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
```

```
# 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]:

```
'''
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)
```

```
            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


'''
3. Confusion matrix plot
'''
def 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')



'''
# 4. Create the 2D CNN model
'''
def 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(0.1)
#     opt = keras.optimizers.RMSprop(lr=0.00001, decay=1e-6)
    model.compile(optimizer=opt, loss=losses.categorical_crossentropy, metrics=['acc'])
    model.summary()
    return model

'''
# 5. Other functions
'''
class get_results:
    '''
    We're going to create a class (blueprint template) for generating the results based on the various model
    So instead of repeating the functions each time, we assign the results into on object with its associate
    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(0.1)
        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=200)
```

```
100%|██████████| 1440/1440 [01:25<00:00, 16.87it/s]
2022-10-18 08:41:02.169116: 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.
2022-10-18 08:41:02.169326: 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 1/200

2022-10-18 08:41:02.481950: W tensorflow/core/platform/profile_utils/cpu_utils.cc:128] Failed to get CPU freq
uency: 0 Hz
2022-10-18 08:41:02.935544: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
2022-10-18 08:41:06.047052: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.

68/68 - 4s - loss: 2.6178 - acc: 0.1528 - val_loss: 19.1283 - val_acc: 0.0667 - 4s/epoch - 57ms/step
Epoch 2/200
```

```
68/68 - 3s - loss: 2.2782 - acc: 0.1880 - val_loss: 3.7007 - val_acc: 0.0778 - 3s/epoch - 47ms/step
Epoch 3/200
68/68 - 3s - loss: 2.1883 - acc: 0.1954 - val_loss: 2.0560 - val_acc: 0.2222 - 3s/epoch - 47ms/step
Epoch 4/200
68/68 - 3s - loss: 2.0808 - acc: 0.1954 - val_loss: 5.1535 - val_acc: 0.0667 - 3s/epoch - 47ms/step
Epoch 5/200
68/68 - 3s - loss: 2.0665 - acc: 0.2056 - val_loss: 2.3944 - val_acc: 0.1889 - 3s/epoch - 47ms/step
Epoch 6/200
68/68 - 3s - loss: 2.0302 - acc: 0.2417 - val_loss: 2.0797 - val_acc: 0.1944 - 3s/epoch - 48ms/step
Epoch 7/200
68/68 - 3s - loss: 2.0665 - acc: 0.2120 - val_loss: 2.4760 - val_acc: 0.2306 - 3s/epoch - 47ms/step
Epoch 8/200
68/68 - 3s - loss: 2.0420 - acc: 0.2241 - val_loss: 1.9390 - val_acc: 0.2333 - 3s/epoch - 48ms/step
Epoch 9/200
68/68 - 3s - loss: 1.9872 - acc: 0.2343 - val_loss: 1.9547 - val_acc: 0.2306 - 3s/epoch - 47ms/step
Epoch 10/200
68/68 - 3s - loss: 2.0059 - acc: 0.2370 - val_loss: 2.6919 - val_acc: 0.1528 - 3s/epoch - 47ms/step
Epoch 11/200
68/68 - 3s - loss: 1.9379 - acc: 0.2722 - val_loss: 2.3116 - val_acc: 0.1778 - 3s/epoch - 48ms/step
Epoch 12/200
68/68 - 3s - loss: 1.9133 - acc: 0.2537 - val_loss: 1.8995 - val_acc: 0.2556 - 3s/epoch - 47ms/step
Epoch 13/200
68/68 - 3s - loss: 1.8955 - acc: 0.2611 - val_loss: 2.1285 - val_acc: 0.2472 - 3s/epoch - 48ms/step
Epoch 14/200
68/68 - 3s - loss: 1.9416 - acc: 0.2519 - val_loss: 1.7863 - val_acc: 0.2667 - 3s/epoch - 46ms/step
Epoch 15/200
68/68 - 3s - loss: 1.9057 - acc: 0.2741 - val_loss: 2.3459 - val_acc: 0.1889 - 3s/epoch - 47ms/step
Epoch 16/200
68/68 - 3s - loss: 1.8843 - acc: 0.2778 - val_loss: 1.9028 - val_acc: 0.2417 - 3s/epoch - 48ms/step
Epoch 17/200
68/68 - 3s - loss: 1.8426 - acc: 0.2852 - val_loss: 2.2404 - val_acc: 0.1861 - 3s/epoch - 48ms/step
Epoch 18/200
68/68 - 3s - loss: 1.8603 - acc: 0.3009 - val_loss: 1.7128 - val_acc: 0.2972 - 3s/epoch - 49ms/step
Epoch 19/200
68/68 - 3s - loss: 1.8066 - acc: 0.2852 - val_loss: 1.8845 - val_acc: 0.2667 - 3s/epoch - 47ms/step
Epoch 20/200
68/68 - 3s - loss: 1.8307 - acc: 0.3120 - val_loss: 2.3143 - val_acc: 0.2278 - 3s/epoch - 47ms/step
Epoch 21/200
68/68 - 3s - loss: 1.8529 - acc: 0.2806 - val_loss: 1.9901 - val_acc: 0.1778 - 3s/epoch - 48ms/step
Epoch 22/200
68/68 - 3s - loss: 1.9327 - acc: 0.2852 - val_loss: 1.5852 - val_acc: 0.3250 - 3s/epoch - 48ms/step
Epoch 23/200
```

```
68/68 - 3s - loss: 1.8390 - acc: 0.2870 - val_loss: 2.2660 - val_acc: 0.2417 - 3s/epoch - 48ms/step
Epoch 24/200
68/68 - 3s - loss: 1.7774 - acc: 0.3222 - val_loss: 2.5606 - val_acc: 0.2306 - 3s/epoch - 48ms/step
Epoch 25/200
68/68 - 3s - loss: 1.7492 - acc: 0.3167 - val_loss: 2.0761 - val_acc: 0.1694 - 3s/epoch - 47ms/step
Epoch 26/200
68/68 - 3s - loss: 1.7720 - acc: 0.3120 - val_loss: 1.6043 - val_acc: 0.3611 - 3s/epoch - 49ms/step
Epoch 27/200
68/68 - 5s - loss: 1.7688 - acc: 0.3306 - val_loss: 2.3698 - val_acc: 0.2361 - 5s/epoch - 74ms/step
Epoch 28/200
68/68 - 6s - loss: 1.7844 - acc: 0.3296 - val_loss: 2.0298 - val_acc: 0.2944 - 6s/epoch - 85ms/step
Epoch 29/200
68/68 - 8s - loss: 1.7666 - acc: 0.3204 - val_loss: 2.9376 - val_acc: 0.1806 - 8s/epoch - 120ms/step
Epoch 30/200
68/68 - 5s - loss: 1.8388 - acc: 0.3185 - val_loss: 1.6166 - val_acc: 0.3750 - 5s/epoch - 76ms/step
Epoch 31/200
68/68 - 4s - loss: 1.8358 - acc: 0.3213 - val_loss: 1.9082 - val_acc: 0.3139 - 4s/epoch - 66ms/step
Epoch 32/200
68/68 - 3s - loss: 1.6884 - acc: 0.3389 - val_loss: 1.7914 - val_acc: 0.2972 - 3s/epoch - 50ms/step
Epoch 33/200
68/68 - 4s - loss: 1.6769 - acc: 0.3685 - val_loss: 2.2842 - val_acc: 0.2639 - 4s/epoch - 54ms/step
Epoch 34/200
68/68 - 4s - loss: 1.7355 - acc: 0.3722 - val_loss: 1.5699 - val_acc: 0.4000 - 4s/epoch - 58ms/step
Epoch 35/200
68/68 - 3s - loss: 1.6817 - acc: 0.3620 - val_loss: 2.4321 - val_acc: 0.2139 - 3s/epoch - 50ms/step
Epoch 36/200
68/68 - 3s - loss: 1.6534 - acc: 0.3806 - val_loss: 2.7276 - val_acc: 0.2361 - 3s/epoch - 40ms/step
Epoch 37/200
68/68 - 3s - loss: 1.6920 - acc: 0.3593 - val_loss: 2.0867 - val_acc: 0.2556 - 3s/epoch - 38ms/step
Epoch 38/200
68/68 - 3s - loss: 1.6846 - acc: 0.3806 - val_loss: 1.6418 - val_acc: 0.3778 - 3s/epoch - 39ms/step
Epoch 39/200
68/68 - 3s - loss: 1.5885 - acc: 0.4093 - val_loss: 2.7510 - val_acc: 0.1972 - 3s/epoch - 42ms/step
Epoch 40/200
68/68 - 3s - loss: 1.5671 - acc: 0.4019 - val_loss: 1.6870 - val_acc: 0.3639 - 3s/epoch - 39ms/step
Epoch 41/200
68/68 - 3s - loss: 1.5514 - acc: 0.4315 - val_loss: 1.8881 - val_acc: 0.3472 - 3s/epoch - 42ms/step
Epoch 42/200
68/68 - 3s - loss: 1.5944 - acc: 0.3963 - val_loss: 1.8619 - val_acc: 0.2639 - 3s/epoch - 43ms/step
Epoch 43/200
68/68 - 3s - loss: 1.5607 - acc: 0.4556 - val_loss: 2.7916 - val_acc: 0.2472 - 3s/epoch - 43ms/step
Epoch 44/200
```

```
68/68 - 3s - loss: 1.5524 - acc: 0.4287 - val_loss: 1.3339 - val_acc: 0.4917 - 3s/epoch - 46ms/step
Epoch 45/200
68/68 - 3s - loss: 1.5111 - acc: 0.4361 - val_loss: 2.7685 - val_acc: 0.2944 - 3s/epoch - 45ms/step
Epoch 46/200
68/68 - 3s - loss: 1.5044 - acc: 0.4333 - val_loss: 2.2323 - val_acc: 0.2278 - 3s/epoch - 40ms/step
Epoch 47/200
68/68 - 3s - loss: 1.5023 - acc: 0.4278 - val_loss: 2.1989 - val_acc: 0.3083 - 3s/epoch - 41ms/step
Epoch 48/200
68/68 - 3s - loss: 1.5319 - acc: 0.4361 - val_loss: 1.7002 - val_acc: 0.3389 - 3s/epoch - 40ms/step
Epoch 49/200
68/68 - 3s - loss: 1.5303 - acc: 0.4241 - val_loss: 1.4998 - val_acc: 0.4639 - 3s/epoch - 39ms/step
Epoch 50/200
68/68 - 3s - loss: 1.5578 - acc: 0.4278 - val_loss: 1.6381 - val_acc: 0.3500 - 3s/epoch - 45ms/step
Epoch 51/200
68/68 - 3s - loss: 1.5328 - acc: 0.4259 - val_loss: 1.5328 - val_acc: 0.4472 - 3s/epoch - 41ms/step
Epoch 52/200
68/68 - 3s - loss: 1.4873 - acc: 0.4296 - val_loss: 1.6924 - val_acc: 0.4056 - 3s/epoch - 40ms/step
Epoch 53/200
68/68 - 3s - loss: 1.5793 - acc: 0.4148 - val_loss: 1.8481 - val_acc: 0.3361 - 3s/epoch - 39ms/step
Epoch 54/200
68/68 - 3s - loss: 1.5934 - acc: 0.3935 - val_loss: 1.5187 - val_acc: 0.4306 - 3s/epoch - 39ms/step
Epoch 55/200
68/68 - 3s - loss: 1.5222 - acc: 0.4296 - val_loss: 1.8161 - val_acc: 0.3472 - 3s/epoch - 37ms/step
Epoch 56/200
68/68 - 3s - loss: 1.4012 - acc: 0.4593 - val_loss: 2.0822 - val_acc: 0.3361 - 3s/epoch - 39ms/step
Epoch 57/200
68/68 - 3s - loss: 1.4257 - acc: 0.4565 - val_loss: 2.1821 - val_acc: 0.3500 - 3s/epoch - 38ms/step
Epoch 58/200
68/68 - 3s - loss: 1.3960 - acc: 0.4620 - val_loss: 1.7749 - val_acc: 0.3444 - 3s/epoch - 38ms/step
Epoch 59/200
68/68 - 3s - loss: 1.4164 - acc: 0.4435 - val_loss: 1.4713 - val_acc: 0.4444 - 3s/epoch - 39ms/step
Epoch 60/200
68/68 - 3s - loss: 1.4317 - acc: 0.4546 - val_loss: 1.4801 - val_acc: 0.4611 - 3s/epoch - 38ms/step
Epoch 61/200
68/68 - 3s - loss: 1.4320 - acc: 0.4704 - val_loss: 1.5431 - val_acc: 0.4667 - 3s/epoch - 39ms/step
Epoch 62/200
68/68 - 3s - loss: 1.4297 - acc: 0.4454 - val_loss: 1.9747 - val_acc: 0.3861 - 3s/epoch - 38ms/step
Epoch 63/200
68/68 - 3s - loss: 1.4841 - acc: 0.4509 - val_loss: 1.4295 - val_acc: 0.4333 - 3s/epoch - 39ms/step
Epoch 64/200
68/68 - 3s - loss: 1.4286 - acc: 0.4269 - val_loss: 1.4387 - val_acc: 0.4694 - 3s/epoch - 37ms/step
Epoch 65/200
```

```
68/68 - 3s - loss: 1.4628 - acc: 0.4463 - val_loss: 1.4026 - val_acc: 0.4972 - 3s/epoch - 39ms/step
Epoch 66/200
68/68 - 3s - loss: 1.3702 - acc: 0.4759 - val_loss: 1.8152 - val_acc: 0.3694 - 3s/epoch - 38ms/step
Epoch 67/200
68/68 - 3s - loss: 1.4441 - acc: 0.4583 - val_loss: 2.3550 - val_acc: 0.2917 - 3s/epoch - 39ms/step
Epoch 68/200
68/68 - 3s - loss: 1.4298 - acc: 0.4787 - val_loss: 2.6813 - val_acc: 0.3389 - 3s/epoch - 39ms/step
Epoch 69/200
68/68 - 3s - loss: 1.6669 - acc: 0.4056 - val_loss: 1.5710 - val_acc: 0.3861 - 3s/epoch - 38ms/step
Epoch 70/200
68/68 - 3s - loss: 1.4532 - acc: 0.4398 - val_loss: 1.4380 - val_acc: 0.4222 - 3s/epoch - 39ms/step
Epoch 71/200
68/68 - 3s - loss: 1.4130 - acc: 0.4750 - val_loss: 1.8972 - val_acc: 0.4028 - 3s/epoch - 38ms/step
Epoch 72/200
68/68 - 3s - loss: 1.4187 - acc: 0.4880 - val_loss: 1.6185 - val_acc: 0.3667 - 3s/epoch - 38ms/step
Epoch 73/200
68/68 - 3s - loss: 1.4173 - acc: 0.4481 - val_loss: 1.8679 - val_acc: 0.3361 - 3s/epoch - 38ms/step
Epoch 74/200
68/68 - 3s - loss: 1.3450 - acc: 0.4815 - val_loss: 1.4515 - val_acc: 0.4444 - 3s/epoch - 38ms/step
Epoch 75/200
68/68 - 3s - loss: 1.3387 - acc: 0.4944 - val_loss: 1.7793 - val_acc: 0.4500 - 3s/epoch - 41ms/step
Epoch 76/200
68/68 - 3s - loss: 1.3721 - acc: 0.4769 - val_loss: 1.2318 - val_acc: 0.5056 - 3s/epoch - 39ms/step
Epoch 77/200
68/68 - 3s - loss: 1.3472 - acc: 0.4657 - val_loss: 1.3935 - val_acc: 0.4778 - 3s/epoch - 39ms/step
Epoch 78/200
68/68 - 3s - loss: 1.3477 - acc: 0.4935 - val_loss: 1.6112 - val_acc: 0.4528 - 3s/epoch - 38ms/step
Epoch 79/200
68/68 - 3s - loss: 1.3178 - acc: 0.5176 - val_loss: 2.0010 - val_acc: 0.3750 - 3s/epoch - 39ms/step
Epoch 80/200
68/68 - 3s - loss: 1.3570 - acc: 0.4731 - val_loss: 2.5099 - val_acc: 0.2500 - 3s/epoch - 38ms/step
Epoch 81/200
68/68 - 3s - loss: 1.3954 - acc: 0.5000 - val_loss: 1.6112 - val_acc: 0.4250 - 3s/epoch - 38ms/step
Epoch 82/200
68/68 - 3s - loss: 1.4362 - acc: 0.4463 - val_loss: 1.6592 - val_acc: 0.3750 - 3s/epoch - 38ms/step
Epoch 83/200
68/68 - 3s - loss: 1.2928 - acc: 0.5139 - val_loss: 2.1054 - val_acc: 0.4028 - 3s/epoch - 38ms/step
Epoch 84/200
68/68 - 3s - loss: 1.3915 - acc: 0.4796 - val_loss: 1.6385 - val_acc: 0.4472 - 3s/epoch - 38ms/step
Epoch 85/200
68/68 - 3s - loss: 1.3084 - acc: 0.4917 - val_loss: 2.1112 - val_acc: 0.3056 - 3s/epoch - 38ms/step
Epoch 86/200
```

```
68/68 - 3s - loss: 1.3045 - acc: 0.4972 - val_loss: 1.8152 - val_acc: 0.4389 - 3s/epoch - 38ms/step
Epoch 87/200
68/68 - 3s - loss: 1.3270 - acc: 0.5037 - val_loss: 1.3558 - val_acc: 0.5444 - 3s/epoch - 38ms/step
Epoch 88/200
68/68 - 3s - loss: 1.3049 - acc: 0.4796 - val_loss: 1.9700 - val_acc: 0.3917 - 3s/epoch - 38ms/step
Epoch 89/200
68/68 - 3s - loss: 1.2901 - acc: 0.4963 - val_loss: 1.3768 - val_acc: 0.4889 - 3s/epoch - 38ms/step
Epoch 90/200
68/68 - 3s - loss: 1.3245 - acc: 0.5176 - val_loss: 1.4398 - val_acc: 0.4972 - 3s/epoch - 39ms/step
Epoch 91/200
68/68 - 3s - loss: 1.2681 - acc: 0.5157 - val_loss: 1.8175 - val_acc: 0.4028 - 3s/epoch - 38ms/step
Epoch 92/200
68/68 - 3s - loss: 1.2793 - acc: 0.5213 - val_loss: 1.7038 - val_acc: 0.3972 - 3s/epoch - 39ms/step
Epoch 93/200
68/68 - 3s - loss: 1.3551 - acc: 0.4954 - val_loss: 1.6209 - val_acc: 0.4722 - 3s/epoch - 38ms/step
Epoch 94/200
68/68 - 3s - loss: 1.4654 - acc: 0.4750 - val_loss: 1.3557 - val_acc: 0.4556 - 3s/epoch - 38ms/step
Epoch 95/200
68/68 - 3s - loss: 1.3147 - acc: 0.5065 - val_loss: 1.9236 - val_acc: 0.4028 - 3s/epoch - 38ms/step
Epoch 96/200
68/68 - 3s - loss: 1.2675 - acc: 0.5306 - val_loss: 1.8052 - val_acc: 0.3806 - 3s/epoch - 38ms/step
Epoch 97/200
68/68 - 3s - loss: 1.2576 - acc: 0.5093 - val_loss: 1.5301 - val_acc: 0.4167 - 3s/epoch - 38ms/step
Epoch 98/200
68/68 - 3s - loss: 1.2492 - acc: 0.5472 - val_loss: 1.4067 - val_acc: 0.4444 - 3s/epoch - 38ms/step
Epoch 99/200
68/68 - 3s - loss: 1.1821 - acc: 0.5278 - val_loss: 1.9020 - val_acc: 0.3972 - 3s/epoch - 39ms/step
Epoch 100/200
68/68 - 3s - loss: 1.2340 - acc: 0.5361 - val_loss: 1.4769 - val_acc: 0.4722 - 3s/epoch - 39ms/step
Epoch 101/200
68/68 - 3s - loss: 1.2761 - acc: 0.5148 - val_loss: 1.6666 - val_acc: 0.4750 - 3s/epoch - 38ms/step
Epoch 102/200
68/68 - 3s - loss: 1.4062 - acc: 0.5000 - val_loss: 1.3852 - val_acc: 0.4889 - 3s/epoch - 38ms/step
Epoch 103/200
68/68 - 3s - loss: 1.2726 - acc: 0.5333 - val_loss: 1.8815 - val_acc: 0.4611 - 3s/epoch - 39ms/step
Epoch 104/200
68/68 - 3s - loss: 1.2985 - acc: 0.5139 - val_loss: 1.6417 - val_acc: 0.4167 - 3s/epoch - 39ms/step
Epoch 105/200
68/68 - 3s - loss: 1.3417 - acc: 0.5157 - val_loss: 1.6570 - val_acc: 0.4000 - 3s/epoch - 38ms/step
Epoch 106/200
68/68 - 3s - loss: 1.2480 - acc: 0.5389 - val_loss: 1.1378 - val_acc: 0.5944 - 3s/epoch - 39ms/step
Epoch 107/200
```

```
68/68 - 3s - loss: 1.2055 - acc: 0.5352 - val_loss: 1.2238 - val_acc: 0.5722 - 3s/epoch - 38ms/step
Epoch 108/200
68/68 - 3s - loss: 1.2270 - acc: 0.5380 - val_loss: 1.6200 - val_acc: 0.4611 - 3s/epoch - 38ms/step
Epoch 109/200
68/68 - 3s - loss: 1.1828 - acc: 0.5639 - val_loss: 1.3892 - val_acc: 0.5139 - 3s/epoch - 38ms/step
Epoch 110/200
68/68 - 3s - loss: 1.1994 - acc: 0.5472 - val_loss: 2.7315 - val_acc: 0.3722 - 3s/epoch - 38ms/step
Epoch 111/200
68/68 - 3s - loss: 1.1956 - acc: 0.5537 - val_loss: 1.4706 - val_acc: 0.5000 - 3s/epoch - 39ms/step
Epoch 112/200
68/68 - 3s - loss: 1.1560 - acc: 0.5843 - val_loss: 1.5031 - val_acc: 0.4944 - 3s/epoch - 42ms/step
Epoch 113/200
68/68 - 3s - loss: 1.1691 - acc: 0.5889 - val_loss: 1.6257 - val_acc: 0.4444 - 3s/epoch - 38ms/step
Epoch 114/200
68/68 - 3s - loss: 1.1680 - acc: 0.5491 - val_loss: 1.4974 - val_acc: 0.4917 - 3s/epoch - 39ms/step
Epoch 115/200
68/68 - 3s - loss: 1.1792 - acc: 0.5704 - val_loss: 1.7385 - val_acc: 0.4833 - 3s/epoch - 39ms/step
Epoch 116/200
68/68 - 3s - loss: 1.2009 - acc: 0.5519 - val_loss: 1.5815 - val_acc: 0.4667 - 3s/epoch - 38ms/step
Epoch 117/200
68/68 - 3s - loss: 1.1787 - acc: 0.5778 - val_loss: 1.8552 - val_acc: 0.4167 - 3s/epoch - 38ms/step
Epoch 118/200
68/68 - 3s - loss: 1.2198 - acc: 0.5648 - val_loss: 1.2816 - val_acc: 0.5194 - 3s/epoch - 38ms/step
Epoch 119/200
68/68 - 3s - loss: 1.1744 - acc: 0.5630 - val_loss: 1.9112 - val_acc: 0.4556 - 3s/epoch - 38ms/step
Epoch 120/200
68/68 - 3s - loss: 1.2561 - acc: 0.5583 - val_loss: 1.5013 - val_acc: 0.4972 - 3s/epoch - 39ms/step
Epoch 121/200
68/68 - 3s - loss: 1.2233 - acc: 0.5759 - val_loss: 1.5101 - val_acc: 0.4417 - 3s/epoch - 38ms/step
Epoch 122/200
68/68 - 3s - loss: 1.2081 - acc: 0.5593 - val_loss: 1.5364 - val_acc: 0.5000 - 3s/epoch - 39ms/step
Epoch 123/200
68/68 - 3s - loss: 1.2226 - acc: 0.5491 - val_loss: 1.3646 - val_acc: 0.4917 - 3s/epoch - 39ms/step
Epoch 124/200
68/68 - 3s - loss: 1.2433 - acc: 0.5380 - val_loss: 1.2384 - val_acc: 0.5722 - 3s/epoch - 38ms/step
Epoch 125/200
68/68 - 3s - loss: 1.3128 - acc: 0.5241 - val_loss: 1.4558 - val_acc: 0.4917 - 3s/epoch - 39ms/step
Epoch 126/200
68/68 - 3s - loss: 1.1579 - acc: 0.5657 - val_loss: 1.4729 - val_acc: 0.5167 - 3s/epoch - 39ms/step
Epoch 127/200
68/68 - 3s - loss: 1.1744 - acc: 0.5694 - val_loss: 1.5444 - val_acc: 0.5111 - 3s/epoch - 38ms/step
Epoch 128/200
```

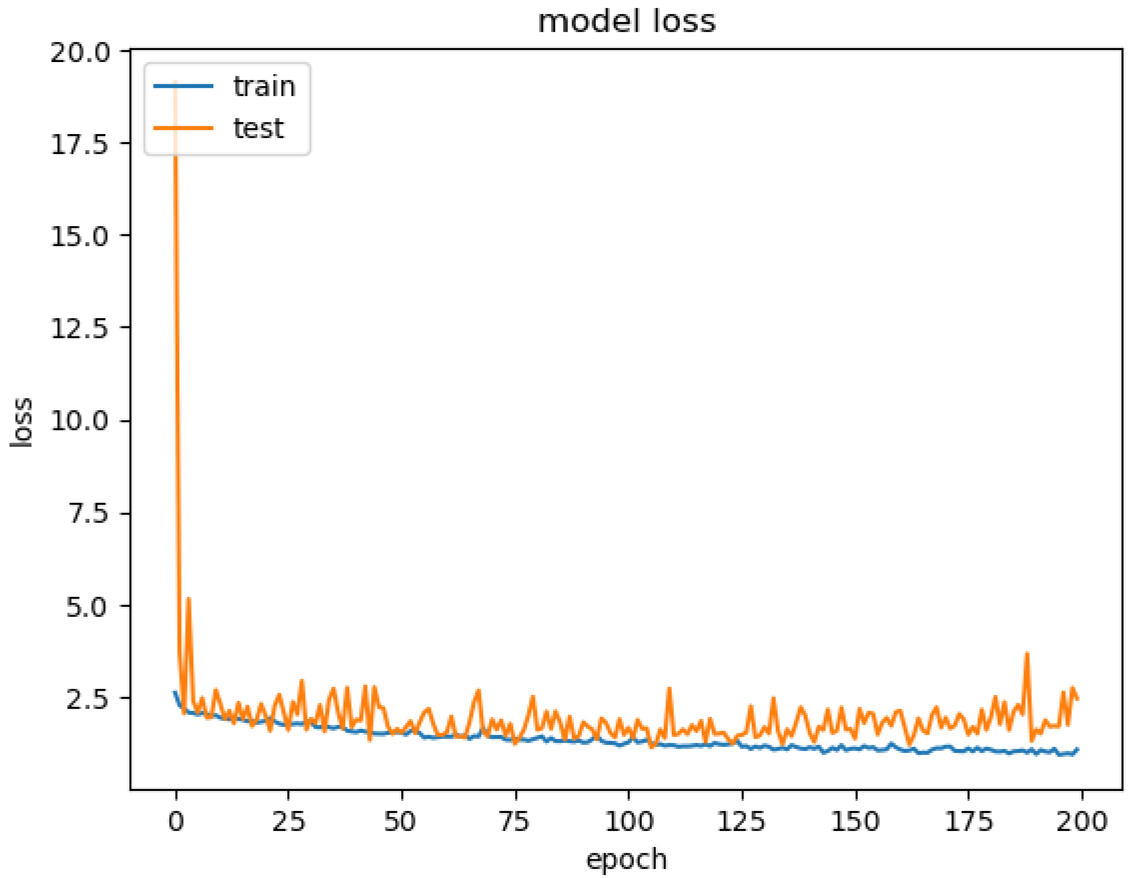
```
68/68 - 3s - loss: 1.0971 - acc: 0.5963 - val_loss: 2.2543 - val_acc: 0.3167 - 3s/epoch - 38ms/step
Epoch 129/200
68/68 - 3s - loss: 1.1651 - acc: 0.5833 - val_loss: 1.4124 - val_acc: 0.4778 - 3s/epoch - 39ms/step
Epoch 130/200
68/68 - 3s - loss: 1.1222 - acc: 0.6028 - val_loss: 1.4629 - val_acc: 0.5222 - 3s/epoch - 38ms/step
Epoch 131/200
68/68 - 3s - loss: 1.1844 - acc: 0.5778 - val_loss: 1.6855 - val_acc: 0.4722 - 3s/epoch - 39ms/step
Epoch 132/200
68/68 - 3s - loss: 1.1545 - acc: 0.5602 - val_loss: 1.5108 - val_acc: 0.4639 - 3s/epoch - 38ms/step
Epoch 133/200
68/68 - 3s - loss: 1.0744 - acc: 0.6009 - val_loss: 2.4666 - val_acc: 0.3667 - 3s/epoch - 38ms/step
Epoch 134/200
68/68 - 3s - loss: 1.0957 - acc: 0.6185 - val_loss: 1.5805 - val_acc: 0.4417 - 3s/epoch - 38ms/step
Epoch 135/200
68/68 - 3s - loss: 1.1293 - acc: 0.5769 - val_loss: 1.2236 - val_acc: 0.5556 - 3s/epoch - 38ms/step
Epoch 136/200
68/68 - 3s - loss: 1.0831 - acc: 0.6148 - val_loss: 1.6269 - val_acc: 0.4722 - 3s/epoch - 39ms/step
Epoch 137/200
68/68 - 3s - loss: 1.1975 - acc: 0.5833 - val_loss: 1.4425 - val_acc: 0.4944 - 3s/epoch - 39ms/step
Epoch 138/200
68/68 - 3s - loss: 1.1482 - acc: 0.5944 - val_loss: 1.8073 - val_acc: 0.4556 - 3s/epoch - 38ms/step
Epoch 139/200
68/68 - 3s - loss: 1.1095 - acc: 0.6148 - val_loss: 2.2234 - val_acc: 0.3806 - 3s/epoch - 39ms/step
Epoch 140/200
68/68 - 3s - loss: 1.0920 - acc: 0.6204 - val_loss: 2.0043 - val_acc: 0.4417 - 3s/epoch - 39ms/step
Epoch 141/200
68/68 - 3s - loss: 1.1418 - acc: 0.5898 - val_loss: 1.5235 - val_acc: 0.4778 - 3s/epoch - 37ms/step
Epoch 142/200
68/68 - 3s - loss: 1.1033 - acc: 0.5815 - val_loss: 1.2835 - val_acc: 0.6056 - 3s/epoch - 39ms/step
Epoch 143/200
68/68 - 3s - loss: 1.1591 - acc: 0.5713 - val_loss: 1.6957 - val_acc: 0.4528 - 3s/epoch - 38ms/step
Epoch 144/200
68/68 - 3s - loss: 0.9944 - acc: 0.6269 - val_loss: 1.6091 - val_acc: 0.4889 - 3s/epoch - 38ms/step
Epoch 145/200
68/68 - 3s - loss: 1.0397 - acc: 0.6324 - val_loss: 2.1645 - val_acc: 0.4250 - 3s/epoch - 39ms/step
Epoch 146/200

68/68 - 3s - loss: 1.1274 - acc: 0.5907 - val_loss: 1.5290 - val_acc: 0.4528 - 3s/epoch - 38ms/step
Epoch 147/200
68/68 - 3s - loss: 1.0662 - acc: 0.6065 - val_loss: 1.5913 - val_acc: 0.5056 - 3s/epoch - 38ms/step
Epoch 148/200
68/68 - 3s - loss: 1.1986 - acc: 0.5657 - val_loss: 2.2167 - val_acc: 0.4167 - 3s/epoch - 38ms/step
```

```
Epoch 149/200
68/68 - 3s - loss: 1.0590 - acc: 0.6241 - val_loss: 1.6192 - val_acc: 0.4806 - 3s/epoch - 38ms/step
Epoch 150/200
68/68 - 3s - loss: 1.1007 - acc: 0.6167 - val_loss: 1.6353 - val_acc: 0.5222 - 3s/epoch - 39ms/step
Epoch 151/200
68/68 - 3s - loss: 1.1127 - acc: 0.6037 - val_loss: 1.3733 - val_acc: 0.5306 - 3s/epoch - 38ms/step
Epoch 152/200
68/68 - 3s - loss: 1.0841 - acc: 0.6083 - val_loss: 2.1821 - val_acc: 0.3583 - 3s/epoch - 38ms/step
Epoch 153/200
68/68 - 3s - loss: 1.1755 - acc: 0.6139 - val_loss: 1.7938 - val_acc: 0.4667 - 3s/epoch - 38ms/step
Epoch 154/200
68/68 - 3s - loss: 1.1226 - acc: 0.5926 - val_loss: 2.0728 - val_acc: 0.4389 - 3s/epoch - 37ms/step
Epoch 155/200
68/68 - 3s - loss: 1.1448 - acc: 0.5861 - val_loss: 2.0459 - val_acc: 0.3861 - 3s/epoch - 39ms/step
Epoch 156/200
68/68 - 3s - loss: 1.0569 - acc: 0.6222 - val_loss: 1.4880 - val_acc: 0.5361 - 3s/epoch - 38ms/step
Epoch 157/200
68/68 - 3s - loss: 1.0655 - acc: 0.6185 - val_loss: 1.8039 - val_acc: 0.4639 - 3s/epoch - 39ms/step
Epoch 158/200
68/68 - 3s - loss: 1.0844 - acc: 0.6046 - val_loss: 1.9389 - val_acc: 0.4389 - 3s/epoch - 38ms/step
Epoch 159/200
68/68 - 3s - loss: 1.2419 - acc: 0.5454 - val_loss: 1.7214 - val_acc: 0.4667 - 3s/epoch - 39ms/step
Epoch 160/200
68/68 - 3s - loss: 1.1404 - acc: 0.5954 - val_loss: 2.0916 - val_acc: 0.3944 - 3s/epoch - 39ms/step
Epoch 161/200
68/68 - 3s - loss: 1.0788 - acc: 0.6102 - val_loss: 2.1285 - val_acc: 0.3611 - 3s/epoch - 38ms/step
Epoch 162/200
68/68 - 3s - loss: 1.0400 - acc: 0.6352 - val_loss: 1.6637 - val_acc: 0.4861 - 3s/epoch - 39ms/step
Epoch 163/200
68/68 - 3s - loss: 1.0564 - acc: 0.6185 - val_loss: 1.2125 - val_acc: 0.5833 - 3s/epoch - 38ms/step
Epoch 164/200
68/68 - 3s - loss: 1.1033 - acc: 0.6102 - val_loss: 1.4693 - val_acc: 0.5250 - 3s/epoch - 39ms/step
Epoch 165/200
68/68 - 3s - loss: 0.9797 - acc: 0.6435 - val_loss: 1.9270 - val_acc: 0.4139 - 3s/epoch - 38ms/step
Epoch 166/200
68/68 - 3s - loss: 0.9973 - acc: 0.6556 - val_loss: 1.5946 - val_acc: 0.5278 - 3s/epoch - 39ms/step
Epoch 167/200
68/68 - 3s - loss: 0.9920 - acc: 0.6444 - val_loss: 1.5192 - val_acc: 0.5278 - 3s/epoch - 38ms/step
Epoch 168/200
68/68 - 3s - loss: 1.0753 - acc: 0.6185 - val_loss: 2.0209 - val_acc: 0.4528 - 3s/epoch - 39ms/step
Epoch 169/200
68/68 - 3s - loss: 1.1183 - acc: 0.5796 - val_loss: 2.2250 - val_acc: 0.3944 - 3s/epoch - 40ms/step
```

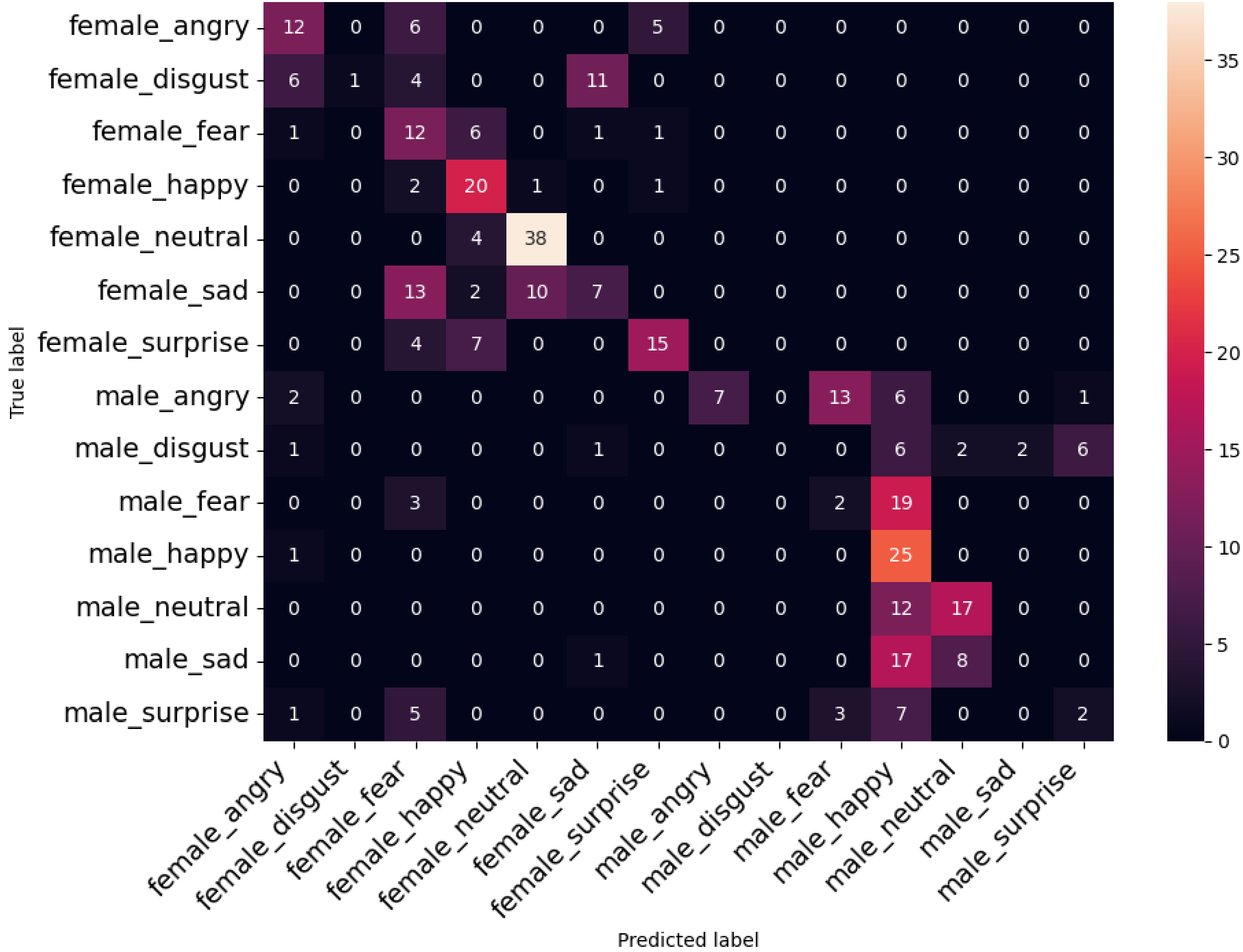
```
Epoch 170/200
68/68 - 3s - loss: 1.1135 - acc: 0.6222 - val_loss: 1.6722 - val_acc: 0.4833 - 3s/epoch - 39ms/step
Epoch 171/200
68/68 - 3s - loss: 1.1577 - acc: 0.6028 - val_loss: 1.9429 - val_acc: 0.4500 - 3s/epoch - 39ms/step
Epoch 172/200
68/68 - 3s - loss: 1.1628 - acc: 0.5889 - val_loss: 1.6536 - val_acc: 0.4472 - 3s/epoch - 38ms/step
Epoch 173/200
68/68 - 3s - loss: 1.0444 - acc: 0.6370 - val_loss: 1.7245 - val_acc: 0.4667 - 3s/epoch - 38ms/step
Epoch 174/200
68/68 - 3s - loss: 1.0371 - acc: 0.6472 - val_loss: 2.0406 - val_acc: 0.4333 - 3s/epoch - 38ms/step
Epoch 175/200
68/68 - 3s - loss: 1.0306 - acc: 0.6306 - val_loss: 1.8996 - val_acc: 0.4889 - 3s/epoch - 39ms/step
Epoch 176/200
68/68 - 3s - loss: 1.1133 - acc: 0.5833 - val_loss: 1.4824 - val_acc: 0.5361 - 3s/epoch - 38ms/step
Epoch 177/200
68/68 - 3s - loss: 1.0278 - acc: 0.6315 - val_loss: 1.6905 - val_acc: 0.4944 - 3s/epoch - 37ms/step
Epoch 178/200
68/68 - 3s - loss: 1.1301 - acc: 0.6111 - val_loss: 1.5090 - val_acc: 0.5333 - 3s/epoch - 39ms/step
Epoch 179/200
68/68 - 3s - loss: 1.0376 - acc: 0.6417 - val_loss: 2.1607 - val_acc: 0.4306 - 3s/epoch - 38ms/step
Epoch 180/200
68/68 - 3s - loss: 1.1055 - acc: 0.6111 - val_loss: 1.6086 - val_acc: 0.4972 - 3s/epoch - 39ms/step
Epoch 181/200
68/68 - 3s - loss: 1.0752 - acc: 0.6046 - val_loss: 1.9564 - val_acc: 0.4194 - 3s/epoch - 38ms/step
Epoch 182/200
68/68 - 3s - loss: 1.0270 - acc: 0.6241 - val_loss: 2.5063 - val_acc: 0.3583 - 3s/epoch - 38ms/step
Epoch 183/200
68/68 - 3s - loss: 1.0222 - acc: 0.6306 - val_loss: 1.7615 - val_acc: 0.4778 - 3s/epoch - 39ms/step
Epoch 184/200
68/68 - 3s - loss: 1.0442 - acc: 0.6398 - val_loss: 2.3656 - val_acc: 0.3444 - 3s/epoch - 38ms/step
Epoch 185/200
68/68 - 3s - loss: 0.9734 - acc: 0.6574 - val_loss: 1.6045 - val_acc: 0.4972 - 3s/epoch - 38ms/step
Epoch 186/200
68/68 - 3s - loss: 1.0282 - acc: 0.6463 - val_loss: 2.0988 - val_acc: 0.4611 - 3s/epoch - 38ms/step
Epoch 187/200
68/68 - 3s - loss: 1.0376 - acc: 0.6389 - val_loss: 2.2873 - val_acc: 0.3833 - 3s/epoch - 38ms/step
Epoch 188/200
68/68 - 3s - loss: 1.0586 - acc: 0.6204 - val_loss: 2.0299 - val_acc: 0.4167 - 3s/epoch - 39ms/step
Epoch 189/200
68/68 - 3s - loss: 0.9972 - acc: 0.6481 - val_loss: 3.6733 - val_acc: 0.3250 - 3s/epoch - 38ms/step
Epoch 190/200
68/68 - 3s - loss: 1.0855 - acc: 0.6259 - val_loss: 1.3134 - val_acc: 0.5944 - 3s/epoch - 38ms/step
```

```
Epoch 191/200
68/68 - 3s - loss: 0.9548 - acc: 0.6380 - val_loss: 1.6190 - val_acc: 0.5111 - 3s/epoch - 39ms/step
Epoch 192/200
68/68 - 3s - loss: 1.0666 - acc: 0.6102 - val_loss: 1.5101 - val_acc: 0.5250 - 3s/epoch - 51ms/step
Epoch 193/200
68/68 - 3s - loss: 1.0245 - acc: 0.6389 - val_loss: 1.8742 - val_acc: 0.4472 - 3s/epoch - 37ms/step
Epoch 194/200
68/68 - 3s - loss: 1.0057 - acc: 0.6537 - val_loss: 1.7021 - val_acc: 0.4778 - 3s/epoch - 38ms/step
Epoch 195/200
68/68 - 3s - loss: 1.1004 - acc: 0.6204 - val_loss: 1.7187 - val_acc: 0.4778 - 3s/epoch - 40ms/step
Epoch 196/200
68/68 - 3s - loss: 0.9375 - acc: 0.6583 - val_loss: 1.7081 - val_acc: 0.5222 - 3s/epoch - 39ms/step
Epoch 197/200
68/68 - 3s - loss: 0.9578 - acc: 0.6481 - val_loss: 2.6274 - val_acc: 0.3917 - 3s/epoch - 38ms/step
Epoch 198/200
68/68 - 3s - loss: 0.9776 - acc: 0.6472 - val_loss: 1.7502 - val_acc: 0.4861 - 3s/epoch - 38ms/step
Epoch 199/200
68/68 - 3s - loss: 0.9467 - acc: 0.6944 - val_loss: 2.7499 - val_acc: 0.3972 - 3s/epoch - 38ms/step
Epoch 200/200
68/68 - 3s - loss: 1.0799 - acc: 0.6019 - val_loss: 2.4617 - val_acc: 0.4389 - 3s/epoch - 38ms/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)
```



```
2022-10-18 08:50:26.439145: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.

accuracy: 43.89%
23/23 - 0s - 264ms/epoch - 11ms/step

2022-10-18 08:50:26.798801: I tensorflow/core/grappler/optimizers/custom_graph_optimizer_registry.cc:114] Plu
gin optimizer for device_type GPU is enabled.
```


True label
Predicted label
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
12 0 6 0 0 0 5 0 0 0 0 0 0 0
6 1 4 0 0 11 0 0 0 0 0 0 0 0
1 0 12 6 0 1 1 0 0 0 0 0 0 0
0 0 2 20 1 0 1 0 0 0 0 0 0 0
0 0 0 4 38 0 0 0 0 0 0 0 0 0
0 0 13 2 10 7 0 0 0 0 0 0 0 0
0 0 4 7 0 0 15 0 0 0 0 0 0 0
2 0 0 0 0 0 0 7 0 13 6 0 0 1
1 0 0 0 0 1 0 0 0 0 6 2 2 6
0 0 3 0 0 0 0 0 0 2 19 0 0 0
1 0 0 0 0 0 0 0 0 0 25 0 0 0
0 0 0 0 0 0 0 0 0 0 12 17 0 0
0 0 0 0 0 1 0 0 0 0 17 8 0 0
1 0 5 0 0 0 0 0 0 3 7 0 0 2
35
30
25
20
15
10
5
0

In [ ]:
```
1
```