In [1]:

```
# 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
```

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

In [2]:

```
ref = pd.read_csv("Data_path.csv")
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]:

```
Data Augmentation method

: 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


Extracting the MFCC feature as an image (Matrix format).

: 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)
```

```
        else:
            offset = 0
        data = np.pad(data, (offset, int(input_length) - len(data) - offset), "constant")

    # Augmentation?
    if aug == 1:
        data = speedNpitch(data)

    # which feature?
    if mfcc == 1:
        # MFCC extraction
        MFCC = librosa.feature.mfcc(data, sr=sampling_rate, n_mfcc=n_mfcc)
        MFCC = np.expand_dims(MFCC, axis=-1)
        X[cnt,] = MFCC

    else:
        # Log-melspectogram
        melspec = librosa.feature.melspectrogram(data, n_mels = n_melspec)
        logspec = librosa.amplitude_to_db(melspec)
        logspec = np.expand_dims(logspec, axis=-1)
        X[cnt,] = logspec

    cnt += 1

return X
```

Confusion matrix plot

```
print_confusion_matrix(confusion_matrix, class_names, figsize = (10,7), fontsize=14):
'''Prints a confusion matrix, as returned by sklearn.metrics.confusion_matrix, as a heatmap.

Arguments
---------
confusion_matrix: numpy.ndarray
    The numpy.ndarray object returned from a call to sklearn.metrics.confusion_matrix.
    Similarly constructed ndarrays can also be used.
class_names: list
    An ordered list of class names, in the order they index the given confusion matrix.
figsize: tuple
    A 2-long tuple, the first value determining the horizontal size of the ouputted figure,
```

```
    the second determining the vertical size. Defaults to (10,7).
fontsize: int
    Font size for axes labels. Defaults to 14.

Returns
-------
matplotlib.figure.Figure
    The resulting confusion matrix figure
'''
df_cm = pd.DataFrame(
    confusion_matrix, index=class_names, columns=class_names,
)
fig = plt.figure(figsize=figsize)
try:
    heatmap = sns.heatmap(df_cm, annot=True, fmt="d")
except ValueError:
    raise ValueError("Confusion matrix values must be integers.")

heatmap.yaxis.set_ticklabels(heatmap.yaxis.get_ticklabels(), rotation=0, ha='right', fontsize=fontsize)
heatmap.xaxis.set_ticklabels(heatmap.xaxis.get_ticklabels(), rotation=45, ha='right', fontsize=fontsize)
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
. Create the 2D CNN model

get_2d_conv_model(n):
''' Create a standard deep 2D convolutional neural network'''
nclass = 14
inp = Input(shape=(n,216,1))  #2D matrix of 30 MFCC bands by 216 audio length.
x = Convolution2D(32, (4,10), padding="same")(inp)
x = BatchNormalization()(x)
x = Activation("relu")(x)
x = MaxPool2D()(x)
x = Dropout(rate=0.2)(x)

x = Convolution2D(32, (4,10), padding="same")(x)
x = BatchNormalization()(x)
x = Activation("relu")(x)
x = MaxPool2D()(x)
```

```
    x = Dropout(rate=0.2)(x)

    x = Convolution2D(32, (4,10), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = MaxPool2D()(x)
    x = Dropout(rate=0.2)(x)

    x = Convolution2D(32, (4,10), padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = MaxPool2D()(x)
    x = Dropout(rate=0.2)(x)

    x = Flatten()(x)
    x = Dense(64)(x)
    x = Dropout(rate=0.2)(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)
    x = Dropout(rate=0.2)(x)

    out = Dense(nclass, activation=softmax)(x)
    model = models.Model(inputs=inp, outputs=out)

    opt = optimizers.Adam(0.0001)
      opt = keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
    model.compile(optimizer=opt, loss=losses.categorical_crossentropy, metrics=['acc'])
    model.summary()
    return model


;. Other functions

ass get_results:
    '''
    We're going to create a class (blueprint template) for generating the results based on the various model approa
    So instead of repeating the functions each time, we assign the results into on object with its associated varia
    depending on each combination:
        1) MFCC with no augmentation
        2) MFCC with augmentation
        3) Logmelspec with no augmentation
        4) Logmelspec with augmentation
```

```
    '''

    def __init__(self, model_history, model ,X_test, y_test, labels):
        self.model_history = model_history
        self.model = model
        self.X_test = X_test
        self.y_test = y_test
        self.labels = labels

    def create_plot(self, model_history):
        '''Check the logloss of both train and validation, make sure they are close and have plateau'''
        plt.plot(model_history.history['loss'])
        plt.plot(model_history.history['val_loss'])
        plt.title('model loss')
        plt.ylabel('loss')
        plt.xlabel('epoch')
        plt.legend(['train', 'test'], loc='upper left')
        plt.show()

    def create_results(self, model):
        '''predict on test set and get accuracy results'''
        opt = optimizers.Adam(0.0001)
        model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
        score = model.evaluate(X_test, y_test, verbose=0)
        print("%s: %.2f%%" % (model.metrics_names[1], score[1]*100))

    def confusion_results(self, X_test, y_test, labels, model):
        '''plot confusion matrix results'''
        preds = model.predict(X_test,
                                 batch_size=16,
                                 verbose=2)
        preds=preds.argmax(axis=1)
        preds = preds.astype(int).flatten()
        preds = (lb.inverse_transform((preds)))

        actual = y_test.argmax(axis=1)
        actual = actual.astype(int).flatten()
        actual = (lb.inverse_transform((actual)))

        classes = labels
        classes.sort()
```

```
    c = confusion_matrix(actual, preds)
    print_confusion_matrix(c, class_names = classes)

def accuracy_results_gender(self, X_test, y_test, labels, model):
    '''Print out the accuracy score and confusion matrix heat map of the Gender classification results'''

    preds = model.predict(X_test,
                     batch_size=16,
                     verbose=2)
    preds=preds.argmax(axis=1)
    preds = preds.astype(int).flatten()
    preds = (lb.inverse_transform((preds)))

    actual = y_test.argmax(axis=1)
    actual = actual.astype(int).flatten()
    actual = (lb.inverse_transform((actual)))

    # print(accuracy_score(actual, preds))

    actual = pd.DataFrame(actual).replace({'female_angry':'female'
               , 'female_disgust':'female'
               , 'female_fear':'female'
               , 'female_happy':'female'
               , 'female_sad':'female'
               , 'female_surprise':'female'
               , 'female_neutral':'female'
               , 'male_angry':'male'
               , 'male_fear':'male'
               , 'male_happy':'male'
               , 'male_sad':'male'
               , 'male_surprise':'male'
               , 'male_neutral':'male'
               , 'male_disgust':'male'
              })
    preds = pd.DataFrame(preds).replace({'female_angry':'female'
           , 'female_disgust':'female'
           , 'female_fear':'female'
           , 'female_happy':'female'
           , 'female_sad':'female'
           , 'female_surprise':'female'
           , 'female_neutral':'female'
           , 'male_angry':'male'
```

```
            , 'male_fear':'male'
            , 'male_happy':'male'
            , 'male_sad':'male'
            , 'male_surprise':'male'
            , 'male_neutral':'male'
            , 'male_disgust':'male'
           })

classes = actual.loc[:,0].unique()
classes.sort()

c = confusion_matrix(actual, preds)
print(accuracy_score(actual, preds))
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 [00:26<00:00, 55.17it/s]
2022-10-13 19:42:08.105728: 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.
2022-10-13 19:42:08.105853: 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
_________________________________________________________________
Epoch 1/100

2022-10-13 19:42:08.261097: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU freq
uency: 0 Hz
2022-10-13 19:42:08.534538: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
2022-10-13 19:42:10.242488: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.

68/68 - 2s - loss: 2.8575 - acc: 0.0981 - val_loss: 2.6124 - val_acc: 0.0806 - 2s/epoch - 32ms/step
Epoch 2/100
```

```
68/68 - 2s - loss: 2.5870 - acc: 0.1620 - val_loss: 2.6181 - val_acc: 0.0861 - 2s/epoch - 23ms/step
Epoch 3/100
68/68 - 2s - loss: 2.4155 - acc: 0.2083 - val_loss: 2.5662 - val_acc: 0.1667 - 2s/epoch - 22ms/step
Epoch 4/100
68/68 - 1s - loss: 2.2862 - acc: 0.2500 - val_loss: 2.4796 - val_acc: 0.1833 - 1s/epoch - 22ms/step
Epoch 5/100
68/68 - 1s - loss: 2.2068 - acc: 0.2796 - val_loss: 2.4008 - val_acc: 0.2111 - 1s/epoch - 21ms/step
Epoch 6/100
68/68 - 1s - loss: 2.1402 - acc: 0.2815 - val_loss: 2.3866 - val_acc: 0.2056 - 1s/epoch - 22ms/step
Epoch 7/100
68/68 - 1s - loss: 2.1160 - acc: 0.2815 - val_loss: 2.3418 - val_acc: 0.2083 - 1s/epoch - 21ms/step
Epoch 8/100
68/68 - 2s - loss: 2.0489 - acc: 0.3139 - val_loss: 2.3240 - val_acc: 0.2222 - 2s/epoch - 22ms/step
Epoch 9/100
68/68 - 2s - loss: 1.9827 - acc: 0.3417 - val_loss: 2.3093 - val_acc: 0.2083 - 2s/epoch - 22ms/step
Epoch 10/100
68/68 - 1s - loss: 1.9247 - acc: 0.3556 - val_loss: 2.2715 - val_acc: 0.2028 - 1s/epoch - 21ms/step
Epoch 11/100
68/68 - 2s - loss: 1.8982 - acc: 0.3676 - val_loss: 2.2331 - val_acc: 0.2444 - 2s/epoch - 22ms/step
Epoch 12/100
68/68 - 1s - loss: 1.8776 - acc: 0.3806 - val_loss: 2.2547 - val_acc: 0.2139 - 1s/epoch - 21ms/step
Epoch 13/100
68/68 - 1s - loss: 1.8184 - acc: 0.3880 - val_loss: 2.2064 - val_acc: 0.2528 - 1s/epoch - 22ms/step
Epoch 14/100
68/68 - 2s - loss: 1.7730 - acc: 0.4130 - val_loss: 2.2813 - val_acc: 0.2111 - 2s/epoch - 23ms/step
Epoch 15/100
68/68 - 1s - loss: 1.7377 - acc: 0.4241 - val_loss: 2.1901 - val_acc: 0.2333 - 1s/epoch - 22ms/step
Epoch 16/100
68/68 - 2s - loss: 1.7029 - acc: 0.4213 - val_loss: 2.1914 - val_acc: 0.2333 - 2s/epoch - 22ms/step
Epoch 17/100
68/68 - 1s - loss: 1.6724 - acc: 0.4426 - val_loss: 2.1166 - val_acc: 0.2556 - 1s/epoch - 21ms/step
Epoch 18/100
68/68 - 1s - loss: 1.6226 - acc: 0.4833 - val_loss: 2.0280 - val_acc: 0.3028 - 1s/epoch - 21ms/step
Epoch 19/100
68/68 - 1s - loss: 1.6125 - acc: 0.4796 - val_loss: 2.0490 - val_acc: 0.2667 - 1s/epoch - 22ms/step
Epoch 20/100
68/68 - 2s - loss: 1.5606 - acc: 0.4889 - val_loss: 2.0013 - val_acc: 0.3222 - 2s/epoch - 22ms/step
Epoch 21/100
68/68 - 1s - loss: 1.5100 - acc: 0.5111 - val_loss: 2.0951 - val_acc: 0.2778 - 1s/epoch - 22ms/step
Epoch 22/100
68/68 - 1s - loss: 1.4888 - acc: 0.5204 - val_loss: 2.1151 - val_acc: 0.2556 - 1s/epoch - 22ms/step
Epoch 23/100
```

```
68/68 - 1s - loss: 1.4235 - acc: 0.5389 - val_loss: 1.9536 - val_acc: 0.3389 - 1s/epoch - 21ms/step
Epoch 24/100
68/68 - 1s - loss: 1.3786 - acc: 0.5463 - val_loss: 2.0118 - val_acc: 0.3111 - 1s/epoch - 21ms/step
Epoch 25/100
68/68 - 2s - loss: 1.3707 - acc: 0.5611 - val_loss: 1.8299 - val_acc: 0.4000 - 2s/epoch - 22ms/step
Epoch 26/100
68/68 - 1s - loss: 1.3425 - acc: 0.5722 - val_loss: 1.7724 - val_acc: 0.3972 - 1s/epoch - 22ms/step
Epoch 27/100
68/68 - 2s - loss: 1.3013 - acc: 0.5824 - val_loss: 1.7219 - val_acc: 0.4389 - 2s/epoch - 22ms/step
Epoch 28/100
68/68 - 1s - loss: 1.2913 - acc: 0.5889 - val_loss: 1.7850 - val_acc: 0.4111 - 1s/epoch - 21ms/step
Epoch 29/100
68/68 - 1s - loss: 1.2365 - acc: 0.6213 - val_loss: 1.8480 - val_acc: 0.3500 - 1s/epoch - 22ms/step
Epoch 30/100
68/68 - 1s - loss: 1.2134 - acc: 0.6167 - val_loss: 1.6670 - val_acc: 0.4472 - 1s/epoch - 22ms/step
Epoch 31/100
68/68 - 1s - loss: 1.1607 - acc: 0.6389 - val_loss: 1.7579 - val_acc: 0.4028 - 1s/epoch - 22ms/step
Epoch 32/100
68/68 - 1s - loss: 1.1715 - acc: 0.6315 - val_loss: 1.6814 - val_acc: 0.4583 - 1s/epoch - 22ms/step
Epoch 33/100
68/68 - 1s - loss: 1.1155 - acc: 0.6620 - val_loss: 1.7892 - val_acc: 0.4056 - 1s/epoch - 22ms/step
Epoch 34/100
68/68 - 1s - loss: 1.0896 - acc: 0.6769 - val_loss: 1.6164 - val_acc: 0.4639 - 1s/epoch - 21ms/step
Epoch 35/100
68/68 - 1s - loss: 1.0619 - acc: 0.7139 - val_loss: 1.6551 - val_acc: 0.4889 - 1s/epoch - 22ms/step
Epoch 36/100
68/68 - 1s - loss: 1.0350 - acc: 0.7009 - val_loss: 1.6281 - val_acc: 0.5028 - 1s/epoch - 22ms/step
Epoch 37/100
68/68 - 1s - loss: 0.9856 - acc: 0.7315 - val_loss: 1.6014 - val_acc: 0.4861 - 1s/epoch - 22ms/step
Epoch 38/100
68/68 - 1s - loss: 0.9756 - acc: 0.7176 - val_loss: 1.5790 - val_acc: 0.5472 - 1s/epoch - 21ms/step
Epoch 39/100
68/68 - 1s - loss: 0.9644 - acc: 0.7120 - val_loss: 1.6234 - val_acc: 0.4778 - 1s/epoch - 22ms/step
Epoch 40/100
68/68 - 1s - loss: 0.9255 - acc: 0.7380 - val_loss: 1.5927 - val_acc: 0.4778 - 1s/epoch - 22ms/step
Epoch 41/100
68/68 - 1s - loss: 0.8855 - acc: 0.7454 - val_loss: 1.5860 - val_acc: 0.5167 - 1s/epoch - 22ms/step
Epoch 42/100
68/68 - 1s - loss: 0.8378 - acc: 0.7852 - val_loss: 1.7497 - val_acc: 0.3750 - 1s/epoch - 22ms/step
Epoch 43/100
68/68 - 2s - loss: 0.8463 - acc: 0.7824 - val_loss: 1.5845 - val_acc: 0.4472 - 2s/epoch - 22ms/step
Epoch 44/100
```

```
68/68 - 1s - loss: 0.8030 - acc: 0.8065 - val_loss: 1.4710 - val_acc: 0.5750 - 1s/epoch - 21ms/step
Epoch 45/100
68/68 - 1s - loss: 0.8221 - acc: 0.7722 - val_loss: 1.4963 - val_acc: 0.5444 - 1s/epoch - 21ms/step
Epoch 46/100
68/68 - 1s - loss: 0.8074 - acc: 0.7815 - val_loss: 1.4809 - val_acc: 0.5417 - 1s/epoch - 21ms/step
Epoch 47/100
68/68 - 2s - loss: 0.7826 - acc: 0.7954 - val_loss: 1.3635 - val_acc: 0.6167 - 2s/epoch - 23ms/step
Epoch 48/100
68/68 - 2s - loss: 0.7277 - acc: 0.8241 - val_loss: 1.4748 - val_acc: 0.5056 - 2s/epoch - 22ms/step
Epoch 49/100
68/68 - 1s - loss: 0.7155 - acc: 0.8306 - val_loss: 1.4257 - val_acc: 0.5722 - 1s/epoch - 21ms/step
Epoch 50/100
68/68 - 1s - loss: 0.7025 - acc: 0.8287 - val_loss: 1.3656 - val_acc: 0.5583 - 1s/epoch - 21ms/step
Epoch 51/100
68/68 - 1s - loss: 0.7045 - acc: 0.8259 - val_loss: 1.3103 - val_acc: 0.6000 - 1s/epoch - 20ms/step
Epoch 52/100
68/68 - 1s - loss: 0.6790 - acc: 0.8306 - val_loss: 1.5550 - val_acc: 0.5056 - 1s/epoch - 22ms/step
Epoch 53/100
68/68 - 1s - loss: 0.6698 - acc: 0.8407 - val_loss: 1.3281 - val_acc: 0.6056 - 1s/epoch - 21ms/step
Epoch 54/100
68/68 - 1s - loss: 0.6464 - acc: 0.8491 - val_loss: 1.2967 - val_acc: 0.6028 - 1s/epoch - 21ms/step
Epoch 55/100
68/68 - 1s - loss: 0.6386 - acc: 0.8426 - val_loss: 1.2950 - val_acc: 0.6194 - 1s/epoch - 22ms/step
Epoch 56/100
68/68 - 2s - loss: 0.6003 - acc: 0.8556 - val_loss: 1.3186 - val_acc: 0.6056 - 2s/epoch - 22ms/step
Epoch 57/100
68/68 - 2s - loss: 0.5964 - acc: 0.8481 - val_loss: 1.2719 - val_acc: 0.6389 - 2s/epoch - 24ms/step
Epoch 58/100
68/68 - 1s - loss: 0.5678 - acc: 0.8731 - val_loss: 1.2898 - val_acc: 0.6361 - 1s/epoch - 20ms/step
Epoch 59/100
68/68 - 1s - loss: 0.5342 - acc: 0.8917 - val_loss: 1.3336 - val_acc: 0.5972 - 1s/epoch - 22ms/step
Epoch 60/100
68/68 - 1s - loss: 0.5500 - acc: 0.8639 - val_loss: 1.4650 - val_acc: 0.5167 - 1s/epoch - 22ms/step
Epoch 61/100
68/68 - 1s - loss: 0.5001 - acc: 0.9000 - val_loss: 1.3582 - val_acc: 0.5972 - 1s/epoch - 21ms/step
Epoch 62/100
68/68 - 2s - loss: 0.4877 - acc: 0.9046 - val_loss: 1.2042 - val_acc: 0.6472 - 2s/epoch - 22ms/step
Epoch 63/100
68/68 - 1s - loss: 0.4837 - acc: 0.9093 - val_loss: 1.3504 - val_acc: 0.5944 - 1s/epoch - 22ms/step
Epoch 64/100
68/68 - 2s - loss: 0.4778 - acc: 0.8972 - val_loss: 1.1829 - val_acc: 0.6389 - 2s/epoch - 22ms/step
Epoch 65/100
```

```
68/68 - 1s - loss: 0.4835 - acc: 0.8981 - val_loss: 1.1444 - val_acc: 0.6500 - 1s/epoch - 22ms/step
Epoch 66/100
68/68 - 1s - loss: 0.4524 - acc: 0.9019 - val_loss: 1.1783 - val_acc: 0.6361 - 1s/epoch - 22ms/step
Epoch 67/100
68/68 - 1s - loss: 0.4295 - acc: 0.9167 - val_loss: 1.2335 - val_acc: 0.6278 - 1s/epoch - 22ms/step
Epoch 68/100
68/68 - 1s - loss: 0.4374 - acc: 0.9037 - val_loss: 1.3876 - val_acc: 0.5389 - 1s/epoch - 21ms/step
Epoch 69/100
68/68 - 1s - loss: 0.4219 - acc: 0.9028 - val_loss: 1.1054 - val_acc: 0.6444 - 1s/epoch - 22ms/step
Epoch 70/100
68/68 - 1s - loss: 0.3702 - acc: 0.9407 - val_loss: 1.1940 - val_acc: 0.6083 - 1s/epoch - 22ms/step
Epoch 71/100
68/68 - 1s - loss: 0.3713 - acc: 0.9343 - val_loss: 1.0696 - val_acc: 0.6944 - 1s/epoch - 21ms/step
Epoch 72/100
68/68 - 2s - loss: 0.3750 - acc: 0.9296 - val_loss: 1.0556 - val_acc: 0.6944 - 2s/epoch - 22ms/step
Epoch 73/100
68/68 - 1s - loss: 0.3716 - acc: 0.9315 - val_loss: 1.1430 - val_acc: 0.6556 - 1s/epoch - 22ms/step
Epoch 74/100
68/68 - 1s - loss: 0.3628 - acc: 0.9306 - val_loss: 1.0396 - val_acc: 0.7111 - 1s/epoch - 21ms/step
Epoch 75/100
68/68 - 1s - loss: 0.3344 - acc: 0.9407 - val_loss: 1.0764 - val_acc: 0.6806 - 1s/epoch - 21ms/step
Epoch 76/100
68/68 - 1s - loss: 0.3348 - acc: 0.9398 - val_loss: 1.1865 - val_acc: 0.6139 - 1s/epoch - 20ms/step
Epoch 77/100
68/68 - 1s - loss: 0.3073 - acc: 0.9509 - val_loss: 1.1418 - val_acc: 0.6222 - 1s/epoch - 21ms/step
Epoch 78/100
68/68 - 1s - loss: 0.2973 - acc: 0.9574 - val_loss: 1.1402 - val_acc: 0.6139 - 1s/epoch - 22ms/step
Epoch 79/100
68/68 - 1s - loss: 0.2858 - acc: 0.9565 - val_loss: 1.1390 - val_acc: 0.6139 - 1s/epoch - 21ms/step
Epoch 80/100
68/68 - 1s - loss: 0.2935 - acc: 0.9528 - val_loss: 1.0372 - val_acc: 0.6778 - 1s/epoch - 21ms/step
Epoch 81/100
68/68 - 1s - loss: 0.2848 - acc: 0.9556 - val_loss: 1.3880 - val_acc: 0.5333 - 1s/epoch - 21ms/step
Epoch 82/100
68/68 - 1s - loss: 0.2766 - acc: 0.9583 - val_loss: 1.0724 - val_acc: 0.6528 - 1s/epoch - 22ms/step
Epoch 83/100
68/68 - 2s - loss: 0.2673 - acc: 0.9583 - val_loss: 1.0822 - val_acc: 0.6472 - 2s/epoch - 23ms/step
Epoch 84/100
68/68 - 1s - loss: 0.2570 - acc: 0.9667 - val_loss: 0.9465 - val_acc: 0.7333 - 1s/epoch - 22ms/step
Epoch 85/100
68/68 - 1s - loss: 0.2445 - acc: 0.9611 - val_loss: 1.1037 - val_acc: 0.6250 - 1s/epoch - 21ms/step
Epoch 86/100
```

```
68/68 - 1s - loss: 0.2414 - acc: 0.9694 - val_loss: 0.9817 - val_acc: 0.6972 - 1s/epoch - 22ms/step
Epoch 87/100
68/68 - 1s - loss: 0.2534 - acc: 0.9565 - val_loss: 1.0400 - val_acc: 0.6944 - 1s/epoch - 22ms/step
Epoch 88/100
68/68 - 1s - loss: 0.2492 - acc: 0.9556 - val_loss: 1.0416 - val_acc: 0.6889 - 1s/epoch - 22ms/step
Epoch 89/100
68/68 - 1s - loss: 0.2295 - acc: 0.9667 - val_loss: 1.0198 - val_acc: 0.6472 - 1s/epoch - 22ms/step
Epoch 90/100
68/68 - 2s - loss: 0.2277 - acc: 0.9630 - val_loss: 0.9437 - val_acc: 0.7417 - 2s/epoch - 22ms/step
Epoch 91/100
68/68 - 1s - loss: 0.2046 - acc: 0.9741 - val_loss: 0.9185 - val_acc: 0.7167 - 1s/epoch - 21ms/step
Epoch 92/100
68/68 - 1s - loss: 0.1955 - acc: 0.9741 - val_loss: 0.8940 - val_acc: 0.7194 - 1s/epoch - 22ms/step
Epoch 93/100
68/68 - 1s - loss: 0.1877 - acc: 0.9778 - val_loss: 0.9000 - val_acc: 0.7472 - 1s/epoch - 22ms/step
Epoch 94/100
68/68 - 1s - loss: 0.1849 - acc: 0.9806 - val_loss: 1.1208 - val_acc: 0.6417 - 1s/epoch - 21ms/step
Epoch 95/100
68/68 - 1s - loss: 0.1817 - acc: 0.9731 - val_loss: 1.0721 - val_acc: 0.6611 - 1s/epoch - 22ms/step
Epoch 96/100
68/68 - 1s - loss: 0.1948 - acc: 0.9704 - val_loss: 1.0057 - val_acc: 0.6944 - 1s/epoch - 21ms/step
Epoch 97/100
68/68 - 2s - loss: 0.1725 - acc: 0.9778 - val_loss: 0.8932 - val_acc: 0.7194 - 2s/epoch - 22ms/step
Epoch 98/100
68/68 - 1s - loss: 0.1785 - acc: 0.9731 - val_loss: 0.9481 - val_acc: 0.6889 - 1s/epoch - 21ms/step
Epoch 99/100
68/68 - 1s - loss: 0.1688 - acc: 0.9750 - val_loss: 0.8602 - val_acc: 0.7500 - 1s/epoch - 21ms/step
Epoch 100/100
68/68 - 1s - loss: 0.1693 - acc: 0.9806 - val_loss: 0.8983 - val_acc: 0.7083 - 1s/epoch - 22ms/step
```

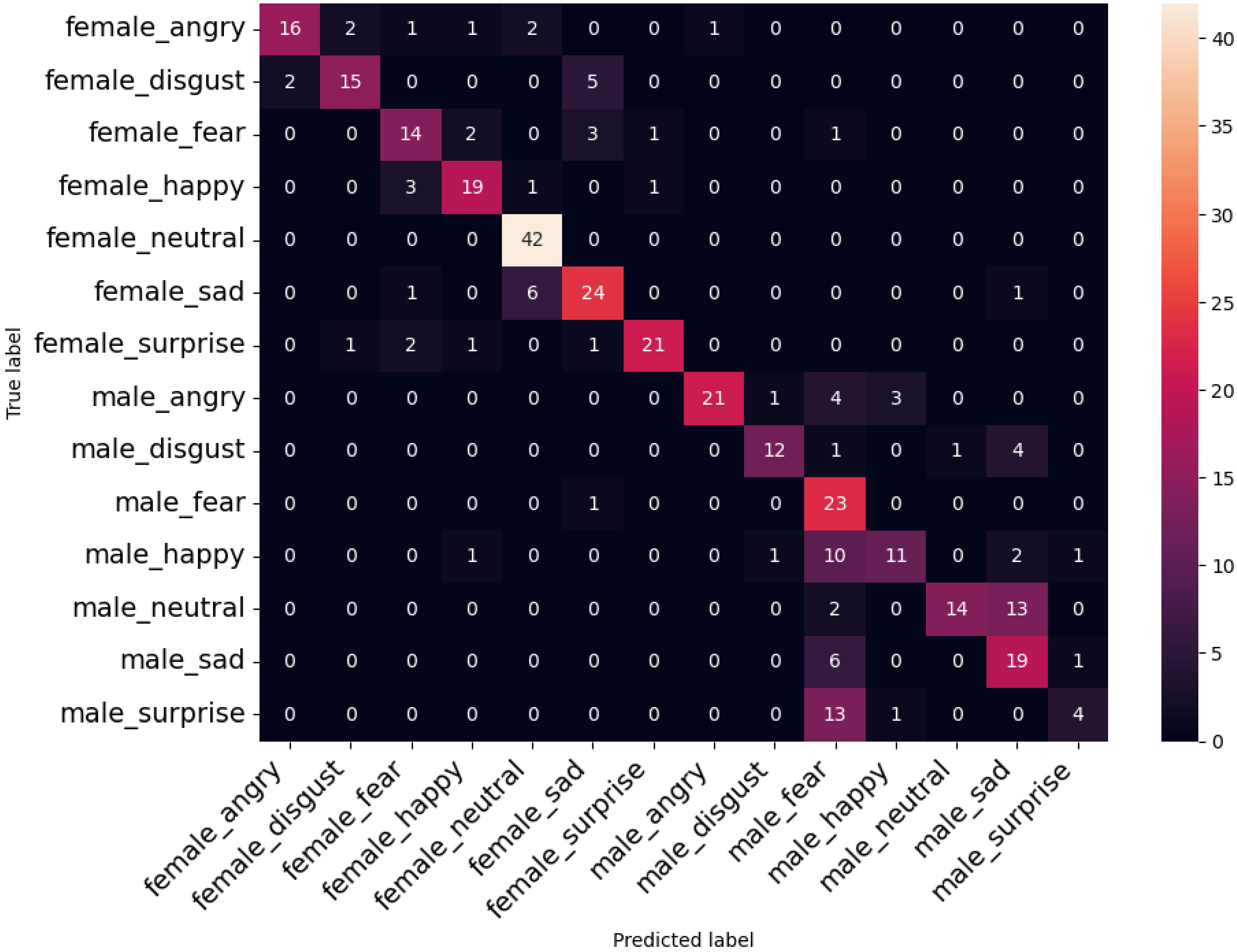
In [5]:

```
results = get_results(model_history,model,X_test,y_test, ref.labels.unique())
results.create_plot(model_history)
results.create_results(model)
results.confusion_results(X_test, y_test, ref.labels.unique(), model)
```

```
2022-10-13 19:44:36.478779: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.

accuracy: 70.83%
23/23 - 0s - 192ms/epoch - 8ms/step

2022-10-13 19:44:36.708834: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
```


female_angry
female_disgust
female_fear
female_happy
female_neutral
female_sad
female_surprise
male_angry
male_disgust
male_fear
male_happy
male_neutral
male_sad
male_surprise
True label
Predicted label
16 2 1 1 2 0 0 1 0 0 0 0 0 0
2 15 0 0 0 5 0 0 0 0 0 0 0 0
0 0 14 2 0 3 1 0 0 1 0 0 0 0
0 0 3 19 1 0 1 0 0 0 0 0 0 0
0 0 0 0 42 0 0 0 0 0 0 0 0 0
0 0 1 0 6 24 0 0 0 0 0 0 1 0
0 1 2 1 0 1 21 0 0 0 0 0 0 0
0 0 0 0 0 0 0 21 1 4 3 0 0 0
0 0 0 0 0 0 0 0 12 1 0 1 4 0
0 0 0 0 0 1 0 0 0 23 0 0 0 0
0 0 0 1 0 0 0 0 1 10 11 0 2 1
0 0 0 0 0 0 0 0 0 2 0 14 13 0
0 0 0 0 0 0 0 0 0 6 0 0 19 1
0 0 0 0 0 0 0 0 0 13 1 0 0 4
0
5
10
15
20
25
30
35
40

In [ ]:

```
```