

In []:

```
import numpy as np
```

In []:

```
import tensorflow as tf
```

Data loading from csv file

In []:

```
with open("/content/drive/MyDrive/Colab Notebooks/dataset2/fer2013.csv") as f:  
    content = f.readlines()
```

```
lines = np.array(content)
```

In []:

```
num_of_instances = lines.size  
print("number of instances: ", num_of_instances)  
print("instance length: ", len(lines[1].split(",")[1].split(" ")))
```

```
number of instances: 35888  
instance length: 2304
```

In []:

```
num_classes = 7 #angry, disgust, fear, happy, sad, surprise, neutral
```

In []:

```
x_train, y_train, x_test, y_test = [], [], [], []
```

In []:

```
for i in range(1, num_of_instances):  
    emotion, img, usage = lines[i].split(",")  
    val = img.split(" ")  
    pixels = np.array(val, 'float32')  
    emotion = tf.keras.utils.to_categorical(emotion, num_classes)  
    if 'Training' in usage:  
        y_train.append(emotion)  
        x_train.append(pixels)  
    elif 'PublicTest' in usage:  
        y_test.append(emotion)  
        x_test.append(pixels)
```

In []:

```
x_train = np.array(x_train, 'float32')  
y_train = np.array(y_train, 'float32')  
x_test = np.array(x_test, 'float32')  
y_test = np.array(y_test, 'float32')
```

In []:

```
x_train /= 255 #normalize inputs between [0, 1]  
x_test /= 255  
  
x_train = x_train.reshape(x_train.shape[0], 48, 48, 1)  
x_train = x_train.astype('float32')  
x_test = x_test.reshape(x_test.shape[0], 48, 48, 1)  
x_test = x_test.astype('float32')
```

Checking Dims

In []:

```
print(x_train.shape)  
print(y_train.shape)  
print(x_test.shape)  
print(y_test.shape)
```

```
(28709, 48, 48, 1)  
(28709, 7)  
(3589, 48, 48, 1)  
(3589, 7)
```

Weigths initializer

not used

In []:

```
#initializer = tf.keras.initializers.RandomNormal(mean=0., stddev=0.00001, seed=1234646445567576789)
```

vgg16 Model

In []:

```
# Build the model
emo_model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(64, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(64, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size =2, strides =2, padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(128, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(128, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size =2, strides =2, padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(256, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(256, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(256, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size =2, strides =2, padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(512, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(512, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(512, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(512, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(512, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Conv2D(512, kernel_size =3, activation='relu', padding = 'same', kernel_regularizer=tf.keras.regularizers.l2(0.0001)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.MaxPool2D(pool_size =2, strides =2, padding = 'same'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(units = 4096, activation = 'relu', kernel_initializer='he_normal'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(units = 4096, activation = 'relu', kernel_initializer='he_normal'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(units = 1000, activation = 'relu', kernel_initializer='he_normal'),
    tf.keras.layers.Dense(units = 7, activation = 'softmax')
])
```

In []:

```
emo_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization (Batch Normalization)	(None, 48, 48, 64)	256
conv2d_1 (Conv2D)	(None, 48, 48, 64)	36928
batch_normalization_1 (Batch Normalization)	(None, 48, 48, 64)	256
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0

batch_normalization_2	(Batch (None, 24, 24, 64)	256
conv2d_2	(Conv2D) (None, 24, 24, 128)	73856
batch_normalization_3	(Batch (None, 24, 24, 128)	512
conv2d_3	(Conv2D) (None, 24, 24, 128)	147584
batch_normalization_4	(Batch (None, 24, 24, 128)	512
max_pooling2d_1	(MaxPooling2 (None, 12, 12, 128)	0
batch_normalization_5	(Batch (None, 12, 12, 128)	512
conv2d_4	(Conv2D) (None, 12, 12, 256)	295168
batch_normalization_6	(Batch (None, 12, 12, 256)	1024
conv2d_5	(Conv2D) (None, 12, 12, 256)	590080
batch_normalization_7	(Batch (None, 12, 12, 256)	1024
conv2d_6	(Conv2D) (None, 12, 12, 256)	590080
batch_normalization_8	(Batch (None, 12, 12, 256)	1024
max_pooling2d_2	(MaxPooling2 (None, 6, 6, 256)	0
batch_normalization_9	(Batch (None, 6, 6, 256)	1024
conv2d_7	(Conv2D) (None, 6, 6, 512)	1180160
batch_normalization_10	(Batch (None, 6, 6, 512)	2048
conv2d_8	(Conv2D) (None, 6, 6, 512)	2359808
batch_normalization_11	(Batch (None, 6, 6, 512)	2048
conv2d_9	(Conv2D) (None, 6, 6, 512)	2359808
batch_normalization_12	(Batch (None, 6, 6, 512)	2048
max_pooling2d_3	(MaxPooling2 (None, 3, 3, 512)	0
batch_normalization_13	(Batch (None, 3, 3, 512)	2048
conv2d_10	(Conv2D) (None, 3, 3, 512)	2359808
batch_normalization_14	(Batch (None, 3, 3, 512)	2048
conv2d_11	(Conv2D) (None, 3, 3, 512)	2359808
batch_normalization_15	(Batch (None, 3, 3, 512)	2048
conv2d_12	(Conv2D) (None, 3, 3, 512)	2359808
batch_normalization_16	(Batch (None, 3, 3, 512)	2048
max_pooling2d_4	(MaxPooling2 (None, 2, 2, 512)	0
batch_normalization_17	(Batch (None, 2, 2, 512)	2048
flatten	(Flatten) (None, 2048)	0
dense	(Dense) (None, 4096)	8392704
dropout	(Dropout) (None, 4096)	0
dense_1	(Dense) (None, 4096)	16781312
dropout_1	(Dropout) (None, 4096)	0
dense_2	(Dense) (None, 1000)	4097000
dense_3	(Dense) (None, 7)	7007

Total params: 44,014,343
Trainable params: 44,002,951
Non-trainable params: 11,392

Input Output test

In []:

```
predictions = emo_model(x_train[1:2]).numpy()
print(predictions)
```

```
[[0.14290261 0.14285195 0.1429394  0.14268515 0.14284706 0.14280333
  0.14297047]]
```

Learning rate decay

not used

In []:

```
#lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(initial_learning_rate=0.1,decay_steps=100,decay
```

Optimizer Loss Function and Metrics

In []:

```
sgd = tf.keras.optimizers.SGD(
    learning_rate=0.0001, momentum=0.85, nesterov=True
)
loss_fn = tf.keras.losses.CategoricalCrossentropy(from_logits=True)
```

Compiling

In []:

```
emo_model.compile(optimizer=sgd,
                  loss=loss_fn,
                  metrics=['accuracy'])
```

Checkpoints

In []:

```
checkpoint_path = "/content/drive/MyDrive/Colab Notebooks/checkpoints/model5/cp.ckpt"

cp_callback = tf.keras.callbacks.ModelCheckpoint(filepath=checkpoint_path,monitor='accuracy', save_freq='epoch')
```

Data Augmentation

In []:

```
##### Include Little Data Augmentation
batch_size = 113 # try several values

train_DataGen = tf.keras.preprocessing.image.ImageDataGenerator(zoom_range=0.2,
                                                                width_shift_range=0.1,
                                                                height_shift_range=0.1,
                                                                horizontal_flip=True)

train_set_conv = train_DataGen.flow(x_train, y_train, batch_size=batch_size) # train_lab is categorical
```

Running ...

last 30 epoch

In []:

```
#history = emo_model.fit(train_set_conv,batch_size=batch_size,callbacks=[cp_callback],shuffle = True)
history = emo_model.fit(train_set_conv,epochs=30,steps_per_epoch=x_train.shape[0]/batch_size,validation_data=
```

```
Epoch 1/30
254/254 [=====] - 20s 78ms/step - loss: 0.5493 - accuracy: 0.9879 - val_loss: 2.5386
- val_accuracy: 0.6913
```

Epoch 00001: accuracy did not improve from 0.98823

```
Epoch 2/30
254/254 [=====] - 20s 78ms/step - loss: 0.5491 - accuracy: 0.9875 - val_loss: 2.5364
- val_accuracy: 0.6910
```

Epoch 00002: accuracy did not improve from 0.98823

```
Epoch 3/30
254/254 [=====] - 20s 78ms/step - loss: 0.5495 - accuracy: 0.9884 - val_loss: 2.5354
- val_accuracy: 0.6904
```

Epoch 00003: accuracy improved from 0.98823 to 0.98844, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model5/cp.ckpt
Epoch 4/30
254/254 [=====] - 21s 82ms/step - loss: 0.5474 - accuracy: 0.9886 - val_loss: 2.5387
- val_accuracy: 0.6910

Epoch 00004: accuracy improved from 0.98844 to 0.98861, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model5/cp.ckpt
Epoch 5/30
254/254 [=====] - 21s 82ms/step - loss: 0.5472 - accuracy: 0.9888 - val_loss: 2.5371
- val_accuracy: 0.6910

Epoch 00005: accuracy improved from 0.98861 to 0.98878, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model5/cp.ckpt
Epoch 6/30
254/254 [=====] - 21s 83ms/step - loss: 0.5472 - accuracy: 0.9885 - val_loss: 2.5446
- val_accuracy: 0.6924

Epoch 00006: accuracy did not improve from 0.98878
Epoch 7/30
254/254 [=====] - 20s 79ms/step - loss: 0.5471 - accuracy: 0.9880 - val_loss: 2.5409
- val_accuracy: 0.6930

Epoch 00007: accuracy did not improve from 0.98878
Epoch 8/30
254/254 [=====] - 20s 79ms/step - loss: 0.5450 - accuracy: 0.9890 - val_loss: 2.5496
- val_accuracy: 0.6910

Epoch 00008: accuracy improved from 0.98878 to 0.98899, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model5/cp.ckpt
Epoch 9/30
254/254 [=====] - 21s 83ms/step - loss: 0.5440 - accuracy: 0.9889 - val_loss: 2.5501
- val_accuracy: 0.6896

Epoch 00009: accuracy did not improve from 0.98899
Epoch 10/30
254/254 [=====] - 20s 80ms/step - loss: 0.5472 - accuracy: 0.9880 - val_loss: 2.5509
- val_accuracy: 0.6913

Epoch 00010: accuracy did not improve from 0.98899
Epoch 11/30
254/254 [=====] - 20s 79ms/step - loss: 0.5449 - accuracy: 0.9884 - val_loss: 2.5505
- val_accuracy: 0.6921

Epoch 00011: accuracy did not improve from 0.98899
Epoch 12/30
254/254 [=====] - 20s 80ms/step - loss: 0.5444 - accuracy: 0.9883 - val_loss: 2.5535
- val_accuracy: 0.6924

Epoch 00012: accuracy did not improve from 0.98899
Epoch 13/30
254/254 [=====] - 20s 80ms/step - loss: 0.5434 - accuracy: 0.9887 - val_loss: 2.5627
- val_accuracy: 0.6930

Epoch 00013: accuracy did not improve from 0.98899
Epoch 14/30
254/254 [=====] - 20s 79ms/step - loss: 0.5439 - accuracy: 0.9881 - val_loss: 2.5653
- val_accuracy: 0.6916

Epoch 00014: accuracy did not improve from 0.98899
Epoch 15/30
254/254 [=====] - 20s 79ms/step - loss: 0.5428 - accuracy: 0.9884 - val_loss: 2.5677
- val_accuracy: 0.6932

Epoch 00015: accuracy did not improve from 0.98899
Epoch 16/30
254/254 [=====] - 20s 79ms/step - loss: 0.5416 - accuracy: 0.9888 - val_loss: 2.5646
- val_accuracy: 0.6907

Epoch 00016: accuracy did not improve from 0.98899
Epoch 17/30
254/254 [=====] - 20s 79ms/step - loss: 0.5418 - accuracy: 0.9885 - val_loss: 2.5652
- val_accuracy: 0.6896

Epoch 00017: accuracy did not improve from 0.98899
Epoch 18/30
254/254 [=====] - 20s 80ms/step - loss: 0.5421 - accuracy: 0.9880 - val_loss: 2.5620
- val_accuracy: 0.6900

254/254 [=====] - 20s 80ms/step - loss: 0.5421 - accuracy: 0.9889 - val_loss: 2.5622
- val_accuracy: 0.6888

Epoch 00018: accuracy did not improve from 0.98899

Epoch 19/30

254/254 [=====] - 20s 80ms/step - loss: 0.5417 - accuracy: 0.9887 - val_loss: 2.5604
- val_accuracy: 0.6916

Epoch 00019: accuracy did not improve from 0.98899

Epoch 20/30

254/254 [=====] - 20s 80ms/step - loss: 0.5397 - accuracy: 0.9889 - val_loss: 2.5659
- val_accuracy: 0.6927

Epoch 00020: accuracy did not improve from 0.98899

Epoch 21/30

254/254 [=====] - 21s 81ms/step - loss: 0.5413 - accuracy: 0.9884 - val_loss: 2.5648
- val_accuracy: 0.6924

Epoch 00021: accuracy did not improve from 0.98899

Epoch 22/30

254/254 [=====] - 20s 78ms/step - loss: 0.5391 - accuracy: 0.9895 - val_loss: 2.5710
- val_accuracy: 0.6932

Epoch 00022: accuracy improved from 0.98899 to 0.98952, saving model to /content/drive/MyDrive/Colab Notebooks
/checkpoints/model5/cp.ckpt

Epoch 23/30

254/254 [=====] - 21s 83ms/step - loss: 0.5389 - accuracy: 0.9887 - val_loss: 2.5733
- val_accuracy: 0.6938

Epoch 00023: accuracy did not improve from 0.98952

Epoch 24/30

254/254 [=====] - 20s 78ms/step - loss: 0.5399 - accuracy: 0.9883 - val_loss: 2.5703
- val_accuracy: 0.6932

Epoch 00024: accuracy did not improve from 0.98952

Epoch 25/30

254/254 [=====] - 20s 79ms/step - loss: 0.5391 - accuracy: 0.9886 - val_loss: 2.5571
- val_accuracy: 0.6932

Epoch 00025: accuracy did not improve from 0.98952

Epoch 26/30

254/254 [=====] - 20s 78ms/step - loss: 0.5393 - accuracy: 0.9885 - val_loss: 2.5690
- val_accuracy: 0.6918

Epoch 00026: accuracy did not improve from 0.98952

Epoch 27/30

254/254 [=====] - 20s 79ms/step - loss: 0.5384 - accuracy: 0.9886 - val_loss: 2.5758
- val_accuracy: 0.6907

Epoch 00027: accuracy did not improve from 0.98952

Epoch 28/30

254/254 [=====] - 20s 79ms/step - loss: 0.5390 - accuracy: 0.9885 - val_loss: 2.5783
- val_accuracy: 0.6930

Epoch 00028: accuracy did not improve from 0.98952

Epoch 29/30

254/254 [=====] - 20s 79ms/step - loss: 0.5370 - accuracy: 0.9887 - val_loss: 2.5797
- val_accuracy: 0.6938

Epoch 00029: accuracy did not improve from 0.98952

Epoch 30/30

254/254 [=====] - 20s 78ms/step - loss: 0.5366 - accuracy: 0.9889 - val_loss: 2.5911
- val_accuracy: 0.6916

Epoch 00030: accuracy did not improve from 0.98952

First 165 epoch

learning rate 0.01

Meet - 15Z017 - ML Class - x ch4.pdf x vgg16 3 - Colaboratory x vgg16 5 - Colaboratory x ResNet_50 - Jupyter Notebook x +

colab.research.google.com/drive/1U4Ifp48m1MhIV8ERWk_qJM6zbtmfxkn#scrollTo=2Hs2kVjwmAn3

vgg16 5

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
Epoch 00158: accuracy did not improve from 0.91609
Epoch 159/300
254/254 [=====] - 21s 83ms/step - loss: 1.0932 - accuracy: 0.8714 - val_loss: 1.9236 - val_accuracy: 0.6467

Epoch 00159: accuracy did not improve from 0.91609
Epoch 160/300
254/254 [=====] - 21s 82ms/step - loss: 1.0257 - accuracy: 0.9021 - val_loss: 1.9325 - val_accuracy: 0.6662

Epoch 00160: accuracy did not improve from 0.91609
Epoch 161/300
254/254 [=====] - 21s 83ms/step - loss: 0.9889 - accuracy: 0.9140 - val_loss: 1.9725 - val_accuracy: 0.6551

Epoch 00161: accuracy did not improve from 0.91609
Epoch 162/300
254/254 [=====] - 21s 82ms/step - loss: 0.9651 - accuracy: 0.9186 - val_loss: 2.1304 - val_accuracy: 0.6417

Epoch 00162: accuracy did not improve from 0.91609
Epoch 163/300
254/254 [=====] - 21s 83ms/step - loss: 0.9686 - accuracy: 0.9178 - val_loss: 2.0681 - val_accuracy: 0.6183

Epoch 00163: accuracy did not improve from 0.91609
Epoch 164/300
254/254 [=====] - 21s 83ms/step - loss: 1.0136 - accuracy: 0.9025 - val_loss: 2.0123 - val_accuracy: 0.6578

Epoch 00164: accuracy did not improve from 0.91609
Epoch 165/300
40/254 [==>.....] - ETA: 16s - loss: 0.9397 - accuracy: 0.9291

import matplotlib.pyplot as plt
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.xlabel('accuracy')
```

Type here to search

02:11 PM 02-03-2021

Second 90 epoch

learning rate 0.001

Meet - 15Z017 - ML Class - x vgg16 3 - Colaboratory x vgg16 5 - Colaboratory x ResNet_50 - Jupyter Notebook x +

colab.research.google.com/drive/1U4Ifp48m1MhIV8ERWk_qJM6zbtmfxkn#scrollTo=bsVNfOuYqCQn

vgg16 5

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
Epoch 83/300
254/254 [=====] - 21s 81ms/step - loss: 0.5790 - accuracy: 0.9847 - val_loss: 2.4830 - val_accuracy: 0.6941

...

Epoch 00883: accuracy did not improve from 0.98617
Epoch 84/300
254/254 [=====] - 21s 81ms/step - loss: 0.5734 - accuracy: 0.9851 - val_loss: 2.5154 - val_accuracy: 0.6918

Epoch 00884: accuracy did not improve from 0.98617
Epoch 85/300
254/254 [=====] - 21s 81ms/step - loss: 0.5733 - accuracy: 0.9844 - val_loss: 2.4906 - val_accuracy: 0.6904

Epoch 00885: accuracy did not improve from 0.98617
Epoch 86/300
254/254 [=====] - 20s 80ms/step - loss: 0.5688 - accuracy: 0.9866 - val_loss: 2.5031 - val_accuracy: 0.6877

Epoch 00886: accuracy improved from 0.98617 to 0.98624, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model15/cp.ckpt
Epoch 87/300
254/254 [=====] - 22s 85ms/step - loss: 0.5644 - accuracy: 0.9861 - val_loss: 2.5354 - val_accuracy: 0.6890

Epoch 00887: accuracy did not improve from 0.98624
Epoch 88/300
254/254 [=====] - 21s 81ms/step - loss: 0.5723 - accuracy: 0.9842 - val_loss: 2.5361 - val_accuracy: 0.6916

Epoch 00888: accuracy did not improve from 0.98624
Epoch 89/300
254/254 [=====] - 20s 80ms/step - loss: 0.5658 - accuracy: 0.9847 - val_loss: 2.5243 - val_accuracy: 0.6952

Epoch 00889: accuracy did not improve from 0.98624
Epoch 90/300
41/254 [==>.....] - ETA: 16s - loss: 0.5555 - accuracy: 0.9888

[ ] import matplotlib.pyplot as plt
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.xlabel('accuracy')
```

Type here to search

02:43 PM 02-03-2021

Third 15 epoch

learning rate 0.0001

Google Meet x vgg16.3 - Colaboratory x vgg16.5 - Colaboratory x ResNet_50 - Jupyter Notebook x +

colab.research.google.com/drive/1U4Ifp48m1MhIV8ERWk_qJM6zbtmfxkn#scrollTo=bsVNF0uYqCQn

vgg16.5

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

RAM Disk Editing

```
Epoch 6/300
254/254 [=====] - 20s 78ms/step - loss: 0.5554 - accuracy: 0.9866 - val_loss: 2.5103 - val_accuracy: 0.6938

Epoch 00006: accuracy did not improve from 0.98770
Epoch 7/300
254/254 [=====] - 20s 79ms/step - loss: 0.5564 - accuracy: 0.9870 - val_loss: 2.5094 - val_accuracy: 0.6932

Epoch 00007: accuracy did not improve from 0.98770
Epoch 8/300
254/254 [=====] - 20s 79ms/step - loss: 0.5561 - accuracy: 0.9863 - val_loss: 2.5127 - val_accuracy: 0.6943

Epoch 00008: accuracy did not improve from 0.98770
Epoch 9/300
254/254 [=====] - 20s 79ms/step - loss: 0.5538 - accuracy: 0.9869 - val_loss: 2.5147 - val_accuracy: 0.6918

Epoch 00009: accuracy did not improve from 0.98770
Epoch 10/300
254/254 [=====] - 20s 79ms/step - loss: 0.5510 - accuracy: 0.9884 - val_loss: 2.5183 - val_accuracy: 0.6927

Epoch 00010: accuracy improved from 0.98770 to 0.98795, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model15/cp.ckpt
Epoch 11/300
254/254 [=====] - 21s 82ms/step - loss: 0.5533 - accuracy: 0.9869 - val_loss: 2.5111 - val_accuracy: 0.6932

Epoch 00011: accuracy did not improve from 0.98795
Epoch 12/300
254/254 [=====] - 20s 79ms/step - loss: 0.5542 - accuracy: 0.9870 - val_loss: 2.5276 - val_accuracy: 0.6921

Epoch 00012: accuracy did not improve from 0.98795
Epoch 13/300
254/254 [=====] - 20s 80ms/step - loss: 0.5532 - accuracy: 0.9870 - val_loss: 2.5147 - val_accuracy: 0.6899

Epoch 00013: accuracy did not improve from 0.98795
Epoch 14/300
254/254 [=====] - 20s 78ms/step - loss: 0.5487 - accuracy: 0.9890 - val_loss: 2.5257 - val_accuracy: 0.6888

Epoch 00014: accuracy improved from 0.98795 to 0.98823, saving model to /content/drive/MyDrive/Colab Notebooks/checkpoints/model15/cp.ckpt
Epoch 15/300
145/254 [=====>.....] - ETA: 8s - loss: 0.5475 - accuracy: 0.9892
```

Type here to search

02:49 PM 02-03-2021

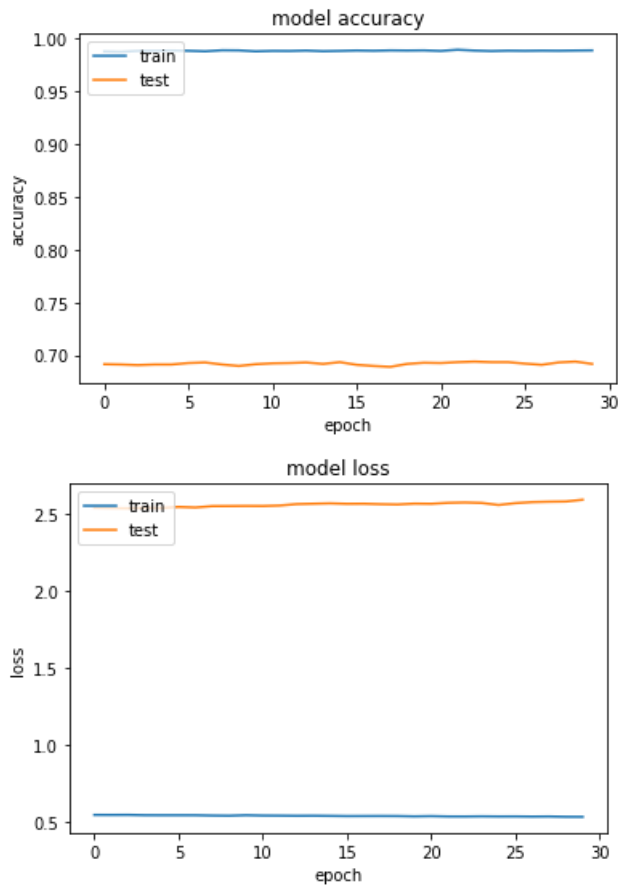
Train Test performance in last 30 epoch

In []:

```
import matplotlib.pyplot as plt
# list all data in history
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```



Train Test results

In []:

```
train_score = emo_model.evaluate(x_train, y_train, verbose=0)
print('Train loss:', train_score[0])
print('Train accuracy:', 100*train_score[1])

test_score = emo_model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', test_score[0])
print('Test accuracy:', 100*test_score[1])
```

```
Train loss: 0.5329243540763855
Train accuracy: 99.08391237258911
Test loss: 2.5911366939544678
Test accuracy: 69.155752658844
```

Real world Test

In []:

```
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()
```

In []:

```
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator

file = '/content/drive/MyDrive/Colab Notebooks/test/test6.png'
true_image = image.load_img(file)
img = image.load_img(file, grayscale=True, target_size=(48, 48))

x = image.img_to_array(img)
x = np.expand_dims(x, axis = 0)
```

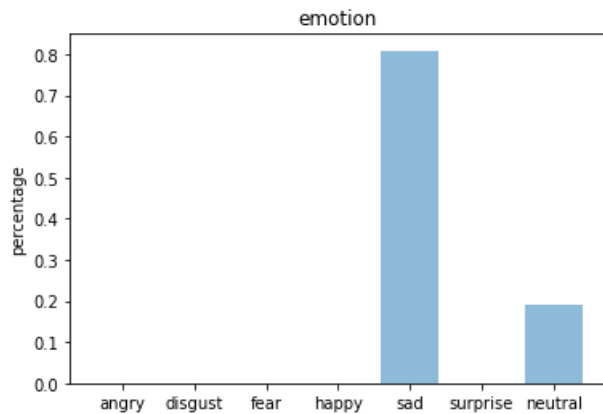
```
x /= 255
```

```
custom = emo_model.predict(x)
emotion_analysis(custom[0])
print(custom[0])
```

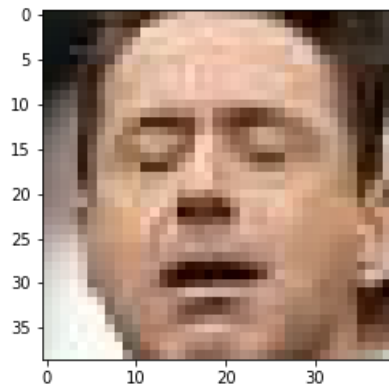
```
x = np.array(x, 'float32')
x = x.reshape([48, 48]);
```

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

/usr/local/lib/python3.7/dist-packages/keras_preprocessing/image/utils.py:107: UserWarning: grayscale is deprecated. Please use color_mode = "grayscale"
warnings.warn('grayscale is deprecated. Please use '



```
[7.3928933e-04 9.5471400e-08 2.1783708e-05 2.7109706e-06 8.0887306e-01
5.0759081e-07 1.9036256e-01]
```



Saving model

```
print(emo_model.save('/content/drive/MyDrive/Colab Notebooks/checkpoints/SuccessModel1'))
```

INFO:tensorflow:Assets written to: /content/drive/MyDrive/Colab Notebooks/checkpoints/SuccessModel1/assets
None

In []: