In [1]:

```
!pip install keras-metrics
!pip install pandas
```

```
Requirement already satisfied: keras-metrics in c:\users\tushar\.conda\envs
\project1\lib\site-packages (1.1.0)
Requirement already satisfied: Keras>=2.1.5 in c:\users\tushar\.conda\envs\p
roject1\lib\site-packages (from keras-metrics) (2.9.0)
Requirement already satisfied: pandas in c:\users\tushar\.conda\envs\project
1\lib\site-packages (1.4.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\tushar\.conda\envs\p
roject1\lib\site-packages (from pandas) (2022.1)
Requirement already satisfied: numpy>=1.18.5 in c:\users\tushar\.conda\envs
\project1\lib\site-packages (from pandas) (1.21.2)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\tushar\.co
nda\envs\project1\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\tushar\.conda\envs\proje
ct1\lib\site-packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

In [2]:

```
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten , Dense , Conv2D , MaxPool2D ,  Dropout, Activa
from tensorflow.keras.preprocessing.image import ImageDataGenerator,img_to_array, load_img
from keras.preprocessing import image
from tensorflow.keras.models import Model
import PIL
import PIL.Image
from PIL import Image
import cv2
from numpy import asarray
import numpy
import pandas as pd

import keras_metrics
print(tf.__version__)
```

```
2.9.0
```

In [3]:

```
import numpy as np
import matplotlib.pyplot as plt
```

In [4]:

```
img_width = 64
img_height = 64
```

In [5]:

```
datagen = ImageDataGenerator(rescale=1/255.0, validation_split=0.3)
```

In [6]:

```
train_data_generator = datagen.flow_from_directory(directory='C:/Users/TUSHAR/malaria/cell_
                           target_size = (img_width, img_height),
                                                   class_mode = 'binary',
                                                   batch_size = 16,
                                                   subset = 'training'

                                                   )
```

```
Found 19292 images belonging to 2 classes.
```

In [7]:

```
validation_data_generator = datagen.flow_from_directory(directory='C:/Users/TUSHAR/malaria/
                                                   target_size = (img_width, img_height),
                                                   class_mode = 'binary',
                                                   batch_size = 16,
                                                   subset = 'validation'

                                                   )
```

```
Found 8266 images belonging to 2 classes.
```

In [8]:

```
path1 = "C:/Users/TUSHAR/malaria/cell_images/Parasitized/C100P61ThinF_IMG_20150918_144823_c
Load_image1 = tf.keras.preprocessing.image.load_img(path1)
print("Parasitized Cell :")
Load_image1
```

```
Parasitized Cell :
```

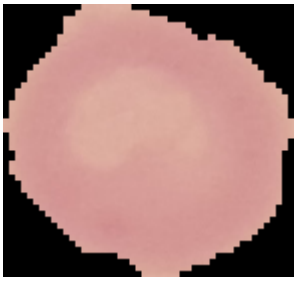
Out[8]:

In [9]:

```
path2 = "C:/Users/TUSHAR/malaria/cell_images/Uninfected/C100P61ThinF_IMG_20150918_144104_ce
print("Uninfected Cell :")
Load_image2 = tf.keras.preprocessing.image.load_img(path2)
Load_image2
```

```
Uninfected Cell :
```

Out[9]:



In [10]:

```
train_data_generator.labels
```

Out[10]:

```
array([0, 0, 0, ..., 1, 1, 1])
```

CNN model

In [11]:

```
model = Sequential()

model.add(Conv2D(32,(3,3), input_shape = (img_width,img_height,3), activation = "relu"))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))
model.add(Conv2D(32,(3,3), activation = "relu"))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.2))
model.add(Conv2D(32,(3,3), activation = "relu"))
model.add(MaxPool2D(2,2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(64, activation = "relu"))
model.add(Dropout(0.5))

model.add(Dense(1, activation = "sigmoid"))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 dropout (Dropout)           (None, 31, 31, 32)        0

 conv2d_1 (Conv2D)           (None, 29, 29, 32)        9248

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 dropout_1 (Dropout)         (None, 14, 14, 32)        0

 conv2d_2 (Conv2D)           (None, 12, 12, 32)        9248

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 32)         0
 2D)

 dropout_2 (Dropout)         (None, 6, 6, 32)          0

 flatten (Flatten)           (None, 1152)              0

 dense (Dense)               (None, 64)                73792

 dropout_3 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 93,249
Trainable params: 93,249
Non-trainable params: 0
_________________________________________________________________
```

In [12]:

```
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=["accuracy",tf.keras.metr
          tf.keras.metrics.Precision(),tf.keras.metrics.Recall(), tf.keras.metrics.FalseNeg
          tf.keras.metrics.FalsePositives(),tf.keras.metrics.TrueNegatives(),tf.keras.metri
```

In [13]:

```
history = model.fit_generator(generator=train_data_generator,
                              steps_per_epoch = len(train_data_generator),
                              epochs = 15,
                              validation_data = validation_data_generator,
                              validation_steps = len(validation_data_generator))
```

```
Epoch 1/15

C:\Users\TUSHAR\AppData\Local\Temp/ipykernel_22420/2835613124.py:1: UserWarn
ing: `Model.fit_generator` is deprecated and will be removed in a future ver
sion. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(generator=train_data_generator,

1206/1206 [==============================] - 55s 42ms/step - loss: 0.4742
- accuracy: 0.7326 - auc: 0.8414 - precision: 0.7413 - recall: 0.7147 - fa
lse_negatives: 2752.0000 - false_positives: 2406.0000 - true_negatives: 72
40.0000 - true_positives: 6894.0000 - val_loss: 0.1621 - val_accuracy: 0.9
519 - val_auc: 0.9863 - val_precision: 0.9288 - val_recall: 0.9787 - val_f
alse_negatives: 88.0000 - val_false_positives: 310.0000 - val_true_negativ
es: 3823.0000 - val_true_positives: 4045.0000
Epoch 2/15
1206/1206 [==============================] - 59s 49ms/step - loss: 0.1883
- accuracy: 0.9418 - auc: 0.9742 - precision: 0.9273 - recall: 0.9588 - fa
lse_negatives: 397.0000 - false_positives: 725.0000 - true_negatives: 892
1.0000 - true_positives: 9249.0000 - val_loss: 0.1451 - val_accuracy: 0.95
32 - val_auc: 0.9859 - val_precision: 0.9460 - val_recall: 0.9613 - val_fa
lse_negatives: 160.0000 - val_false_positives: 227.0000 - val_true_negativ
es: 3906.0000 - val_true_positives: 3973.0000
Epoch 3/15
1206/1206 [==============================] - 41s 34ms/step - loss: 0.1638
- accuracy: 0.9497 - auc: 0.9796 - precision: 0.9369 - recall: 0.9642 - fa
lse_negatives: 345.0000 - false_positives: 626.0000 - true_negatives: 902
0.0000 - true_positives: 9301.0000 - val_loss: 0.1441 - val_accuracy: 0.95
15 - val_auc: 0.9887 - val_precision: 0.9264 - val_recall: 0.9809 - val_fa
lse_negatives: 79.0000 - val_false_positives: 322.0000 - val_true_negative
s: 3811.0000 - val_true_positives: 4054.0000
Epoch 4/15
1206/1206 [==============================] - 39s 33ms/step - loss: 0.1598
- accuracy: 0.9504 - auc: 0.9810 - precision: 0.9377 - recall: 0.9649 - fa
lse_negatives: 339.0000 - false_positives: 618.0000 - true_negatives: 902
8.0000 - true_positives: 9307.0000 - val_loss: 0.1305 - val_accuracy: 0.95
69 - val_auc: 0.9892 - val_precision: 0.9476 - val_recall: 0.9673 - val_fa
lse_negatives: 135.0000 - val_false_positives: 221.0000 - val_true_negativ
es: 3912.0000 - val_true_positives: 3998.0000
Epoch 5/15
1206/1206 [==============================] - 64s 53ms/step - loss: 0.1509
- accuracy: 0.9537 - auc: 0.9826 - precision: 0.9417 - recall: 0.9672 - fa
lse_negatives: 316.0000 - false_positives: 578.0000 - true_negatives: 906
8.0000 - true_positives: 9330.0000 - val_loss: 0.1390 - val_accuracy: 0.95
57 - val_auc: 0.9861 - val_precision: 0.9475 - val_recall: 0.9649 - val_fa
lse_negatives: 145.0000 - val_false_positives: 221.0000 - val_true_negativ
es: 3912.0000 - val_true_positives: 3988.0000
Epoch 6/15
1206/1206 [==============================] - 141s 117ms/step - loss: 0.149
1 - accuracy: 0.9533 - auc: 0.9833 - precision: 0.9404 - recall: 0.9680 -
false_negatives: 309.0000 - false_positives: 592.0000 - true_negatives: 90
54.0000 - true_positives: 9337.0000 - val_loss: 0.1415 - val_accuracy: 0.9
521 - val_auc: 0.9873 - val_precision: 0.9271 - val_recall: 0.9814 - val_f
alse_negatives: 77.0000 - val_false_positives: 319.0000 - val_true_negativ
```

```
es: 3814.0000 - val_true_positives: 4056.0000
Epoch 7/15
1206/1206 [==============================] - 107s 89ms/step - loss: 0.1477
- accuracy: 0.9535 - auc: 0.9839 - precision: 0.9409 - recall: 0.9678 - fa
lse_negatives: 311.0000 - false_positives: 586.0000 - true_negatives: 906
0.0000 - true_positives: 9335.0000 - val_loss: 0.1316 - val_accuracy: 0.95
60 - val_auc: 0.9887 - val_precision: 0.9456 - val_recall: 0.9676 - val_fa
lse_negatives: 134.0000 - val_false_positives: 230.0000 - val_true_negativ
es: 3903.0000 - val_true_positives: 3999.0000
Epoch 8/15
1206/1206 [==============================] - 166s 138ms/step - loss: 0.143
3 - accuracy: 0.9538 - auc: 0.9844 - precision: 0.9424 - recall: 0.9667 -
false_negatives: 321.0000 - false_positives: 570.0000 - true_negatives: 90
76.0000 - true_positives: 9325.0000 - val_loss: 0.1277 - val_accuracy: 0.9
585 - val_auc: 0.9893 - val_precision: 0.9529 - val_recall: 0.9647 - val_f
alse_negatives: 146.0000 - val_false_positives: 197.0000 - val_true_negati
ves: 3936.0000 - val_true_positives: 3987.0000
Epoch 9/15
1206/1206 [==============================] - 367s 304ms/step - loss: 0.135
8 - accuracy: 0.9558 - auc: 0.9863 - precision: 0.9449 - recall: 0.9682 -
false_negatives: 307.0000 - false_positives: 545.0000 - true_negatives: 91
01.0000 - true_positives: 9339.0000 - val_loss: 0.1310 - val_accuracy: 0.9
563 - val_auc: 0.9885 - val_precision: 0.9523 - val_recall: 0.9608 - val_f
alse_negatives: 162.0000 - val_false_positives: 199.0000 - val_true_negati
ves: 3934.0000 - val_true_positives: 3971.0000
Epoch 10/15
1206/1206 [==============================] - 93s 77ms/step - loss: 0.1342
- accuracy: 0.9565 - auc: 0.9865 - precision: 0.9445 - recall: 0.9700 - fa
lse_negatives: 289.0000 - false_positives: 550.0000 - true_negatives: 909
6.0000 - true_positives: 9357.0000 - val_loss: 0.1338 - val_accuracy: 0.95
66 - val_auc: 0.9875 - val_precision: 0.9508 - val_recall: 0.9630 - val_fa
lse_negatives: 153.0000 - val_false_positives: 206.0000 - val_true_negativ
es: 3927.0000 - val_true_positives: 3980.0000
Epoch 11/15
1206/1206 [==============================] - 99s 82ms/step - loss: 0.1365
- accuracy: 0.9571 - auc: 0.9860 - precision: 0.9454 - recall: 0.9702 - fa
lse_negatives: 287.0000 - false_positives: 540.0000 - true_negatives: 910
6.0000 - true_positives: 9359.0000 - val_loss: 0.1289 - val_accuracy: 0.95
50 - val_auc: 0.9891 - val_precision: 0.9524 - val_recall: 0.9579 - val_fa
lse_negatives: 174.0000 - val_false_positives: 198.0000 - val_true_negativ
es: 3935.0000 - val_true_positives: 3959.0000
Epoch 12/15
1206/1206 [==============================] - 102s 85ms/step - loss: 0.1354
- accuracy: 0.9563 - auc: 0.9868 - precision: 0.9467 - recall: 0.9670 - fa
lse_negatives: 318.0000 - false_positives: 525.0000 - true_negatives: 912
1.0000 - true_positives: 9328.0000 - val_loss: 0.1338 - val_accuracy: 0.95
51 - val_auc: 0.9889 - val_precision: 0.9533 - val_recall: 0.9572 - val_fa
lse_negatives: 177.0000 - val_false_positives: 194.0000 - val_true_negativ
es: 3939.0000 - val_true_positives: 3956.0000
Epoch 13/15
1206/1206 [==============================] - 99s 82ms/step - loss: 0.1313
- accuracy: 0.9578 - auc: 0.9871 - precision: 0.9456 - recall: 0.9715 - fa
lse_negatives: 275.0000 - false_positives: 539.0000 - true_negatives: 910
7.0000 - true_positives: 9371.0000 - val_loss: 0.1350 - val_accuracy: 0.95
28 - val_auc: 0.9893 - val_precision: 0.9337 - val_recall: 0.9748 - val_fa
lse_negatives: 104.0000 - val_false_positives: 286.0000 - val_true_negativ
es: 3847.0000 - val_true_positives: 4029.0000
Epoch 14/15
1206/1206 [==============================] - 64s 53ms/step - loss: 0.1278
- accuracy: 0.9582 - auc: 0.9879 - precision: 0.9476 - recall: 0.9700 - fa
lse_negatives: 289.0000 - false_positives: 517.0000 - true_negatives: 912
```

```
9.0000 - true_positives: 9357.0000 - val_loss: 0.1320 - val_accuracy: 0.95
29 - val_auc: 0.9888 - val_precision: 0.9341 - val_recall: 0.9746 - val_fa
lse_negatives: 105.0000 - val_false_positives: 284.0000 - val_true_negativ
es: 3849.0000 - val_true_positives: 4028.0000
Epoch 15/15
1206/1206 [==============================] - 67s 56ms/step - loss: 0.1300
- accuracy: 0.9561 - auc: 0.9879 - precision: 0.9453 - recall: 0.9682 - fa
lse_negatives: 307.0000 - false_positives: 540.0000 - true_negatives: 910
6.0000 - true_positives: 9339.0000 - val_loss: 0.1331 - val_accuracy: 0.95
43 - val_auc: 0.9879 - val_precision: 0.9499 - val_recall: 0.9591 - val_fa
lse_negatives: 169.0000 - val_false_positives: 209.0000 - val_true_negativ
es: 3924.0000 - val_true_positives: 3964.0000
```

In [14]:

```
history.history
```

Out[14]:


```
{'loss': [0.47415444254875183,
  0.18833310902118683,
  0.16384179890155792,
  0.15979844331741333,
  0.1508999615907669,
  0.14909353852272034,
  0.14773476123809814,
  0.1433371901512146,
  0.13582460582256317,
  0.13415615260601044,
  0.13647086918354034,
  0.1353720873594284,
  0.13129423558712006,
  0.12782011926174164,
  0.129964217543602],
 'accuracy': [0.7326352596282959,
  0.941841185092926,
  0.9496682286262512,
  0.9503939747810364,
  0.9536595344543457,
  0.9532967209815979,
  0.9535040259361267,
  0.9538150429725647,
  0.9558365941047668,
  0.9565104842185974,
  0.9571325182914734,
  0.9563031196594238,
  0.9578063488006592,
  0.9582210183143616,
  0.956095814704895],
 'auc': [0.8413510322570801,
  0.9741533398628235,
  0.979616641998291,
  0.9810283780097961,
  0.9825879335403442,
  0.9832656383514404,
  0.9839394092559814,
  0.9843878149986267,
  0.9862507581710815,
  0.9865179061889648,
  0.9860384464263916,
  0.9867541790008545,
  0.9871281981468201,
  0.9879489541053772,
  0.9879450798034668],
 'precision': [0.7412903308868408,
  0.9273110032081604,
  0.9369396567344666,
  0.937732994556427,
  0.9416633248329163,
  0.9403766989707947,
  0.940933346748352,
  0.9423951506614685,
  0.9448603987693787,
  0.9444836974143982,
```

```
 0.9454490542411804,
 0.9467167258262634,
 0.945610523223877,
 0.9476402401924133,
 0.9453386068344116],
'recall': [0.7147004008293152,
 0.9588430523872375,
 0.9642338752746582,
 0.9648559093475342,
 0.9672403335571289,
 0.9679660201072693,
 0.9677586555480957,
 0.9667219519615173,
 0.9681733250617981,
 0.9700393676757812,
 0.9702467322349548,
 0.9670329689979553,
 0.9714908003807068,
 0.9700393676757812,
 0.9681733250617981],
'false_negatives': [2752.0,
 397.0,
 345.0,
 339.0,
 316.0,
 309.0,
 311.0,
 321.0,
 307.0,
 289.0,
 287.0,
 318.0,
 275.0,
 289.0,
 307.0],
'false_positives': [2406.0,
 725.0,
 626.0,
 618.0,
 578.0,
 592.0,
 586.0,
 570.0,
 545.0,
 550.0,
 540.0,
 525.0,
 539.0,
 517.0,
 540.0],
'true_negatives': [7240.0,
 8921.0,
 9020.0,
 9028.0,
 9068.0,
 9054.0,
 9060.0,
 9076.0,
 9101.0,
 9096.0,
 9106.0,
```

```
 9121.0,
 9107.0,
 9129.0,
 9106.0],
'true_positives': [6894.0,
 9249.0,
 9301.0,
 9307.0,
 9330.0,
 9337.0,
 9335.0,
 9325.0,
 9339.0,
 9357.0,
 9359.0,
 9328.0,
 9371.0,
 9357.0,
 9339.0],
'val_loss': [0.16208207607269287,
 0.1451147496700287,
 0.1440781056880951,
 0.13049225509166718,
 0.13899841904640198,
 0.1414981633424759,
 0.1315917819738388,
 0.12770162522792816,
 0.13103757798671722,
 0.13384872674942017,
 0.1289084404706955,
 0.1338093876838684,
 0.13501965999603271,
 0.1320132464170456,
 0.13309067487716675],
'val_accuracy': [0.951850950717926,
 0.9531816840171814,
 0.9514880180358887,
 0.956932008266449,
 0.9557222127914429,
 0.9520928859710693,
 0.955964207649231,
 0.9585047364234924,
 0.9563271403312683,
 0.9565690755844116,
 0.9549963474273682,
 0.9551173448562622,
 0.952818751335144,
 0.9529397487640381,
 0.9542704820632935],
'val_auc': [0.9863092303276062,
 0.9858543872833252,
 0.9886935353279114,
 0.9891721606254578,
 0.9861369132995605,
 0.9873274564743042,
 0.9886631369590759,
 0.9893069267272949,
 0.9884518980979919,
 0.9875177145004272,
 0.9890713095664978,
 0.9888871908187866,
```

```
 0.9892838001251221,
 0.9887883067131042,
 0.9878962635993958],
'val_precision': [0.9288174510002136,
 0.9459523558616638,
 0.92641681432724,
 0.9476179480552673,
 0.9474934935569763,
 0.9270856976509094,
 0.9456136226654053,
 0.9529158473014832,
 0.952278196811676,
 0.9507883191108704,
 0.9523695111274719,
 0.9532530307769775,
 0.9337195754051208,
 0.9341372847557068,
 0.9499161243438721],
'val_recall': [0.9787079691886902,
 0.9612872004508972,
 0.9808855652809143,
 0.9673360586166382,
 0.9649165272712708,
 0.9813694357872009,
 0.9675780534744263,
 0.9646745920181274,
 0.9608032703399658,
 0.9629808664321899,
 0.957899808883667,
 0.9571739435195923,
 0.9748367071151733,
 0.9745947122573853,
 0.9591096043586731],
'val_false_negatives': [88.0,
 160.0,
 79.0,
 135.0,
 145.0,
 77.0,
 134.0,
 146.0,
 162.0,
 153.0,
 174.0,
 177.0,
 104.0,
 105.0,
 169.0],
'val_false_positives': [310.0,
 227.0,
 322.0,
 221.0,
 221.0,
 319.0,
 230.0,
 197.0,
 199.0,
 206.0,
 198.0,
 194.0,
 286.0,
```

```
  284.0,
  209.0],
 'val_true_negatives': [3823.0,
  3906.0,
  3811.0,
  3912.0,
  3912.0,
  3814.0,
  3903.0,
  3936.0,
  3934.0,
  3927.0,
  3935.0,
  3939.0,
  3847.0,
  3849.0,
  3924.0],
 'val_true_positives': [4045.0,
  3973.0,
  4054.0,
  3998.0,
  3988.0,
  4056.0,
  3999.0,
  3987.0,
  3971.0,
  3980.0,
  3959.0,
  3956.0,
  4029.0,
  4028.0,
  3964.0]}
```

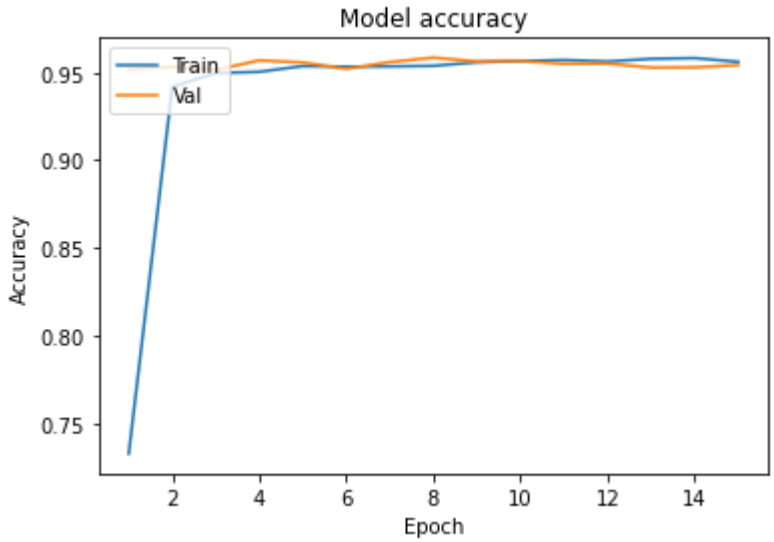
Plotting Graphs and Outputs

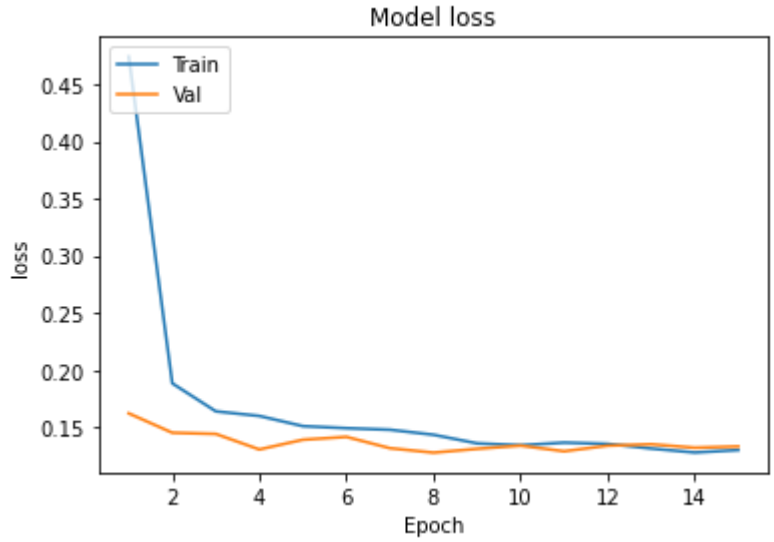
In [15]:

```
def plot_learningCurve(history,epoch):
  #plot learning and validation accuracy values
  epoch_range=range(1,epoch+1)
  plt.plot(epoch_range,history.history['accuracy'])
  plt.plot(epoch_range,history.history['val_accuracy'])
  plt.title("Model accuracy")
  plt.ylabel("Accuracy")
  plt.xlabel("Epoch")
  plt.legend(['Train','Val'], loc = 'upper left')
  plt.show()

   #plot learning and validation loss values
  plt.plot(epoch_range,history.history['loss'])
  plt.plot(epoch_range,history.history['val_loss'])
  plt.title("Model loss")
  plt.ylabel("loss")
  plt.xlabel("Epoch")
  plt.legend(['Train','Val'], loc = 'upper left')
  plt.show()
```

In [16]:

```
plot_learningCurve(history,15)
```





In [17]:

```
img_path=r"C:/Users/TUSHAR/malaria/cell_images/Parasitized/C100P61ThinF_IMG_20150918_144104
img = tf.keras.preprocessing.image.load_img(img_path, target_size=(64,64))
img_tensor = img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor /= 255.
```

In [18]:

```
for j in range(0,1):
  activation_model = Model(inputs=model.inputs, outputs=model.layers[6].output)
  activation = activation_model(img_tensor)
  print(model.get_layer(index = 6).name)
  plt.figure(figsize=(20,20))
  print(activation)
```

```
conv2d_2
tf.Tensor(
[[[[0.         0.         0.         ... 0.         0.
    0.21733752]
   [0.         0.04873561 0.         ... 0.         0.
    0.07749383]
   [0.         0.         0.         ... 0.         0.
    0.        ]
   ...
   [0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]]

  [[0.         0.         0.         ... 0.         0.
    0.00191365]
   [0.         0.04760345 0.         ... 0.         0.
    0.        ]
   [0.         0.         0.05389107 ... 0.         0.
    0.01801428]
   ...
   [0.         0.         0.         ... 0.04600011 0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]]

  [[0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.05187155 0.
    0.        ]
   [0.         0.         0.         ... 0.04332941 0.
    0.        ]
   ...
   [0.         0.         0.         ... 0.04511498 0.
    0.        ]
   [0.         0.         0.         ... 0.04887653 0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]]

  ...

  [[0.         0.         0.         ... 0.04574909 0.
    0.        ]
   [0.         0.         0.         ... 0.04036599 0.
    0.        ]
   [0.         0.         0.         ... 0.05022356 0.
```

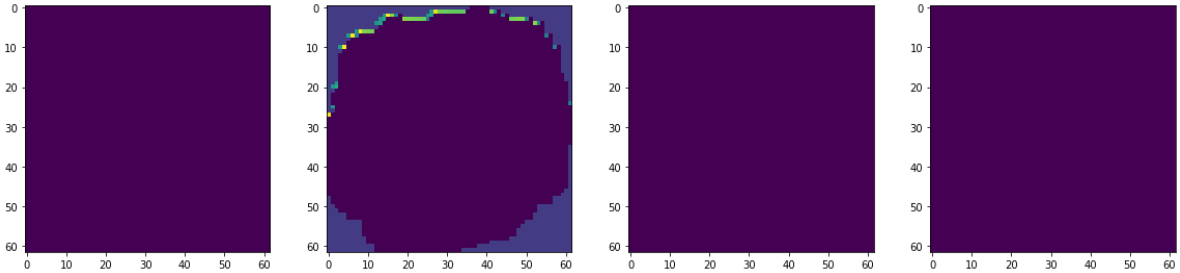
```
    0.        ]
   ...
   [0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.03720443
    0.        ]
   [0.         0.09789867 0.         ... 0.         0.
    0.        ]]

  [[0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.00135227 0.
    0.0030197 ]
   [0.         0.         0.         ... 0.03559976 0.
    0.        ]
   ...
   [0.         0.         0.         ... 0.         0.0756468
    0.        ]
   [0.         0.21931729 0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]]

  [[0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]
   ...
   [0.         0.10218927 0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]
   [0.         0.         0.         ... 0.         0.
    0.        ]]]], shape=(1, 12, 12, 32), dtype=float32)
```

```
<Figure size 1440x1440 with 0 Axes>
```

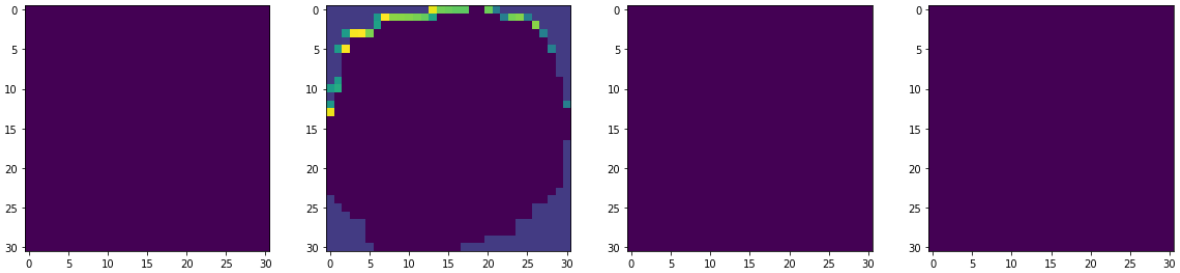
In [19]:

```
for j in range(0,9):
  activation_model = Model(inputs=model.inputs, outputs=model.layers[j].output)
  activation = activation_model(img_tensor)
  print(model.get_layer(index = j).name)
  plt.figure(figsize=(20,20))

  for i in range(4):
      plt.subplot(4,4,i+1)

      plt.imshow(activation[0,:,:,i])
  plt.show()
```

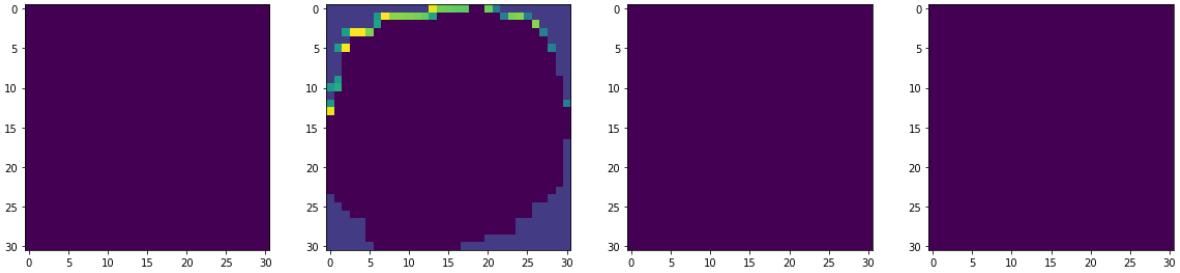
conv2d



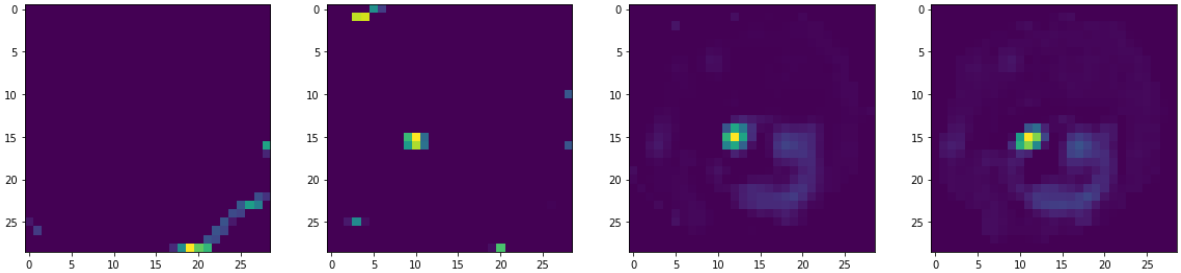
max_pooling2d



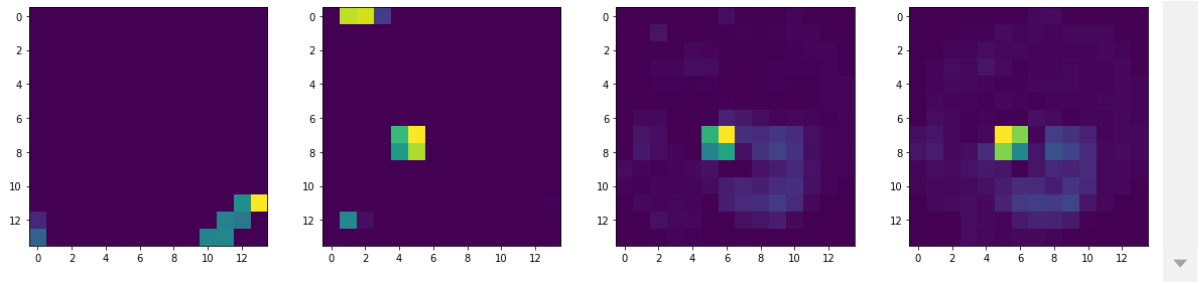
dropout



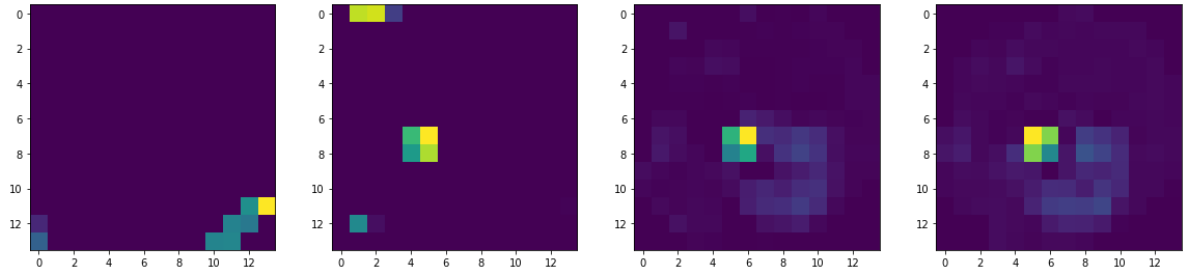
conv2d_1



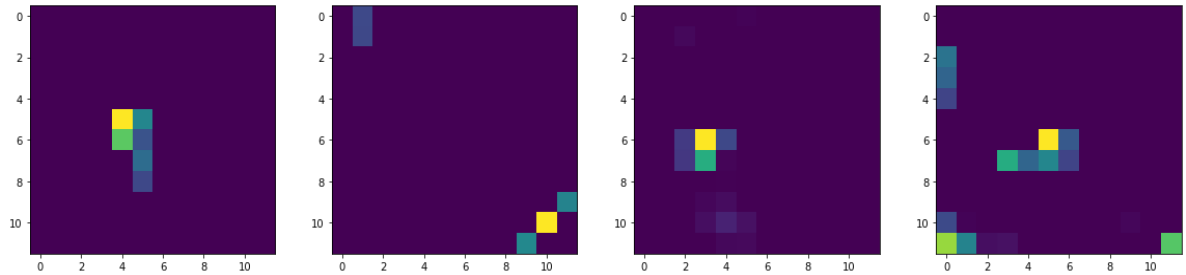
max_pooling2d_1



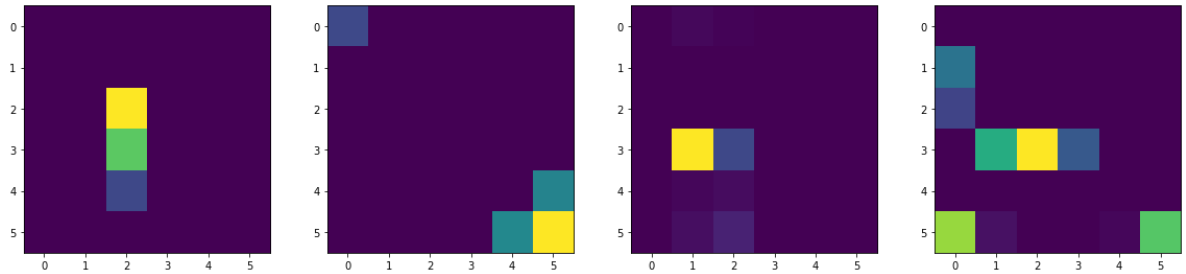
dropout_1



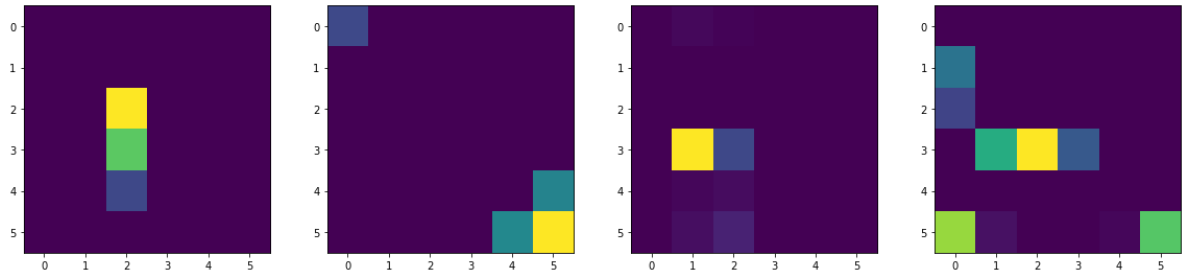
conv2d_2



max_pooling2d_2



dropout_2



In [20]:

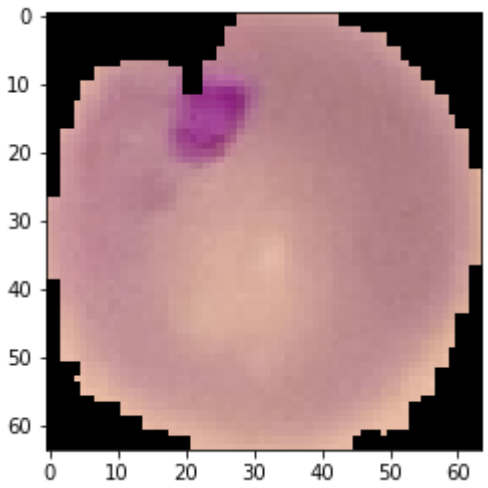
```
for j in range(9,13):
  activation_model = Model(inputs=model.inputs, outputs=model.layers[j].output)
  activation = activation_model(img_tensor)
  print(model.get_layer(index = j).name)
  plt.figure(figsize=(20,20))
  print(activation)
```

```
flatten
tf.Tensor([[0.         0.04873561 0.         ... 0.         0.         0.
]], shape=(1, 1152), dtype=float32)
dense
tf.Tensor(
[[ 2.5145137  0.         0.         3.5105863  1.022135   0.
  10.98222    8.899346  12.516857   6.5334935 12.425389   0.
   4.4840484  0.         8.664883   6.1059036  9.143598   3.592224
   0.         0.2321178  6.1526623  0.         0.         8.119906
   0.         0.         3.9622989  4.517168  10.827438   5.329316
   8.078925   0.         9.650024   2.4534864  6.7880063  0.
   0.         0.         0.         0.         4.137784   0.
  12.048561   0.         4.934613   3.8733768  6.6572866  0.
   0.         0.         0.         9.834607   7.148009   0.
   0.         8.771756   0.         0.         0.         9.867221
   0.         0.        11.100216   0.       ]], shape=(1, 64), dtype=float3
2)
dropout_3
tf.Tensor(
[[ 2.5145137  0.         0.         3.5105863  1.022135   0.
  10.98222    8.899346  12.516857   6.5334935 12.425389   0.
   4.4840484  0.         8.664883   6.1059036  9.143598   3.592224
   0.         0.2321178  6.1526623  0.         0.         8.119906
   0.         0.         3.9622989  4.517168  10.827438   5.329316
   8.078925   0.         9.650024   2.4534864  6.7880063  0.
   0.         0.         0.         0.         4.137784   0.
  12.048561   0.         4.934613   3.8733768  6.6572866  0.
   0.         0.         0.         9.834607   7.148009   0.
   0.         8.771756   0.         0.         0.         9.867221
   0.         0.        11.100216   0.       ]], shape=(1, 64), dtype=float3
2)
dense_1
tf.Tensor([[0.00016342]], shape=(1, 1), dtype=float32)

<Figure size 1440x1440 with 0 Axes>

<Figure size 1440x1440 with 0 Axes>

<Figure size 1440x1440 with 0 Axes>

<Figure size 1440x1440 with 0 Axes>
```

In [21]:

```
image_path="C:/Users/TUSHAR/Desktop/malaria/predict/C33P1thinF_IMG_20150619_121229a_cell_17
img = tf.keras.preprocessing.image.load_img(image_path, target_size=(64, 64))
plt.imshow(img)
img = np.expand_dims(img, axis=0)
predictions = (model.predict(img) > 0.5).astype("int32")
plt.show()
```

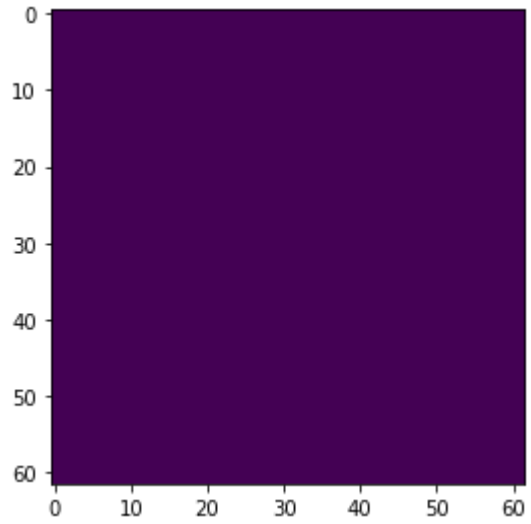
```
1/1 [==============================] - 6s 6s/step
```



In [22]:

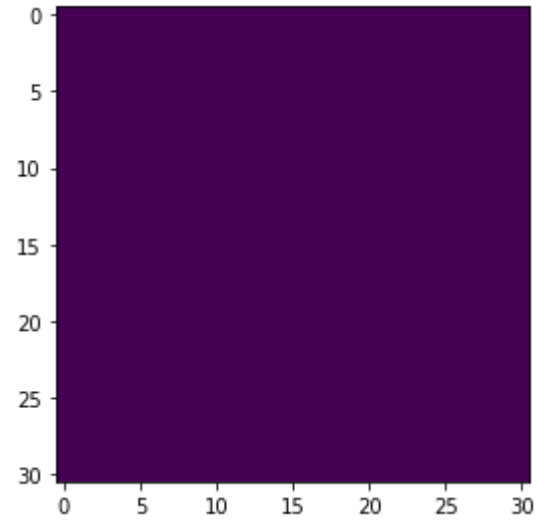
```
img = tf.keras.preprocessing.image.load_img(image_path, target_size=(64,64))
img_tensor = img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
img_tensor /= 255.
# for j in range(0,1):
#   activation_model = Model(inputs=model.inputs, outputs=model.layers[6].output)
#   activation = activation_model(img_tensor)
#   print(model.get_layer(index = 6).name)
#   plt.figure(figsize=(20,20))
#   print(activation)
for j in range(0,9):
  activation_model = Model(inputs=model.inputs, outputs=model.layers[j].output)
  activation = activation_model(img_tensor)
  print(model.get_layer(index = j).name)
  plt.figure(figsize=(20,20))

  for i in range(1):
      plt.subplot(4,4,i+1)

      plt.imshow(activation[0,:,:,i])
  plt.show()
for j in range(9,13):
  activation_model = Model(inputs=model.inputs, outputs=model.layers[j].output)
  activation = activation_model(img_tensor)
  print(model.get_layer(index = j).name)
  plt.figure(figsize=(20,20))
  print(activation)
if(predictions[0]==1):
  print(" -------------Uninfected-------------")
else :
  print("-------------Parasitized-------------")
```

conv2d

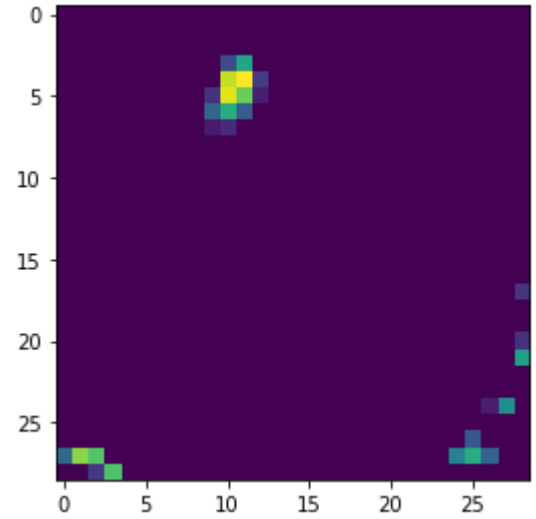


max_pooling2d

dropout

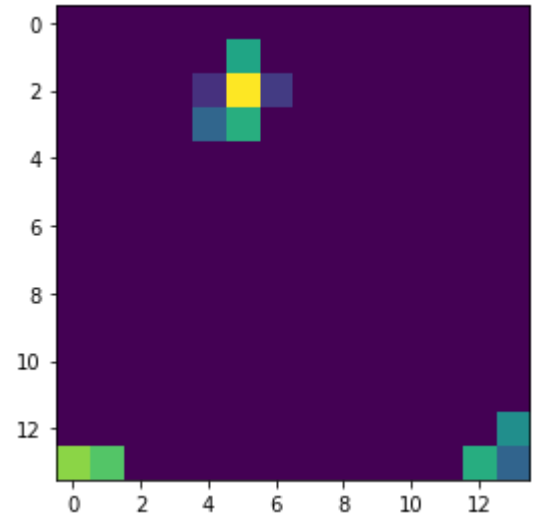


conv2d_1

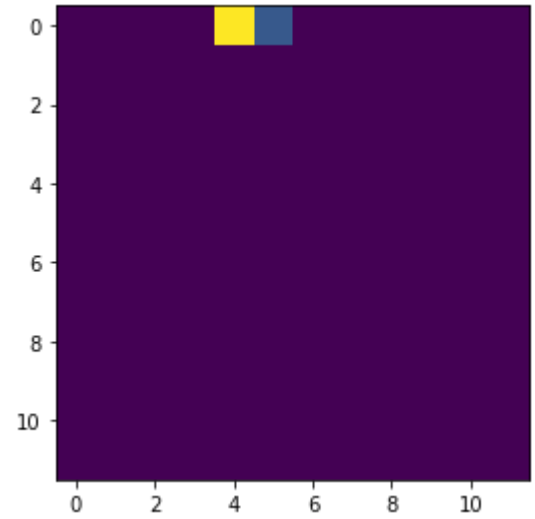


max_pooling2d_1

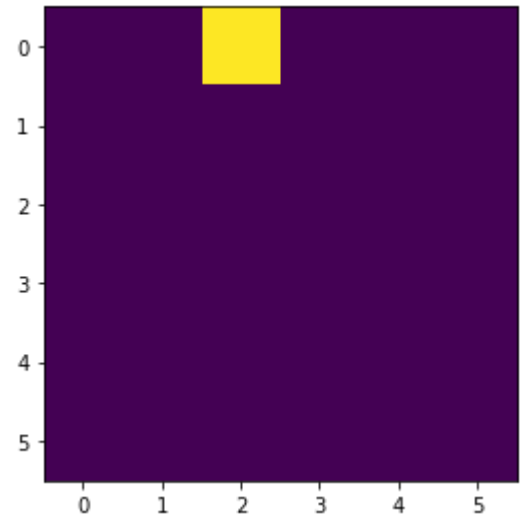
dropout_1



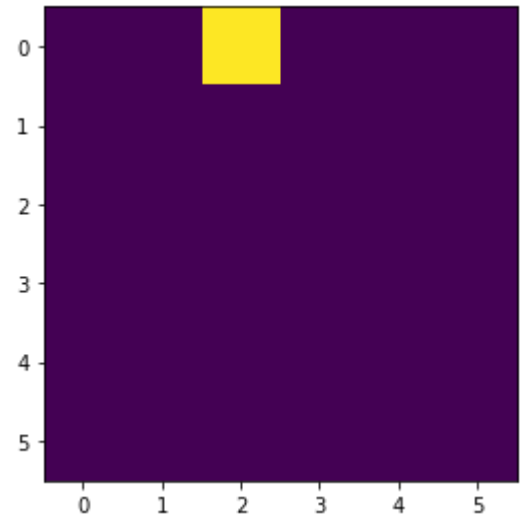
conv2d_2



max_pooling2d_2



dropout_2




```
flatten
tf.Tensor([[0.         0.         0.         ... 0.00937566 0.07398548 0.
]], shape=(1, 1152), dtype=float32)
dense
tf.Tensor(
[[ 0.          0.          0.          4.9162736   6.884733    0.
   5.6071925   0.          6.4320836   5.870165    5.7063856   0.
   0.98942536  0.          6.561062    0.          1.9658072   0.
   0.          0.          8.268598    0.          0.          8.97612
   0.          0.          1.7449192   6.681889    7.789742    0.
   3.0165615   0.          8.309432   10.303375    0.          0.
   0.          0.          0.          0.          0.          0.
   5.382281    6.21482     4.808842    7.6315074   4.365401    0.
   0.          0.          0.          0.          6.42188     0.
   0.          7.157146    0.          0.          0.          7.446033
   0.          0.          0.          0.        ]], shape=(1, 64), dtype=fl
oat32)
dropout_3
tf.Tensor(
[[ 0.          0.          0.          4.9162736   6.884733    0.
   5.6071925   0.          6.4320836   5.870165    5.7063856   0.
   0.98942536  0.          6.561062    0.          1.9658072   0.
   0.          0.          8.268598    0.          0.          8.97612
   0.          0.          1.7449192   6.681889    7.789742    0.
   3.0165615   0.          8.309432   10.303375    0.          0.
   0.          0.          0.          0.          0.          0.
   5.382281    6.21482     4.808842    7.6315074   4.365401    0.
   0.          0.          0.          0.          6.42188     0.
   0.          7.157146    0.          0.          0.          7.446033
   0.          0.          0.          0.        ]], shape=(1, 64), dtype=fl
oat32)
dense_1
tf.Tensor([[0.00398245]], shape=(1, 1), dtype=float32)
-------------Parasitized-------------


<Figure size 1440x1440 with 0 Axes>

<Figure size 1440x1440 with 0 Axes>

<Figure size 1440x1440 with 0 Axes>

<Figure size 1440x1440 with 0 Axes>
```