In [1]:
```
# Created by: Jessica Gallo
# Date Created: 5/7/2021
# Last Modified: 5/13/2021
# SVM for Face Recognition/Classification
# RGB -> Gaussian Filter -> Histogram Equalization -> Sobel Filter
```

In [2]:
```
# -----
# GPU |
# ----

import tensorflow as tf
print(tf.test.gpu_device_name())
import tensorflow
print(tensorflow.__version__)
import keras
print(keras.__version__)
```

```
2.1.0
2.3.1

Using TensorFlow backend.
```

In [3]:
```
# --------
# IMPORTS |
# --------
import os, shutil
import glob
import pandas as pd
import matplotlib.pyplot as plt
from tensorflow.keras import layers
from tensorflow.keras import models
from tensorflow.keras import optimizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from keras.preprocessing import image  # preprocessing a single image
import numpy as np  # preprocessing a single image
from keras.applications import VGG16  # defining the loss tensor for filter visualiation
from keras import backend as K  # defining the loss tensor for filter visualization
import random
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, auc
from sklearn.metrics import roc_auc_score
from sklearn.metrics import classification_report
import seaborn as sns
```

In [4]:
```
# --------------
# LOAD Dataset |
# -------------

# 113 total pictures in each folder
# 91 total pictures in each folder for training
# 22 total pictures in each folder for testing
# Path to dataset

dataset_dir = './Documents/MATLAB/RGB_Enhanced_Dataset'

os.listdir('./Documents/MATLAB/RGB_Enhanced_Dataset') # returns list
```

Out[4]:
```
['test', 'train']
```

In [5]:
```
# ---------------------
# Mapping directories |
# --------------------

train_dir = './Documents/MATLAB/RGB_Enhanced_Dataset/train'
test_dir = './Documents/MATLAB/RGB_Enhanced_Dataset/test'
```

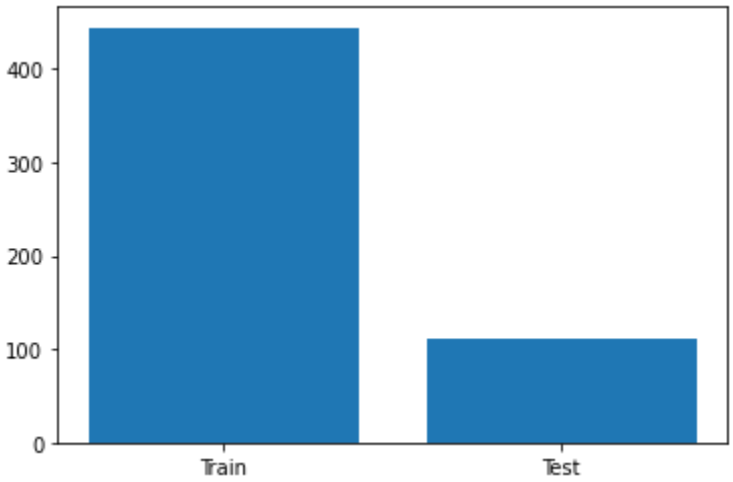
In [6]:
```
# ------------------------------------------
# Listing Amount of Images for Each Expression |
# ------------------------------------------

train_count=[]
val_count=[]
whole_count=[]

print('TRAINING SET')
files= os.listdir(".//Documents//MATLAB//RGB_Enhanced_Dataset//train")
for type in files:
    count = os.listdir('.//Documents//MATLAB//RGB_Enhanced_Dataset//train//'+type+'/')
    print(type+ "  "+ str(len(count)))
    train_count.append(len(count))
print()

print('TEST SET')
files= os.listdir('.//Documents//MATLAB//RGB_Enhanced_Dataset//test')

for type in files:
    count = os.listdir('.//Documents//MATLAB//RGB_Enhanced_Dataset//test//'+type+'/')
    print(type+ "  "+ str(len(count)))
    whole_count.append(len(count))
```

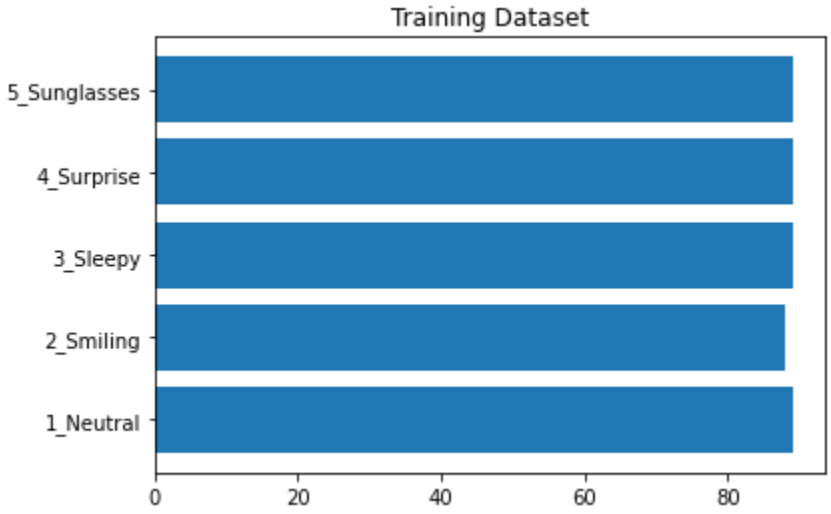
```
TRAINING SET
1_Neutral  89
2_Smiling  88
3_Sleepy  89
4_Surprise  89
5_Sunglasses  89

TEST SET
1_Neutral  22
2_Smiling  22
3_Sleepy  22
4_Surprise  22
5_Sunglasses  22
```

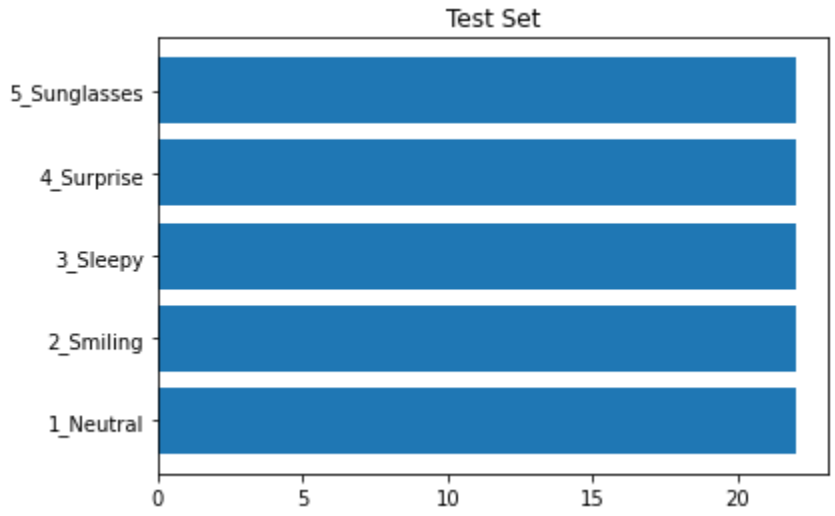
In [7]:
```
# -----------------------------------
# Number of Samples in Each Directory |
# -----------------------------------

# returns a number of items inside the folder
def getNumber(path):
    s = 0
    for i in os.listdir(path):
        if i !='.DS_Store':
            s += len(os.listdir(os.path.join(path,i)))
    return s

n_train = getNumber(train_dir)
n_test = getNumber(test_dir)

print('Number of Samples Train:', n_train)
print('Number of Samples Test:', n_test)
```

```
Number of Samples Train: 444
Number of Samples Test: 110
```

In [8]:
```
# ----------------------------------------
# Plots to Show Data Distribution/Amounts |
# ----------------------------------------

# General Datasets
fig, ax = plt.subplots()
ax.bar(['Train', 'Test'], [n_train, n_test])
plt.show()

# Each expression aracter in training set
plt.barh(files, train_count,)
plt.title('Training Dataset')
plt.show()

# Each expression in test set
plt.barh(files, whole_count,)
plt.title('Test Set')
plt.show()
```



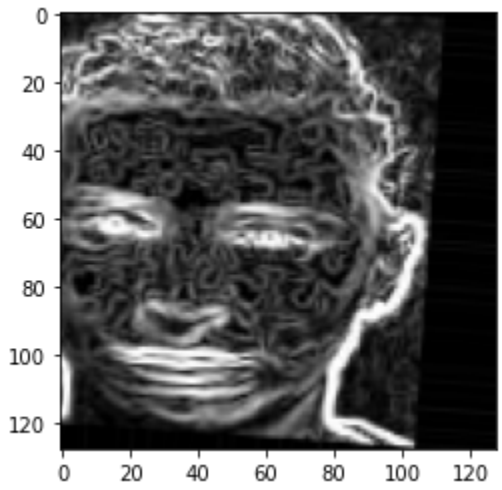


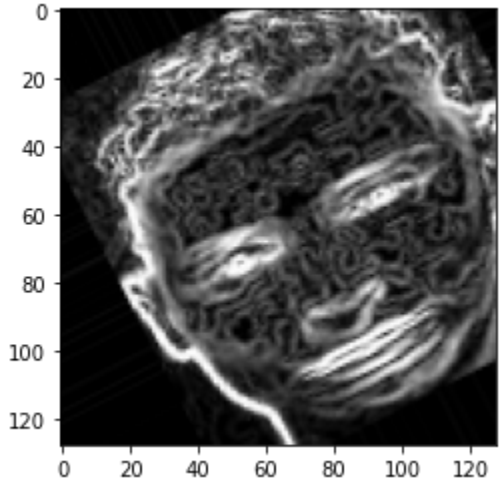


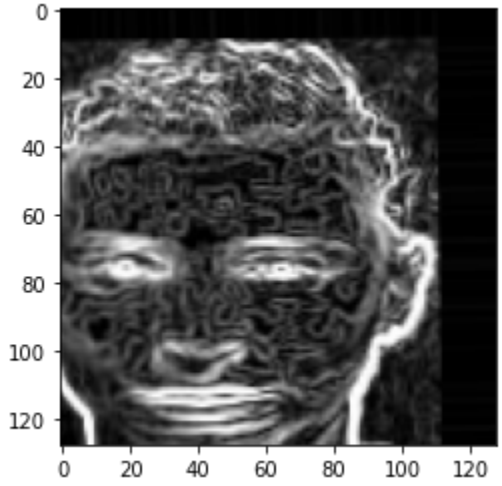
In [9]:
```
# --------------------
# ImageDataGenerator |
# -------------------

# TEST
test_datagen = ImageDataGenerator(rescale=1./255)

# TRAIN
train_datagen = ImageDataGenerator(rescale= 1./255,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   horizontal_flip = True)
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size = (128, 128),
                                                    batch_size = 32,
                                                    class_mode = 'categorical')
# VALIDATION
validation_datagen = ImageDataGenerator(rescale= 1./255,
                                        rotation_range = 40,
                                        width_shift_range = 0.2,
                                        height_shift_range = 0.2,
                                        shear_range = 0.2,
                                        horizontal_flip = True)
validation_generator = test_datagen.flow_from_directory(test_dir,
                                                        target_size = (128, 128),
                                                        batch_size = 32,
                                                        class_mode = 'categorical',
                                                        shuffle=False)
```

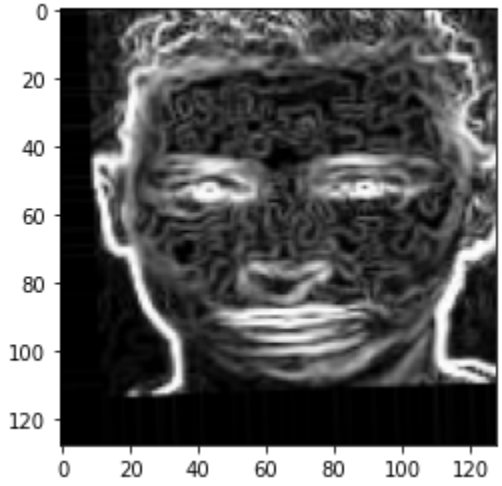
```
Found 444 images belonging to 5 classes.
Found 110 images belonging to 5 classes.
```

In [10]:
```
# ------------------------------------
# Displaing Randomly Augmented Images |
# ------------------------------------

from keras.preprocessing import image
train_dir = './/Documents//MATLAB//RGB_Enhanced_Dataset//train//1_Neutral'

fnames = [os.path.join(train_dir, fname) for
          fname in os.listdir(train_dir)]

img_path = fnames[6]  #choosing an image to augment

img = image.load_img(img_path, target_size = (128, 128))

x = image.img_to_array(img)

x = x.reshape((1,) + x.shape)

i = 0
for batch in train_datagen.flow(x, batch_size=1):
    plt.figure(i)
    imgplot = plt.imshow(image.array_to_img(batch[0]))
    i+=1
    if i%4 == 0:
        break

plt.show()
```




0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120


0
20
40
60
80
100
120
0
20
40
60
80
100
120

In [14]:
```
# MODEL 1
# -------
# ReLU activation function
# Adam optimizer
# 4 Conv
# 1 Batch Norm
# 4 MaxPool
# 1 Dropout
# 3 Dense
# 1 Flatten
# Batchsize 32

model = tensorflow.keras.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu',
                        input_shape = (128, 128, 3),
                        kernel_initializer = 'glorot_normal',
                        bias_initializer = 'zeros'))
model.add(layers.BatchNormalization())
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(128, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Flatten())
model.add(layers.Dropout(.5))
model.add(layers.Dense(512, activation='relu',
                       kernel_initializer = 'glorot_normal',
                       bias_initializer = 'zeros'))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_1 (Batch (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 6, 6, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 4608)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 4608)              0
```

```
_________________________________________________________________
dense_3 (Dense)              (None, 512)               2359808
_________________________________________________________________
dense_4 (Dense)              (None, 512)               262656
_________________________________________________________________
dense_5 (Dense)              (None, 5)                 2565
=================================================================
Total params: 2,865,989
Trainable params: 2,865,925
Non-trainable params: 64
_________________________________________________________________
```

In [15]:

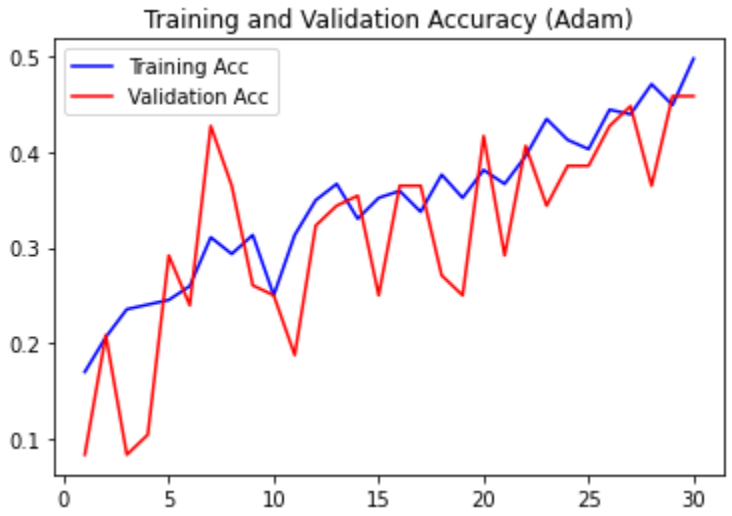
```
adam = optimizers.Adam(lr = 0.001)

model.compile(loss = "categorical_crossentropy",
              optimizer=adam,
              metrics=["acc"])
```

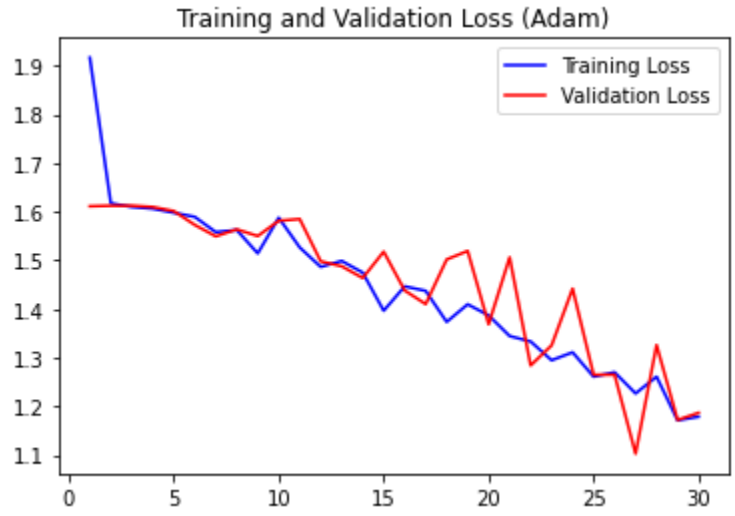
In [16]:
```
batch_size=32

# Training model with adagrad, 30 epochs and shuffling data
history_adam = model.fit_generator(train_generator,
                                   steps_per_epoch=int(444/batch_size),
                                   epochs=30,
                                   validation_data=validation_generator,
                                   validation_steps=int(110/batch_size),
                                   shuffle=True)
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 13 steps, validate for 3 steps
Epoch 1/30
13/13 [==============================] - 21s 2s/step - loss: 1.9139 - acc: 0.1699 - va
l_loss: 1.6111 - val_acc: 0.0833
Epoch 2/30
13/13 [==============================] - 17s 1s/step - loss: 1.6174 - acc: 0.2063 - va
l_loss: 1.6122 - val_acc: 0.2083
Epoch 3/30
13/13 [==============================] - 17s 1s/step - loss: 1.6096 - acc: 0.2354 - va
l_loss: 1.6123 - val_acc: 0.0833
Epoch 4/30
13/13 [==============================] - 18s 1s/step - loss: 1.6059 - acc: 0.2403 - va
l_loss: 1.6095 - val_acc: 0.1042
Epoch 5/30
13/13 [==============================] - 17s 1s/step - loss: 1.5978 - acc: 0.2451 - va
l_loss: 1.6012 - val_acc: 0.2917
Epoch 6/30
13/13 [==============================] - 17s 1s/step - loss: 1.5891 - acc: 0.2597 - va
l_loss: 1.5727 - val_acc: 0.2396
Epoch 7/30
13/13 [==============================] - 17s 1s/step - loss: 1.5578 - acc: 0.3107 - va
l_loss: 1.5493 - val_acc: 0.4271
Epoch 8/30
13/13 [==============================] - 17s 1s/step - loss: 1.5622 - acc: 0.2933 - va
l_loss: 1.5636 - val_acc: 0.3646
Epoch 9/30
13/13 [==============================] - 17s 1s/step - loss: 1.5144 - acc: 0.3131 - va
l_loss: 1.5499 - val_acc: 0.2604
Epoch 10/30
13/13 [==============================] - 10s 752ms/step - loss: 1.5888 - acc: 0.2500 -
val_loss: 1.5813 - val_acc: 0.2500
Epoch 11/30
13/13 [==============================] - 8s 578ms/step - loss: 1.5253 - acc: 0.3131 -
val_loss: 1.5848 - val_acc: 0.1875
Epoch 12/30
13/13 [==============================] - 10s 743ms/step - loss: 1.4866 - acc: 0.3495 -
val_loss: 1.4979 - val_acc: 0.3229
Epoch 13/30
13/13 [==============================] - 10s 733ms/step - loss: 1.4982 - acc: 0.3665 -
val_loss: 1.4878 - val_acc: 0.3438
Epoch 14/30
```

```
13/13 [==============================] - 8s 652ms/step - loss: 1.4756 - acc: 0.3301 -
val_loss: 1.4636 - val_acc: 0.3542
Epoch 15/30
13/13 [==============================] - 8s 613ms/step - loss: 1.3970 - acc: 0.3519 -
val_loss: 1.5178 - val_acc: 0.2500
Epoch 16/30
13/13 [==============================] - 9s 682ms/step - loss: 1.4459 - acc: 0.3592 -
val_loss: 1.4380 - val_acc: 0.3646
Epoch 17/30
13/13 [==============================] - 10s 797ms/step - loss: 1.4367 - acc: 0.3374 -
val_loss: 1.4100 - val_acc: 0.3646
Epoch 18/30
13/13 [==============================] - 9s 679ms/step - loss: 1.3735 - acc: 0.3762 -
val_loss: 1.5018 - val_acc: 0.2708
Epoch 19/30
13/13 [==============================] - 8s 604ms/step - loss: 1.4102 - acc: 0.3519 -
val_loss: 1.5192 - val_acc: 0.2500
Epoch 20/30
13/13 [==============================] - 8s 621ms/step - loss: 1.3858 - acc: 0.3811 -
val_loss: 1.3685 - val_acc: 0.4167
Epoch 21/30
13/13 [==============================] - 9s 669ms/step - loss: 1.3434 - acc: 0.3665 -
val_loss: 1.5061 - val_acc: 0.2917
Epoch 22/30
13/13 [==============================] - 8s 611ms/step - loss: 1.3317 - acc: 0.3956 -
val_loss: 1.2839 - val_acc: 0.4062
Epoch 23/30
13/13 [==============================] - 9s 665ms/step - loss: 1.2943 - acc: 0.4345 -
val_loss: 1.3253 - val_acc: 0.3438
Epoch 24/30
13/13 [==============================] - 9s 682ms/step - loss: 1.3110 - acc: 0.4126 -
val_loss: 1.4418 - val_acc: 0.3854
Epoch 25/30
13/13 [==============================] - 8s 634ms/step - loss: 1.2613 - acc: 0.4029 -
val_loss: 1.2648 - val_acc: 0.3854
Epoch 26/30
13/13 [==============================] - 9s 730ms/step - loss: 1.2704 - acc: 0.4442 -
val_loss: 1.2649 - val_acc: 0.4271
Epoch 27/30
13/13 [==============================] - 9s 710ms/step - loss: 1.2252 - acc: 0.4393 -
val_loss: 1.1026 - val_acc: 0.4479
Epoch 28/30
13/13 [==============================] - 8s 636ms/step - loss: 1.2611 - acc: 0.4709 -
val_loss: 1.3259 - val_acc: 0.3646
Epoch 29/30
13/13 [==============================] - 8s 616ms/step - loss: 1.1703 - acc: 0.4490 -
val_loss: 1.1719 - val_acc: 0.4583
Epoch 30/30
13/13 [==============================] - 9s 654ms/step - loss: 1.1776 - acc: 0.4976 -
val_loss: 1.1865 - val_acc: 0.4583
```

In [17]:
```
# ---------------------------------------------
# Visualizing Train/Validation Loss & Accuracy |
# ---------------------------------------------

acc_adam = history_adam.history['acc']
val_acc_adam = history_adam.history['val_acc']
loss_adam = history_adam.history['loss']
val_loss_adam = history_adam.history['val_loss']

epochs_adam = range(1,len(acc_adam) +1)

# Plot of accuracy
plt.plot(epochs_adam, acc_adam, color='blue', label='Training Acc')
plt.plot(epochs_adam, val_acc_adam, color='red', label='Validation Acc')
plt.title('Training and Validation Accuracy (Adam)')
plt.legend()

plt.figure()

# Plot of loss
plt.plot(epochs_adam, loss_adam, color='blue', label='Training Loss')
plt.plot(epochs_adam, val_loss_adam, color='red', label='Validation Loss')
plt.title('Training and Validation Loss (Adam)')
plt.legend()
plt.figure()

plt.show()
```





<Figure size 432x288 with 0 Axes>

In [18]:
```
# ---------------
# SAVE THE MODEL |
# ---------------
model.save('DIP_Proj_modelAdam.h5')
```

In [24]:
```
# ------------------
# Confusion Matrix |
# ------------------

num_of_train_samples = 444
#num_of_test_samples = 416 # steps per epoch
batch_size=32
steps_per_epoch=num_of_train_samples // batch_size
#validation_generator.reset()

Y_pred = model.predict_grenerator(validation_generator, steps_per_epoch)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix for Model with Adam Optimizer')
print(confusion_matrix(validation_generator.classes, y_pred))
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-24-2188fe66f14c> in <module>
      9 #validation_generator.reset()
     10
---> 11 Y_pred = model.predict_grenerator(validation_generator, steps_per_epoch)
     12 y_pred = np.argmax(Y_pred, axis=1)
     13 print('Confusion Matrix for Model with Adam Optimizer')

AttributeError: 'Sequential' object has no attribute 'predict_grenerator'
```

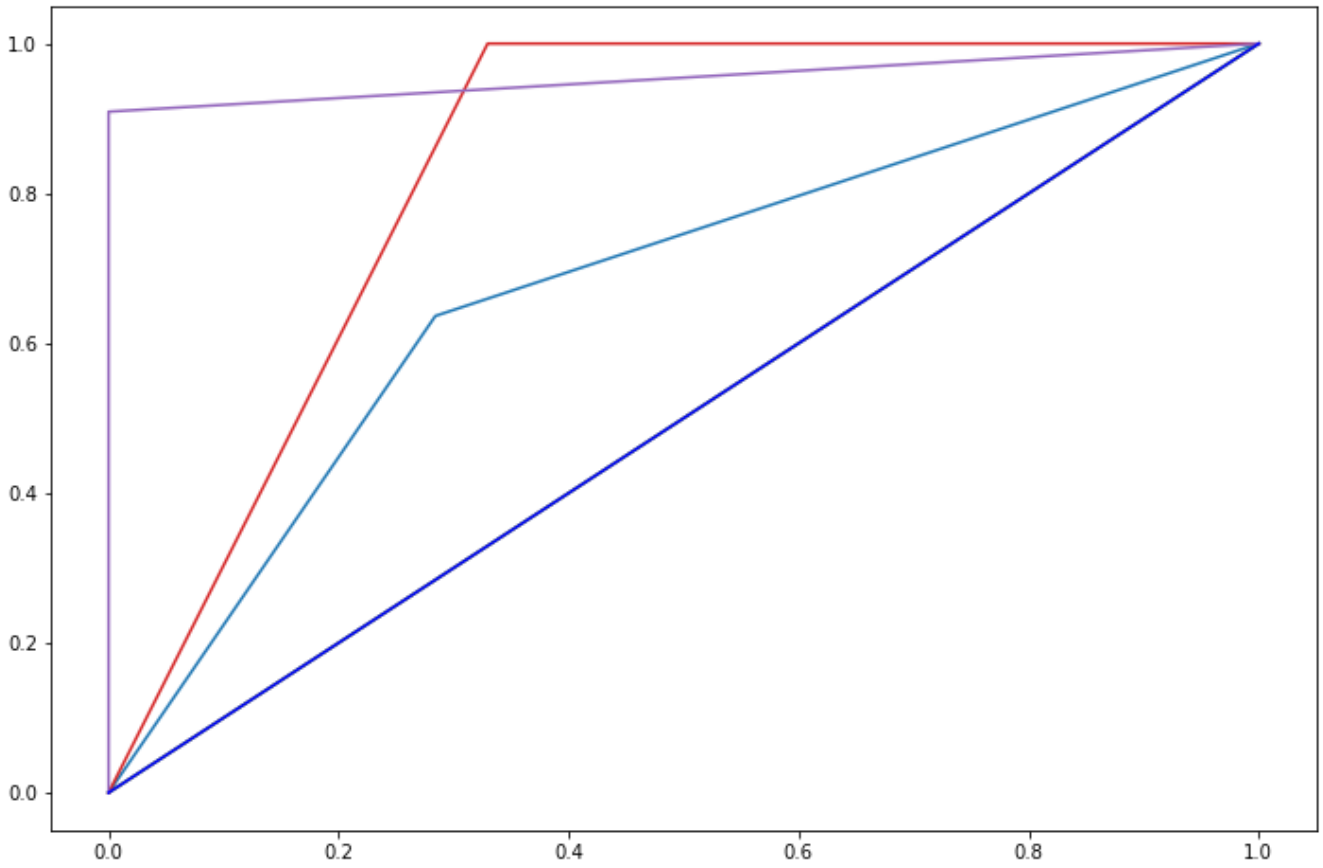
In [25]:
```
# ----------------------
# Classification Report |
# ----------------------

print('Classification Report for Model with Adam Optimizer')
target_names = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']
print(classification_report(validation_generator.classes, y_pred, target_names=target_names
```

```
Classification Report for Model with Adam Optimizer
              precision    recall  f1-score   support

   1_Neutral       0.36      0.64      0.46        22
   2_Smiling       0.00      0.00      0.00        22
    3_Sleepy       0.00      0.00      0.00        22
  4_Surprise       0.43      1.00      0.60        22
5_Sunglasses       1.00      0.91      0.95        22

    accuracy                           0.51       110
   macro avg       0.36      0.51      0.40       110
weighted avg       0.36      0.51      0.40       110


C:\Users\User\anaconda3\envs\tensorflow\lib\site-packages\sklearn\metrics\_classificatio
n.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
```

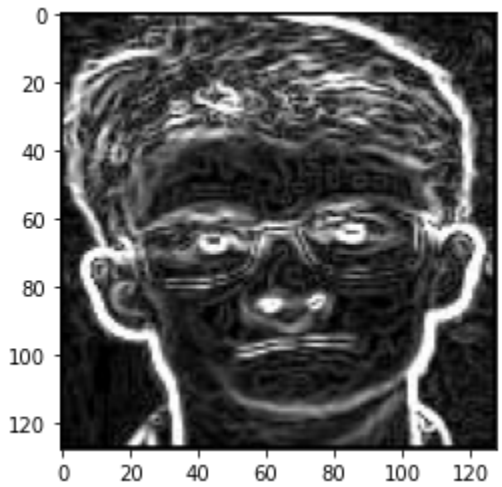
In [22]:
```
# ---------------
# ROC/AUC Score |
# --------------

from sklearn.preprocessing import LabelBinarizer

# set plot figure size
fig, c_ax = plt.subplots(1,1, figsize = (12, 8))

def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)

    for (idx, c_label) in enumerate(target_names): # target_names: no of the labels
        fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), y_pred[:,idx])
        c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
    c_ax.plot(fpr, fpr, 'b-', label = 'Random Guessing')
    return roc_auc_score(y_test, y_pred, average=average)

validation_generator.reset() # resetting generator
y_pred = model.predict_generator(validation_generator, verbose = True)
y_pred = np.argmax(y_pred, axis=1)
multiclass_roc_auc_score(validation_generator.classes, y_pred)
```

```
WARNING:tensorflow:From <ipython-input-22-470a0699b29d>:23: Model.predict_generator (from
tensorflow.python.keras.engine.training) is deprecated and will be removed in a future ve
rsion.
Instructions for updating:
Please use Model.predict, which supports generators.
4/4 [==============================] - 0s 107ms/step
```

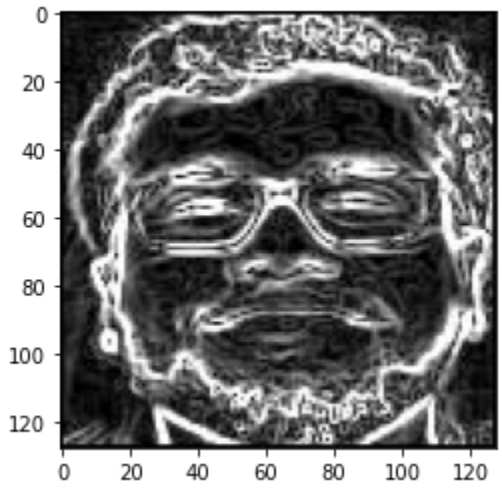
Out[22]:
```
0.6931818181818181
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

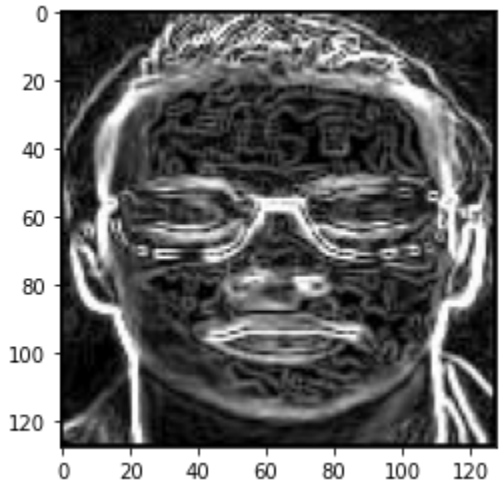
In [28]:
```
# ----------------------------------------
# Displaying 12 images with the prediction |
# ----------------------------------------

# 1 image from each class

labels = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']

i=0
for names in labels:
    pathToFolder = test_dir +'/'+names+'/'
    fnames = [os.path.join(pathToFolder, fname) for
          fname in os.listdir(pathToFolder)]

    # generate random number btwn (0,30)
    randNum = random.randint(0, 20)
    img_path = fnames[randNum]
    tmp_img = image.load_img(img_path, target_size = (128, 128))
    tmp_img = image.img_to_array(tmp_img)
    tmp_img = np.expand_dims(tmp_img, axis = 0)

    tmp_img /=255.
    plt.imshow(tmp_img[0])
    plt.show()
    # predict
    result = model.predict(tmp_img)
    train_generator.class_indices
    print('Actual value: ',i,'\tPredicted value: ', np.argmax(result))
    i+=1

print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size = (128, 128),
                                                  batch_size = 32,
                                                  class_mode = "categorical",
                                                  shuffle=False)
```



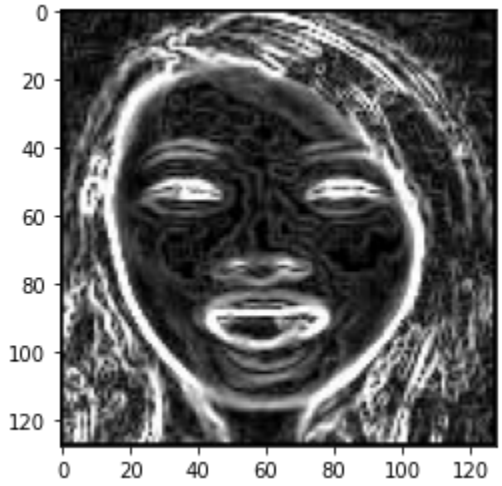

```
Actual value:  0        Predicted value:  0
```



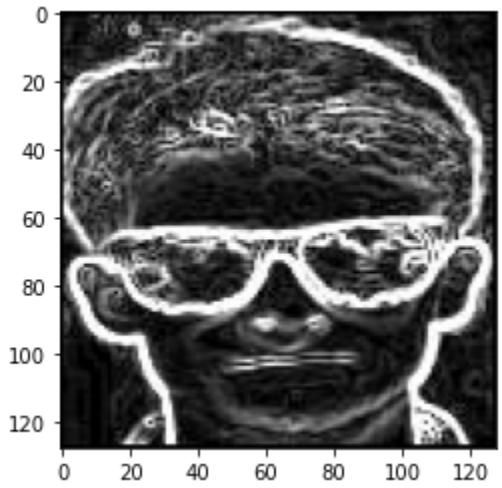

```
Actual value:  1        Predicted value:  3
```




```
Actual value:  2        Predicted value:  3
```




```
Actual value:  3        Predicted value:  3
```




```
Actual value:  4        Predicted value:  4
Total number of images for "testing":
Found 110 images belonging to 5 classes.
```

In [186]:
```
# MODEL 2
# -------
# ReLU activation function
# Adadagrad optimizer
# 4 Conv
# 1 Batch Norm
# 4 MaxPool
# 1 Dropout
# 3 Dense
# 1 Flatten
# Batchsize 32

model2 = tensorflow.keras.Sequential()
model2.add(layers.Conv2D(32, (3,3), activation='relu',
                         input_shape = (128, 128, 3),
                         kernel_initializer = 'glorot_normal',
                         bias_initializer = 'zeros'))
model2.add(layers.BatchNormalization())
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.Conv2D(64, (3,3), activation='relu'))
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.Conv2D(128, (3,3), activation='relu'))
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.Conv2D(128, (3,3), activation='relu'))
model2.add(layers.MaxPooling2D((2,2)))
model2.add(layers.Flatten())
model2.add(layers.Dropout(.5))
model2.add(layers.Dense(32, activation='relu',
                        kernel_initializer = 'glorot_normal',
                        bias_initializer = 'zeros'))
model2.add(layers.Dense(64, activation='relu'))
model2.add(layers.Dense(5, activation='softmax'))
model2.summary()
```

```
Model: "sequential_39"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_155 (Conv2D)          (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_39 (Batc (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_155 (MaxPoolin (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_156 (Conv2D)          (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_156 (MaxPoolin (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_157 (Conv2D)          (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_157 (MaxPoolin (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_158 (Conv2D)          (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_158 (MaxPoolin (None, 6, 6, 128)         0
_________________________________________________________________
flatten_39 (Flatten)         (None, 4608)              0
_________________________________________________________________
dropout_39 (Dropout)         (None, 4608)              0
```

```
_________________________________________________________________
dense_115 (Dense)            (None, 32)                147488
_________________________________________________________________
dense_116 (Dense)            (None, 64)                2112
_________________________________________________________________
dense_117 (Dense)            (None, 5)                 325
=================================================================
Total params: 390,885
Trainable params: 390,821
Non-trainable params: 64
_________________________________________________________________
```

In [187]:

```
adagrad = optimizers.Adagrad(lr=0.01)

model2.compile(loss = "categorical_crossentropy",
               optimizer=adagrad,
               metrics=["acc"])
```

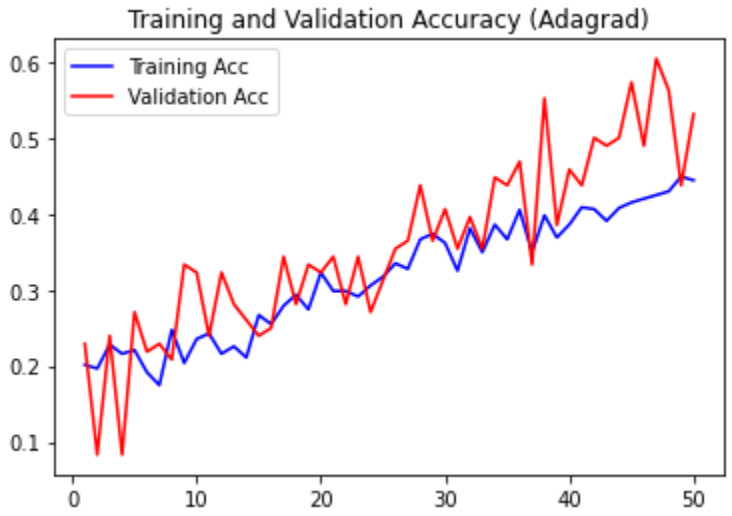
In [188]:
```
batch_size=32

# Training model with adagrad, 30 epochs and shuffling data
history_adagrad = model2.fit_generator(train_generator,
                                       steps_per_epoch=int(444/batch_size),
                                       epochs=50,
                                       validation_data=validation_generator,
                                       validation_steps=int(110/batch_size),
                                       shuffle=True)
```

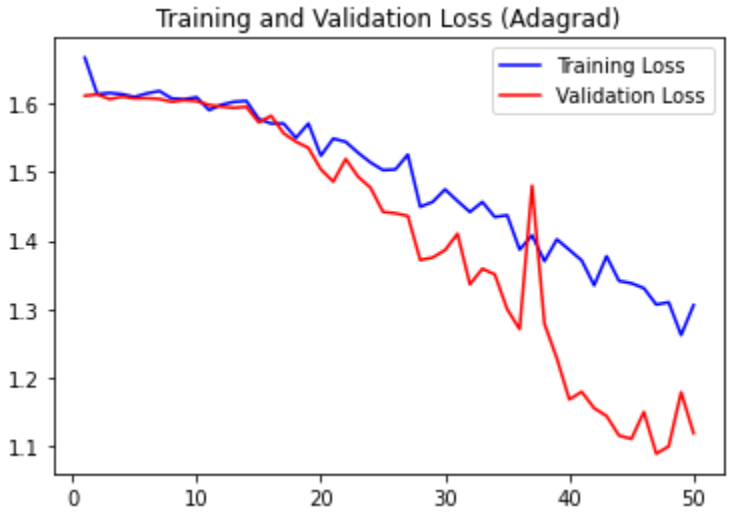
```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 13 steps, validate for 3 steps
Epoch 1/50
13/13 [==============================] - 8s 628ms/step - loss: 1.6666 - acc: 0.2015 -
val_loss: 1.6109 - val_acc: 0.2292
Epoch 2/50
13/13 [==============================] - 8s 592ms/step - loss: 1.6132 - acc: 0.1966 -
val_loss: 1.6129 - val_acc: 0.0833
Epoch 3/50
13/13 [==============================] - 8s 597ms/step - loss: 1.6147 - acc: 0.2282 -
val_loss: 1.6059 - val_acc: 0.2396
Epoch 4/50
13/13 [==============================] - 8s 613ms/step - loss: 1.6131 - acc: 0.2160 -
val_loss: 1.6097 - val_acc: 0.0833
Epoch 5/50
13/13 [==============================] - 8s 589ms/step - loss: 1.6090 - acc: 0.2209 -
val_loss: 1.6070 - val_acc: 0.2708
Epoch 6/50
13/13 [==============================] - 8s 598ms/step - loss: 1.6145 - acc: 0.1917 -
val_loss: 1.6074 - val_acc: 0.2188
Epoch 7/50
13/13 [==============================] - 8s 596ms/step - loss: 1.6180 - acc: 0.1748 -
val_loss: 1.6061 - val_acc: 0.2292
Epoch 8/50
13/13 [==============================] - 8s 604ms/step - loss: 1.6068 - acc: 0.2476 -
val_loss: 1.6021 - val_acc: 0.2083
Epoch 9/50
13/13 [==============================] - 8s 593ms/step - loss: 1.6059 - acc: 0.2039 -
val_loss: 1.6044 - val_acc: 0.3333
Epoch 10/50
13/13 [==============================] - 8s 598ms/step - loss: 1.6087 - acc: 0.2354 -
val_loss: 1.6030 - val_acc: 0.3229
Epoch 11/50
13/13 [==============================] - 8s 598ms/step - loss: 1.5897 - acc: 0.2427 -
val_loss: 1.5974 - val_acc: 0.2396
Epoch 12/50
13/13 [==============================] - 8s 590ms/step - loss: 1.5979 - acc: 0.2160 -
val_loss: 1.5950 - val_acc: 0.3229
Epoch 13/50
13/13 [==============================] - 8s 591ms/step - loss: 1.6020 - acc: 0.2257 -
val_loss: 1.5932 - val_acc: 0.2812
Epoch 14/50
```

```
13/13 [==============================] - 8s 593ms/step - loss: 1.6033 - acc: 0.2112 -
val_loss: 1.5948 - val_acc: 0.2604
Epoch 15/50
13/13 [==============================] - 8s 594ms/step - loss: 1.5769 - acc: 0.2670 -
val_loss: 1.5721 - val_acc: 0.2396
Epoch 16/50
13/13 [==============================] - 8s 596ms/step - loss: 1.5703 - acc: 0.2549 -
val_loss: 1.5817 - val_acc: 0.2500
Epoch 17/50
13/13 [==============================] - 8s 602ms/step - loss: 1.5699 - acc: 0.2791 -
val_loss: 1.5562 - val_acc: 0.3438
Epoch 18/50
13/13 [==============================] - 8s 594ms/step - loss: 1.5495 - acc: 0.2937 -
val_loss: 1.5440 - val_acc: 0.2812
Epoch 19/50
13/13 [==============================] - 8s 591ms/step - loss: 1.5701 - acc: 0.2743 -
val_loss: 1.5350 - val_acc: 0.3333
Epoch 20/50
13/13 [==============================] - 8s 597ms/step - loss: 1.5237 - acc: 0.3228 -
val_loss: 1.5037 - val_acc: 0.3229
Epoch 21/50
13/13 [==============================] - 8s 594ms/step - loss: 1.5486 - acc: 0.2985 -
val_loss: 1.4859 - val_acc: 0.3438
Epoch 22/50
13/13 [==============================] - 8s 600ms/step - loss: 1.5433 - acc: 0.2985 -
val_loss: 1.5188 - val_acc: 0.2812
Epoch 23/50
13/13 [==============================] - 8s 600ms/step - loss: 1.5277 - acc: 0.2913 -
val_loss: 1.4930 - val_acc: 0.3438
Epoch 24/50
13/13 [==============================] - 8s 594ms/step - loss: 1.5138 - acc: 0.3058 -
val_loss: 1.4770 - val_acc: 0.2708
Epoch 25/50
13/13 [==============================] - 8s 611ms/step - loss: 1.5027 - acc: 0.3173 -
val_loss: 1.4419 - val_acc: 0.3125
Epoch 26/50
13/13 [==============================] - 8s 631ms/step - loss: 1.5028 - acc: 0.3350 -
val_loss: 1.4398 - val_acc: 0.3542
Epoch 27/50
13/13 [==============================] - 8s 608ms/step - loss: 1.5241 - acc: 0.3277 -
val_loss: 1.4360 - val_acc: 0.3646
Epoch 28/50
13/13 [==============================] - 8s 590ms/step - loss: 1.4486 - acc: 0.3665 -
val_loss: 1.3718 - val_acc: 0.4375
Epoch 29/50
13/13 [==============================] - 8s 593ms/step - loss: 1.4553 - acc: 0.3738 -
val_loss: 1.3754 - val_acc: 0.3646
Epoch 30/50
13/13 [==============================] - 8s 586ms/step - loss: 1.4732 - acc: 0.3617 -
val_loss: 1.3863 - val_acc: 0.4062
Epoch 31/50
13/13 [==============================] - 8s 594ms/step - loss: 1.4567 - acc: 0.3252 -
val_loss: 1.4100 - val_acc: 0.3542
Epoch 32/50
13/13 [==============================] - 8s 589ms/step - loss: 1.4424 - acc: 0.3811 -
val_loss: 1.3363 - val_acc: 0.3958
Epoch 33/50
13/13 [==============================] - 8s 594ms/step - loss: 1.4560 - acc: 0.3495 -
val_loss: 1.3591 - val_acc: 0.3542
```

```
Epoch 34/50
13/13 [==============================] - 8s 603ms/step - loss: 1.4350 - acc: 0.3859 -
val_loss: 1.3509 - val_acc: 0.4479
Epoch 35/50
13/13 [==============================] - 8s 603ms/step - loss: 1.4361 - acc: 0.3665 -
val_loss: 1.3002 - val_acc: 0.4375
Epoch 36/50
13/13 [==============================] - 8s 598ms/step - loss: 1.3867 - acc: 0.4053 -
val_loss: 1.2712 - val_acc: 0.4688
Epoch 37/50
13/13 [==============================] - 8s 597ms/step - loss: 1.4081 - acc: 0.3495 -
val_loss: 1.4801 - val_acc: 0.3333
Epoch 38/50
13/13 [==============================] - 8s 601ms/step - loss: 1.3693 - acc: 0.3981 -
val_loss: 1.2789 - val_acc: 0.5521
Epoch 39/50
13/13 [==============================] - 8s 606ms/step - loss: 1.4014 - acc: 0.3689 -
val_loss: 1.2286 - val_acc: 0.3854
Epoch 40/50
13/13 [==============================] - 9s 685ms/step - loss: 1.3862 - acc: 0.3859 -
val_loss: 1.1689 - val_acc: 0.4583
Epoch 41/50
13/13 [==============================] - 8s 602ms/step - loss: 1.3711 - acc: 0.4087 -
val_loss: 1.1800 - val_acc: 0.4375
Epoch 42/50
13/13 [==============================] - 8s 614ms/step - loss: 1.3349 - acc: 0.4062 -
val_loss: 1.1561 - val_acc: 0.5000
Epoch 43/50
13/13 [==============================] - 10s 760ms/step - loss: 1.3782 - acc: 0.3908 -
val_loss: 1.1448 - val_acc: 0.4896
Epoch 44/50
13/13 [==============================] - 9s 690ms/step - loss: 1.3409 - acc: 0.4078 -
val_loss: 1.1163 - val_acc: 0.5000
Epoch 45/50
13/13 [==============================] - 8s 598ms/step - loss: 1.3373 - acc: 0.4150 -
val_loss: 1.1115 - val_acc: 0.5729
Epoch 46/50
13/13 [==============================] - 8s 601ms/step - loss: 1.3310 - acc: 0.4199 -
val_loss: 1.1506 - val_acc: 0.4896
Epoch 47/50
13/13 [==============================] - 8s 640ms/step - loss: 1.3069 - acc: 0.4248 -
val_loss: 1.0899 - val_acc: 0.6042
Epoch 48/50
13/13 [==============================] - 8s 601ms/step - loss: 1.3100 - acc: 0.4296 -
val_loss: 1.1005 - val_acc: 0.5625
Epoch 49/50
13/13 [==============================] - 8s 609ms/step - loss: 1.2631 - acc: 0.4490 -
val_loss: 1.1794 - val_acc: 0.4375
Epoch 50/50
13/13 [==============================] - 8s 647ms/step - loss: 1.3043 - acc: 0.4442 -
val_loss: 1.1197 - val_acc: 0.5312
```

In [189]:
```
# ---------------------------------------------
# Visualizing Train/Validation Loss & Accuracy |
# ---------------------------------------------

acc_adagrad = history_adagrad.history['acc']
val_acc_adagrad = history_adagrad.history['val_acc']
loss_adagrad = history_adagrad.history['loss']
val_loss_adagrad = history_adagrad.history['val_loss']

epochs_adagrad = range(1,len(acc_adagrad) +1)

# Plot of accuracy
plt.plot(epochs_adagrad, acc_adagrad, color='blue', label='Training Acc')
plt.plot(epochs_adagrad, val_acc_adagrad, color='red', label='Validation Acc')
plt.title('Training and Validation Accuracy (Adagrad)')
plt.legend()

plt.figure()

# Plot of loss
plt.plot(epochs_adagrad, loss_adagrad, color='blue', label='Training Loss')
plt.plot(epochs_adagrad, val_loss_adagrad, color='red', label='Validation Loss')
plt.title('Training and Validation Loss (Adagrad)')
plt.legend()
plt.figure()

plt.show()
```



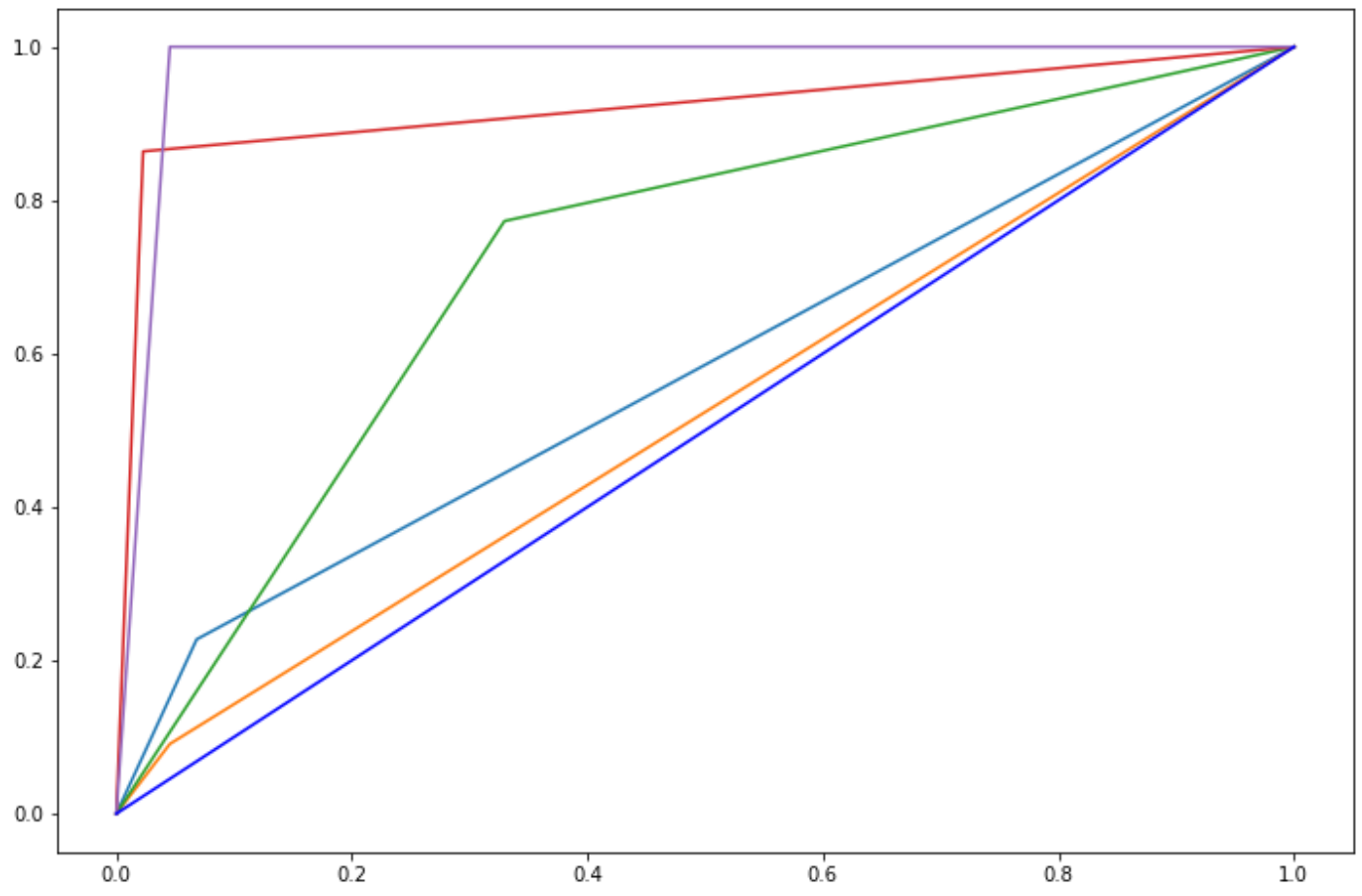


<Figure size 432x288 with 0 Axes>

In [190]:
```
# ---------------
# ROC/AUC Score |
# --------------
from sklearn.preprocessing import LabelBinarizer

# set plot figure size
fig, c_ax = plt.subplots(1,1, figsize = (12, 8))

def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)

    for (idx, c_label) in enumerate(target_names): # target_names: no of the labels
        fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), y_pred[:,idx])
        c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
    c_ax.plot(fpr, fpr, 'b-', label = 'Random Guessing')
    return roc_auc_score(y_test, y_pred, average=average)

validation_generator.reset() # resetting generator
y_pred = model2.predict_generator(validation_generator, verbose = True)
y_pred = np.argmax(y_pred, axis=1)
multiclass_roc_auc_score(validation_generator.classes, y_pred)
```

```
4/4 [==============================] - 0s 98ms/step
```

Out[190]:
```
0.7443181818181819
```



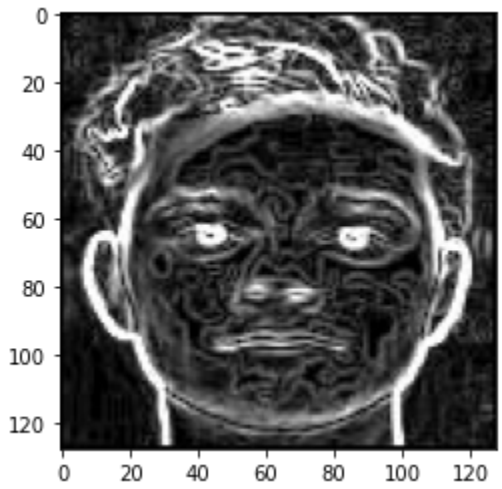
In [191]:
```
# ----------------------
# Classification Report |
# ----------------------

print('Classification Report for Model with Adagrad')
target_names = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']
print(classification_report(validation_generator.classes, y_pred, target_names=target_names
```

```
Classification Report for Model with Adagrad
              precision    recall  f1-score   support

   1_Neutral       0.45      0.23      0.30        22
   2_Smiling       0.33      0.09      0.14        22
    3_Sleepy       0.37      0.77      0.50        22
  4_Surprise       0.90      0.86      0.88        22
5_Sunglasses       0.85      1.00      0.92        22

    accuracy                           0.59       110
   macro avg       0.58      0.59      0.55       110
weighted avg       0.58      0.59      0.55       110
```

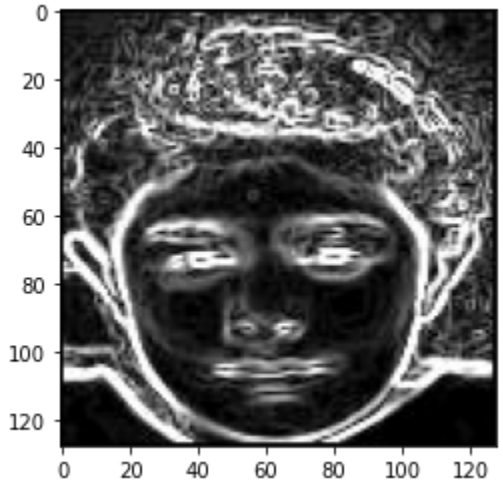
In [192]:
```
# ------------------
# Confusion Matrix |
# ------------------

num_of_train_samples = 444
#num_of_test_samples = 416 # steps per epoch
batch_size=32
steps_per_epoch=num_of_train_samples // batch_size
#validation_generator.reset()

Y_pred = model2.predict_generator(validation_generator, steps_per_epoch)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix fo Model with Adagrad')
print(confusion_matrix(validation_generator.classes, y_pred))
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 13 batches). You may need to use the repeat() function when building your dataset.
Confusion Matrix fo Model with Adagrad
[[ 5  2 14  1  0]
 [ 3  2 14  1  2]
 [ 2  1 17  0  2]
 [ 1  1  1 19  0]
 [ 0  0  0  0 22]]
```

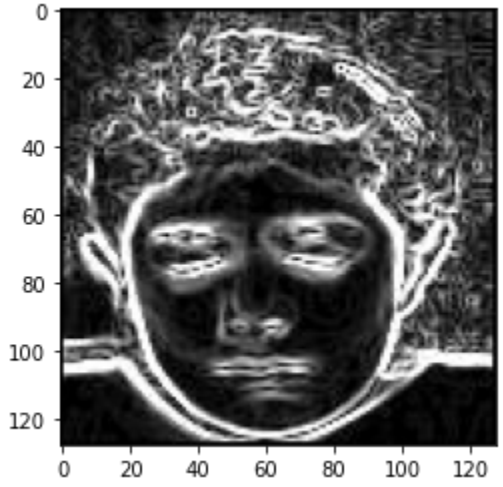
In [193]:
```
# ----------------------------------------
# Displaying 12 Images with the Prediction |
# ----------------------------------------

# 1 image from each class

labels = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']

i=0
for names in labels:
    pathToFolder = test_dir +'/'+names+'/'
    fnames = [os.path.join(pathToFolder, fname) for
          fname in os.listdir(pathToFolder)]

    # generate random number btwn (0,30)
    randNum = random.randint(0, 20)
    img_path = fnames[randNum]
    tmp_img = image.load_img(img_path, target_size = (128, 128))
    tmp_img = image.img_to_array(tmp_img)
    tmp_img = np.expand_dims(tmp_img, axis = 0)

    tmp_img /=255.
    plt.imshow(tmp_img[0])
    plt.show()
    # predict
    result = model2.predict(tmp_img)
    train_generator.class_indices
    print('Actual value: ',i,'\tPredicted value: ', np.argmax(result))
    i+=1

print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size = (128, 128),
                                                  batch_size = 32,
                                                  class_mode = "categorical",
                                                  shuffle=False)
```



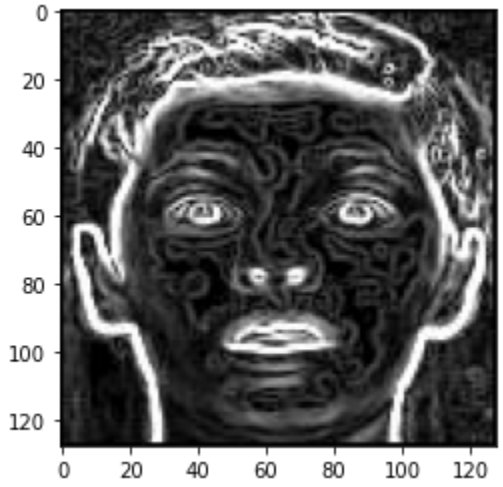
Actual value: 0   Predicted value: 2



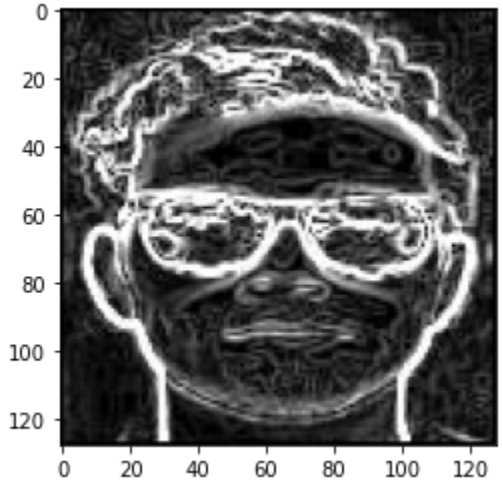
Actual value: 1   Predicted value: 0



Actual value: 2   Predicted value: 0




```
Actual value:  3        Predicted value:  2
```




```
Actual value:  4        Predicted value:  4
Total number of images for "testing":
Found 110 images belonging to 5 classes.
```

In [133]:
```
# MODEL 3
# -------
# ReLU activation function
# Rmsprop optimizer
# 4 Conv
# 1 Batch Norm
# 4 MaxPool
# 1 Dropout
# 3 Dense
# 1 Flatten
# Batchsize 32

model3 = tensorflow.keras.Sequential()
model3.add(layers.Conv2D(32, (3,3), activation='relu',
                         input_shape = (128, 128, 3),
                         kernel_initializer = 'glorot_normal',
                         bias_initializer = 'zeros'))
model3.add(layers.BatchNormalization())
model3.add(layers.MaxPooling2D((2,2)))
model3.add(layers.Conv2D(64, (3,3), activation='relu'))
model3.add(layers.MaxPooling2D((2,2)))
model3.add(layers.Conv2D(128, (3,3), activation='relu'))
model3.add(layers.MaxPooling2D((2,2)))
model3.add(layers.Conv2D(128, (3,3), activation='relu'))
model3.add(layers.MaxPooling2D((2,2)))
model3.add(layers.Flatten())
model3.add(layers.Dropout(.5))
model3.add(layers.Dense(64, activation='relu',
                        kernel_initializer = 'glorot_normal',
                        bias_initializer = 'zeros'))
model3.add(layers.Dense(128, activation='relu'))
model3.add(layers.Dense(5, activation='softmax'))
model3.summary()
```

```
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_91 (Conv2D)           (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_23 (Batc (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_91 (MaxPooling (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_92 (Conv2D)           (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_92 (MaxPooling (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_93 (Conv2D)           (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_93 (MaxPooling (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_94 (Conv2D)           (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_94 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________________
flatten_23 (Flatten)         (None, 4608)              0
_________________________________________________________________
dropout_23 (Dropout)         (None, 4608)              0
```

```
_________________________________________________________________
dense_67 (Dense)             (None, 64)                294976
_________________________________________________________________
dense_68 (Dense)             (None, 128)               8320
_________________________________________________________________
dense_69 (Dense)             (None, 5)                 645
=================================================================
Total params: 544,901
Trainable params: 544,837
Non-trainable params: 64
_________________________________________________________________
```

In [134]:

```
rmsprop = optimizers.RMSprop(lr=0.001, rho=0.9)

model3.compile(loss = "categorical_crossentropy",
               optimizer=rmsprop,
               metrics=["acc"])
```

In [135]:
```
# Training model with adagrad, 30 epochs and shuffling data
batch_size=32

history_rsm = model3.fit_generator(train_generator,
                                   steps_per_epoch=int(444/batch_size),
                                   epochs=50,
                                   validation_data=validation_generator,
                                   validation_steps=int(110/batch_size),
                                   shuffle=True)
```

```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 13 steps, validate for 3 steps
Epoch 1/50
13/13 [==============================] - 9s 694ms/step - loss: 1.6852 - acc: 0.2039 -
val_loss: 1.6122 - val_acc: 0.0833
Epoch 2/50
13/13 [==============================] - 8s 634ms/step - loss: 1.6243 - acc: 0.1845 -
val_loss: 1.6109 - val_acc: 0.0833
Epoch 3/50
13/13 [==============================] - 9s 690ms/step - loss: 1.6147 - acc: 0.2257 -
val_loss: 1.6082 - val_acc: 0.2292
Epoch 4/50
13/13 [==============================] - 8s 594ms/step - loss: 1.6061 - acc: 0.2500 -
val_loss: 1.6060 - val_acc: 0.2708
Epoch 5/50
13/13 [==============================] - 8s 596ms/step - loss: 1.5989 - acc: 0.2743 -
val_loss: 1.6158 - val_acc: 0.1042
Epoch 6/50
13/13 [==============================] - 8s 606ms/step - loss: 1.5667 - acc: 0.2500 -
val_loss: 1.5626 - val_acc: 0.3021
Epoch 7/50
13/13 [==============================] - 8s 599ms/step - loss: 1.5565 - acc: 0.2937 -
val_loss: 1.5526 - val_acc: 0.3125
Epoch 8/50
13/13 [==============================] - 8s 599ms/step - loss: 1.5201 - acc: 0.3053 -
val_loss: 1.6186 - val_acc: 0.2083
Epoch 9/50
13/13 [==============================] - 8s 596ms/step - loss: 1.5536 - acc: 0.3197 -
val_loss: 1.5098 - val_acc: 0.3646
Epoch 10/50
13/13 [==============================] - 8s 607ms/step - loss: 1.4868 - acc: 0.3398 -
val_loss: 1.4654 - val_acc: 0.4062
Epoch 11/50
13/13 [==============================] - 8s 593ms/step - loss: 1.4285 - acc: 0.3641 -
val_loss: 1.7033 - val_acc: 0.1458
Epoch 12/50
13/13 [==============================] - 8s 602ms/step - loss: 1.5643 - acc: 0.3155 -
val_loss: 1.4140 - val_acc: 0.4167
Epoch 13/50
13/13 [==============================] - 8s 591ms/step - loss: 1.4411 - acc: 0.3654 -
val_loss: 1.3967 - val_acc: 0.5104
Epoch 14/50
```

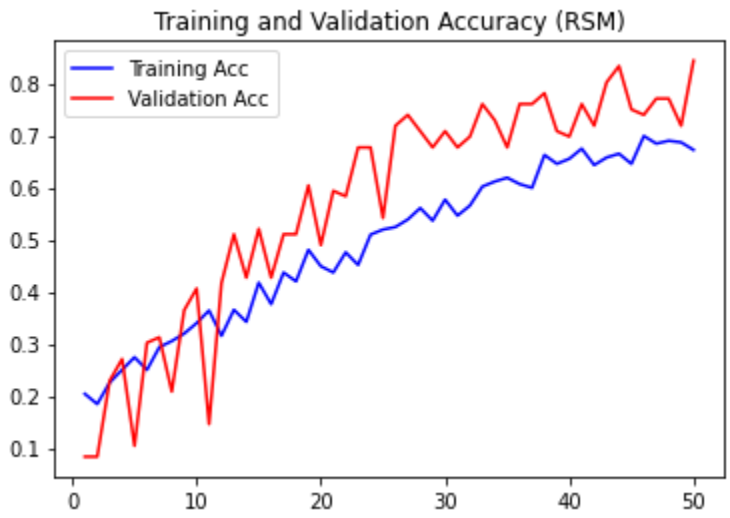
```
13/13 [==============================] - 8s 590ms/step - loss: 1.4523 - acc: 0.3422 -
val_loss: 1.3572 - val_acc: 0.4271
Epoch 15/50
13/13 [==============================] - 8s 609ms/step - loss: 1.4170 - acc: 0.4175 -
val_loss: 1.2582 - val_acc: 0.5208
Epoch 16/50
13/13 [==============================] - 8s 598ms/step - loss: 1.3619 - acc: 0.3762 -
val_loss: 1.2577 - val_acc: 0.4271
Epoch 17/50
13/13 [==============================] - 8s 591ms/step - loss: 1.3416 - acc: 0.4369 -
val_loss: 1.1675 - val_acc: 0.5104
Epoch 18/50
13/13 [==============================] - 8s 591ms/step - loss: 1.3363 - acc: 0.4199 -
val_loss: 1.2245 - val_acc: 0.5104
Epoch 19/50
13/13 [==============================] - 8s 600ms/step - loss: 1.2582 - acc: 0.4806 -
val_loss: 1.1004 - val_acc: 0.6042
Epoch 20/50
13/13 [==============================] - 8s 596ms/step - loss: 1.2771 - acc: 0.4490 -
val_loss: 1.1394 - val_acc: 0.4896
Epoch 21/50
13/13 [==============================] - 8s 598ms/step - loss: 1.2949 - acc: 0.4369 -
val_loss: 1.0682 - val_acc: 0.5938
Epoch 22/50
13/13 [==============================] - 8s 596ms/step - loss: 1.2143 - acc: 0.4757 -
val_loss: 1.0315 - val_acc: 0.5833
Epoch 23/50
13/13 [==============================] - 8s 611ms/step - loss: 1.2302 - acc: 0.4515 -
val_loss: 0.9393 - val_acc: 0.6771
Epoch 24/50
13/13 [==============================] - 8s 594ms/step - loss: 1.1804 - acc: 0.5097 -
val_loss: 0.9813 - val_acc: 0.6771
Epoch 25/50
13/13 [==============================] - 8s 596ms/step - loss: 1.0919 - acc: 0.5194 -
val_loss: 0.9657 - val_acc: 0.5417
Epoch 26/50
13/13 [==============================] - 8s 600ms/step - loss: 1.1032 - acc: 0.5243 -
val_loss: 0.8919 - val_acc: 0.7188
Epoch 27/50
13/13 [==============================] - 8s 620ms/step - loss: 1.0707 - acc: 0.5388 -
val_loss: 0.8532 - val_acc: 0.7396
Epoch 28/50
13/13 [==============================] - 8s 597ms/step - loss: 1.1345 - acc: 0.5607 -
val_loss: 0.8259 - val_acc: 0.7083
Epoch 29/50
13/13 [==============================] - 8s 600ms/step - loss: 1.0928 - acc: 0.5364 -
val_loss: 0.8089 - val_acc: 0.6771
Epoch 30/50
13/13 [==============================] - 8s 601ms/step - loss: 1.0828 - acc: 0.5769 -
val_loss: 0.7362 - val_acc: 0.7083
Epoch 31/50
13/13 [==============================] - 8s 599ms/step - loss: 1.0360 - acc: 0.5461 -
val_loss: 0.7606 - val_acc: 0.6771
Epoch 32/50
13/13 [==============================] - 8s 609ms/step - loss: 0.9954 - acc: 0.5655 -
val_loss: 0.6879 - val_acc: 0.6979
Epoch 33/50
13/13 [==============================] - 8s 592ms/step - loss: 0.9260 - acc: 0.6019 -
val_loss: 0.6352 - val_acc: 0.7604
```

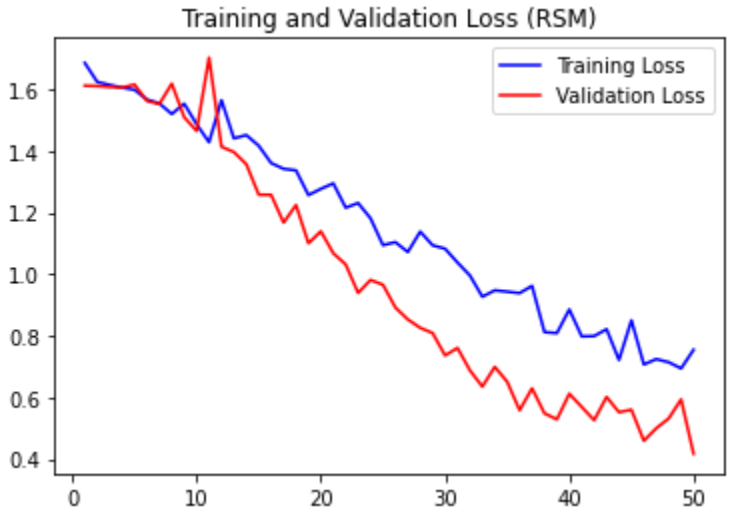
```
Epoch 34/50
13/13 [==============================] - 8s 596ms/step - loss: 0.9456 - acc: 0.6117 -
val_loss: 0.6991 - val_acc: 0.7292
Epoch 35/50
13/13 [==============================] - 8s 598ms/step - loss: 0.9411 - acc: 0.6189 -
val_loss: 0.6508 - val_acc: 0.6771
Epoch 36/50
13/13 [==============================] - 8s 612ms/step - loss: 0.9424 - acc: 0.6068 -
val_loss: 0.5583 - val_acc: 0.7604
Epoch 37/50
13/13 [==============================] - 8s 639ms/step - loss: 0.9586 - acc: 0.5995 -
val_loss: 0.6296 - val_acc: 0.7604
Epoch 38/50
13/13 [==============================] - 8s 634ms/step - loss: 0.8114 - acc: 0.6626 -
val_loss: 0.5484 - val_acc: 0.7812
Epoch 39/50
13/13 [==============================] - 8s 599ms/step - loss: 0.8081 - acc: 0.6456 -
val_loss: 0.5287 - val_acc: 0.7083
Epoch 40/50
13/13 [==============================] - 8s 610ms/step - loss: 0.8846 - acc: 0.6553 -
val_loss: 0.6120 - val_acc: 0.6979
Epoch 41/50
13/13 [==============================] - 8s 592ms/step - loss: 0.7966 - acc: 0.6748 -
val_loss: 0.5689 - val_acc: 0.7604
Epoch 42/50
13/13 [==============================] - 8s 592ms/step - loss: 0.7980 - acc: 0.6432 -
val_loss: 0.5261 - val_acc: 0.7188
Epoch 43/50
13/13 [==============================] - 8s 595ms/step - loss: 0.8226 - acc: 0.6578 -
val_loss: 0.6015 - val_acc: 0.8021
Epoch 44/50
13/13 [==============================] - 8s 596ms/step - loss: 0.7187 - acc: 0.6650 -
val_loss: 0.5520 - val_acc: 0.8333
Epoch 45/50
13/13 [==============================] - 8s 598ms/step - loss: 0.8481 - acc: 0.6456 -
val_loss: 0.5600 - val_acc: 0.7500
Epoch 46/50
13/13 [==============================] - 8s 596ms/step - loss: 0.7062 - acc: 0.6990 -
val_loss: 0.4593 - val_acc: 0.7396
Epoch 47/50
13/13 [==============================] - 8s 592ms/step - loss: 0.7224 - acc: 0.6845 -
val_loss: 0.4998 - val_acc: 0.7708
Epoch 48/50
13/13 [==============================] - 8s 601ms/step - loss: 0.7143 - acc: 0.6899 -
val_loss: 0.5317 - val_acc: 0.7708
Epoch 49/50
13/13 [==============================] - 8s 600ms/step - loss: 0.6922 - acc: 0.6869 -
val_loss: 0.5935 - val_acc: 0.7188
Epoch 50/50
13/13 [==============================] - 8s 590ms/step - loss: 0.7539 - acc: 0.6723 -
val_loss: 0.4173 - val_acc: 0.8438
```

In [40]:

```
# ---------------
# SAVE THE MODEL |
# ---------------
model3.save('DIP_Proj_modelRSM.h5')
```

In [136]:
```
# ---------------------------------------------
# Visualizing Train/Validation Loss & Accuracy |
# ---------------------------------------------

acc_rsm = history_rsm.history['acc']
val_acc_rsm = history_rsm.history['val_acc']
loss_rsm = history_rsm.history['loss']
val_loss_rsm = history_rsm.history['val_loss']

epochs_rsm = range(1,len(acc_rsm) +1)

# Plot of accuracy
plt.plot(epochs_rsm, acc_rsm, color='blue', label='Training Acc')
plt.plot(epochs_rsm, val_acc_rsm, color='red', label='Validation Acc')
plt.title('Training and Validation Accuracy (RSM)')
plt.legend()

plt.figure()

# Plot of loss
plt.plot(epochs_rsm, loss_rsm, color='blue', label='Training Loss')
plt.plot(epochs_rsm, val_loss_rsm, color='red', label='Validation Loss')
plt.title('Training and Validation Loss (RSM)')
plt.legend()
plt.figure()

plt.show()
```



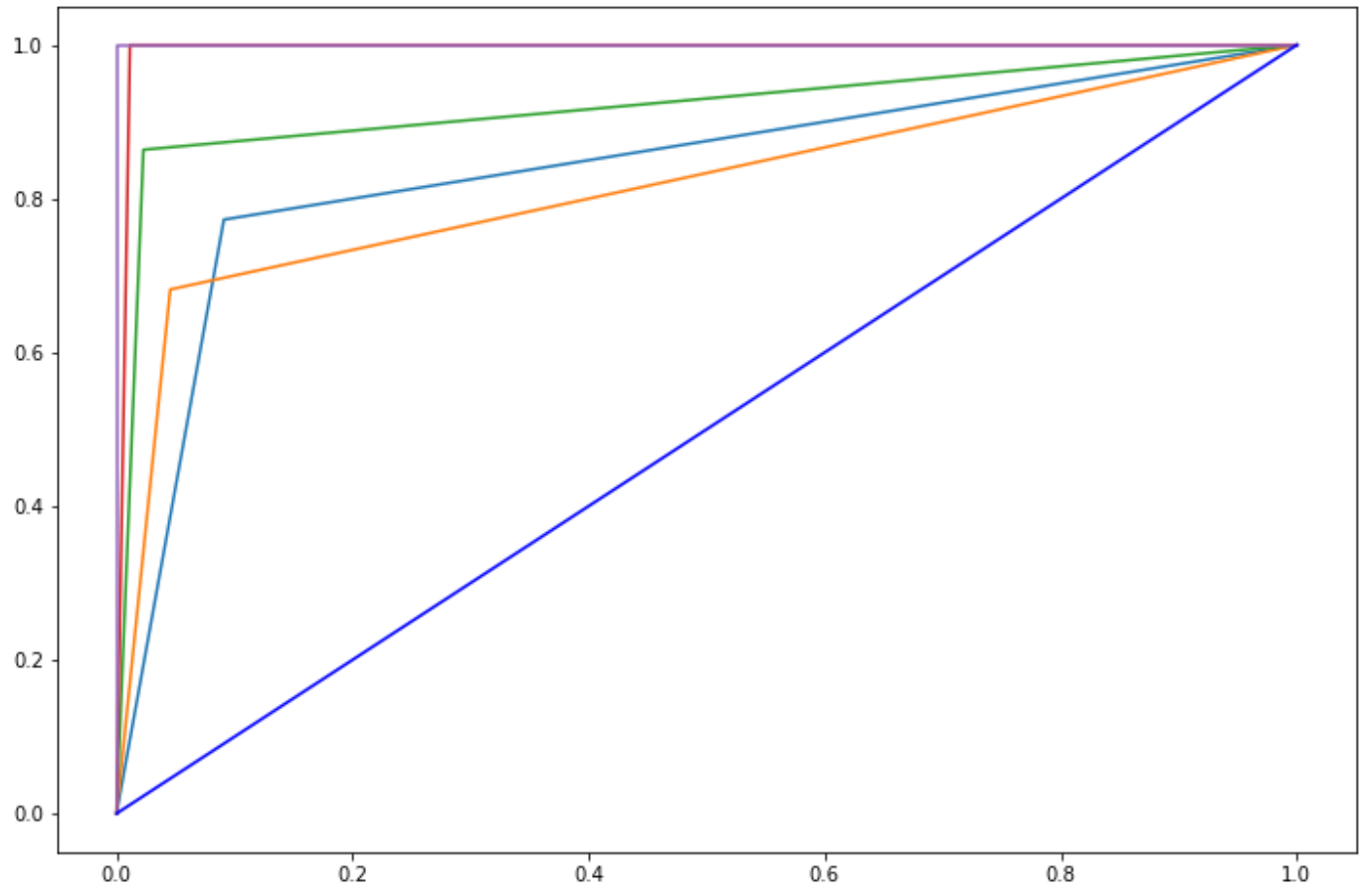


<Figure size 432x288 with 0 Axes>

In [137]:
```
# --------------
# ROC/AUC Score |
# --------------

from sklearn.preprocessing import LabelBinarizer

# set plot figure size
fig, c_ax = plt.subplots(1,1, figsize = (12, 8))

def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)

    for (idx, c_label) in enumerate(target_names): # target_names: no of the labels
        fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), y_pred[:,idx])
        c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
    c_ax.plot(fpr, fpr, 'b-', label = 'Random Guessing')
    return roc_auc_score(y_test, y_pred, average=average)

validation_generator.reset() # resetting generator
y_pred = model3.predict_generator(validation_generator, verbose = True)
y_pred = np.argmax(y_pred, axis=1)
multiclass_roc_auc_score(validation_generator.classes, y_pred)
```

```
4/4 [==============================] - 0s 100ms/step
```

Out[137]:
```
0.9147727272727273
```



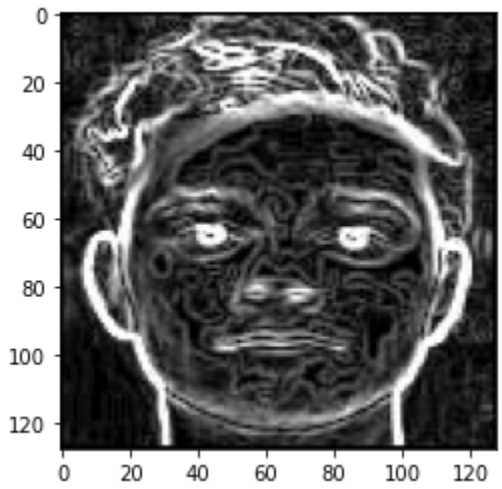
In [138]:
```
# ----------------------
# Classification Report |
# ----------------------

print('Classification Report for Model with RSM')
target_names = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']
print(classification_report(validation_generator.classes, y_pred, target_names=target_names
```

```
Classification Report for Model with RSM
              precision    recall  f1-score   support

   1_Neutral       0.68      0.77      0.72        22
   2_Smiling       0.79      0.68      0.73        22
    3_Sleepy       0.90      0.86      0.88        22
  4_Surprise       0.96      1.00      0.98        22
5_Sunglasses       1.00      1.00      1.00        22

    accuracy                           0.86       110
   macro avg       0.87      0.86      0.86       110
weighted avg       0.87      0.86      0.86       110
```

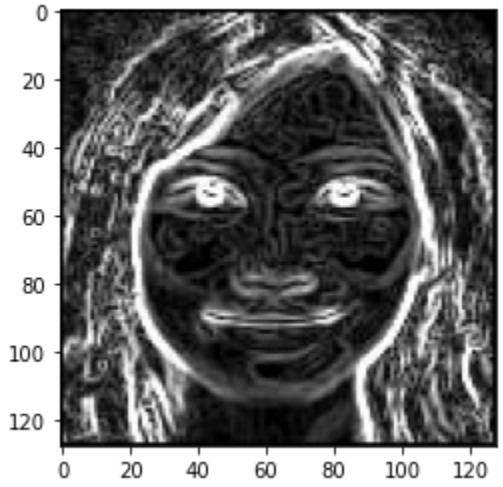
In [139]:
```
# ------------------
# Confusion Matrix |
# ------------------

num_of_train_samples = 444
#num_of_test_samples = 416 # steps per epoch
batch_size=32
steps_per_epoch=num_of_train_samples // batch_size
#validation_generator.reset()

Y_pred = model3.predict_generator(validation_generator, steps_per_epoch)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix for Model with RSM')
print(confusion_matrix(validation_generator.classes, y_pred))
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 13 batches). You may need to use the repeat() function when building your dataset.
Confusion Matrix for Model with RSM
[[17  4  1  0  0]
 [ 5 15  1  1  0]
 [ 3  0 19  0  0]
 [ 0  0  0 22  0]
 [ 0  0  0  0 22]]
```

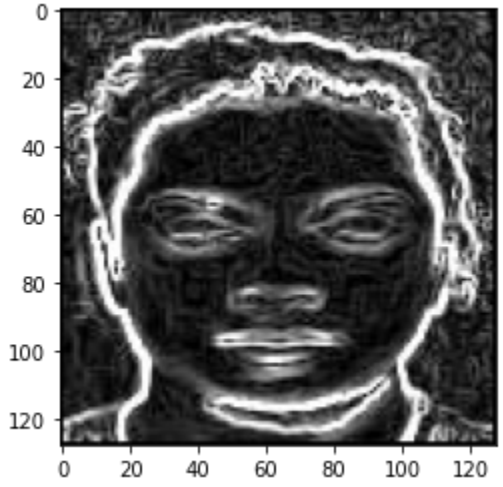
In [140]:
```
# ----------------------------------------
# Displaying 12 Images with the Prediction |
# ----------------------------------------

# 1 image from each class

labels = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']

i=0
for names in labels:
    pathToFolder = test_dir +'/'+names+'/'
    fnames = [os.path.join(pathToFolder, fname) for
          fname in os.listdir(pathToFolder)]

    # generate random number btwn (0,30)
    randNum = random.randint(0, 20)
    img_path = fnames[randNum]
    tmp_img = image.load_img(img_path, target_size = (128, 128))
    tmp_img = image.img_to_array(tmp_img)
    tmp_img = np.expand_dims(tmp_img, axis = 0)

    tmp_img /=255.
    plt.imshow(tmp_img[0])
    plt.show()
    # predict
    result = model3.predict(tmp_img)
    train_generator.class_indices
    print('Actual value: ',i,'\tPredicted value: ', np.argmax(result))
    i+=1

print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size = (128, 128),
                                                  batch_size = 32,
                                                  class_mode = "categorical",
                                                  shuffle=False)
```



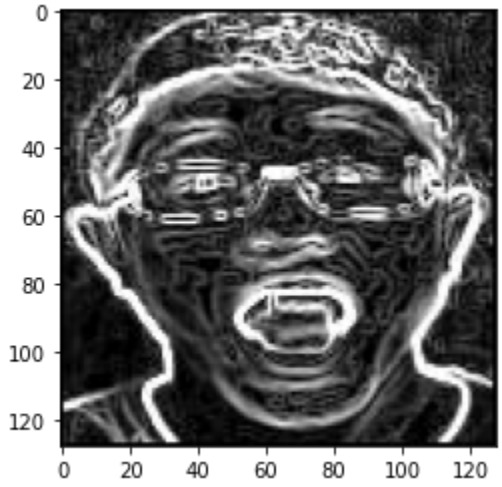
Actual value: 0   Predicted value: 0



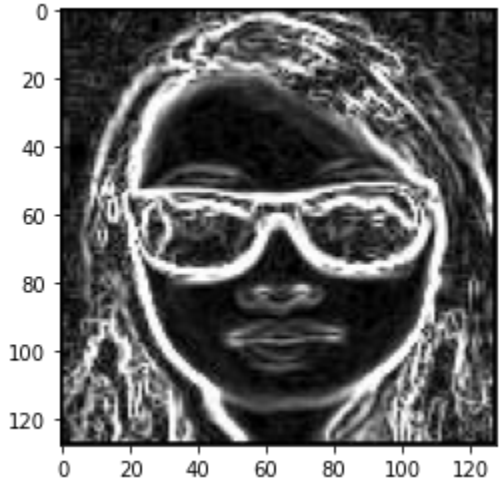
Actual value: 1   Predicted value: 0



Actual value: 2   Predicted value: 2




```
Actual value:  3        Predicted value:  3
```




```
Actual value:  4        Predicted value:  4
Total number of images for "testing":
Found 110 images belonging to 5 classes.
```

In [165]:
```
# MODEL 4
# -------
# ReLU activation function
# Rmsprop optimizer
# 4 Conv
# 1 Batch Norm
# 4 MaxPool
# 1 Dropout
# 3 Dense
# 1 Flatten
# Batchsize 32

model4 = tensorflow.keras.Sequential()
model4.add(layers.Conv2D(32, (3,3), activation='relu',
                         input_shape = (128, 128, 3),
                         kernel_initializer = 'glorot_normal',
                         bias_initializer = 'zeros'))
model4.add(layers.BatchNormalization())
model4.add(layers.MaxPooling2D((2,2)))
model4.add(layers.Conv2D(64, (3,3), activation='relu'))
model4.add(layers.MaxPooling2D((2,2)))
model4.add(layers.Conv2D(128, (3,3), activation='relu'))
model4.add(layers.MaxPooling2D((2,2)))
model4.add(layers.Conv2D(128, (3,3), activation='relu'))
model4.add(layers.MaxPooling2D((2,2)))
model4.add(layers.Flatten())
model4.add(layers.Dropout(.5))
model4.add(layers.Dense(128, activation='relu',
                        kernel_initializer = 'glorot_normal',
                        bias_initializer = 'zeros'))
model4.add(layers.Dense(512, activation='relu'))
model4.add(layers.Dense(5, activation='softmax'))
model4.summary()
```

```
Model: "sequential_32"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_127 (Conv2D)          (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_32 (Batc (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_127 (MaxPoolin (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_128 (Conv2D)          (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_128 (MaxPoolin (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_129 (Conv2D)          (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_129 (MaxPoolin (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_130 (Conv2D)          (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_130 (MaxPoolin (None, 6, 6, 128)         0
_________________________________________________________________
flatten_32 (Flatten)         (None, 4608)              0
_________________________________________________________________
dropout_32 (Dropout)         (None, 4608)              0
```

```
_________________________________________________________________
dense_94 (Dense)             (None, 128)               589952
_________________________________________________________________
dense_95 (Dense)             (None, 512)               66048
_________________________________________________________________
dense_96 (Dense)             (None, 5)                 2565
=================================================================
Total params: 899,525
Trainable params: 899,461
Non-trainable params: 64
_________________________________________________________________
```

In [166]:

```
# SGD optimizer with Nesterov momentum
sgd = optimizers.SGD(lr=0.001, decay=1e-2, momentum=0.9, nesterov=True)

model4.compile(loss = "categorical_crossentropy",
               optimizer=sgd,
               metrics=["acc"])
```

In [167]:
```
# Training model with adagrad, 50 epochs and shuffling data
batch_size=32

history_sgd = model4.fit_generator(train_generator,
                                   steps_per_epoch=int(444/batch_size),
                                   epochs=30,
                                   validation_data=validation_generator,
                                   validation_steps=int(110/batch_size),
                                   shuffle=True)
```

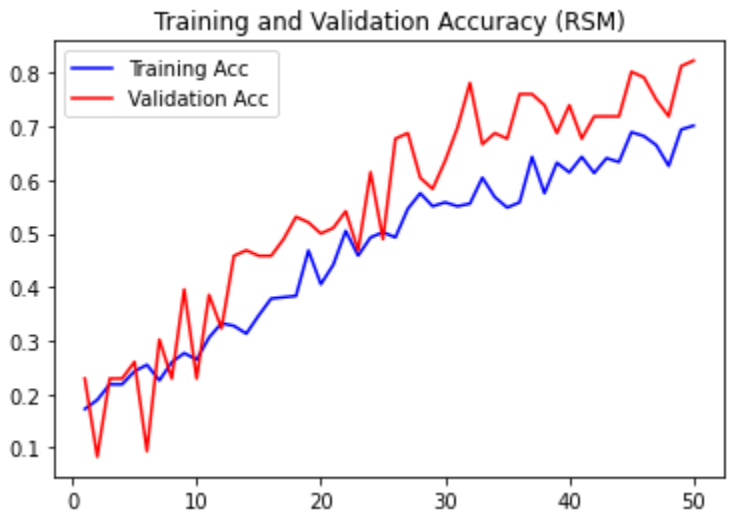
```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 13 steps, validate for 3 steps
Epoch 1/30
13/13 [==============================] - 8s 639ms/step - loss: 1.7144 - acc: 0.1432 -
val_loss: 1.6109 - val_acc: 0.2396
Epoch 2/30
13/13 [==============================] - 8s 596ms/step - loss: 1.6335 - acc: 0.2039 -
val_loss: 1.6086 - val_acc: 0.2500
Epoch 3/30
13/13 [==============================] - 8s 595ms/step - loss: 1.6168 - acc: 0.1893 -
val_loss: 1.6049 - val_acc: 0.2292
Epoch 4/30
13/13 [==============================] - 8s 612ms/step - loss: 1.6093 - acc: 0.2160 -
val_loss: 1.6054 - val_acc: 0.3125
Epoch 5/30
13/13 [==============================] - 8s 594ms/step - loss: 1.6158 - acc: 0.2184 -
val_loss: 1.6046 - val_acc: 0.3021
Epoch 6/30
13/13 [==============================] - 8s 596ms/step - loss: 1.6140 - acc: 0.2184 -
val_loss: 1.6026 - val_acc: 0.2917
Epoch 7/30
13/13 [==============================] - 8s 598ms/step - loss: 1.6045 - acc: 0.2112 -
val_loss: 1.6031 - val_acc: 0.3021
Epoch 8/30
13/13 [==============================] - 8s 599ms/step - loss: 1.6070 - acc: 0.2330 -
val_loss: 1.6041 - val_acc: 0.2396
Epoch 9/30
13/13 [==============================] - 8s 605ms/step - loss: 1.6045 - acc: 0.2428 -
val_loss: 1.6018 - val_acc: 0.2812
Epoch 10/30
13/13 [==============================] - 8s 595ms/step - loss: 1.6037 - acc: 0.2330 -
val_loss: 1.6007 - val_acc: 0.3021
Epoch 11/30
13/13 [==============================] - 8s 593ms/step - loss: 1.6040 - acc: 0.2306 -
val_loss: 1.6016 - val_acc: 0.2604
Epoch 12/30
13/13 [==============================] - 8s 627ms/step - loss: 1.6137 - acc: 0.2160 -
val_loss: 1.6004 - val_acc: 0.3021
Epoch 13/30
13/13 [==============================] - 8s 623ms/step - loss: 1.5989 - acc: 0.2257 -
val_loss: 1.5974 - val_acc: 0.3333
Epoch 14/30
```

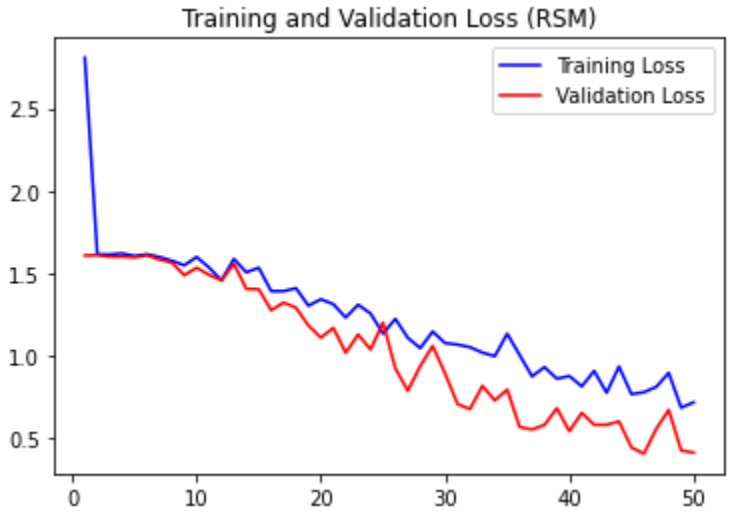
```
13/13 [==============================] - 8s 602ms/step - loss: 1.5897 - acc: 0.2621 -
val_loss: 1.5950 - val_acc: 0.3021
Epoch 15/30
13/13 [==============================] - 8s 595ms/step - loss: 1.5809 - acc: 0.2961 -
val_loss: 1.5959 - val_acc: 0.2604
Epoch 16/30
13/13 [==============================] - 8s 609ms/step - loss: 1.5949 - acc: 0.2330 -
val_loss: 1.5924 - val_acc: 0.3333
Epoch 17/30
13/13 [==============================] - 8s 596ms/step - loss: 1.5795 - acc: 0.2816 -
val_loss: 1.5909 - val_acc: 0.3333
Epoch 18/30
13/13 [==============================] - 8s 600ms/step - loss: 1.5852 - acc: 0.2718 -
val_loss: 1.5863 - val_acc: 0.3750
Epoch 19/30
13/13 [==============================] - 8s 595ms/step - loss: 1.5711 - acc: 0.2694 -
val_loss: 1.5847 - val_acc: 0.3750
Epoch 20/30
13/13 [==============================] - 8s 627ms/step - loss: 1.5911 - acc: 0.2379 -
val_loss: 1.5827 - val_acc: 0.3333
Epoch 21/30
13/13 [==============================] - 8s 606ms/step - loss: 1.5737 - acc: 0.2694 -
val_loss: 1.5809 - val_acc: 0.3438
Epoch 22/30
13/13 [==============================] - 8s 596ms/step - loss: 1.5683 - acc: 0.2816 -
val_loss: 1.5740 - val_acc: 0.3646
Epoch 23/30
13/13 [==============================] - 8s 594ms/step - loss: 1.5559 - acc: 0.3058 -
val_loss: 1.5666 - val_acc: 0.3958
Epoch 24/30
13/13 [==============================] - 8s 596ms/step - loss: 1.6032 - acc: 0.2282 -
val_loss: 1.5697 - val_acc: 0.3438
Epoch 25/30
13/13 [==============================] - 8s 602ms/step - loss: 1.5769 - acc: 0.2816 -
val_loss: 1.5649 - val_acc: 0.3854
Epoch 26/30
13/13 [==============================] - 8s 599ms/step - loss: 1.5547 - acc: 0.3083 -
val_loss: 1.5579 - val_acc: 0.4062
Epoch 27/30
13/13 [==============================] - 8s 605ms/step - loss: 1.5698 - acc: 0.2646 -
val_loss: 1.5505 - val_acc: 0.4062
Epoch 28/30
13/13 [==============================] - 8s 620ms/step - loss: 1.5599 - acc: 0.3058 -
val_loss: 1.5503 - val_acc: 0.3750
Epoch 29/30
13/13 [==============================] - 9s 681ms/step - loss: 1.5570 - acc: 0.2743 -
val_loss: 1.5543 - val_acc: 0.3229
Epoch 30/30
13/13 [==============================] - 9s 671ms/step - loss: 1.5566 - acc: 0.2694 -
val_loss: 1.5486 - val_acc: 0.3542
```

In [51]:

```
# ---------------
# SAVE THE MODEL |
# ---------------
model4.save('DIP_Proj_modelRSM2.h5')
```

In [52]:
```
# ---------------------------------------------
# Visualizing Train/Validation Loss & Accuracy |
# ---------------------------------------------

acc_sgd = history_sgd.history['acc']
val_acc_sgd = history_sgd.history['val_acc']
loss_sgd = history_sgd.history['loss']
val_loss_sgd = history_sgd.history['val_loss']

epochs_sgd = range(1,len(acc_sgd) +1)

# Plot of accuracy
plt.plot(epochs_sgd, acc_sgd, color='blue', label='Training Acc')
plt.plot(epochs_sgd, val_acc_sgd, color='red', label='Validation Acc')
plt.title('Training and Validation Accuracy (RSM)')
plt.legend()

plt.figure()

# Plot of loss
plt.plot(epochs_sgd, loss_sgd, color='blue', label='Training Loss')
plt.plot(epochs_sgd, val_loss_sgd, color='red', label='Validation Loss')
plt.title('Training and Validation Loss (RSM)')
plt.legend()
plt.figure()

plt.show()
```



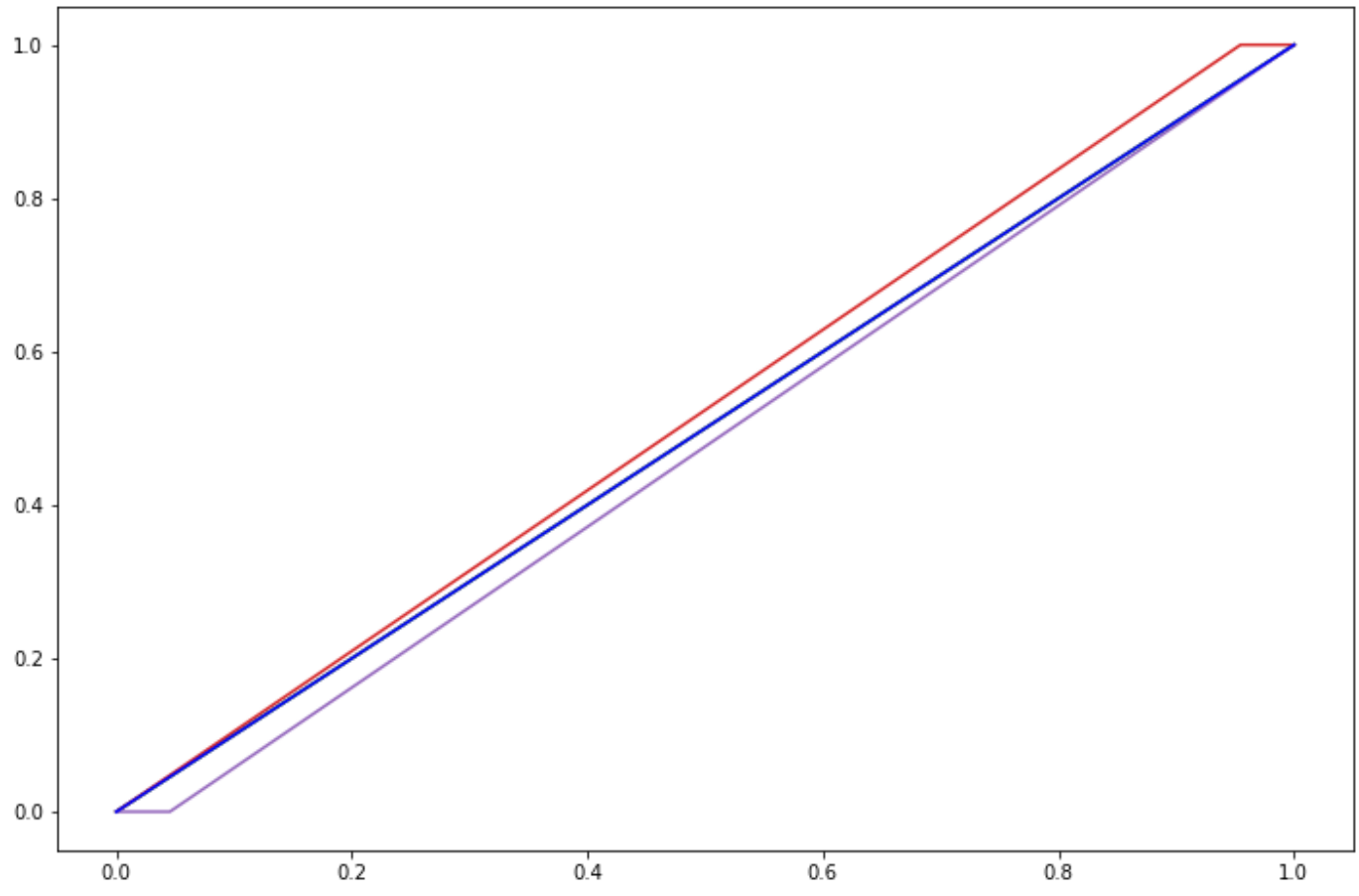


<Figure size 432x288 with 0 Axes>

In [53]:
```
# ---------------
# ROC/AUC Score |
# --------------

from sklearn.preprocessing import LabelBinarizer

# set plot figure size
fig, c_ax = plt.subplots(1,1, figsize = (12, 8))

def multiclass_roc_auc_score(y_test, y_pred, average="macro"):
    lb = LabelBinarizer()
    lb.fit(y_test)
    y_test = lb.transform(y_test)
    y_pred = lb.transform(y_pred)

    for (idx, c_label) in enumerate(target_names): # target_names: no of the labels
        fpr, tpr, thresholds = roc_curve(y_test[:,idx].astype(int), y_pred[:,idx])
        c_ax.plot(fpr, tpr, label = '%s (AUC:%0.2f)'  % (c_label, auc(fpr, tpr)))
    c_ax.plot(fpr, fpr, 'b-', label = 'Random Guessing')
    return roc_auc_score(y_test, y_pred, average=average)

validation_generator.reset() # resetting generator
y_pred = model3.predict_generator(validation_generator, verbose = True)
y_pred = np.argmax(y_pred, axis=1)
multiclass_roc_auc_score(validation_generator.classes, y_pred)
```

```
4/4 [==============================] - 0s 111ms/step
```

Out[53]:
```
0.5
```



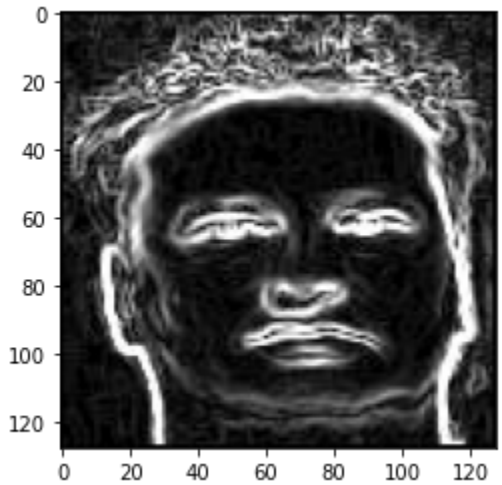
In [54]:
```
# ----------------------
# Classification Report |
# ----------------------

print('Classification Report for Model with RSM')
target_names = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']
print(classification_report(validation_generator.classes, y_pred, target_names=target_names
```

```
Classification Report for Model with RSM
              precision    recall  f1-score   support

   1_Neutral       0.00      0.00      0.00        22
   2_Smiling       0.00      0.00      0.00        22
    3_Sleepy       0.00      0.00      0.00        22
  4_Surprise       0.21      1.00      0.34        22
5_Sunglasses       0.00      0.00      0.00        22

    accuracy                           0.20       110
   macro avg       0.04      0.20      0.07       110
weighted avg       0.04      0.20      0.07       110


C:\Users\User\anaconda3\envs\tensorflow\lib\site-packages\sklearn\metrics\_classificatio
n.py:1221: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to control this be
havior.
  _warn_prf(average, modifier, msg_start, len(result))
```

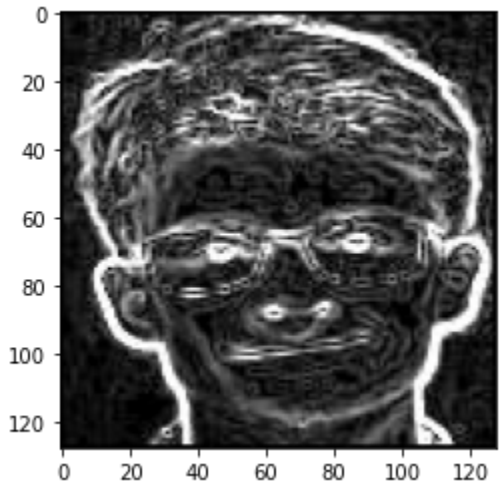
In [55]:
```
# ------------------
# Confusion Matrix |
# ------------------

num_of_train_samples = 444
#num_of_test_samples = 416 # steps per epoch
batch_size=32
steps_per_epoch=num_of_train_samples // batch_size
#validation_generator.reset()

Y_pred = model3.predict_generator(validation_generator, steps_per_epoch)
y_pred = np.argmax(Y_pred, axis=1)
print('Confusion Matrix for Model with RSM')
print(confusion_matrix(validation_generator.classes, y_pred))
```

```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 13 batches). You may need to use the repeat() function when building your dataset.
Confusion Matrix for Model with RSM
[[ 0  0  0 20  2]
 [ 0  0  0 21  1]
 [ 0  0  0 21  1]
 [ 0  0  0 22  0]
 [ 0  0  0 22  0]]
```

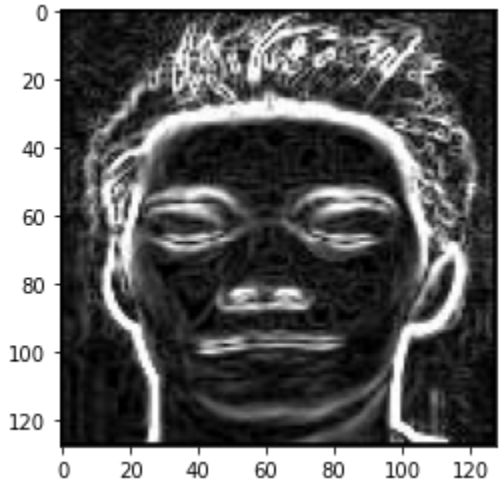
In [56]:
```
# ----------------------------------------
# Displaying 12 Images with the Prediction |
# ----------------------------------------

# 1 image from each class

labels = ['1_Neutral', '2_Smiling', '3_Sleepy', '4_Surprise', '5_Sunglasses']

i=0
for names in labels:
    pathToFolder = test_dir +'/'+names+'/'
    fnames = [os.path.join(pathToFolder, fname) for
          fname in os.listdir(pathToFolder)]

    # generate random number btwn (0,30)
    randNum = random.randint(0, 20)
    img_path = fnames[randNum]
    tmp_img = image.load_img(img_path, target_size = (128, 128))
    tmp_img = image.img_to_array(tmp_img)
    tmp_img = np.expand_dims(tmp_img, axis = 0)

    tmp_img /=255.
    plt.imshow(tmp_img[0])
    plt.show()
    # predict
    result = model3.predict(tmp_img)
    train_generator.class_indices
    print('Actual value: ',i,'\tPredicted value: ', np.argmax(result))
    i+=1

print('Total number of images for "testing":')
test_generator = test_datagen.flow_from_directory(test_dir,
                                                  target_size = (128, 128),
                                                  batch_size = 32,
                                                  class_mode = "categorical",
                                                  shuffle=False)
```



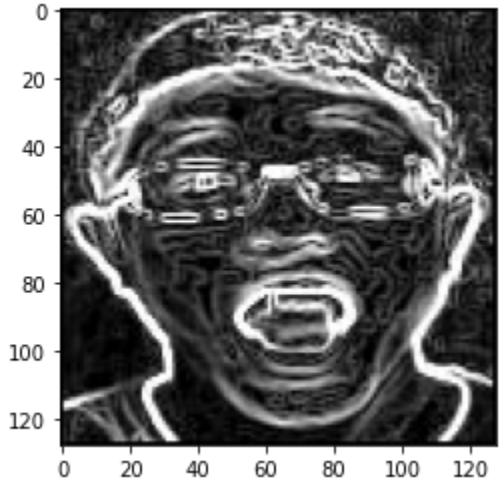

```
Actual value:  0        Predicted value:  3
```



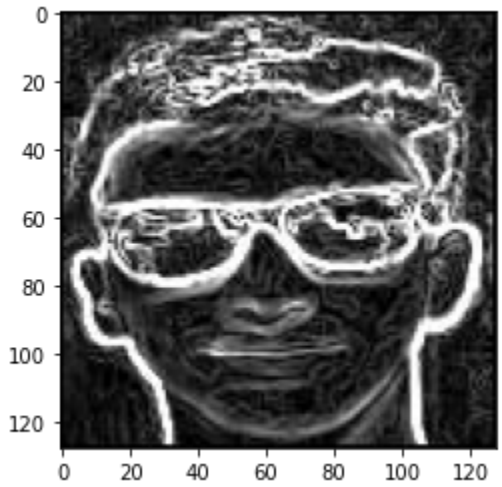
Actual value: 1   Predicted value: 3



Actual value: 2   Predicted value: 3



Actual value: 3   Predicted value: 3




```
Actual value:  4        Predicted value:  3
Total number of images for "testing":
Found 110 images belonging to 5 classes.
```

In [ ]: