMG_20150918_144104_cell_168.png',
 'C100P61ThinF_IMG_20150918_144104_cell_169.png',
 'C100P61ThinF_IMG_20150918_144104_cell_170.png',
 'C100P61ThinF_IMG_20150918_144104_cell_171.png',
 'C100P61ThinF_IMG_20150918_144348_cell_138.png',
 'C100P61ThinF_IMG_20150918_144348_cell_139.png',
 'C100P61ThinF_IMG_20150918_144348_cell_140.png',
 'C100P61ThinF_IMG_20150918_144348_cell_141.png',
 'C100P61ThinF_IMG_20150918_144348_cell_142.png',
 'C100P61ThinF_IMG_20150918_144348_cell_143.png',
 'C100P61ThinF_IMG_20150918_144823_cell_157.png',
 'C100P61ThinF_IMG_20150918_144823_cell_158.png',
 'C100P61ThinF_IMG_20150918_144823_cell_159.png',
```

In [7]:
```
plt.imshow(imread(train_dir+'\\parasitized\\C100P61ThinF_IMG_20150918_144104_
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x1e3075813c8>
```

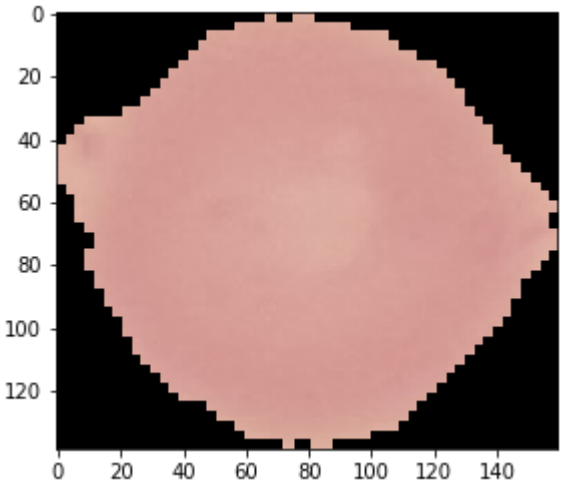


In [8]:
```
os.listdir(train_dir+'\\uninfected')
```

```
'C100P61ThinF_IMG_20150918_145042_cell_7.png',
'C100P61ThinF_IMG_20150918_145042_cell_81.png',
'C100P61ThinF_IMG_20150918_145042_cell_94.png',
'C100P61ThinF_IMG_20150918_145422_cell_12.png',
'C100P61ThinF_IMG_20150918_145422_cell_136.png',
'C100P61ThinF_IMG_20150918_145422_cell_157.png',
'C100P61ThinF_IMG_20150918_145422_cell_21.png',
'C100P61ThinF_IMG_20150918_145422_cell_3.png',
'C100P61ThinF_IMG_20150918_145422_cell_37.png',
'C100P61ThinF_IMG_20150918_145422_cell_85.png',
'C100P61ThinF_IMG_20150918_145422_cell_86.png',
'C100P61ThinF_IMG_20150918_145609_cell_101.png',
'C100P61ThinF_IMG_20150918_145609_cell_113.png',
'C100P61ThinF_IMG_20150918_145609_cell_121.png',
'C100P61ThinF_IMG_20150918_145609_cell_30.png',
'C100P61ThinF_IMG_20150918_145609_cell_46.png',
'C100P61ThinF_IMG_20150918_145609_cell_48.png',
'C100P61ThinF_IMG_20150918_145609_cell_75.png',
'C100P61ThinF_IMG_20150918_145609_cell_91.png',
'C100P61ThinF_IMG_20150918_145609_cell_99.png',
```

In [9]:
```
plt.imshow(imread(train_dir+'\\uninfected\\C100P61ThinF_IMG_20150918_144104_c
```

Out[9]: <matplotlib.image.AxesImage at 0x1e30871a508>



In [10]:
```
dim1=[]
dim2=[]
col=[]
for i in os.listdir(train_dir+'\\parasitized'):
    img = imread(train_dir+'\\parasitized\\'+i)
    d1, d2, c1 = img.shape
    dim1.append(d1)
    dim2.append(d2)
    col.append(c1)
```

In [11]:
```
print(np.mean(dim1), np.mean(dim2), np.mean(col))
```

```
134.360205144643 133.66447632021797 3.0
```

Building the CNN Model

In [12]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
from keras.callbacks import EarlyStopping
```

```
Using TensorFlow backend.
```

In [13]:
```
model = Sequential()
```

In [14]:
```
model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same', activation='r
model.add(MaxPool2D())
model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same', activation='r
model.add(MaxPool2D())
model.add(Conv2D(filters=64, kernel_size=(3,3), padding='same', activation='r
model.add(MaxPool2D())
model.add(Flatten())

model.add(Dense(128, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accurac
```

In [15]:
```
from keras.preprocessing.image import ImageDataGenerator
```

In [16]:
```
datagen = ImageDataGenerator(
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        rotation_range=20, width_shift_range=0.10, height_shift_range=0.10, f
```

In [17]:
```
train_generator = datagen.flow_from_directory(
        train_dir,
        target_size=(130, 130),
        batch_size=32,
        class_mode='binary')

validation_generator = datagen.flow_from_directory(
        test_dir,
        target_size=(130, 130),
        batch_size=32,
        class_mode='binary', shuffle=False)
```

```
Found 24958 images belonging to 2 classes.
Found 2600 images belonging to 2 classes.
```

In [18]:
```
early_stop = EarlyStopping(patience=2)
```

Training the Model

In [19]:
```
model.fit_generator(
        train_generator,
        epochs=10,
        validation_data=validation_generator,callbacks=[early_stop])
```

```
Epoch 1/10
780/780 [==============================] - 1221s 2s/step - loss: 0.3115 - a
ccuracy: 0.8722 - val_loss: 0.1194 - val_accuracy: 0.9408
Epoch 2/10
780/780 [==============================] - 1197s 2s/step - loss: 0.1743 - a
ccuracy: 0.9472 - val_loss: 0.0331 - val_accuracy: 0.9473
Epoch 3/10
780/780 [==============================] - 1193s 2s/step - loss: 0.1652 - a
ccuracy: 0.9492 - val_loss: 0.0384 - val_accuracy: 0.9508
Epoch 4/10
780/780 [==============================] - 1207s 2s/step - loss: 0.1562 - a
ccuracy: 0.9505 - val_loss: 0.0298 - val_accuracy: 0.9500
Epoch 5/10
780/780 [==============================] - 1203s 2s/step - loss: 0.1539 - a
ccuracy: 0.9513 - val_loss: 0.0563 - val_accuracy: 0.9462
Epoch 6/10
780/780 [==============================] - 1198s 2s/step - loss: 0.1510 - a
ccuracy: 0.9512 - val_loss: 0.1015 - val_accuracy: 0.9465
```

Out[19]:
```
<keras.callbacks.callbacks.History at 0x1e314663548>
```

Training History

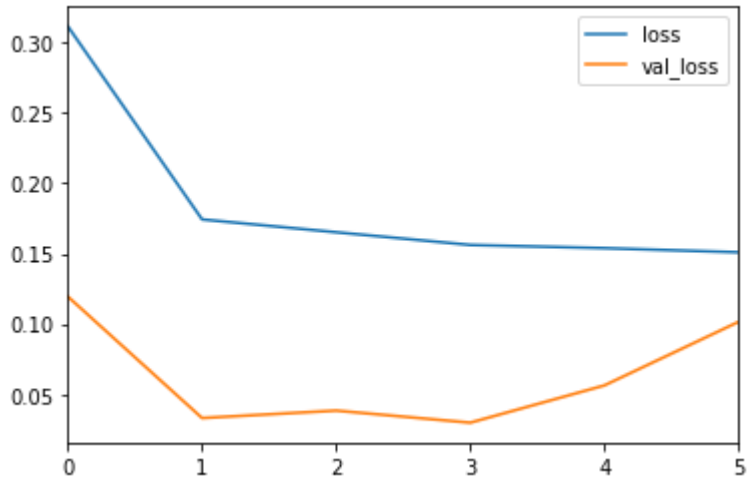
In [20]:
```
history = pd.DataFrame(model.history.history)
```

In [21]:
```
history.columns
```

Out[21]:
```
Index(['val_loss', 'val_accuracy', 'loss', 'accuracy'], dtype='object')
```

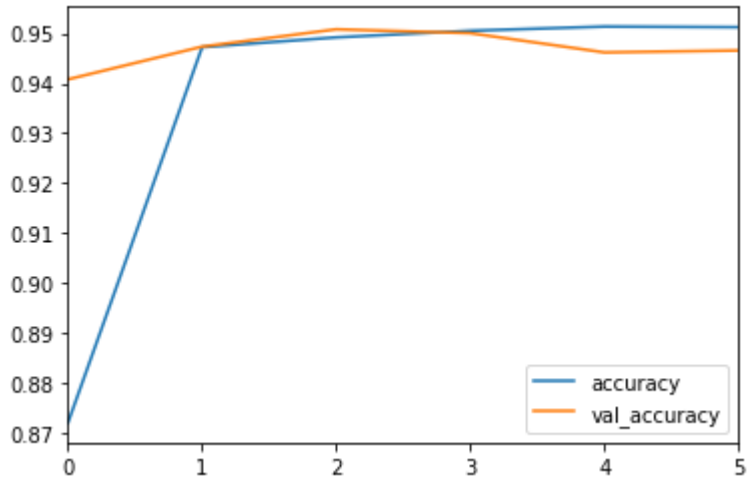
In [22]:
```
history[['loss', 'val_loss']].plot()
```

Out[22]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e3146d6688>
```



In [23]:
```
history[['accuracy', 'val_accuracy']].plot()
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1e3147ae408>
```



Model Evaluation

In [24]:
```
model.evaluate_generator(validation_generator)
```

Out[24]:
```
[0.07550478726625443, 0.9507692456245422]
```

In [25]:
```
y_pred = model.predict_generator(validation_generator)
```

In [26]:
```
from sklearn.metrics import classification_report, confusion_matrix
```

In [63]:
```
y_pred = y_pred >0.5
```

In [64]:
```
y_pred.shape
```

Out[64]: (2600, 1)

In [65]:
```
y_test = validation_generator.classes
```

In [66]:
```
y_test.shape
```

Out[66]: (2600,)

In [67]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      0.94      0.95      1300
           1       0.94      0.96      0.95      1300

    accuracy                           0.95      2600
   macro avg       0.95      0.95      0.95      2600
weighted avg       0.95      0.95      0.95      2600
```

In [68]:
```
print(confusion_matrix(y_test, y_pred))
```

```
[[1216   84]
 [  47 1253]]
```

** We can still improve the performance by tuning Input shape, number of kernels and shape of kernel**