New Section

```
from google.colab import files

uploaded = files.upload()

for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(
      name=fn, length=len(uploaded[fn])))
```

Choose Files archive (1).zip

- **archive (1).zip**(application/x-zip-compressed) - 63252113 bytes, last modified: 7/30/2023 - 100% done

```
Saving archive (1).zip to archive (1) (1).zip
User uploaded file "archive (1) (1).zip" with length 63252113 bytes
```

```
from zipfile import ZipFile
file_name = "archive (1).zip"

with ZipFile(file_name, 'r') as zip:
  zip.extractall()
  print("Done")
```

```
Done
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
import numpy as np
import cv2
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.optimizers import Adam
from keras.layers import MaxPooling2D
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
```

```
train_dir = 'train'
val_dir = 'test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=64,
```

```
        color_mode="grayscale",
        class_mode='categorical')
```

```
    Found 28709 images belonging to 7 classes.
    Found 7178 images belonging to 7 classes.
```

```
emotion_model = Sequential()
emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
```

```
emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001, decay=1e-6),metrics=['accurac
emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=60,
        validation_data=validation_generator,
        validation_steps=7178 // 64)
```

```
    <ipython-input-7-e3e2c1d7010d>:2: UserWarning: `Model.fit_generator` is deprecated and will be remov
      emotion_model_info = emotion_model.fit_generator(
    Epoch 1/60
    448/448 [==============================] - 17s 34ms/step - loss: 1.6249 - accuracy: 0.3670 - val_los
    Epoch 2/60
    448/448 [==============================] - 14s 30ms/step - loss: 1.5327 - accuracy: 0.4125 - val_los
    Epoch 3/60
    448/448 [==============================] - 14s 32ms/step - loss: 1.4633 - accuracy: 0.4383 - val_los
    Epoch 4/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.4102 - accuracy: 0.4603 - val_los
    Epoch 5/60
    448/448 [==============================] - 14s 32ms/step - loss: 1.3584 - accuracy: 0.4829 - val_los
    Epoch 6/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.3113 - accuracy: 0.5049 - val_los
    Epoch 7/60
    448/448 [==============================] - 15s 32ms/step - loss: 1.2683 - accuracy: 0.5204 - val_los
    Epoch 8/60
    448/448 [==============================] - 14s 32ms/step - loss: 1.2377 - accuracy: 0.5329 - val_los
    Epoch 9/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.2003 - accuracy: 0.5484 - val_los
    Epoch 10/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.1723 - accuracy: 0.5611 - val_los
    Epoch 11/60
    448/448 [==============================] - 14s 32ms/step - loss: 1.1466 - accuracy: 0.5712 - val_los
    Epoch 12/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.1198 - accuracy: 0.5817 - val_los
    Epoch 13/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.0913 - accuracy: 0.5939 - val_los
    Epoch 14/60
    448/448 [==============================] - 14s 31ms/step - loss: 1.0724 - accuracy: 0.6012 - val_los
    Epoch 15/60
```

```
448/448 [==============================] - 14s 32ms/step - loss: 1.0448 - accuracy: 0.6099 - val_los
Epoch 16/60
448/448 [==============================] - 14s 32ms/step - loss: 1.0221 - accuracy: 0.6237 - val_los
Epoch 17/60
448/448 [==============================] - 14s 32ms/step - loss: 0.9951 - accuracy: 0.6317 - val_los
Epoch 18/60
448/448 [==============================] - 14s 31ms/step - loss: 0.9778 - accuracy: 0.6384 - val_los
Epoch 19/60
448/448 [==============================] - 14s 30ms/step - loss: 0.9486 - accuracy: 0.6515 - val_los
Epoch 20/60
448/448 [==============================] - 13s 30ms/step - loss: 0.9314 - accuracy: 0.6569 - val_los
Epoch 21/60
448/448 [==============================] - 14s 31ms/step - loss: 0.9065 - accuracy: 0.6655 - val_los
Epoch 22/60
448/448 [==============================] - 14s 30ms/step - loss: 0.8824 - accuracy: 0.6772 - val_los
Epoch 23/60
448/448 [==============================] - 13s 30ms/step - loss: 0.8615 - accuracy: 0.6811 - val_los
Epoch 24/60
448/448 [==============================] - 14s 32ms/step - loss: 0.8379 - accuracy: 0.6961 - val_los
Epoch 25/60
448/448 [==============================] - 17s 38ms/step - loss: 0.8156 - accuracy: 0.7010 - val_los
Epoch 26/60
448/448 [==============================] - 14s 32ms/step - loss: 0.7936 - accuracy: 0.7110 - val_los
Epoch 27/60
448/448 [==============================] - 14s 31ms/step - loss: 0.7760 - accuracy: 0.7191 - val_los
```

```
emotion_model.save('model.h5')
```

```
from keras.models import load_model
emotion_model = load_model('model.h5')
```

```
def emotion_analysis(emotions):
    objects = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
    y_pos = np.arange(len(objects))

    plt.bar(y_pos, emotions, align='center', alpha=0.5)
    plt.xticks(y_pos, objects)
    plt.ylabel('percentage')
    plt.title('emotion')

    plt.show()
```

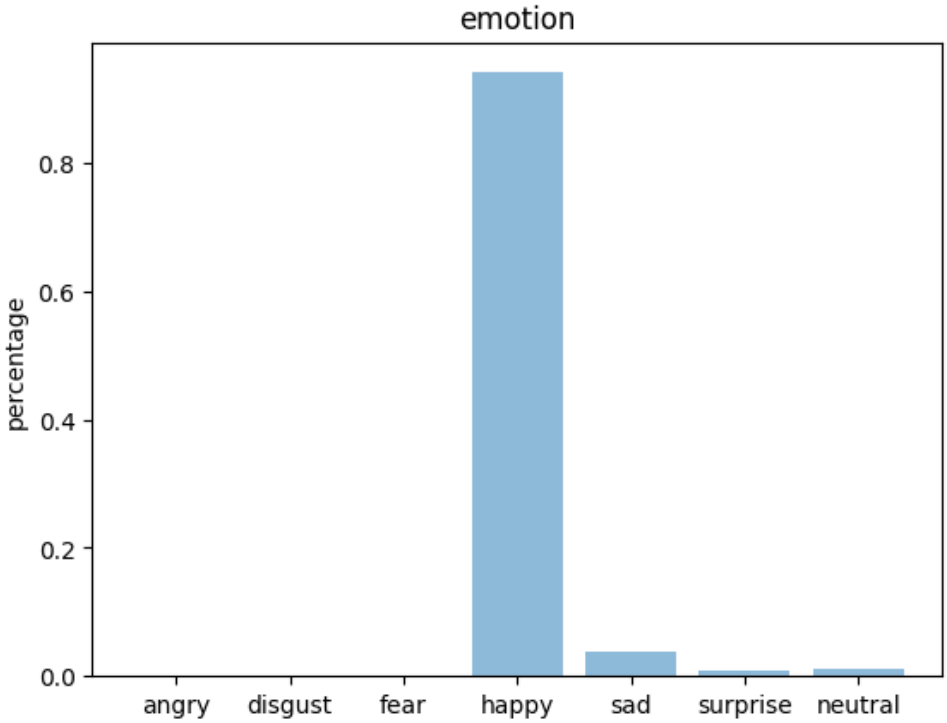
```
from IPython.display import display, Javascript
from google.colab.output import eval_js
from base64 import b64decode

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video: true});

      document.body.appendChild(div);
```

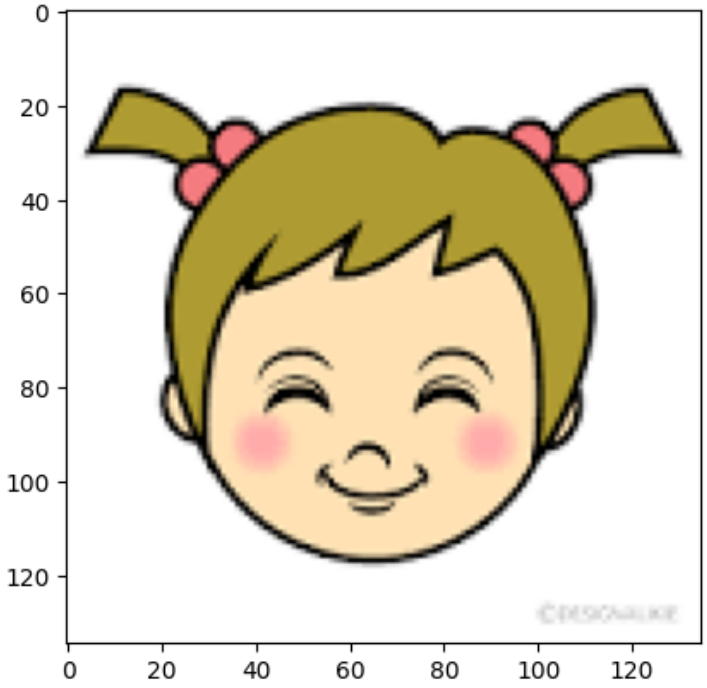
```
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.
      google.colab.output.setIframeHeight(document.documentElement.scrollHeight, true);

      // Wait for Capture to be clicked.
      await new Promise((resolve) => capture.onclick = resolve);

      const canvas = document.createElement('canvas');
      canvas.width = video.videoWidth;
      canvas.height = video.videoHeight;
      canvas.getContext('2d').drawImage(video, 0, 0);
      stream.getVideoTracks()[0].stop();
      div.remove();
      return canvas.toDataURL('image/jpeg', quality);
    }
    ''')
  display(js)
  data = eval_js('takePhoto({})'.format(quality))
  binary = b64decode(data.split(',')[1])
  with open(filename, 'wb') as f:
    f.write(binary)
  return filename


take_photo()
```

```
'photo.jpg'
```

```
from tensorflow.keras.utils import load_img
from keras.models import load_model
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.utils import load_img
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.utils import img_to_array
import numpy as np
import matplotlib.pyplot as plt
file = '/content/images (1).png'
true_image =load_img(file)
img =load_img(file, color_mode="grayscale", target_size=(48, 48))
x =img_to_array(img)
x = np.expand_dims(x, axis = 0)
x /= 255

custom = emotion_model.predict(x)
emotion_analysis(custom[0])

x = np.array(x, 'float32')
x = x.reshape([48, 48]);


plt.imshow(true_image)
plt.show()
```

```
1/1 [==============================] - 0s 21ms/step
```





✓ 1s completed at 6:20 PM