In [1]:
```python
# Keras
import keras
from keras import regularizers
from keras.preprocessing import sequence
from keras.preprocessing.text import Tokenizer
# from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential, Model, model_from_json
from keras.layers import Dense, Embedding, LSTM
from keras.layers import Input, Flatten, Dropout, Activation, BatchNormalization
from keras.layers import Conv1D, MaxPooling1D, AveragePooling1D
from keras.utils import np_utils, to_categorical
from keras.callbacks import (EarlyStopping, LearningRateScheduler,
                             ModelCheckpoint, TensorBoard, ReduceLROnPlateau)
from keras import losses, models, optimizers
from keras.activations import relu, softmax
from keras.layers import (Convolution2D, GlobalAveragePooling2D, BatchNormalization, Flatten, Dropout,
                          GlobalMaxPool2D, MaxPool2D, concatenate, Activation, Input, Dense)

# sklearn
from sklearn.metrics import confusion_matrix, accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Other
from tqdm import tqdm, tqdm_pandas
import scipy
from scipy.stats import skew
import librosa
import librosa.display
import json
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from matplotlib.pyplot import specgram
import pandas as pd
import seaborn as sns
import glob
import os
import sys
import IPython.display as ipd  # To play sound in the notebook
import warnings
```

```
42  # ignore warnings
43  if not sys.warnoptions:
44      warnings.simplefilter("ignore")
```

In [2]:
```
1  ref = pd.read_csv("Data_path.csv")
2  ref.head()
```

Out[2]:

	labels	source	path
0	male_surprise	RAVDESS	data/RAVDESS/Actor_01/03-01-08-02-02-01-01.wav
1	male_surprise	RAVDESS	data/RAVDESS/Actor_01/03-01-08-01-01-01-01.wav
2	male_angry	RAVDESS	data/RAVDESS/Actor_01/03-01-05-01-02-01-01.wav
3	male_fear	RAVDESS	data/RAVDESS/Actor_01/03-01-06-01-02-02-01.wav
4	male_fear	RAVDESS	data/RAVDESS/Actor_01/03-01-06-02-01-02-01.wav

In [3]:

```python
'''
1. Data Augmentation method
'''
def speedNpitch(data):
    """
    Speed and Pitch Tuning.
    """
    # you can change low and high here
    length_change = np.random.uniform(low=0.8, high = 1)
    speed_fac = 1.2  / length_change # try changing 1.0 to 2.0 ... =D
    tmp = np.interp(np.arange(0,len(data),speed_fac),np.arange(0,len(data)),data)
    minlen = min(data.shape[0], tmp.shape[0])
    data *= 0
    data[0:minlen] = tmp[0:minlen]
    return data

'''
2. Extracting the MFCC feature as an image (Matrix format).
'''
def prepare_data(df, n, aug, mfcc):
    X = np.empty(shape=(df.shape[0], n, 216, 1))
    input_length = sampling_rate * audio_duration

    cnt = 0
    for fname in tqdm(df.path):
        file_path = fname
        data, _ = librosa.load(file_path, sr=sampling_rate
                               ,res_type="kaiser_fast"
                               ,duration=2.5
                               ,offset=0.5
                               )

        # Random offset / Padding
        if len(data) > input_length:
            max_offset = len(data) - input_length
            offset = np.random.randint(max_offset)
            data = data[offset:(input_length+offset)]
        else:
            if input_length > len(data):
                max_offset = input_length - len(data)
                offset = np.random.randint(max_offset)
```

```python
42              else:
43                  offset = 0
44              data = np.pad(data, (offset, int(input_length) - len(data) - offset), "constant")
45
46          # Augmentation?
47          if aug == 1:
48              data = speedNpitch(data)
49
50          # which feature?
51          if mfcc == 1:
52              # MFCC extraction
53              MFCC = librosa.feature.mfcc(data, sr=sampling_rate, n_mfcc=n_mfcc)
54              MFCC = np.expand_dims(MFCC, axis=-1)
55              X[cnt,] = MFCC
56
57          else:
58              # Log-melspectogram
59              melspec = librosa.feature.melspectrogram(data, n_mels = n_melspec)
60              logspec = librosa.amplitude_to_db(melspec)
61              logspec = np.expand_dims(logspec, axis=-1)
62              X[cnt,] = logspec
63
64          cnt += 1
65
66      return X
67
68
69  '''
70  3. Confusion matrix plot
71  '''
72  def print_confusion_matrix(confusion_matrix, class_names, figsize = (10,7), fontsize=14):
73      '''Prints a confusion matrix, as returned by sklearn.metrics.confusion_matrix, as a heatmap.
74
75      Arguments
76      ---------
77      confusion_matrix: numpy.ndarray
78          The numpy.ndarray object returned from a call to sklearn.metrics.confusion_matrix.
79          Similarly constructed ndarrays can also be used.
80      class_names: list
81          An ordered list of class names, in the order they index the given confusion matrix.
82      figsize: tuple
83          A 2-long tuple, the first value determining the horizontal size of the ouputted figure,
```

```python
 84             the second determining the vertical size. Defaults to (10,7).
 85         fontsize: int
 86             Font size for axes labels. Defaults to 14.
 87
 88         Returns
 89         -------
 90         matplotlib.figure.Figure
 91             The resulting confusion matrix figure
 92         '''
 93         df_cm = pd.DataFrame(
 94             confusion_matrix, index=class_names, columns=class_names,
 95         )
 96         fig = plt.figure(figsize=figsize)
 97         try:
 98             heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
 99         except ValueError:
100             raise ValueError("Confusion matrix values must be integers.")
101
102     heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=fontsize)
103     heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=45, ha='right', fontsize=fontsize
104     plt.ylabel('True label')
105     plt.xlabel('Predicted label')
106
107
108
109 '''
110 # 4. Create the 2D CNN model
111 '''
112 def get_2d_conv_model(n):
113     ''' Create a standard deep 2D convolutional neural network'''
114     nclass = 14
115     inp = Input(shape=(n,216,1))  #2D matrix of 30 MFCC bands by 216 audio length.
116     x = Convolution2D(32, (4,10), padding="same")(inp)
117     x = BatchNormalization()(x)
118     x = Activation("relu")(x)
119     x = MaxPool2D()(x)
120     x = Dropout(rate=0.2)(x)
121
122     x = Convolution2D(32, (4,10), padding="same")(x)
123     x = BatchNormalization()(x)
124     x = Activation("relu")(x)
125     x = MaxPool2D()(x)
```

```python
126         x = Dropout(rate=0.2)(x)
127
128         x = Convolution2D(32, (4,10), padding="same")(x)
129         x = BatchNormalization()(x)
130         x = Activation("relu")(x)
131         x = MaxPool2D()(x)
132         x = Dropout(rate=0.2)(x)
133
134         x = Convolution2D(32, (4,10), padding="same")(x)
135         x = BatchNormalization()(x)
136         x = Activation("relu")(x)
137         x = MaxPool2D()(x)
138         x = Dropout(rate=0.2)(x)
139
140         x = Flatten()(x)
141         x = Dense(64)(x)
142         x = Dropout(rate=0.2)(x)
143         x = BatchNormalization()(x)
144         x = Activation("relu")(x)
145         x = Dropout(rate=0.2)(x)
146
147         out = Dense(nclass, activation=softmax)(x)
148         model = models.Model(inputs=inp, outputs=out)
149
150         opt = optimizers.Adam(0.00001)
151 #         opt = keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
152         model.compile(optimizer=opt, loss=losses.categorical_crossentropy, metrics=['acc'])
153         model.summary()
154         return model
155
156 '''
157 # 5. Other functions
158 '''
159 class get_results:
160     '''
161     We're going to create a class (blueprint template) for generating the results based on the various mode
162     So instead of repeating the functions each time, we assign the results into on object with its associat
163     depending on each combination:
164         1) MFCC with no augmentation
165         2) MFCC with augmentation
166         3) Logmelspec with no augmentation
167         4) Logmelspec with augmentation
```

```python
168        '''
169
170    def __init__(self, model_history, model ,X_test, y_test, labels):
171        self.model_history = model_history
172        self.model = model
173        self.X_test = X_test
174        self.y_test = y_test
175        self.labels = labels
176
177    def create_plot(self, model_history):
178        '''Check the logloss of both train and validation, make sure they are close and have plateau'''
179        plt.plot(model_history.history['loss'])
180        plt.plot(model_history.history['val_loss'])
181        plt.title('model loss')
182        plt.ylabel('loss')
183        plt.xlabel('epoch')
184        plt.legend(['train', 'test'], loc='upper left')
185        plt.show()
186
187    def create_results(self, model):
188        '''predict on test set and get accuracy results'''
189        opt = optimizers.Adam(0.00001)
190        model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
191        score = model.evaluate(X_test, y_test, verbose=0)
192        print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))
193
194    def confusion_results(self, X_test, y_test, labels, model):
195        '''plot confusion matrix results'''
196        preds = model.predict(X_test,
197                              batch_size=16,
198                              verbose=2)
199        preds=preds.argmax(axis=1)
200        preds = preds.astype(int).flatten()
201        preds = (lb.inverse_transform((preds)))
202
203        actual = y_test.argmax(axis=1)
204        actual = actual.astype(int).flatten()
205        actual = (lb.inverse_transform((actual)))
206
207        classes = labels
208        classes.sort()
209
```

```python
210            c = confusion_matrix(actual, preds)
211            print_confusion_matrix(c, class_names = classes)
212
213        def accuracy_results_gender(self, X_test, y_test, labels, model):
214            '''Print out the accuracy score and confusion matrix heat map of the Gender classification results'''
215
216            preds = model.predict(X_test,
217                                  batch_size=16,
218                                  verbose=2)
219            preds=preds.argmax(axis=1)
220            preds = preds.astype(int).flatten()
221            preds = (lb.inverse_transform((preds)))
222
223            actual = y_test.argmax(axis=1)
224            actual = actual.astype(int).flatten()
225            actual = (lb.inverse_transform((actual)))
226
227            # print(accuracy_score(actual, preds))
228
229            actual = pd.DataFrame(actual).replace({'female_angry':'female'
230                        , 'female_disgust':'female'
231                        , 'female_fear':'female'
232                        , 'female_happy':'female'
233                        , 'female_sad':'female'
234                        , 'female_surprise':'female'
235                        , 'female_neutral':'female'
236                        , 'male_angry':'male'
237                        , 'male_fear':'male'
238                        , 'male_happy':'male'
239                        , 'male_sad':'male'
240                        , 'male_surprise':'male'
241                        , 'male_neutral':'male'
242                        , 'male_disgust':'male'
243                    })
244            preds = pd.DataFrame(preds).replace({'female_angry':'female'
245                    , 'female_disgust':'female'
246                    , 'female_fear':'female'
247                    , 'female_happy':'female'
248                    , 'female_sad':'female'
249                    , 'female_surprise':'female'
250                    , 'female_neutral':'female'
251                    , 'male_angry':'male'
```

```python
252                    , 'male_fear':'male'
253                    , 'male_happy':'male'
254                    , 'male_sad':'male'
255                    , 'male_surprise':'male'
256                    , 'male_neutral':'male'
257                    , 'male_disgust':'male'
258                })
259
260        classes = actual.loc[:,0].unique()
261        classes.sort()
262
263        c = confusion_matrix(actual, preds)
264        print(accuracy_score(actual, preds))
265        print_confusion_matrix(c, class_names = classes)
```

In [4]:

```python
sampling_rate=44100
audio_duration=2.5
n_mfcc = 30
mfcc = prepare_data(ref, n = n_mfcc, aug = 0, mfcc = 1)

# Split between train and test
X_train, X_test, y_train, y_test = train_test_split(mfcc
                                        , ref.labels
                                        , test_size=0.25
                                        , shuffle=True
                                        , random_state=42
                                        )


# one hot encode the target
lb = LabelEncoder()
y_train = np_utils.to_categorical(lb.fit_transform(y_train))
y_test = np_utils.to_categorical(lb.fit_transform(y_test))

# Normalization as per the standard NN process
mean = np.mean(X_train, axis=0)
std = np.std(X_train, axis=0)

X_train = (X_train - mean)/std
X_test = (X_test - mean)/std

# Build CNN model
model = get_2d_conv_model(n=n_mfcc)
model_history = model.fit(X_train, y_train, validation_data=(X_test, y_test),
                    batch_size=16, verbose = 2, epochs=100)
```

```
100%|████████████████████████████| 1440/1440 [00:27<00:00, 51.67it/s]
2022-10-17 21:17:19.809997: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:30
6] Could not identify NUMA node of platform GPU ID 0, defaulting to 0. Your kernel may not have been built wi
th NUMA support.
2022-10-17 21:17:19.810113: I tensorflow/core/common_runtime/pluggable_device/pluggable_device_factory.cc:27
2] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 0 MB memory) -> physical Plug
gableDevice (device: 0, name: METAL, pci bus id: <undefined>)

Metal device set to: Apple M2
```

```
Model: "model"


_____
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 30, 216, 1)]      0

 conv2d (Conv2D)             (None, 30, 216, 32)       1312

 batch_normalization (BatchN  (None, 30, 216, 32)      128
 ormalization)

 activation (Activation)     (None, 30, 216, 32)       0

 max_pooling2d (MaxPooling2D  (None, 15, 108, 32)      0
 )

 dropout (Dropout)           (None, 15, 108, 32)       0

 conv2d_1 (Conv2D)           (None, 15, 108, 32)       40992

 batch_normalization_1 (Batc  (None, 15, 108, 32)      128
 hNormalization)

 activation_1 (Activation)   (None, 15, 108, 32)       0

 max_pooling2d_1 (MaxPooling  (None, 7, 54, 32)        0
 2D)

 dropout_1 (Dropout)         (None, 7, 54, 32)         0

 conv2d_2 (Conv2D)           (None, 7, 54, 32)         40992

 batch_normalization_2 (Batc  (None, 7, 54, 32)        128
 hNormalization)

 activation_2 (Activation)   (None, 7, 54, 32)         0

 max_pooling2d_2 (MaxPooling  (None, 3, 27, 32)        0
 2D)

 dropout_2 (Dropout)         (None, 3, 27, 32)         0
```

```
conv2d_3 (Conv2D)            (None, 3, 27, 32)         40992

batch_normalization_3 (Batc  (None, 3, 27, 32)         128
hNormalization)

activation_3 (Activation)    (None, 3, 27, 32)         0

max_pooling2d_3 (MaxPooling  (None, 1, 13, 32)         0
2D)

dropout_3 (Dropout)          (None, 1, 13, 32)         0

flatten (Flatten)            (None, 416)               0

dense (Dense)                (None, 64)                26688

dropout_4 (Dropout)          (None, 64)                0

batch_normalization_4 (Batc  (None, 64)                256
hNormalization)

activation_4 (Activation)    (None, 64)                0

dropout_5 (Dropout)          (None, 64)                0

dense_1 (Dense)              (None, 14)                910

=================================================================
Total params: 152,654
Trainable params: 152,270
Non-trainable params: 384
_____


2022-10-17 21:17:19.960190: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU freq
uency: 0 Hz

Epoch 1/100

2022-10-17 21:17:20.230103: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
2022-10-17 21:17:21.791029: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
```

```
68/68 - 2s - loss: 3.0747 - acc: 0.0750 - val_loss: 2.6810 - val_acc: 0.0806 - 2s/epoch - 30ms/step
Epoch 2/100
68/68 - 1s - loss: 3.0013 - acc: 0.0870 - val_loss: 2.7305 - val_acc: 0.0806 - 1s/epoch - 18ms/step
Epoch 3/100
68/68 - 1s - loss: 2.9593 - acc: 0.0889 - val_loss: 2.7416 - val_acc: 0.0778 - 1s/epoch - 19ms/step
Epoch 4/100
68/68 - 1s - loss: 2.8748 - acc: 0.1000 - val_loss: 2.7273 - val_acc: 0.0889 - 1s/epoch - 18ms/step
Epoch 5/100
68/68 - 1s - loss: 2.8477 - acc: 0.0898 - val_loss: 2.7087 - val_acc: 0.0806 - 1s/epoch - 19ms/step
Epoch 6/100
68/68 - 1s - loss: 2.7820 - acc: 0.1120 - val_loss: 2.6915 - val_acc: 0.0694 - 1s/epoch - 19ms/step
Epoch 7/100
68/68 - 1s - loss: 2.7550 - acc: 0.1120 - val_loss: 2.6857 - val_acc: 0.0639 - 1s/epoch - 19ms/step
Epoch 8/100
68/68 - 1s - loss: 2.7407 - acc: 0.1194 - val_loss: 2.6866 - val_acc: 0.0667 - 1s/epoch - 19ms/step
Epoch 9/100
68/68 - 1s - loss: 2.6761 - acc: 0.1157 - val_loss: 2.6881 - val_acc: 0.0639 - 1s/epoch - 19ms/step
Epoch 10/100
68/68 - 1s - loss: 2.6541 - acc: 0.1278 - val_loss: 2.6991 - val_acc: 0.0639 - 1s/epoch - 18ms/step
Epoch 11/100
68/68 - 1s - loss: 2.6781 - acc: 0.1231 - val_loss: 2.7049 - val_acc: 0.0639 - 1s/epoch - 19ms/step
Epoch 12/100
68/68 - 1s - loss: 2.6270 - acc: 0.1463 - val_loss: 2.7138 - val_acc: 0.0611 - 1s/epoch - 19ms/step
Epoch 13/100
68/68 - 1s - loss: 2.6155 - acc: 0.1417 - val_loss: 2.7138 - val_acc: 0.0639 - 1s/epoch - 19ms/step
Epoch 14/100
68/68 - 1s - loss: 2.5865 - acc: 0.1454 - val_loss: 2.7161 - val_acc: 0.0639 - 1s/epoch - 19ms/step
Epoch 15/100
68/68 - 1s - loss: 2.5646 - acc: 0.1389 - val_loss: 2.7132 - val_acc: 0.0639 - 1s/epoch - 19ms/step
Epoch 16/100
68/68 - 1s - loss: 2.5533 - acc: 0.1370 - val_loss: 2.7169 - val_acc: 0.0583 - 1s/epoch - 18ms/step
Epoch 17/100
68/68 - 1s - loss: 2.5439 - acc: 0.1556 - val_loss: 2.7308 - val_acc: 0.0583 - 1s/epoch - 19ms/step
Epoch 18/100
68/68 - 1s - loss: 2.5120 - acc: 0.1713 - val_loss: 2.7304 - val_acc: 0.0583 - 1s/epoch - 18ms/step
Epoch 19/100
68/68 - 1s - loss: 2.5117 - acc: 0.1769 - val_loss: 2.7274 - val_acc: 0.0556 - 1s/epoch - 18ms/step
Epoch 20/100
68/68 - 1s - loss: 2.4929 - acc: 0.1630 - val_loss: 2.7276 - val_acc: 0.0556 - 1s/epoch - 18ms/step
Epoch 21/100
68/68 - 1s - loss: 2.4799 - acc: 0.1759 - val_loss: 2.7320 - val_acc: 0.0583 - 1s/epoch - 18ms/step
Epoch 22/100
```

```
68/68 - 1s - loss: 2.4704 - acc: 0.1648 - val_loss: 2.7184 - val_acc: 0.0556 - 1s/epoch - 18ms/step
Epoch 23/100
68/68 - 1s - loss: 2.4503 - acc: 0.1806 - val_loss: 2.7146 - val_acc: 0.0611 - 1s/epoch - 18ms/step
Epoch 24/100
68/68 - 1s - loss: 2.4155 - acc: 0.1963 - val_loss: 2.7279 - val_acc: 0.0583 - 1s/epoch - 19ms/step
Epoch 25/100
68/68 - 1s - loss: 2.4263 - acc: 0.1907 - val_loss: 2.7077 - val_acc: 0.0583 - 1s/epoch - 18ms/step
Epoch 26/100
68/68 - 1s - loss: 2.4156 - acc: 0.1824 - val_loss: 2.7075 - val_acc: 0.0583 - 1s/epoch - 19ms/step
Epoch 27/100
68/68 - 1s - loss: 2.3642 - acc: 0.2222 - val_loss: 2.6983 - val_acc: 0.0583 - 1s/epoch - 19ms/step
Epoch 28/100
68/68 - 1s - loss: 2.3845 - acc: 0.2046 - val_loss: 2.6975 - val_acc: 0.0583 - 1s/epoch - 18ms/step
Epoch 29/100
68/68 - 1s - loss: 2.3462 - acc: 0.2167 - val_loss: 2.6973 - val_acc: 0.0611 - 1s/epoch - 19ms/step
Epoch 30/100
68/68 - 1s - loss: 2.3542 - acc: 0.2120 - val_loss: 2.6971 - val_acc: 0.0611 - 1s/epoch - 18ms/step
Epoch 31/100
68/68 - 1s - loss: 2.3218 - acc: 0.2519 - val_loss: 2.6760 - val_acc: 0.0611 - 1s/epoch - 19ms/step
Epoch 32/100
68/68 - 1s - loss: 2.3590 - acc: 0.2176 - val_loss: 2.6692 - val_acc: 0.0611 - 1s/epoch - 18ms/step
Epoch 33/100
68/68 - 1s - loss: 2.3237 - acc: 0.2370 - val_loss: 2.6539 - val_acc: 0.0722 - 1s/epoch - 18ms/step
Epoch 34/100
68/68 - 1s - loss: 2.3286 - acc: 0.2306 - val_loss: 2.6595 - val_acc: 0.0694 - 1s/epoch - 19ms/step
Epoch 35/100
68/68 - 1s - loss: 2.2900 - acc: 0.2398 - val_loss: 2.6572 - val_acc: 0.0694 - 1s/epoch - 18ms/step
Epoch 36/100
68/68 - 1s - loss: 2.2930 - acc: 0.2361 - val_loss: 2.6512 - val_acc: 0.0694 - 1s/epoch - 19ms/step
Epoch 37/100
68/68 - 1s - loss: 2.2882 - acc: 0.2361 - val_loss: 2.6533 - val_acc: 0.0694 - 1s/epoch - 18ms/step
Epoch 38/100
68/68 - 1s - loss: 2.2842 - acc: 0.2389 - val_loss: 2.6265 - val_acc: 0.0694 - 1s/epoch - 19ms/step
Epoch 39/100
68/68 - 1s - loss: 2.2548 - acc: 0.2352 - val_loss: 2.6315 - val_acc: 0.0694 - 1s/epoch - 18ms/step
Epoch 40/100
68/68 - 1s - loss: 2.2602 - acc: 0.2324 - val_loss: 2.6194 - val_acc: 0.0694 - 1s/epoch - 19ms/step
Epoch 41/100
68/68 - 1s - loss: 2.2318 - acc: 0.2611 - val_loss: 2.6094 - val_acc: 0.0722 - 1s/epoch - 18ms/step
Epoch 42/100
68/68 - 1s - loss: 2.2452 - acc: 0.2583 - val_loss: 2.6075 - val_acc: 0.0750 - 1s/epoch - 18ms/step
Epoch 43/100
```

```
68/68 - 1s - loss: 2.2294 - acc: 0.2620 - val_loss: 2.6041 - val_acc: 0.0750 - 1s/epoch - 18ms/step
Epoch 44/100
68/68 - 1s - loss: 2.2315 - acc: 0.2556 - val_loss: 2.5895 - val_acc: 0.0833 - 1s/epoch - 19ms/step
Epoch 45/100
68/68 - 1s - loss: 2.2175 - acc: 0.2444 - val_loss: 2.5847 - val_acc: 0.0861 - 1s/epoch - 18ms/step
Epoch 46/100
68/68 - 1s - loss: 2.2270 - acc: 0.2528 - val_loss: 2.5756 - val_acc: 0.0861 - 1s/epoch - 19ms/step
Epoch 47/100
68/68 - 1s - loss: 2.2031 - acc: 0.2648 - val_loss: 2.5729 - val_acc: 0.0861 - 1s/epoch - 19ms/step
Epoch 48/100
68/68 - 1s - loss: 2.1789 - acc: 0.2741 - val_loss: 2.5716 - val_acc: 0.0861 - 1s/epoch - 19ms/step
Epoch 49/100
68/68 - 1s - loss: 2.1567 - acc: 0.2778 - val_loss: 2.5653 - val_acc: 0.0889 - 1s/epoch - 19ms/step
Epoch 50/100
68/68 - 1s - loss: 2.1816 - acc: 0.2917 - val_loss: 2.5580 - val_acc: 0.0972 - 1s/epoch - 19ms/step
Epoch 51/100
68/68 - 1s - loss: 2.1804 - acc: 0.2648 - val_loss: 2.5543 - val_acc: 0.0944 - 1s/epoch - 19ms/step
Epoch 52/100
68/68 - 1s - loss: 2.1238 - acc: 0.3019 - val_loss: 2.5522 - val_acc: 0.0917 - 1s/epoch - 18ms/step
Epoch 53/100
68/68 - 1s - loss: 2.1275 - acc: 0.3056 - val_loss: 2.5393 - val_acc: 0.0861 - 1s/epoch - 18ms/step
Epoch 54/100
68/68 - 1s - loss: 2.1180 - acc: 0.2963 - val_loss: 2.5172 - val_acc: 0.1139 - 1s/epoch - 19ms/step
Epoch 55/100
68/68 - 1s - loss: 2.1188 - acc: 0.2852 - val_loss: 2.5207 - val_acc: 0.1139 - 1s/epoch - 19ms/step
Epoch 56/100
68/68 - 1s - loss: 2.1291 - acc: 0.2954 - val_loss: 2.5143 - val_acc: 0.1167 - 1s/epoch - 18ms/step
Epoch 57/100
68/68 - 1s - loss: 2.1167 - acc: 0.2954 - val_loss: 2.5122 - val_acc: 0.1167 - 1s/epoch - 19ms/step
Epoch 58/100
68/68 - 1s - loss: 2.0611 - acc: 0.3296 - val_loss: 2.5045 - val_acc: 0.1194 - 1s/epoch - 18ms/step
Epoch 59/100
68/68 - 1s - loss: 2.1113 - acc: 0.2981 - val_loss: 2.4987 - val_acc: 0.1222 - 1s/epoch - 18ms/step
Epoch 60/100
68/68 - 1s - loss: 2.1040 - acc: 0.2963 - val_loss: 2.5112 - val_acc: 0.1194 - 1s/epoch - 19ms/step
Epoch 61/100
68/68 - 1s - loss: 2.0713 - acc: 0.3130 - val_loss: 2.5132 - val_acc: 0.1111 - 1s/epoch - 19ms/step
Epoch 62/100
68/68 - 1s - loss: 2.0991 - acc: 0.3056 - val_loss: 2.4989 - val_acc: 0.1194 - 1s/epoch - 18ms/step
Epoch 63/100
68/68 - 1s - loss: 2.1114 - acc: 0.3083 - val_loss: 2.4898 - val_acc: 0.1222 - 1s/epoch - 19ms/step
Epoch 64/100
```

```
68/68 - 1s - loss: 2.0642 - acc: 0.2935 - val_loss: 2.4757 - val_acc: 0.1361 - 1s/epoch - 19ms/step
Epoch 65/100
68/68 - 1s - loss: 2.0653 - acc: 0.3139 - val_loss: 2.4682 - val_acc: 0.1361 - 1s/epoch - 18ms/step
Epoch 66/100
68/68 - 1s - loss: 2.0630 - acc: 0.3157 - val_loss: 2.4700 - val_acc: 0.1389 - 1s/epoch - 18ms/step
Epoch 67/100
68/68 - 1s - loss: 2.0663 - acc: 0.2972 - val_loss: 2.4730 - val_acc: 0.1389 - 1s/epoch - 19ms/step
Epoch 68/100
68/68 - 1s - loss: 2.0360 - acc: 0.3287 - val_loss: 2.4618 - val_acc: 0.1389 - 1s/epoch - 18ms/step
Epoch 69/100
68/68 - 1s - loss: 2.0402 - acc: 0.3296 - val_loss: 2.4527 - val_acc: 0.1417 - 1s/epoch - 18ms/step
Epoch 70/100
68/68 - 1s - loss: 2.0519 - acc: 0.3241 - val_loss: 2.4527 - val_acc: 0.1417 - 1s/epoch - 19ms/step
Epoch 71/100
68/68 - 1s - loss: 2.0267 - acc: 0.3333 - val_loss: 2.4455 - val_acc: 0.1389 - 1s/epoch - 18ms/step
Epoch 72/100
68/68 - 1s - loss: 1.9898 - acc: 0.3537 - val_loss: 2.4462 - val_acc: 0.1389 - 1s/epoch - 19ms/step
Epoch 73/100
68/68 - 1s - loss: 2.0192 - acc: 0.3343 - val_loss: 2.4413 - val_acc: 0.1389 - 1s/epoch - 18ms/step


Epoch 74/100
68/68 - 1s - loss: 2.0216 - acc: 0.3546 - val_loss: 2.4427 - val_acc: 0.1389 - 1s/epoch - 19ms/step
Epoch 75/100
68/68 - 1s - loss: 2.0146 - acc: 0.3389 - val_loss: 2.4390 - val_acc: 0.1333 - 1s/epoch - 18ms/step
Epoch 76/100
68/68 - 1s - loss: 1.9952 - acc: 0.3417 - val_loss: 2.4341 - val_acc: 0.1417 - 1s/epoch - 19ms/step
Epoch 77/100
68/68 - 1s - loss: 1.9550 - acc: 0.3769 - val_loss: 2.4204 - val_acc: 0.1389 - 1s/epoch - 19ms/step
Epoch 78/100
68/68 - 1s - loss: 1.9937 - acc: 0.3259 - val_loss: 2.4270 - val_acc: 0.1417 - 1s/epoch - 18ms/step
Epoch 79/100
68/68 - 1s - loss: 1.9575 - acc: 0.3491 - val_loss: 2.4182 - val_acc: 0.1389 - 1s/epoch - 18ms/step
Epoch 80/100
68/68 - 1s - loss: 1.9900 - acc: 0.3500 - val_loss: 2.4038 - val_acc: 0.1417 - 1s/epoch - 19ms/step
Epoch 81/100
68/68 - 1s - loss: 1.9595 - acc: 0.3731 - val_loss: 2.4061 - val_acc: 0.1444 - 1s/epoch - 18ms/step
Epoch 82/100
68/68 - 1s - loss: 1.9526 - acc: 0.3602 - val_loss: 2.3983 - val_acc: 0.1444 - 1s/epoch - 18ms/step
Epoch 83/100
68/68 - 1s - loss: 1.9874 - acc: 0.3287 - val_loss: 2.4054 - val_acc: 0.1472 - 1s/epoch - 18ms/step
Epoch 84/100
68/68 - 1s - loss: 1.9621 - acc: 0.3454 - val_loss: 2.3911 - val_acc: 0.1528 - 1s/epoch - 18ms/step
```

```
Epoch 85/100
68/68 - 1s - loss: 1.9320 - acc: 0.3602 - val_loss: 2.3917 - val_acc: 0.1500 - 1s/epoch - 18ms/step
Epoch 86/100
68/68 - 1s - loss: 1.9477 - acc: 0.3657 - val_loss: 2.3832 - val_acc: 0.1556 - 1s/epoch - 18ms/step
Epoch 87/100
68/68 - 1s - loss: 1.9425 - acc: 0.3639 - val_loss: 2.3865 - val_acc: 0.1556 - 1s/epoch - 18ms/step
Epoch 88/100
68/68 - 1s - loss: 1.9368 - acc: 0.3778 - val_loss: 2.3794 - val_acc: 0.1611 - 1s/epoch - 18ms/step
Epoch 89/100
68/68 - 1s - loss: 1.9200 - acc: 0.3787 - val_loss: 2.3624 - val_acc: 0.1917 - 1s/epoch - 18ms/step
Epoch 90/100
68/68 - 1s - loss: 1.9308 - acc: 0.3787 - val_loss: 2.3601 - val_acc: 0.1917 - 1s/epoch - 18ms/step
Epoch 91/100
68/68 - 1s - loss: 1.8975 - acc: 0.3852 - val_loss: 2.3584 - val_acc: 0.1861 - 1s/epoch - 19ms/step
Epoch 92/100
68/68 - 1s - loss: 1.9163 - acc: 0.3630 - val_loss: 2.3527 - val_acc: 0.1944 - 1s/epoch - 19ms/step
Epoch 93/100
68/68 - 1s - loss: 1.8953 - acc: 0.3759 - val_loss: 2.3611 - val_acc: 0.1833 - 1s/epoch - 19ms/step
Epoch 94/100
68/68 - 1s - loss: 1.8814 - acc: 0.3787 - val_loss: 2.3571 - val_acc: 0.1889 - 1s/epoch - 18ms/step
Epoch 95/100
68/68 - 1s - loss: 1.8753 - acc: 0.3954 - val_loss: 2.3476 - val_acc: 0.1917 - 1s/epoch - 18ms/step
Epoch 96/100
68/68 - 1s - loss: 1.8828 - acc: 0.4000 - val_loss: 2.3516 - val_acc: 0.1917 - 1s/epoch - 18ms/step
Epoch 97/100
68/68 - 1s - loss: 1.8983 - acc: 0.3731 - val_loss: 2.3511 - val_acc: 0.1861 - 1s/epoch - 19ms/step
Epoch 98/100
68/68 - 1s - loss: 1.8975 - acc: 0.3685 - val_loss: 2.3551 - val_acc: 0.1833 - 1s/epoch - 18ms/step
Epoch 99/100
68/68 - 1s - loss: 1.8423 - acc: 0.3991 - val_loss: 2.3420 - val_acc: 0.1861 - 1s/epoch - 18ms/step
Epoch 100/100
68/68 - 1s - loss: 1.8691 - acc: 0.3935 - val_loss: 2.3367 - val_acc: 0.1861 - 1s/epoch - 18ms/step
```

```
In [5]:   1  results = get_results(model_history,model,X_test,y_test, ref.labels.unique())
          2  results.create_plot(model_history)
          3  results.create_results(model)
          4  results.confusion_results(X_test, y_test, ref.labels.unique(), model)
```
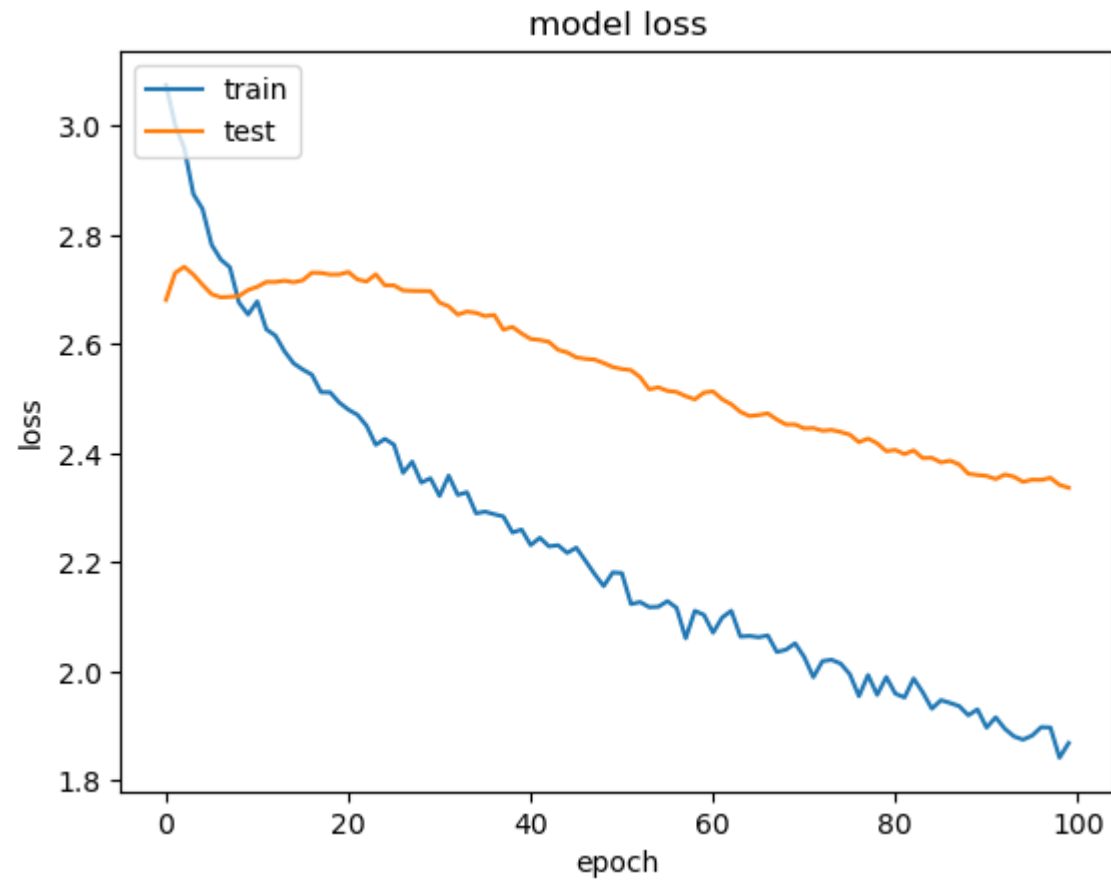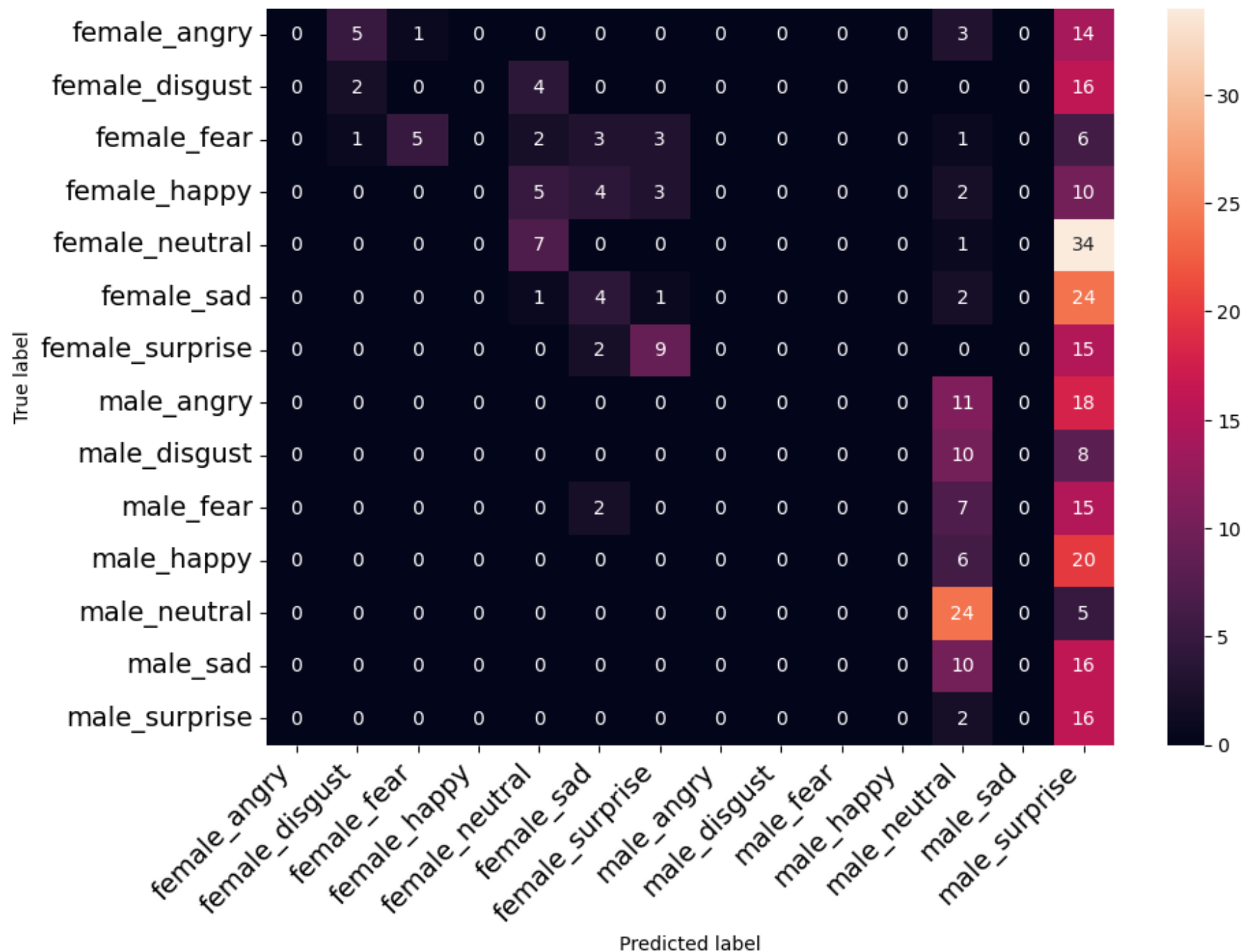
```
2022-10-17 21:19:27.173942: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114]
Plugin optimizer for device_type GPU is enabled.

accuracy: 18.61%
23/23 - 0s - 189ms/epoch - 8ms/step

2022-10-17 21:19:27.411155: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
```

In [ ]:   1