In [3]:

```
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
import numpy as np
import pandas as pd
import os
```

In [4]:

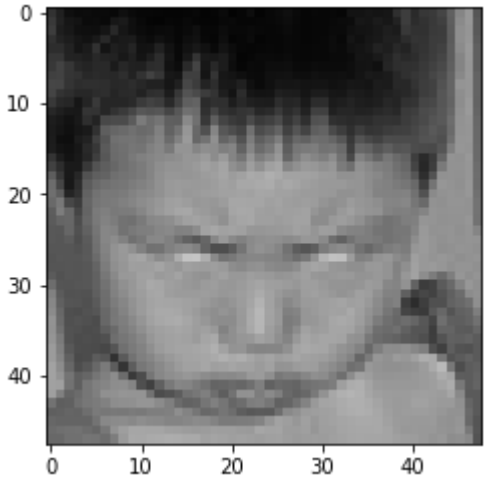
```
Datadirectory = "train/"
```

In [5]:

```
Classes = ["0","1","2","3","4","5","6"]
```

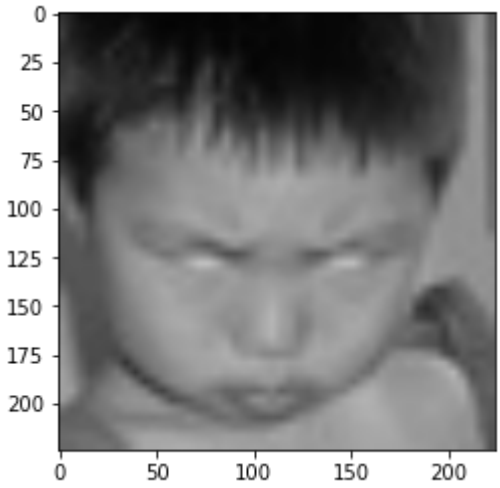
In [6]:

```
for category in Classes:
    path = os.path.join(Datadirectory, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img))
        plt.imshow(cv2.cvtColor(img_array,cv2.COLOR_BGR2RGB))
        plt.show()
        break
    break
```



In [7]:

```
img_size= 224
new_array= cv2.resize(img_array,(img_size,img_size))
plt.imshow(cv2.cvtColor(new_array,cv2.COLOR_BGR2RGB))
plt.show()
```



In [8]:

```
new_array.shape
```

Out[8]:

```
(224, 224, 3)
```

In [9]:

```
training_Data = []

def create_training_Data():
    for category in Classes:
        path = os.path.join(Datadirectory, category)
        class_num= Classes.index(category)
        for img in os.listdir(path):
            try:
                img_array = cv2.imread(os.path.join(path,img))
                new_array= cv2.resize(img_array,(img_size,img_size))
                training_Data.append([new_array,class_num])
            except Exception as e :
                pass
```

In [10]:

```
create_training_Data()
```

In [11]:

```
print(len(training_Data))
```

```
28709
```

In [12]:

```
temp = np.array(training_Data)
```

```
C:\Users\PUSPKA~1\AppData\Local\Temp/ipykernel_7624/2755283514.py:1: Visible
DeprecationWarning: Creating an ndarray from ragged nested sequences (which
is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or
shapes) is deprecated. If you meant to do this, you must specify 'dtype=obje
ct' when creating the ndarray.
  temp = np.array(training_Data)
```

temp.shape()

In [13]:

```
temp.shape
```

Out[13]:

```
(28709, 2)
```

In [14]:

```
import random
random.shuffle(training_Data)
```

In [15]:

```
X = []
y = []

for features,label in training_Data:
    X.append(features)
    y.append(label)

X = np.array(X).reshape(-1 , img_size , img_size,3)
```

In [16]:

```
X.shape
```

Out[16]:

```
(28709, 224, 224, 3)
```

In [27]:

```
y[546]
```

Out[27]:

```
2
```

In [28]:

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

In [29]:

```
Y.shape
```

Out[29]:

```
(28709,)
```

In [30]:

```
#deep learning

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

In [31]:

```
model = tf.keras.applications.MobileNetV2()   ##pre - trained MOdel
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applic
ations/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224.
h5 (https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v
2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224.h5)
14540800/14536120 [==============================] - 781s 54us/step
14548992/14536120 [==============================] - 781s 54us/step
```

In [33]:

```
model.summary()
```

```
_depthwise_relu[0][0]']

 block_9_project_BN (BatchNorma  (None, 14, 14, 64)  256         ['block_9
_project[0][0]']
 lization)

 block_9_add (Add)              (None, 14, 14, 64)   0           ['block_8
_add[0][0]',
                                                                  'block_9
_project_BN[0][0]']

 block_10_expand (Conv2D)       (None, 14, 14, 384)  24576       ['block_9
_add[0][0]']

 block_10_expand_BN (BatchNorma  (None, 14, 14, 384)  1536       ['block_1
0_expand[0][0]']
 lization)

 block_10_expand_relu (ReLU)    (None, 14, 14, 384)  0           ['block_1
0_expand_BN[0][0]']
```

In [35]:

```
#transfer learning
base_input = model.layers[0].input
```

In [38]:

```
base_output = model.layers[-2].output
```

In [39]:

```
base_output
```

Out[39]:

```
<KerasTensor: shape=(None, 1280) dtype=float32 (created by layer 'global_ave
rage_pooling2d')>
```

In [40]:

```
final_output = layers.Dense(128)(base_output)
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(64)(final_output)
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(7,activation = 'softmax')(final_output) #for 7 calsses
```

In [41]:

```
final_output
```

Out[41]:

```
<KerasTensor: shape=(None, 7) dtype=float32 (created by layer 'dense_2')>
```

In [42]:

```
new_model = keras.Model(inputs = base_input , outputs = final_output)
```

In [43]:

```
new_model.summary()
```

```
 block_1_depthwise (DepthwiseCo  (None, 56, 56, 96)  864         ['block_1
_pad[0][0]']
 nv2D)

 block_1_depthwise_BN (BatchNor  (None, 56, 56, 96)  384         ['block_1
_depthwise[0][0]']
 malization)

 block_1_depthwise_relu (ReLU)  (None, 56, 56, 96)   0           ['block_1
_depthwise_BN[0][0]']

 block_1_project (Conv2D)       (None, 56, 56, 24)   2304        ['block_1
_depthwise_relu[0][0]']

 block_1_project_BN (BatchNorma  (None, 56, 56, 24)  96          ['block_1
_project[0][0]']
 lization)

 block_2_expand (Conv2D)        (None, 56, 56, 144)  3456        ['block_1
 project_BN[0][0]']
```

In [45]:

```
new_model.compile(loss ="sparse_categorical_crossentropy", optimizer = "adam" , metrics = [
```

In [46]:

```
new_model.fit(X,Y , epochs = 25)
```

```
Epoch 1/25
898/898 [==============================] - 1564s 2s/step - loss: 1.2534 - ac
curacy: 0.5266
Epoch 2/25
898/898 [==============================] - 1536s 2s/step - loss: 1.0763 - ac
curacy: 0.5976
Epoch 3/25
898/898 [==============================] - 1522s 2s/step - loss: 1.0003 - ac
curacy: 0.6231
Epoch 4/25
898/898 [==============================] - 1512s 2s/step - loss: 0.9433 - ac
curacy: 0.6489
Epoch 5/25
898/898 [==============================] - 1524s 2s/step - loss: 0.8950 - ac
curacy: 0.6687
Epoch 6/25
898/898 [==============================] - 1456s 2s/step - loss: 0.8381 - ac
curacy: 0.6895
Epoch 7/25
898/898 [==============================] - 1465s 2s/step - loss: 0.7916 - ac
curacy: 0.7092
Epoch 8/25
898/898 [==============================] - 1465s 2s/step - loss: 0.7396 - ac
curacy: 0.7282
Epoch 9/25
898/898 [==============================] - 1475s 2s/step - loss: 0.6930 - ac
curacy: 0.7444
Epoch 10/25
898/898 [==============================] - 1471s 2s/step - loss: 0.6336 - ac
curacy: 0.7689
Epoch 11/25
898/898 [==============================] - 1468s 2s/step - loss: 0.5841 - ac
curacy: 0.7882
Epoch 12/25
898/898 [==============================] - 1474s 2s/step - loss: 0.5294 - ac
curacy: 0.8070
Epoch 13/25
898/898 [==============================] - 1467s 2s/step - loss: 0.4877 - ac
curacy: 0.8233
Epoch 14/25
898/898 [==============================] - 1467s 2s/step - loss: 0.4458 - ac
curacy: 0.8412
Epoch 15/25
898/898 [==============================] - 1455s 2s/step - loss: 0.3989 - ac
curacy: 0.8562
Epoch 16/25
898/898 [==============================] - 1467s 2s/step - loss: 0.3632 - ac
curacy: 0.8682
Epoch 17/25
898/898 [==============================] - 1469s 2s/step - loss: 0.3346 - ac
curacy: 0.8821
Epoch 18/25
898/898 [==============================] - 1475s 2s/step - loss: 0.3069 - ac
curacy: 0.8924
Epoch 19/25
898/898 [==============================] - 1470s 2s/step - loss: 0.2739 - ac
curacy: 0.9043
```

```
Epoch 20/25
898/898 [==============================] - 1476s 2s/step - loss: 0.2598 - ac
curacy: 0.9080
Epoch 21/25
898/898 [==============================] - 1478s 2s/step - loss: 0.2453 - ac
curacy: 0.9134
Epoch 22/25
898/898 [==============================] - 1615s 2s/step - loss: 0.2274 - ac
curacy: 0.9201
Epoch 23/25
898/898 [==============================] - 1896s 2s/step - loss: 0.2064 - ac
curacy: 0.9279
Epoch 24/25
898/898 [==============================] - 1794s 2s/step - loss: 0.1998 - ac
curacy: 0.9313
Epoch 25/25
898/898 [==============================] - 1613s 2s/step - loss: 0.1925 - ac
curacy: 0.9320
```

Out[46]:

```
<keras.callbacks.History at 0x24d99480c40>
```

In [47]:

```
new_model.save('my_modelforfacial.h5')
```

In [48]:

```
new_model = tf.keras.models.load_model('my_modelforfacial.h5')
```

In [116]:

```
frame = cv2.imread("img1.jpg")
```

In [117]:

```
frame.shape
```

Out[117]:

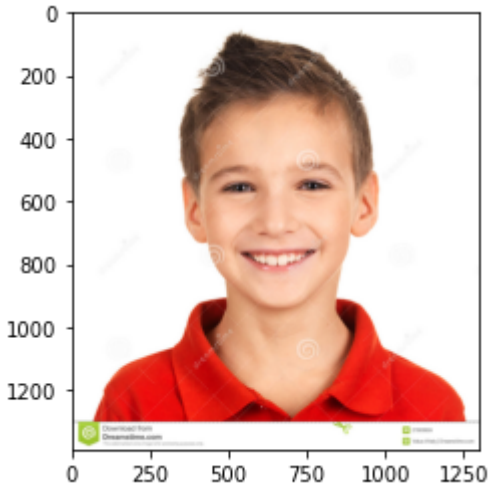
```
(1390, 1300, 3)
```

In [118]:

```
plt.imshow(cv2.cvtColor(frame,cv2.COLOR_BGR2RGB))
```

Out[118]:

```
<matplotlib.image.AxesImage at 0x24da32f7880>
```



In [119]:

```
#face detection on gray images
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_defaul
```

In [120]:

```
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
```

In [121]:

```
gray.shape
```

Out[121]:

```
(1390, 1300)
```

In [122]:

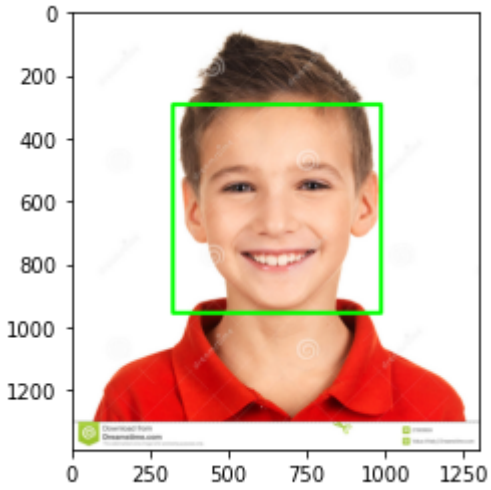
```
faces = faceCascade.detectMultiScale(gray,1.1,4)
for x,y,w,h in faces :
    roi_gray =gray[y:y+h , x:x+w]
    roi_color =frame[y:y+h , x:x+w]
    cv2.rectangle(frame,(x,y) , (x+w,y+h) , (0,255,0), 12)
    faces = faceCascade.detectMultiScale(roi_gray)
    if len(faces) == 0:
        print("face not detected")
    else:
        for (ex,ey,ew,eh) in faces:
            face_roi = roi_color[ey: ey+eh , ex:ex+ew]
```

In [123]:

```
plt.imshow(cv2.cvtColor(frame , cv2.COLOR_BGR2RGB))
```

Out[123]:

```
<matplotlib.image.AxesImage at 0x24da2dd7e80>
```

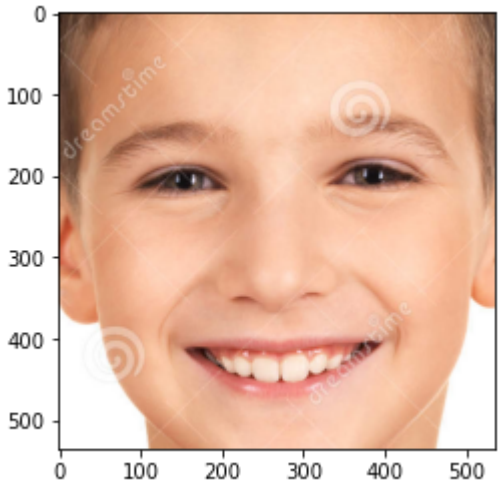


In [124]:

```
plt.imshow(cv2.cvtColor(face_roi , cv2.COLOR_BGR2RGB))
```

Out[124]:

```
<matplotlib.image.AxesImage at 0x24da31d07c0>
```



In [125]:

```
final_image = cv2.resize(face_roi ,(224,224))
final_image = np.expand_dims(final_image,axis =0)
```

In [126]:

```
pred = new_model.predict(final_image)
```

In [127]:

```
pred[0]
```

Out[127]:

```
array([2.2194536e-06, 3.8759610e-11, 3.6978581e-05, 9.9695086e-01,
       2.9643385e-03, 9.2197815e-06, 3.6385667e-05], dtype=float32)
```

In [128]:

```
np.argmax(pred)
```

Out[128]:

```
3
```

In [ ]:

```
import cv2
path = "haarcascade_frontalface_default.xml"
font_scale = 3.5
font = cv2.FONT_HERSHEY_PLAIN

rectangle_bgr = (255,255,255)
img = np.zeros((500,500))

text = "some text in a box ! "
(text_width,text_height) = cv2.getTextSize(text,font,fontScale = font_scale,thickness =  1)

text_offset_x = 10
text_offset_y = img.shape[0] - 25
box_coords = ((text_offset_x,text_offset_y),(text_offset_x + text_width+2,text_offset_y - t
cv2.rectangle(img ,box_coords[0],box_coords[1],rectangle_bgr , cv2.FILLED)
cv2.putText(img , text,(text_offset_x,text_offset_y),font,fontScale = font_scale,color = (0
cap = cv2.VideoCapture(1)

if not cap.isOpened():
    cap = cv2.VideoCapture(0)
if not cap.isOpened():
    raise IOErrror(" cannot open webcam ")

while True:
    ret,frame = cap.read()
    faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_de
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = faceCascade.detectMultiScale(gray,1.1,4)


    for x,y,w,h in faces :
        roi_gray =gray[y:y+h , x:x+w]
        roi_color =frame[y:y+h , x:x+w]
        cv2.rectangle(frame,(x,y) , (x+w,y+h) , (225,0,0), 2)
        faces = faceCascade.detectMultiScale(roi_gray)
        if len(faces) == 0:
            print("face not detected")
        else:
            for (ex,ey,ew,eh) in faces:
                face_roi = roi_color[ey: ey+eh , ex:ex+ew]
    final_image = cv2.resize(face_roi ,(224,224))
    final_image = np.expand_dims(final_image,axis =0)

    font = cv2.FONT_HERSHEY_SIMPLEX

    pred = new_model.predict(final_image)
    font_scale = 1.5
    font = cv2.FONT_HERSHEY_PLAIN

    if(np.argmax(pred)==0):
        status = "ANGRY"
        x1,y1,w1,h1 = 0,0,175,75
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10) , y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPL
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))
    elif(np.argmax(pred)==1):
        status = "DISGUST"
        x1,y1,w1,h1 = 0,0,175,75
```

```
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))
    elif(np.argmax(pred)==2):
        status = "FEAR"
        x1,y1,w1,h1 = 0,0,175,75
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))
    elif(np.argmax(pred)==3):
        status = "HAPPY"
        x1,y1,w1,h1 = 0,0,175,75
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))

    elif(np.argmax(pred)==4):
        status = "NEUTRAL"
        x1,y1,w1,h1 = 0,0,175,75
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))
    elif(np.argmax(pred)==5):
        status = "SAD"
        x1,y1,w1,h1 = 0,0,175,75
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))
    else:
        status = "NEUTRAL"
        x1,y1,w1,h1 = 0,0,175,75
        cv2.rectangle(frame , (x1,x1),(x1+w1 , y1+h1),(0,0,0),-1)
        cv2.putText(frame , status ,(x1 + int(w1/10),y1+int(h1/2)),cv2.FONT_HERSHEY_SIMPLEX
        cv2.putText(frame,status , (100,150),font , 3,(0,0,255),2,cv2.LINE_4)
        cv2.rectangle(frame , (x,y), (x+w , y+h), (0,0,255))




    cv2.imshow("facial expression recognition",frame)
    if cv2.waitKey(2) & 0xFF == ord('q'):
        break



cap.release()
cv2.destroyAllWindows()
```

In [ ]:

```
```

In [ ]: