Spotle AI thon 2020 Face Emotion Detection

September 13, 2020

1 Introduction

1.1 About

Team : The Elite

- Vidhya Subramaniam
- Devleena Banerjee
- Priyank jha
- Karthik chiranjeevi

1.2 Objective

Building A Mood Classifier Based On Facial Expressions - With AI and Computer Vision

The objective of this study is to classify mood of the person from facial expressions Images are categorized in three classes namely sadness, fear and happiness based on the emotion shown in the facial expressions .

1.3 Dataset

The data consists of 48x48 pixel grayscale images of faces. The pixel values are stored in 2304 (48*48) columns. These column names start with pixel. Along with pixel values, there is emotion column that say about mood of the image.

The task is to categorize each face based on the emotion shown in the facial expression in to one of three categories.

Along with pixel values, aithon2020_level2_traning.csv dataset contains another column emotion that say about mood that is present in the image. This is the dataset you will use to train your model.

The final test set, which will be used to determine the winner of the competition, will be published later.

https://spotleai.sgp1.digitaloceanspaces.com/course/zip/aithon2020-level-2.zip

The training dataset is stored inside the data folder.

1.4 Rules of submission

- You have to submit folder after zipped it.

- The name of the folder is aithon2020_yourname_or_teamname
- The folder must not contain any dataset.
- If you have used any python libraries other than in default list, you have to mention in requirements.txt file.
- There must be aithon_level2.py file inside the folder.
- And, a function name 'aithon_level2_api' must be defined inside the file.
- The function will be called to train and test your program.
- The function name, signature and output type is fixed.
- The first argument is file name that contains data for training.
- The second argument is file name that contains data for test.
- The function must return predicted value or emotion for each data in test dataset sequentially in a list. For example, ['sad', 'happy', 'fear', 'fear', ... , 'happy']

By default following, Python libraries are installed in test environment

- Numpy
- Pandas
- OpneCV for Python
- TensorFlows
- Keras
- PyTorch
- Scikit-learn
- Theano
- Matplotlib
- Seaborn

2 Importing packages and data

2.1 Import necessary packages

[1]:
```
!pip install livelossplot
```

```
Collecting livelossplot
  Downloading https://files.pythonhosted.org/packages/0f/08/1884157a3de72d41fa97
cacacafaa49abf00eba53cb7e08615b2b65b4a9d/livelossplot-0.5.3-py3-none-any.whl
Requirement already satisfied: matplotlib; python_version >= "3.6" in
/usr/local/lib/python3.6/dist-packages (from livelossplot) (3.2.2)
Requirement already satisfied: bokeh; python_version >= "3.6" in
/usr/local/lib/python3.6/dist-packages (from livelossplot) (2.1.1)
Requirement already satisfied: ipython in /usr/local/lib/python3.6/dist-packages
(from livelossplot) (5.5.0)
Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib; python_version >=
"3.6"->livelossplot) (2.8.1)
```

```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-
packages (from matplotlib; python_version >= "3.6"->livelossplot) (0.10.0)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.6/dist-
packages (from matplotlib; python_version >= "3.6"->livelossplot) (1.18.5)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib; python_version >=
"3.6"->livelossplot) (1.2.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
/usr/local/lib/python3.6/dist-packages (from matplotlib; python_version >=
"3.6"->livelossplot) (2.4.7)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.6/dist-
packages (from bokeh; python_version >= "3.6"->livelossplot) (20.4)
Requirement already satisfied: pillow>=4.0 in /usr/local/lib/python3.6/dist-
packages (from bokeh; python_version >= "3.6"->livelossplot) (7.0.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.6/dist-
packages (from bokeh; python_version >= "3.6"->livelossplot) (3.13)
Requirement already satisfied: typing-extensions>=3.7.4 in
/usr/local/lib/python3.6/dist-packages (from bokeh; python_version >=
"3.6"->livelossplot) (3.7.4.3)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.6/dist-
packages (from bokeh; python_version >= "3.6"->livelossplot) (5.1.1)
Requirement already satisfied: Jinja2>=2.7 in /usr/local/lib/python3.6/dist-
packages (from bokeh; python_version >= "3.6"->livelossplot) (2.11.2)
Requirement already satisfied: decorator in /usr/local/lib/python3.6/dist-
packages (from ipython->livelossplot) (4.4.2)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in
/usr/local/lib/python3.6/dist-packages (from ipython->livelossplot) (1.0.18)
Requirement already satisfied: pexpect; sys_platform != "win32" in
/usr/local/lib/python3.6/dist-packages (from ipython->livelossplot) (4.8.0)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.6/dist-
packages (from ipython->livelossplot) (0.7.5)
Requirement already satisfied: pygments in /usr/local/lib/python3.6/dist-
packages (from ipython->livelossplot) (2.1.3)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.6/dist-
packages (from ipython->livelossplot) (4.3.3)
Requirement already satisfied: simplegeneric>0.8 in
/usr/local/lib/python3.6/dist-packages (from ipython->livelossplot) (0.8.1)
Requirement already satisfied: setuptools>=18.5 in
/usr/local/lib/python3.6/dist-packages (from ipython->livelossplot) (49.6.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.6/dist-
packages (from python-dateutil>=2.1->matplotlib; python_version >=
"3.6"->livelossplot) (1.15.0)
Requirement already satisfied: MarkupSafe>=0.23 in
/usr/local/lib/python3.6/dist-packages (from Jinja2>=2.7->bokeh; python_version
>= "3.6"->livelossplot) (1.1.1)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.6/dist-packages
(from prompt-toolkit<2.0.0,>=1.0.4->ipython->livelossplot) (0.2.5)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.6/dist-
```

```
packages (from pexpect; sys_platform != "win32"->ipython->livelossplot) (0.6.0)
Requirement already satisfied: ipython-genutils in
/usr/local/lib/python3.6/dist-packages (from
traitlets>=4.2->ipython->livelossplot) (0.2.0)
Installing collected packages: livelossplot
Successfully installed livelossplot-0.5.3
```

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import cv2 as cv
import os
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import tensorflow as tf
from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D,␣
↪Dropout, BatchNormalization, Activation
from tensorflow.keras import Model,Sequential
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import plot_model
from livelossplot import PlotLossesKeras
from tensorflow.keras.models import load_model
from google.colab import files
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

2.2 Downloading the data

Downloading dataset.zip

[3]:
```
!wget https://spotleai.sgp1.digitaloceanspaces.com/course/zip/
↪aithon2020-level-2.zip
```

```
--2020-09-13 05:31:46--
https://spotleai.sgp1.digitaloceanspaces.com/course/zip/aithon2020-level-2.zip
Resolving spotleai.sgp1.digitaloceanspaces.com
(spotleai.sgp1.digitaloceanspaces.com)… 103.253.144.208
Connecting to spotleai.sgp1.digitaloceanspaces.com
(spotleai.sgp1.digitaloceanspaces.com)|103.253.144.208|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 29149906 (28M) [application/zip]
Saving to: ‘aithon2020-level-2.zip’
```

```
aithon2020-level-2. 100%[===================>]  27.80M  6.00MB/s    in 4.6s

2020-09-13 05:31:52 (6.00 MB/s) - 'aithon2020-level-2.zip' saved
[29149906/29149906]
```

Extracting the zip

[4]:
```
!unzip aithon2020-level-2.zip
```

```
Archive:  aithon2020-level-2.zip
   creating: aithon2020-level-2/
  inflating: aithon2020-level-2/.DS_Store
  inflating: __MACOSX/aithon2020-level-2/._.DS_Store
  inflating: aithon2020-level-2/requirements.txt
  inflating: __MACOSX/aithon2020-level-2/._requirements.txt
   creating: aithon2020-level-2/source/
   creating: aithon2020-level-2/data/
  inflating: aithon2020-level-2/aithon_level2.py
   creating: aithon2020-level-2/.idea/
  inflating: aithon2020-level-2/source/classification.py
  inflating: aithon2020-level-2/data/aithon2020_level2_traning.csv
  inflating: __MACOSX/aithon2020-level-2/data/._aithon2020_level2_traning.csv
  inflating: aithon2020-level-2/.idea/workspace.xml
  inflating: aithon2020-level-2/.idea/untitled.iml
  inflating: aithon2020-level-2/.idea/modules.xml
  inflating: aithon2020-level-2/.idea/misc.xml
```

2.3 Import the training data

[5]:
```
train = pd.read_csv('aithon2020-level-2/data/aithon2020_level2_traning.csv')
train.head()
```

[5]:

	emotion	pixel_0	pixel_1	…	pixel_2301	pixel_2302	pixel_2303
0	Fear	231	212	…	88	110	152
1	Fear	55	55	…	34	30	57
2	Sad	20	17	…	99	107	118
3	Happy	4	2	…	3	7	12
4	Fear	255	255	…	79	79	83

[5 rows x 2305 columns]

3 Generating images from given data

3.1 image_from_array() : Return images, when passed an array.

[6]:
```
def image_from_array(arr,width,height,return_image=False):
    """
    Input : Takes in an array, width and height
    Output: Displays an image by reshaping image to
            provided width and height.


    # Press any key to close the window
    # if return_image=True, then the image matrix is returned
    # instead of displaying it.
    """
    # Reshaping the given array
    img = np.array(arr.reshape(width,height),dtype=np.uint8)

    if return_image:
        return img

    else:
        # displaying image ; press any button to close
        cv.imshow('image',img)
        cv.waitKey(0)
        cv.destroyAllWindows()
```

3.2 generate_images(data,labels) : Creates a images folder and stores images

[7]:
```
def generate_images(data,labels):
    """
    Input : Input data matrix of images and lables list
    Output: Store all the images in the /images/ folder
            along with labels.csv
    """
    # Checking if given data matrix and labels match
    assert len(data)==len(labels),"Input array size  labels size"

    # Getting current working directory
    path = os.getcwd()
    label_list = []

    try:
        # Creating a new directory 'images'
        os.mkdir(path+'/images')

    except OSError as error:
```

```
        # If directory already exists
        print(error)
        print("\nDelete the existing images folder & try again")
    store_path = path+'/images/'
    for i in range(len(data)):
        img = image_from_array(data[i],48,48,return_image=True)
        cv.imwrite(store_path+str(i)+'.png',img)
        label_list.append([i,labels[i]])
    label_df = pd.DataFrame(label_list,columns=['Image','Emotion'])
    label_df.to_csv(store_path+'labels.csv',index=False)
```

[ ]:
```
generate_images(train.iloc[:,1:].values,train['emotion'].values)
```

3.3 generate_images_folderwise(x_train,y_train,x_test,y_test)

Creates folders train and test and in each folder class with images

[8]:
```
def generate_images_folderwise(x_train,y_train,x_test,y_test):
    """
    Input : Input data matrix of images, lables list
    Output: Store all the images in the
            /images/<train or test>/<class_folder>
    """
    # Checking if given data matrix and labels match
    assert len(x_train)==len(y_train),"Input array size  labels size"
    assert len(x_test)==len(y_test),"Input array size  labels size"

    # Getting current working directory
    path = os.getcwd()

    temp = 0

    try:
        # Creating a new directory 'images'
        os.mkdir(path+'/images')
        os.mkdir(path+'/images/'+'train')
        os.mkdir(path+'/images/'+'test')

    except OSError as error:
        # If directory already exists
        print(error)
        print("\nDelete the existing images folder & try again")

    store_path = path+'/images/'+'train/'
    class_list = np.unique(y_train)
    for j in class_list:
        temp = 0
        os.mkdir(store_path+str(j))
```

```
        for i in range(len(x_train)):
            if y_train[i]==j:
                temp += 1
                img = image_from_array(x_train[i],48,48,return_image=True)
                cv.imwrite(store_path+str(j+'/')+str(temp)+'.png',img)

    store_path = path+'/images/'+'test/'
    class_list = np.unique(y_test)
    for j in class_list:
        temp = 0
        os.mkdir(store_path+str(j))
        for i in range(len(x_test)):
            if y_test[i]==j:
                temp += 1
                img = image_from_array(x_test[i],48,48,return_image=True)
                cv.imwrite(store_path+str(j+'/')+str(temp)+'.png',img)
```

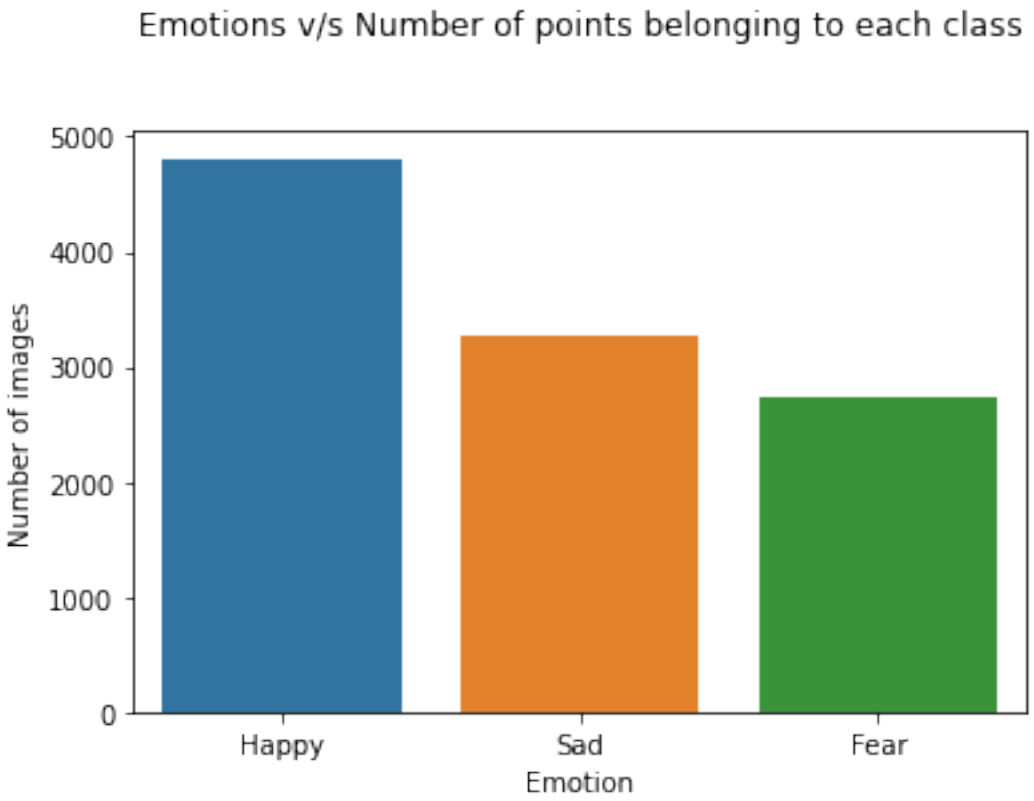
[28]:
```
x = train.iloc[:,1:].values
y = train['emotion'].values
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪stratify=y,random_state=42)
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size=0.
 ↪25,random_state=42)
generate_images_folderwise(x_train,y_train,x_val,y_val)
```

4 Exploratory data analysis

4.1 Checking for class imbalance

[10]:
```
temp = train['emotion'].value_counts()
sns.barplot(x=temp.index,y=temp.values)
plt.title("Emotions v/s Number of points belonging to each class\n\n")
plt.xlabel("Emotion")
plt.ylabel("Number of images")
```

[10]:
```
Text(0, 0.5, 'Number of images')
```

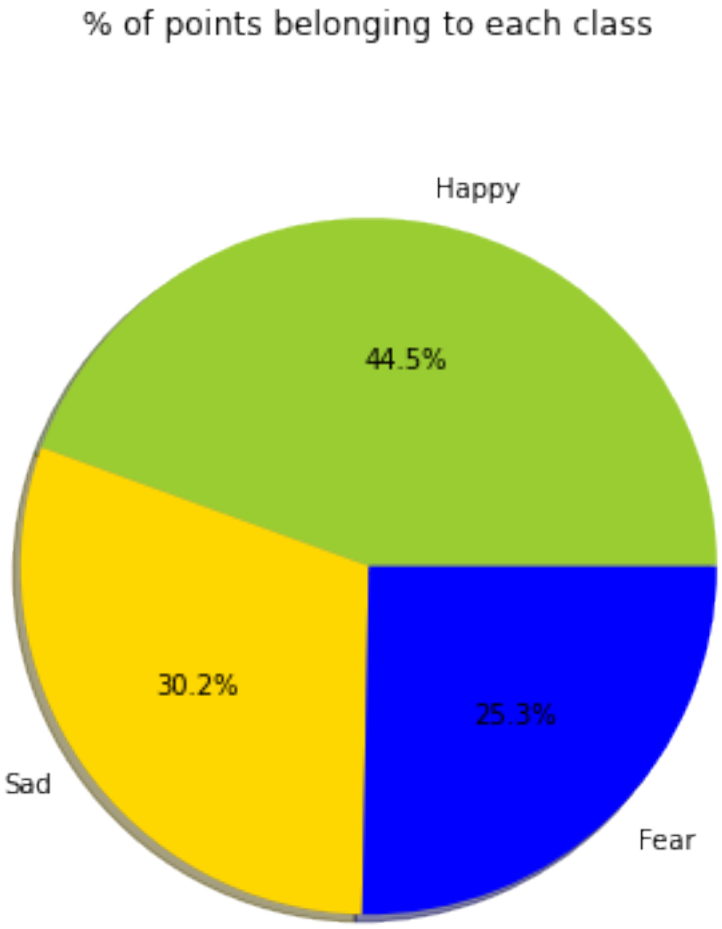


[11]:
```
class_label = train['emotion'].value_counts()
total_points = len(train)
print("Points with class label -> Happy are =  ",class_label.values[0]/
 ↪total_points*100,"%")
print("Points with class label -> Sad are =  ",class_label.values[1]/
 ↪total_points*100,"%")
print("Points with class label -> Fear are =  ",class_label.values[2]/
 ↪total_points*100,"%")
labels = ['Happy','Sad','Fear']
sizes = [44.45,30.23,25.30]
colors = ['yellowgreen', 'gold','blue']
plt.figure(figsize=(6,6))
plt.title("% of points belonging to each class\n\n")
plt.pie(sizes, labels=labels, colors=colors,autopct='%1.1f%%', shadow=True)
```

```
Points with class label -> Happy are =   44.45779791069613 %
Points with class label -> Sad are =   30.239437921789776 %
Points with class label -> Fear are =   25.3027641675141 %
```

[11]:
```
([<matplotlib.patches.Wedge at 0x7f4a242f9b70>,
  <matplotlib.patches.Wedge at 0x7f4a2430a5f8>,
```

```
 <matplotlib.patches.Wedge at 0x7f4a2430aef0>],
[Text(0.19052115567928468, 1.083375137816366, 'Happy'),
 Text(-0.9067898105676032, -0.6226814911740758, 'Sad'),
 Text(0.7703276824781344, -0.7852358000039648, 'Fear')],
[Text(0.10392063037051891, 0.5909318933543813, '44.5%'),
 Text(-0.4946126239459653, -0.339644449731314, '30.2%'),
 Text(0.4201787358971642, -0.42831043636579896, '25.3%')])
```



4.2 Exploring various images from the dataset

The referecne for the code below is taken from : https://www.kaggle.com/xhlulu/eda-simple-keras-cnn-k-mnist

The below code loads 25 random images from dataset, with their label.

[12]:
```
plt.figure(figsize=(10,10))
for i in range(25):
    temp = np.random.randint(0,10000)
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(True)
    plt.imshow(image_from_array(train.iloc[temp,1:].
 ↪values,48,48,return_image=True),cmap="gray")
    plt.xlabel(train['emotion'].values[temp])
```

5 Preparing data for modeling

5.1 Data Augmentation

The reference for below code was taken from : https://www.kaggle.com/sanikamal/multi-class-image-classification-with-augmentation

[13]:
```
path = os.getcwd()

TRAINING_DIR = path+"/images/train/"
training_datagen = ImageDataGenerator(
    rescale = 1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')

VALIDATION_DIR = path+"/images/test/"
validation_datagen = ImageDataGenerator(rescale = 1./255)

train_generator = training_datagen.flow_from_directory(
    TRAINING_DIR,
    color_mode='grayscale',
    target_size = (48,48),
    batch_size = 64,
    class_mode='categorical',
    shuffle=True)

validation_generator = validation_datagen.flow_from_directory(
    VALIDATION_DIR,
    color_mode = 'grayscale',
    target_size = (48,48),
    batch_size = 64,
    class_mode='categorical',
    shuffle=False)
```

```
Found 6489 images belonging to 3 classes.
Found 2164 images belonging to 3 classes.
```

6 Modeling

6.1 A simple CNN model

Loading the initial parameters

[19]:
```
num_classes = 3
epochs = 100
```

[20]:
```
tf.keras.backend.clear_session()
```

[21]:
```
# Creating the model
model = Sequential()

model.add(Conv2D(32, kernel_size=(5, 5),␣
 ↪input_shape=(48,48,1),activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(32, kernel_size=(5, 5),␣
 ↪input_shape=(48,48,1),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2),strides=2))
model.add(Dropout(0.25))


model.add(Conv2D(64, kernel_size=(3, 3),␣
 ↪input_shape=(48,48,1),activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(64, kernel_size=(3, 3),␣
 ↪input_shape=(48,48,1),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(3, 3),strides=2))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3),␣
 ↪input_shape=(48,48,1),activation='relu'))
model.add(BatchNormalization())
model.add(Conv2D(128, kernel_size=(3, 3),␣
 ↪input_shape=(48,48,1),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(1, 1),strides=2))
model.add(Dropout(0.25))



model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

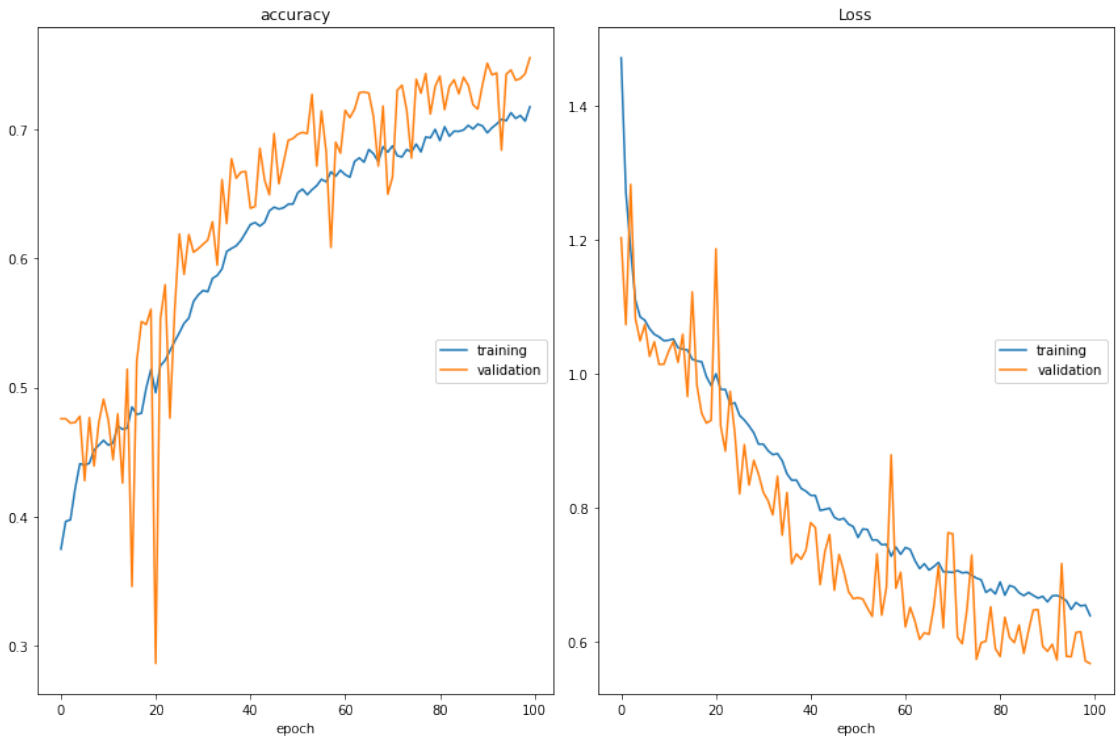
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 44, 44, 32)        832
_________________________________________________________________
batch_normalization (BatchNo (None, 44, 44, 32)        128
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 40, 40, 32)        25632
_________________________________________________________________
batch_normalization_1 (Batch (None, 40, 40, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 20, 20, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 20, 20, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 18, 18, 64)        18496
_________________________________________________________________
batch_normalization_2 (Batch (None, 18, 18, 64)        256
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 16, 16, 64)        36928
_________________________________________________________________
batch_normalization_3 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 5, 5, 128)         73856
_________________________________________________________________
batch_normalization_4 (Batch (None, 5, 5, 128)         512
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 3, 3, 128)         147584
_________________________________________________________________
batch_normalization_5 (Batch (None, 3, 3, 128)         512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
dropout_2 (Dropout)          (None, 2, 2, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 512)               262656
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1539
=================================================================
Total params: 569,315
Trainable params: 568,419
```

```
Non-trainable params: 896
_________________________________________________________________
```

[22]:
```
model.compile(
    loss = 'categorical_crossentropy',
    optimizer=Adam(),
    metrics=['accuracy'])
```

[23]:
```
steps_per_epoch = train_generator.n//train_generator.batch_size
validation_steps = validation_generator.n//validation_generator.batch_size
checkpoint = ModelCheckpoint("model_weights.h5", monitor='val_accuracy',
                             save_weights_only=True, mode='max', verbose=1)
callbacks = [PlotLossesKeras(), checkpoint]

history = model.fit(
    x=train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data = validation_generator,
    validation_steps = validation_steps,
    callbacks=callbacks
)
```




```
accuracy
```

```
        training                 (min:    0.375, max:    0.717, cur:    0.717)
        validation               (min:    0.286, max:    0.755, cur:    0.755)
Loss
        training                 (min:    0.639, max:    1.471, cur:    0.639)
        validation               (min:    0.568, max:    1.282, cur:    0.568)

Epoch 00100: saving model to model_weights.h5
101/101 [==============================] - 6s 57ms/step - loss: 0.6390 -
accuracy: 0.7172 - val_loss: 0.5678 - val_accuracy: 0.7552
```

7 Saving the model

[ ]:
```
model.save('model')
```

```
INFO:tensorflow:Assets written to: model/assets
```

[ ]:
```
!zip -r model.zip  model
```

```
  adding: model/ (stored 0%)
  adding: model/saved_model.pb (deflated 91%)
  adding: model/variables/ (stored 0%)
  adding: model/variables/variables.data-00000-of-00001 (deflated 8%)
  adding: model/variables/variables.index (deflated 74%)
  adding: model/assets/ (stored 0%)
```

[ ]:
```
files.download('model.zip')
```

```
<IPython.core.display.Javascript object>


<IPython.core.display.Javascript object>
```

8 Evaluating the model

[33]:
```
x_test = x_test.astype('float32')/255.0
```

[48]:
```
train_generator.class_indices
```

[48]:
```
{'Fear': 0, 'Happy': 1, 'Sad': 2}
```

[51]:
```
output = []
temp = 0
for i in range(len(x_test)):
  temp = np.argmax(model.predict(x_test[i].reshape(1,48,48,1)),axis=1)
  if temp ==0:
    output.append('Fear')
```

```
  elif temp==1:
    output.append('Happy')
  elif temp==2:
    output.append('Sad')
```

[57]:
```
print("Accuracy on unseen data : ",np.sum(output==y_test)/len(x_test))
```

```
Accuracy on unseen data :  0.7587800369685767
```