facial-expression

May 20, 2024

[5]:

```
from keras.utils import to_categorical
from keras_preprocessing.image import load_img
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
import os
import pandas as pd
import numpy as np
```

[6]:

```
pip install keras_preprocessing
```

```
Defaulting to user installation because normal site-packages is not
writeableNote: you may need to restart the kernel to use updated packages.

Requirement already satisfied: keras_preprocessing in
c:\users\riyau\appdata\roaming\python\python311\site-packages (1.1.2)
Requirement already satisfied: numpy>=1.9.1 in
c:\programdata\anaconda3\lib\site-packages (from keras_preprocessing) (1.24.3)
Requirement already satisfied: six>=1.9.0 in c:\programdata\anaconda3\lib\site-
packages (from keras_preprocessing) (1.16.0)
```

[7]:

```
TRAIN_DTR = "C:/Users/riyau/Downloads/archive (3)/images/train"
TEST_DTR = "C:/Users/riyau/Downloads/archive (3)/images/test"
```

[8]:

```
def createdataframe(dir):
    image_paths = []
    labels = []
    for label in os.listdir(dir):
        for imagename in os.listdir(os.path.join(dir,label)):
            image_paths.append(os.path.join(dir,label,imagename))
            labels.append(label)
        print(label,"completed")
    return image_paths,labels
```

[9]:

```
train = pd.DataFrame()
train['image'], train["labels"] = createdataframe(TRAIN_DTR)
```

```
angry completed
disgust completed
```

```
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

[10]:

```
!pip install pandas
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: pandas in c:\programdata\anaconda3\lib\site-
packages (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in
c:\programdata\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
c:\programdata\anaconda3\lib\site-packages (from pandas) (2022.7)
Requirement already satisfied: numpy>=1.21.0 in
c:\programdata\anaconda3\lib\site-packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\lib\site-
packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
```

[11]:

```
print(train)
```

```
                                                   image    labels
0      C:/Users/riyau/Downloads/archive (3)/images/tr…     angry
1      C:/Users/riyau/Downloads/archive (3)/images/tr…     angry
2      C:/Users/riyau/Downloads/archive (3)/images/tr…     angry
3      C:/Users/riyau/Downloads/archive (3)/images/tr…     angry
4      C:/Users/riyau/Downloads/archive (3)/images/tr…     angry
…                                                    …        …
28816  C:/Users/riyau/Downloads/archive (3)/images/tr…  surprise
28817  C:/Users/riyau/Downloads/archive (3)/images/tr…  surprise
28818  C:/Users/riyau/Downloads/archive (3)/images/tr…  surprise
28819  C:/Users/riyau/Downloads/archive (3)/images/tr…  surprise
28820  C:/Users/riyau/Downloads/archive (3)/images/tr…  surprise

[28821 rows x 2 columns]
```

[12]:

```
test = pd.DataFrame()
test['image'], test["labels"] = createdataframe(TEST_DTR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

[13]:
```
print(test)
print(test['image'])
```

```
                                                 image    labels
0     C:/Users/riyau/Downloads/archive (3)/images/te…     angry
1     C:/Users/riyau/Downloads/archive (3)/images/te…     angry
2     C:/Users/riyau/Downloads/archive (3)/images/te…     angry
3     C:/Users/riyau/Downloads/archive (3)/images/te…     angry
4     C:/Users/riyau/Downloads/archive (3)/images/te…     angry
…                                                  …       …
7061  C:/Users/riyau/Downloads/archive (3)/images/te…  surprise
7062  C:/Users/riyau/Downloads/archive (3)/images/te…  surprise
7063  C:/Users/riyau/Downloads/archive (3)/images/te…  surprise
7064  C:/Users/riyau/Downloads/archive (3)/images/te…  surprise
7065  C:/Users/riyau/Downloads/archive (3)/images/te…  surprise

[7066 rows x 2 columns]
0       C:/Users/riyau/Downloads/archive (3)/images/te…
1       C:/Users/riyau/Downloads/archive (3)/images/te…
2       C:/Users/riyau/Downloads/archive (3)/images/te…
3       C:/Users/riyau/Downloads/archive (3)/images/te…
4       C:/Users/riyau/Downloads/archive (3)/images/te…
                              …
7061    C:/Users/riyau/Downloads/archive (3)/images/te…
7062    C:/Users/riyau/Downloads/archive (3)/images/te…
7063    C:/Users/riyau/Downloads/archive (3)/images/te…
7064    C:/Users/riyau/Downloads/archive (3)/images/te…
7065    C:/Users/riyau/Downloads/archive (3)/images/te…
Name: image, Length: 7066, dtype: object
```

[14]:
```
from tqdm.notebook import tqdm
```

[15]:
```
def extract_features(images):
    features = []
    for image in tqdm(images):
        img = load_img(image,grayscale = True)
        img = np.array(img)
        features.append(img)
    features = np.array(features)
    features = features.reshape(len(features),48,48,1)
    return features
```

[16]:
```
train_features = extract_features(train['image'])
```

```
  0%|          | 0/28821 [00:00<?, ?it/s]
```

```
C:\Users\riyau\AppData\Roaming\Python\Python311\site-
packages\keras_preprocessing\image\utils.py:107: UserWarning: grayscale is
```

```
deprecated. Please use color_mode = "grayscale"
  warnings.warn('grayscale is deprecated. Please use '
```

[17]:
```
test_features = extract_features(test['image'])
```

```
  0%|          | 0/7066 [00:00<?, ?it/s]
```

[18]:
```
x_train = train_features/255.0   #pixels highest value 255
x_test = test_features/255.0
```

[19]:
```
pip install -U scikit-learn
```

```
Defaulting to user installation because normal site-packages is not
writeableNote: you may need to restart the kernel to use updated packages.

Requirement already satisfied: scikit-learn in
c:\users\riyau\appdata\roaming\python\python311\site-packages (1.3.0)
Requirement already satisfied: numpy>=1.17.3 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.24.3)
Requirement already satisfied: scipy>=1.5.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.10.1)
Requirement already satisfied: joblib>=1.1.1 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
c:\programdata\anaconda3\lib\site-packages (from scikit-learn) (2.2.0)
```

[20]:
```
from sklearn.preprocessing import LabelEncoder
```

[21]:
```
le = LabelEncoder()
le.fit(train['labels'])
```

[21]:
```
LabelEncoder()
```

[36]:
```
y_train = le.transform(train['labels'])
y_test = le.transform(test['labels'])
```

[37]:
```
y_train = to_categorical(y_train,num_classes=7)
y_test = to_categorical(y_test,num_classes=7)
```

[38]:
```
model = Sequential()
model.add(Conv2D(128,kernel_size=(3,3),activation='relu',␣
  ↪input_shape=(48,48,1)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(256,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))
model.add(Conv2D(512,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512,kernel_size=(3,3),activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))




model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256,activation='relu'))
model.add(Dropout(0.3))

#output layer
model.add(Dense(7,activation='softmax'))
```

[39]:
```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics =␣
 ↪'accuracy')
```

[40]:
```
model.fit(x=x_train,y=y_train,batch_size=128,epochs=100,validation_data =␣
 ↪(x_test,y_test))
```

```
Epoch 1/100
226/226 [==============================] - 902s 4s/step - loss: 1.8185 -
accuracy: 0.2430 - val_loss: 1.7877 - val_accuracy: 0.2569
Epoch 2/100
226/226 [==============================] - 915s 4s/step - loss: 1.7635 -
accuracy: 0.2659 - val_loss: 1.6905 - val_accuracy: 0.3051
Epoch 3/100
226/226 [==============================] - 1060s 5s/step - loss: 1.6346 -
accuracy: 0.3448 - val_loss: 1.4826 - val_accuracy: 0.4406
Epoch 4/100
226/226 [==============================] - 905s 4s/step - loss: 1.4985 -
accuracy: 0.4133 - val_loss: 1.4126 - val_accuracy: 0.4694
Epoch 5/100
226/226 [==============================] - 864s 4s/step - loss: 1.4276 -
accuracy: 0.4454 - val_loss: 1.2950 - val_accuracy: 0.5105
Epoch 6/100
226/226 [==============================] - 859s 4s/step - loss: 1.3817 -
accuracy: 0.4649 - val_loss: 1.2656 - val_accuracy: 0.5130
Epoch 7/100
226/226 [==============================] - 877s 4s/step - loss: 1.3473 -
```

```
accuracy: 0.4827 - val_loss: 1.2326 - val_accuracy: 0.5255
Epoch 8/100
226/226 [==============================] - 1484s 7s/step - loss: 1.3159 -
accuracy: 0.4960 - val_loss: 1.1937 - val_accuracy: 0.5463
Epoch 9/100
226/226 [==============================] - 1894s 8s/step - loss: 1.2934 -
accuracy: 0.5036 - val_loss: 1.1876 - val_accuracy: 0.5449
Epoch 10/100
226/226 [==============================] - 870s 4s/step - loss: 1.2630 -
accuracy: 0.5201 - val_loss: 1.1683 - val_accuracy: 0.5517
Epoch 11/100
226/226 [==============================] - 923s 4s/step - loss: 1.2502 -
accuracy: 0.5234 - val_loss: 1.1757 - val_accuracy: 0.5521
Epoch 12/100
226/226 [==============================] - 889s 4s/step - loss: 1.2381 -
accuracy: 0.5276 - val_loss: 1.1478 - val_accuracy: 0.5607
Epoch 13/100
226/226 [==============================] - 970s 4s/step - loss: 1.2172 -
accuracy: 0.5361 - val_loss: 1.1796 - val_accuracy: 0.5528
Epoch 14/100
226/226 [==============================] - 845s 4s/step - loss: 1.2086 -
accuracy: 0.5424 - val_loss: 1.1196 - val_accuracy: 0.5787
Epoch 15/100
169/226 [=====================>…] - ETA: 3:28 - loss: 1.1976 - accuracy:
0.5467
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[40], line 1
----> 1 model.fit(x=x_train,y=y_train,batch_size=128,epochs=100,validation_data
  ↪= (x_test,y_test))

File␣
  ↪~\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\traceback_utils.
  ↪py:65, in filter_traceback.<locals>.error_handler(*args, **kwargs)
     63 filtered_tb = None
     64 try:
---> 65     return fn(*args, **kwargs)
     66 except Exception as e:
     67     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\training
  ↪py:1742, in Model.fit(self, x, y, batch_size, epochs, verbose, callbacks,␣
  ↪validation_split, validation_data, shuffle, class_weight, sample_weight,␣
  ↪initial_epoch, steps_per_epoch, validation_steps, validation_batch_size,␣
  ↪validation_freq, max_queue_size, workers, use_multiprocessing)
   1734 with tf.profiler.experimental.Trace(
   1735     "train",
   1736     epoch_num=epoch,
  (…)
  1739     _r=1,
  1740 ):
  1741     callbacks.on_train_batch_begin(step)
-> 1742     tmp_logs = self.train_function(iterator)
  1743     if data_handler.should_sync:
  1744         context.async_wait()

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\util\traceback_utils.
 ↪py:150, in filter_traceback.<locals>.error_handler(*args, **kwargs)
    148 filtered_tb = None
    149 try:
--> 150   return fn(*args, **kwargs)
    151 except Exception as e:
    152   filtered_tb = _process_traceback_frames(e.__traceback__)

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\polymorphic_funct
 ↪py:825, in Function.__call__(self, *args, **kwds)
    822 compiler = "xla" if self._jit_compile else "nonXla"
    824 with OptionalXlaContext(self._jit_compile):
--> 825   result = self._call(*args, **kwds)
    827 new_tracing_count = self.experimental_get_tracing_count()
    828 without_tracing = (tracing_count == new_tracing_count)

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\polymorphic_funct
 ↪py:857, in Function._call(self, *args, **kwds)
    854   self._lock.release()
    855   # In this case we have created variables on the first call, so we run␣
 ↪the
    856   # defunned version which is guaranteed to never create variables.
--> 857   return self._no_variable_creation_fn(*args, **kwds)  # pylint:␣
 ↪disable=not-callable
    858 elif self._variable_creation_fn is not None:
    859   # Release the lock early so that multiple threads can perform the call
    860   # in parallel.
    861   self._lock.release()

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\polymorphic_funct
 ↪py:148, in TracingCompiler.__call__(self, *args, **kwargs)
    145 with self._lock:
    146   (concrete_function,
    147    filtered_flat_args) = self._maybe_define_function(args, kwargs)
--> 148 return concrete_function._call_flat(
    149     filtered_flat_args, captured_inputs=concrete_function.
 ↪captured_inputs)
```

```
File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\polymorphic_functi
 ↪py:1349, in ConcreteFunction._call_flat(self, args, captured_inputs)
   1345 possible_gradient_type = gradients_util.PossibleTapeGradientTypes(args)
   1346 if (possible_gradient_type == gradients_util.POSSIBLE_GRADIENT_TYPES_NONE
   1347     and executing_eagerly):
   1348   # No tape is watching; skip to running the function.
-> 1349   return self._build_call_outputs(self._inference_function(*args))
   1350 forward_backward = self._select_forward_and_backward_functions(
   1351     args,
   1352     possible_gradient_type,
   1353     executing_eagerly)
   1354 forward_function, args_with_tangents = forward_backward.forward()

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\polymorphic_functi
 ↪py:196, in AtomicFunction.__call__(self, *args)
    194 with record.stop_recording():
    195   if self._bound_context.executing_eagerly():
--> 196     outputs = self._bound_context.call_function(
    197         self.name,
    198         list(args),
    199         len(self.function_type.flat_outputs),
    200     )
    201   else:
    202     outputs = make_call_op_in_graph(self, list(args))

File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\context.
 ↪py:1457, in Context.call_function(self, name, tensor_inputs, num_outputs)
   1455 cancellation_context = cancellation.context()
   1456 if cancellation_context is None:
-> 1457   outputs = execute.execute(
   1458       name.decode("utf-8"),
   1459       num_outputs=num_outputs,
   1460       inputs=tensor_inputs,
   1461       attrs=attrs,
   1462       ctx=self,
   1463   )
   1464 else:
   1465   outputs = execute.execute_with_cancellation(
   1466       name.decode("utf-8"),
   1467       num_outputs=num_outputs,
   (…)
   1471       cancellation_manager=cancellation_context,
   1472   )
```

```
File␣
 ↪~\AppData\Roaming\Python\Python311\site-packages\tensorflow\python\eager\execute.
 ↪py:53, in quick_execute(op_name, num_outputs, inputs, attrs, ctx, name)
     51 try:
     52   ctx.ensure_initialized()
---> 53   tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name
     54                                       inputs, attrs, num_outputs)
     55 except core._NotOkStatusException as e:
     56   if name is not None:

KeyboardInterrupt:
```

[54]:

```python
import json
from tensorflow.keras.models import model_from_json

# Assuming you have already defined the 'model' object

# Convert the model architecture to JSON format
model_json = model.to_json()

# Define the name of the JSON file
json_file_name = "emotionaldetector.json"

# Write the model JSON to the file 'emotionaldetector.json'
with open(json_file_name, 'w') as json_file:
    json_file.write(model_json)

# Save the model weights to an h5 file
model.save('emotionaldetector.h5')

# Now, to load the model back with the saved JSON architecture and weights
# Load the model architecture from the JSON file
with open(json_file_name, 'r') as json_file:
    loaded_model_json = json_file.read()

# Create a new model with the loaded JSON architecture
loaded_model = model_from_json(loaded_model_json)

# Load the model weights from the h5 file
loaded_model.load_weights('emotionaldetector.h5')
```

```
C:\Users\riyau\AppData\Roaming\Python\Python311\site-
packages\keras\src\engine\training.py:3000: UserWarning: You are saving your
model as an HDF5 file via `model.save()`. This file format is considered legacy.
We recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')`.
  saving_api.save_model(
```

[55]:

```
from keras.models import model_from_json
```

[56]:

```
import json
from tensorflow.keras.models import model_from_json

# Assuming you have already defined the 'model' object

# Define the name of the JSON file
json_file_name = "emotionaldetector.json"

# Read the model architecture from the existing JSON file
with open(json_file_name, 'r') as json_file:
    model_json = json_file.read()

# Load the model architecture from the JSON data
model = model_from_json(model_json)

# Load the model weights from the h5 file
model.load_weights('emotionaldetector.h5')
```

[59]:

```
label = ['angry','disgust','fear','happy','neutral','sad','surprise']
```

[60]:

```
def ef(image):
    img = load_img(image,grayscale = True)
    feature = np.array(img)
    feature = feature.reshape(1,48,48,1)
    return feature/255.0
```

[70]:

```
import matplotlib.pyplot as plt
%matplotlib inline
```

[ ]:

```

```

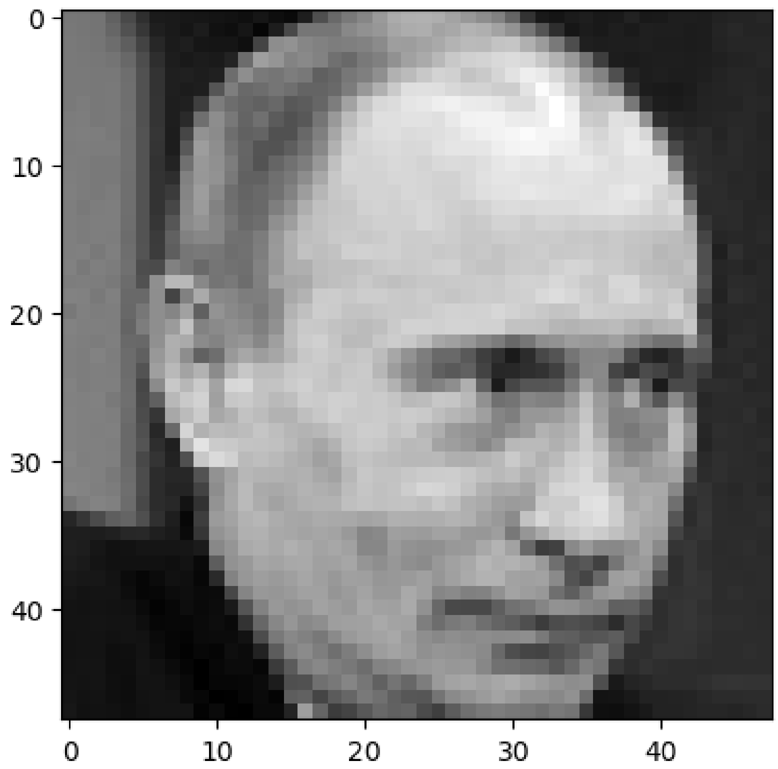
[74]:

```
image = 'C:/Users/riyau/Downloads/archive (3)/images/test/angry/65.jpg'
print('original image is of angry')
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print('model prediction is', pred_label)
plt.imshow(img.reshape(48,48) ,cmap='gray')
```

```
original image is of angry
1/1 [==============================] - 1s 1s/step
model prediction is angry
```

[74]:

```
<matplotlib.image.AxesImage at 0x1b9f8029210>
```



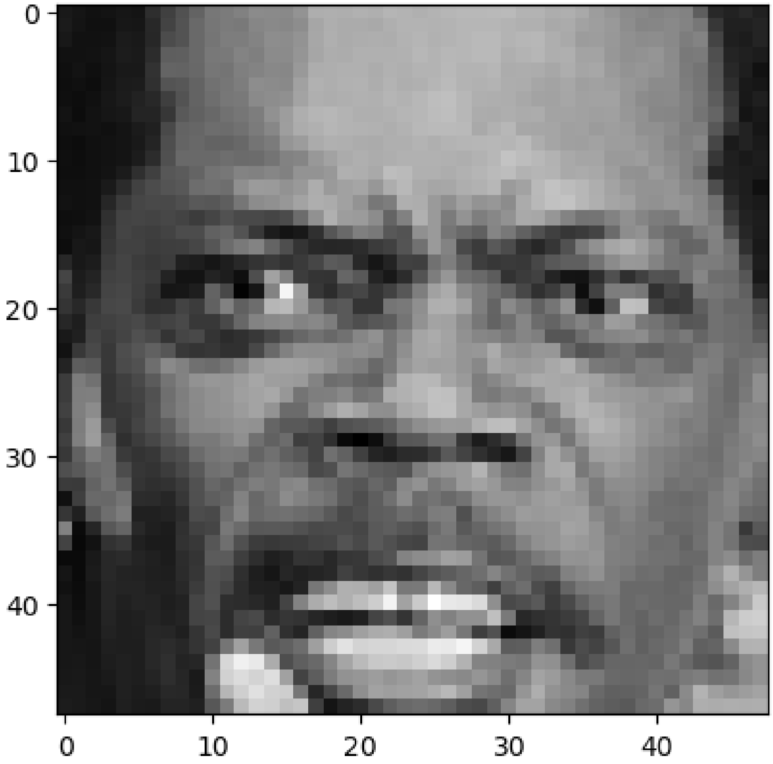
[80]:

```
image = 'C:/Users/riyau/Downloads/archive (3)/images/train/angry/10.jpg'
print('original image is of angry')
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print('model prediction is', pred_label)
plt.imshow(img.reshape(48,48) ,cmap='gray')
```

```
original image is of angry
1/1 [==============================] - 0s 244ms/step
model prediction is angry
```

[80]:

```
<matplotlib.image.AxesImage at 0x1b9fad07090>
```



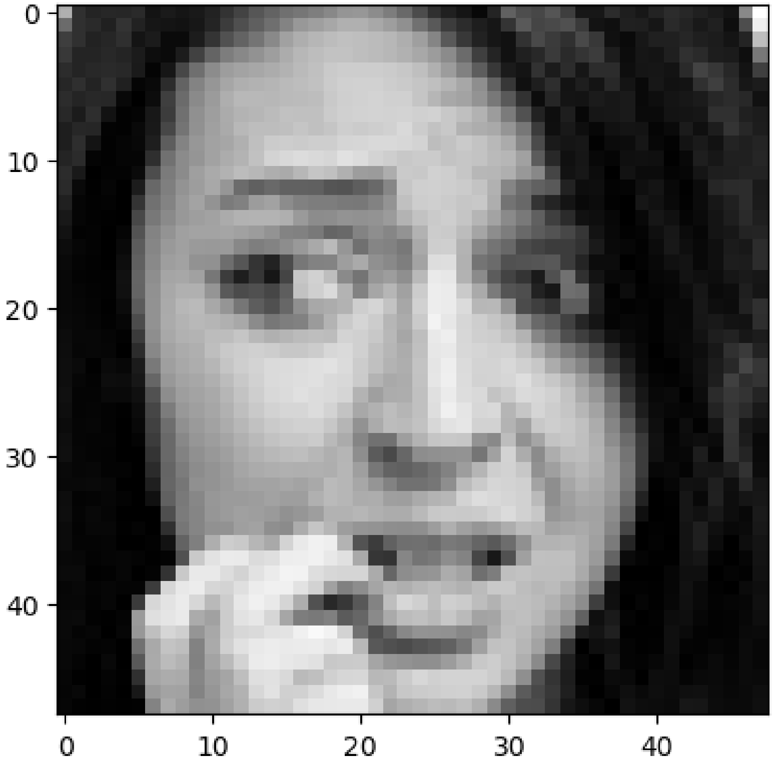
[82]:

```
image = 'C:/Users/riyau/Downloads/archive (3)/images/test/fear/21.jpg'
print('original image is of fear')
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print('model prediction is', pred_label)
plt.imshow(img.reshape(48,48) ,cmap='gray')
```

```
original image is of fear
1/1 [==============================] - 0s 37ms/step
model prediction is fear
```

[82]:

```
<matplotlib.image.AxesImage at 0x1b9fae43790>
```


0
10
20
30
40
0
10
20
30
40

[ ]: