Facial Expression Recognition with Keras

Task 1: Import Libraries

In [2]:

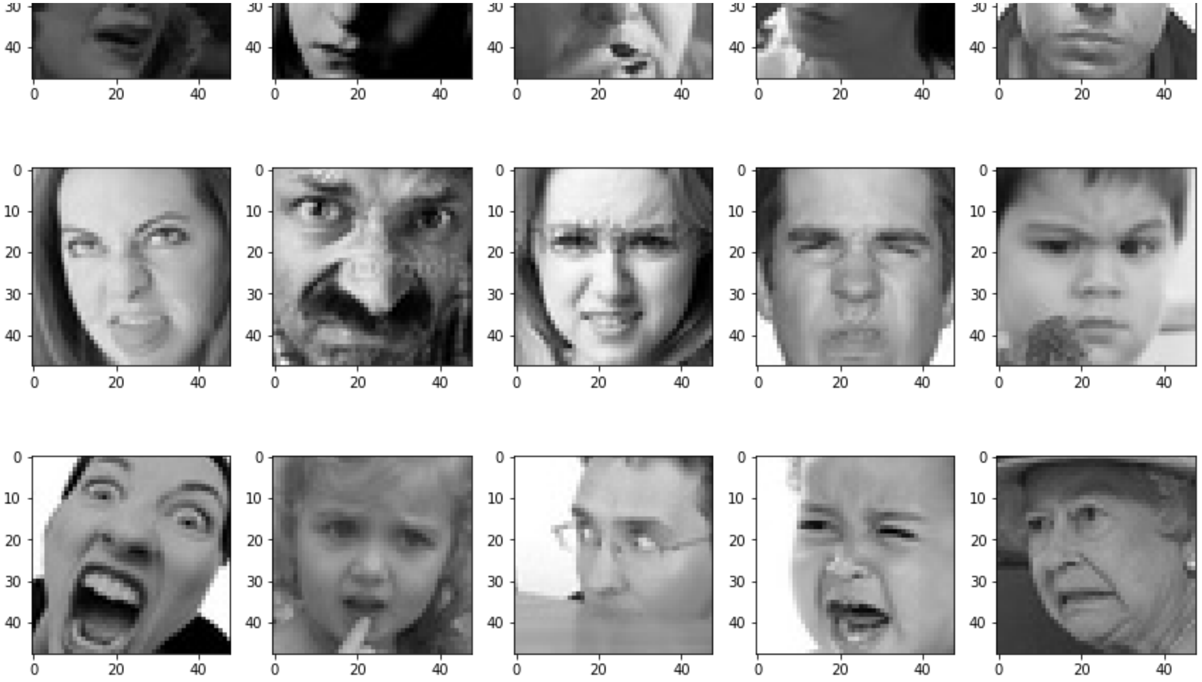
```
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import utils
import os
%matplotlib inline

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Input, Dropout,Flatten, Conv2D
from tensorflow.keras.layers import BatchNormalization, Activation, MaxPooling2D
from tensorflow.keras.models import Model, Sequential
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras.utils import plot_model

from IPython.display import SVG, Image
from livelossplot import PlotLossesTensorFlowKeras
import tensorflow as tf
print("Tensorflow version:", tf.__version__)
```

```
Tensorflow version: 2.1.0
```

Task 2: Plot Sample Image

In [3]:

```
utils.datasets.fer.plot_example_images(plt).show()
```



```
In [5]: for expression in os.listdir("train/"):
            print(str(len(os.listdir("train/" + expression))) + " " + expressio
        n + " images")
```

```
3171 surprise images
7215 happy images
4965 neutral images
3995 angry images
4830 sad images
436 disgust images
4097 fear images
```

Task 3: Generate Training and Validation Batches

```
In [6]: img_size = 48
        batch_size = 64
```

```
datagen_train = ImageDataGenerator(horizontal_flip=True)
train_generator = datagen_train.flow_from_directory("train/", target_si
ze=(img_size,img_size),
                                                    color_mode='graysca
le',
                                                    batch_size=batch_si
ze,
                                                    class_mode='categor
ical',
                                                    shuffle=True)

datagen_validation = ImageDataGenerator(horizontal_flip=True)
validation_generator = datagen_train.flow_from_directory("test/", targe
t_size=(img_size,img_size),
                                                    color_mode='graysca
le',
                                                    batch_size=batch_si
ze,
                                                    class_mode='categor
ical',
                                                    shuffle=True)
# print(type(train_generator))
# print(train_generator)
```

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

Task 4: Create CNN Model

Inspired by Goodfellow, I.J., et.al. (2013). Challenged in representation learning: A report of three machine learning contests. *Neural Networks*, 64, 59-63. doi:10.1016/j.neunet.2014.09.005

In [9]:

```
model = Sequential()
```

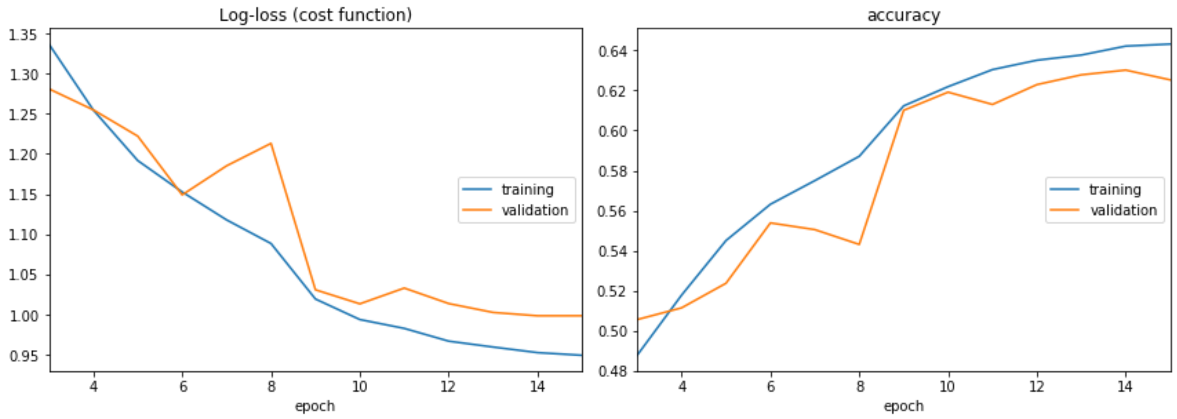
```
# 1 - conv layer
# 64 filters, 3 by 3
model.add(Conv2D(64, (3,3), padding='same', input_shape=(48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# 2 - conv layer
model.add(Conv2D(128, (5,5), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# 3 - conv layer
model.add(Conv2D(512, (3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# 4 - conv layer
model.add(Conv2D(512, (3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

opt = Adam(lr=0.0005)
model.compile(optimizer=opt, loss='categorical_crossentropy', metrics=[
'accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640       
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256       
_________________________________________________________________
activation (Activation)      (None, 48, 48, 64)        0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0         
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 128)       204928    
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 128)       512       
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 128)       0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 512)       590336    
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 512)       2048      
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 512)       0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 512)         0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 512)         0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 512)         2359808
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 512)         2048
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 512)         0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0
_________________________________________________________________
dense (Dense)                (None, 256)               1179904
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591
=================================================================
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968
_________________________________________________________________
```

In [ ]:

Task 6: Train and Evaluate Model

```
In [10]:
epochs = 15
steps_per_epoch = train_generator.n//train_generator.batch_size
validation_steps = validation_generator.n//validation_generator.batch_s
ize

checkpoint = ModelCheckpoint("model_weights.h5", monitor='val_accuracy'
, save_weights_only=True,
                             mode='max', verbose=1)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=
2, min_lr=0.00001,
                              model='auto')
callbacks = [PlotLossesTensorFlowKeras(), checkpoint, reduce_lr]

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



```
Log-loss (cost function):
```

```
training   (min:    0.949, max:    1.798, cur:    0.949)
validation (min:    0.998, max:    1.705, cur:    0.998)

accuracy:
training   (min:    0.309, max:    0.643, cur:    0.643)
validation (min:    0.355, max:    0.630, cur:    0.625)

Epoch 00015: saving model to model_weights.h5
448/448 [==============================] - 27s 61ms/step - loss: 0.94
87 - accuracy: 0.6431 - val_loss: 0.9983 - val_accuracy: 0.6253
```

Task 7: Represent Model as JSON String

In [11]:

```
model_json = model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(model_json)
```

In [ ]: