In [ ]:

```
# 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 [1]:

```
# -----
# 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 [2]:

```
# --------
# 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 [3]:

```
# --------------
# 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 = './Desktop/Thermal2'

os.listdir('./Desktop/Thermal2') # returns list
```

Out[3]:

```
['test', 'train']
```

In [4]:

```
# ---------------------
# Mapping directories |
# --------------------

train_dir = './Desktop/Thermal2/train'
test_dir = './Desktop/Thermal2/test'
```

In [5]:
```python
# ---------------------------------------------
# Listing Amount of Images for Each Expression |
# --------------------------------------------

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

print('TRAINING SET')
files= os.listdir(".//Desktop//Thermal2//train")
for type in files:
    count = os.listdir('.//Desktop//Thermal2//train//'+type+'/')
    print(type+ "  "+ str(len(count)))
    train_count.append(len(count))
print()

print('TEST SET')
files= os.listdir('.//Desktop//Thermal2//test')

for type in files:
    count = os.listdir('.//Desktop//Thermal2//test//'+type+'/')
    print(type+ "  "+ str(len(count)))
    whole_count.append(len(count))
```

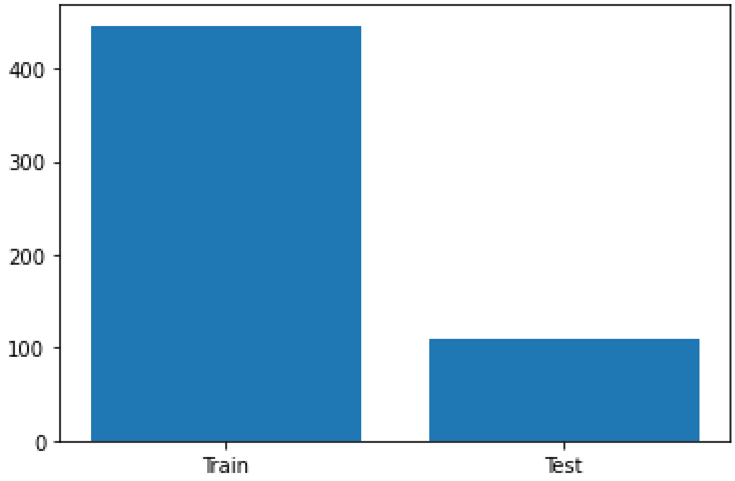
```
TRAINING SET
1_Neutral  89
2_Smiling  89
3_Sleepy  89
4_Surprise  89
5_Sunglasses  90

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

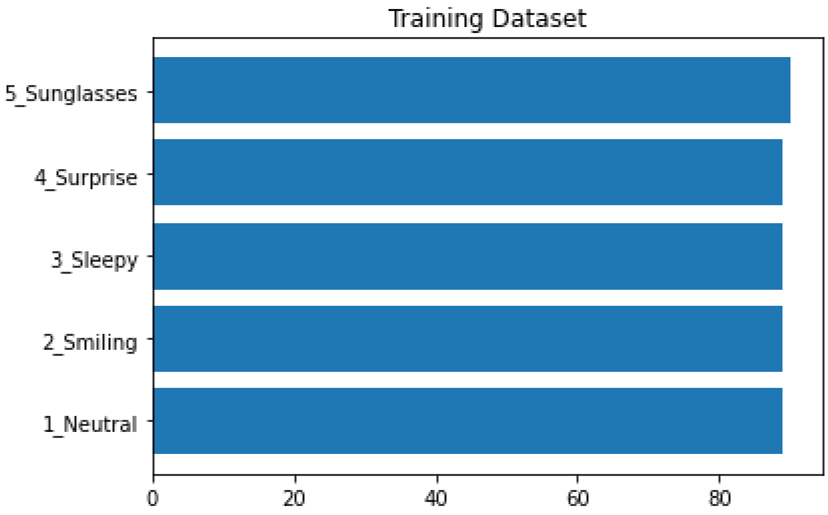
In [6]:
```python
# -------------------------------------
# Number of Samples in Each Directory |
# ------------------------------------

# 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)
```

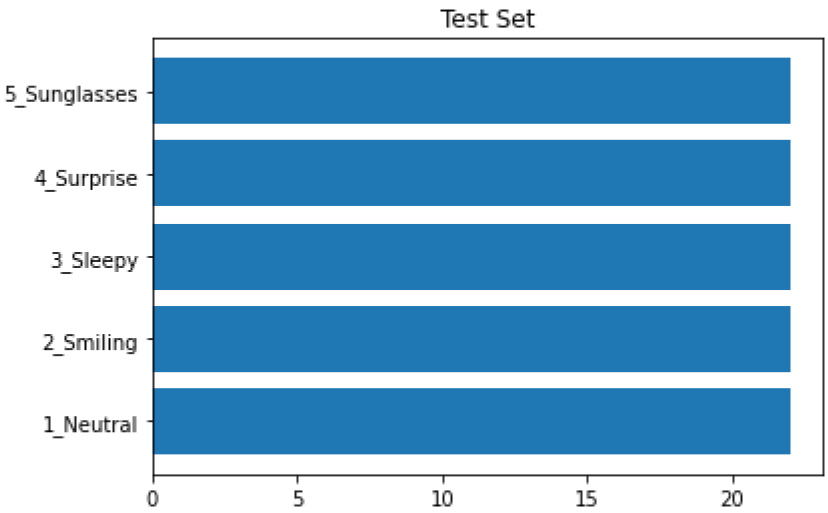
```
Number of Samples Train: 446
Number of Samples Test: 110
```

In [7]:

```
# ----------------------------------------
# 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()
```





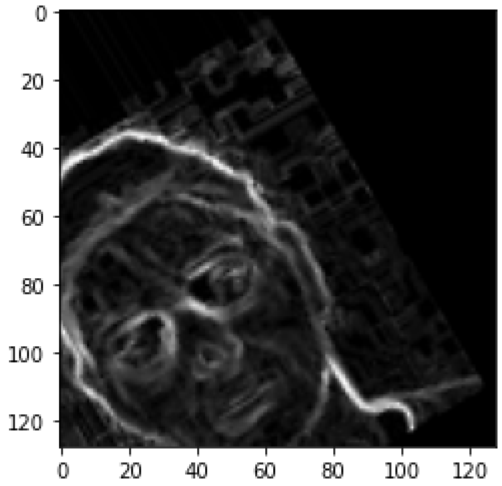


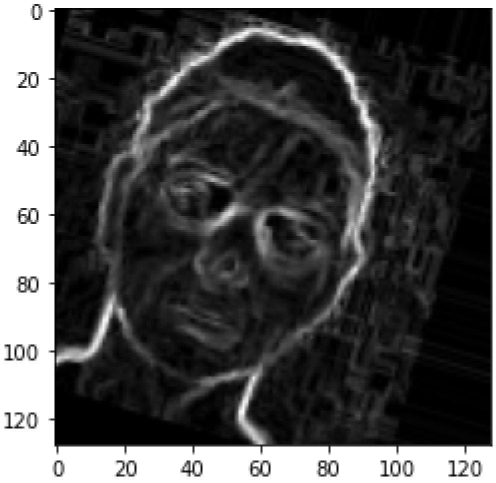
In [8]:
```
# --------------------
# ImageDataGenerator |
# -------------------

# 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)
```

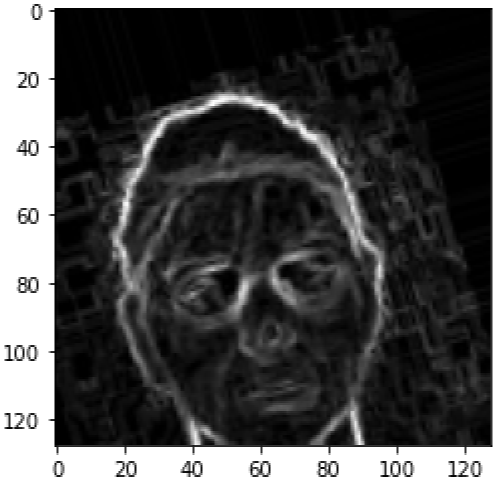
```
Found 446 images belonging to 5 classes.
Found 110 images belonging to 5 classes.
```

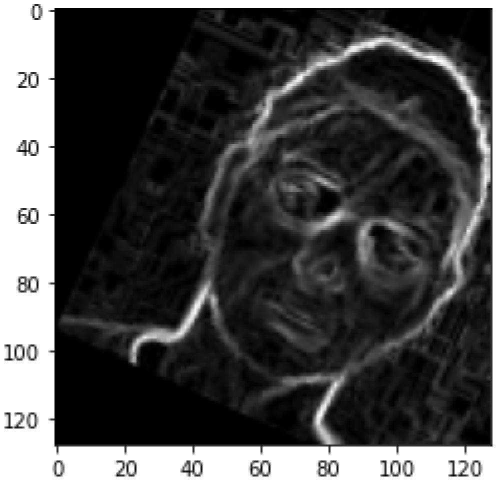
In [9]:

```
# ------------------------------------
# Displaing Randomly Augmented Images |
# -----------------------------------

from keras.preprocessing import image
train_dir = './/Desktop//Thermal2//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 [35]:

```
# 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(32, activation='relu',
                       kernel_initializer = 'glorot_normal',
                       bias_initializer = 'zeros'))
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(5, activation='softmax'))
model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_19 (Conv2D)           (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_5 (Batch (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_21 (Conv2D)           (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________________
flatten_5 (Flatten)          (None, 4608)              0
_________________________________________________________________
dropout_5 (Dropout)          (None, 4608)              0
```

```
_________________________________________________________________
dense_15 (Dense)             (None, 32)                147488
_________________________________________________________________
dense_16 (Dense)             (None, 64)                2112
_________________________________________________________________
dense_17 (Dense)             (None, 5)                 325
=================================================================
Total params: 390,885
Trainable params: 390,821
Non-trainable params: 64
_________________________________________________________________
```

In [36]:

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

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

In [37]:

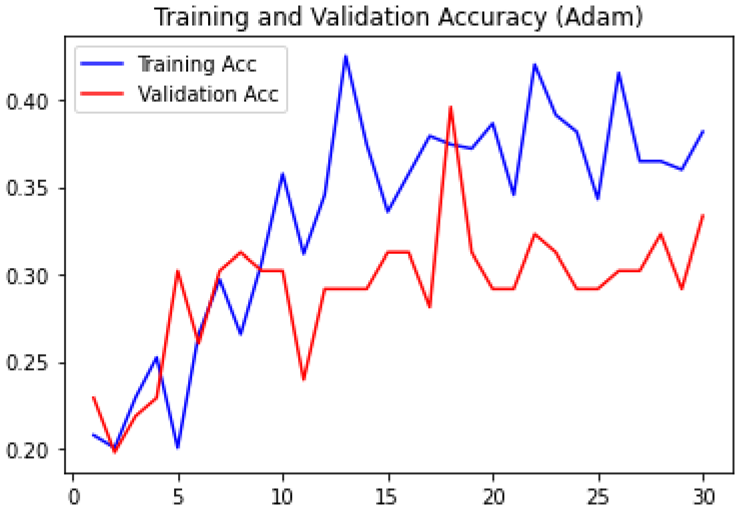
```
batch_size=32

# 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 [==============================] - 10s 760ms/step - loss: 1.6913 - acc: 0.2077 -
val_loss: 1.6076 - val_acc: 0.2292
Epoch 2/30
13/13 [==============================] - 9s 657ms/step - loss: 1.6147 - acc: 0.2005 - v
al_loss: 1.6099 - val_acc: 0.1979
Epoch 3/30
13/13 [==============================] - 9s 665ms/step - loss: 1.6032 - acc: 0.2295 - v
al_loss: 1.6074 - val_acc: 0.2188
Epoch 4/30
13/13 [==============================] - 8s 634ms/step - loss: 1.5971 - acc: 0.2524 - v
al_loss: 1.6049 - val_acc: 0.2292
Epoch 5/30
13/13 [==============================] - 9s 673ms/step - loss: 1.5990 - acc: 0.2005 - v
al_loss: 1.6001 - val_acc: 0.3021
Epoch 6/30
13/13 [==============================] - 8s 623ms/step - loss: 1.5485 - acc: 0.2657 - v
al_loss: 1.5901 - val_acc: 0.2604
Epoch 7/30
13/13 [==============================] - 8s 637ms/step - loss: 1.4785 - acc: 0.2971 - v
al_loss: 1.5634 - val_acc: 0.3021
Epoch 8/30
13/13 [==============================] - 8s 642ms/step - loss: 1.5025 - acc: 0.2657 - v
al_loss: 1.5672 - val_acc: 0.3125
Epoch 9/30
13/13 [==============================] - 8s 641ms/step - loss: 1.4339 - acc: 0.3068 - v
al_loss: 1.5492 - val_acc: 0.3021
Epoch 10/30
13/13 [==============================] - 8s 629ms/step - loss: 1.3908 - acc: 0.3575 - v
al_loss: 1.5288 - val_acc: 0.3021
Epoch 11/30
13/13 [==============================] - 8s 633ms/step - loss: 1.3737 - acc: 0.3116 - v
al_loss: 1.5054 - val_acc: 0.2396
Epoch 12/30
13/13 [==============================] - 8s 647ms/step - loss: 1.3212 - acc: 0.3454 - v
al_loss: 1.4880 - val_acc: 0.2917
Epoch 13/30
13/13 [==============================] - 8s 640ms/step - loss: 1.2217 - acc: 0.4251 - v
al_loss: 1.4388 - val_acc: 0.2917
Epoch 14/30
```

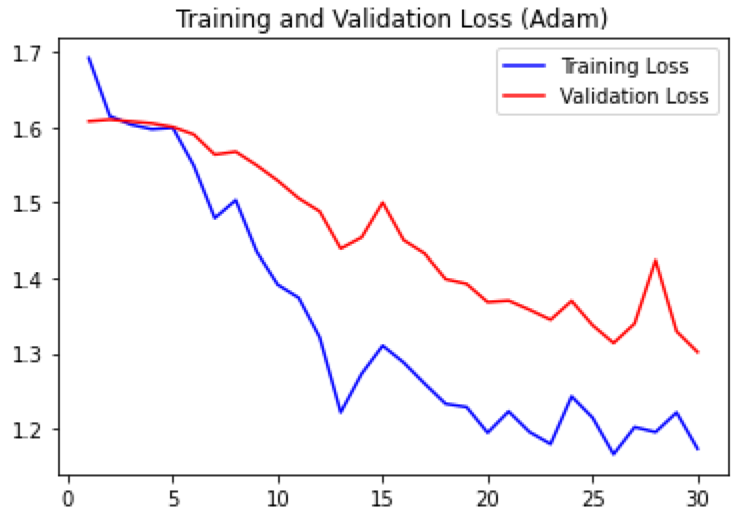
```
13/13 [==============================] - 8s 632ms/step - loss: 1.2727 - acc: 0.3744 - v
al_loss: 1.4538 - val_acc: 0.2917
Epoch 15/30
13/13 [==============================] - 8s 644ms/step - loss: 1.3102 - acc: 0.3357 - v
al_loss: 1.4998 - val_acc: 0.3125
Epoch 16/30
13/13 [==============================] - 8s 636ms/step - loss: 1.2885 - acc: 0.3575 - v
al_loss: 1.4500 - val_acc: 0.3125
Epoch 17/30
13/13 [==============================] - 8s 628ms/step - loss: 1.2599 - acc: 0.3792 - v
al_loss: 1.4322 - val_acc: 0.2812
Epoch 18/30
13/13 [==============================] - 9s 659ms/step - loss: 1.2331 - acc: 0.3744 - v
al_loss: 1.3987 - val_acc: 0.3958
Epoch 19/30
13/13 [==============================] - 9s 694ms/step - loss: 1.2285 - acc: 0.3720 - v
al_loss: 1.3924 - val_acc: 0.3125
Epoch 20/30
13/13 [==============================] - 9s 707ms/step - loss: 1.1949 - acc: 0.3865 - v
al_loss: 1.3683 - val_acc: 0.2917
Epoch 21/30
13/13 [==============================] - 8s 652ms/step - loss: 1.2242 - acc: 0.3454 - v
al_loss: 1.3703 - val_acc: 0.2917
Epoch 22/30
13/13 [==============================] - 8s 637ms/step - loss: 1.1963 - acc: 0.4203 - v
al_loss: 1.3581 - val_acc: 0.3229
Epoch 23/30
13/13 [==============================] - 9s 664ms/step - loss: 1.1800 - acc: 0.3913 - v
al_loss: 1.3449 - val_acc: 0.3125
Epoch 24/30
13/13 [==============================] - 8s 647ms/step - loss: 1.2433 - acc: 0.3816 - v
al_loss: 1.3699 - val_acc: 0.2917
Epoch 25/30
13/13 [==============================] - 8s 638ms/step - loss: 1.2145 - acc: 0.3430 - v
al_loss: 1.3377 - val_acc: 0.2917
Epoch 26/30
13/13 [==============================] - 8s 632ms/step - loss: 1.1663 - acc: 0.4155 - v
al_loss: 1.3140 - val_acc: 0.3021
Epoch 27/30
13/13 [==============================] - 8s 639ms/step - loss: 1.2030 - acc: 0.3647 - v
al_loss: 1.3398 - val_acc: 0.3021
Epoch 28/30
13/13 [==============================] - 8s 632ms/step - loss: 1.1960 - acc: 0.3647 - v
al_loss: 1.4238 - val_acc: 0.3229
Epoch 29/30
13/13 [==============================] - 8s 633ms/step - loss: 1.2215 - acc: 0.3599 - v
al_loss: 1.3296 - val_acc: 0.2917
Epoch 30/30
13/13 [==============================] - 8s 631ms/step - loss: 1.1737 - acc: 0.3816 - v
al_loss: 1.3021 - val_acc: 0.3333
```

In [38]:

```
# --------------------------------------------
# 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()
```



Training and Validation Accuracy (Adam)



```
<Figure size 432x288 with 0 Axes>
```

In [39]:

```
# -----------------------
# 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_name
```

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

   1_Neutral       0.28      0.73      0.40        22
   2_Smiling       0.40      0.09      0.15        22
    3_Sleepy       0.00      0.00      0.00        22
  4_Surprise       0.28      0.32      0.30        22
5_Sunglasses       1.00      1.00      1.00        22

    accuracy                           0.43       110
   macro avg       0.39      0.43      0.37       110
weighted avg       0.39      0.43      0.37       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))
```

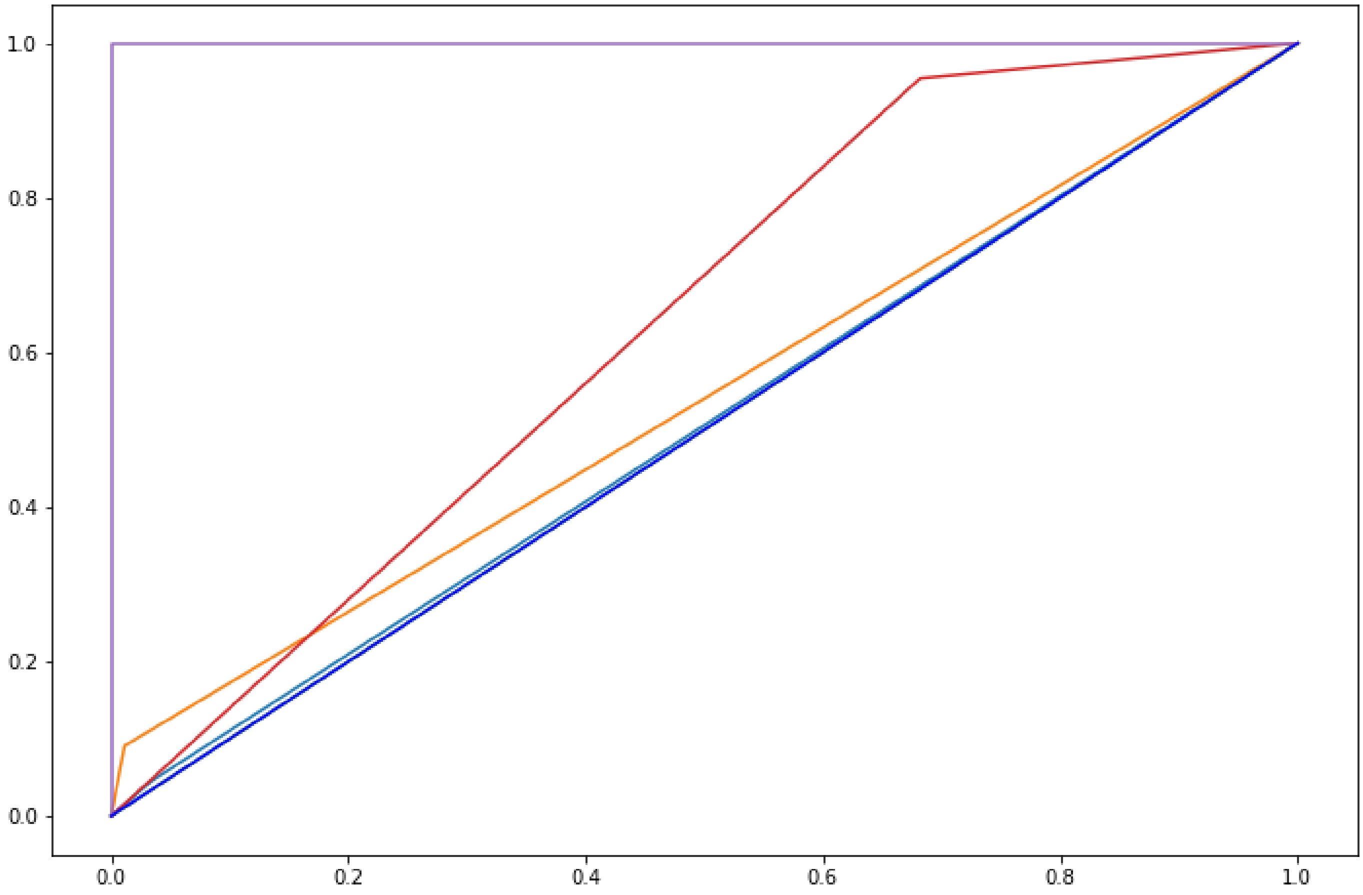
In [40]:

```
# ---------------
# 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)
```

```
4/4 [==============================] - 0s 109ms/step
```

Out[40]:

```
0.6363636363636364
```



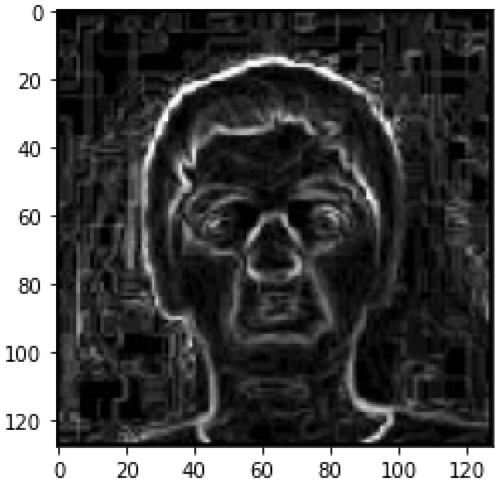
In [41]:

```
# ------------------
# 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))
```

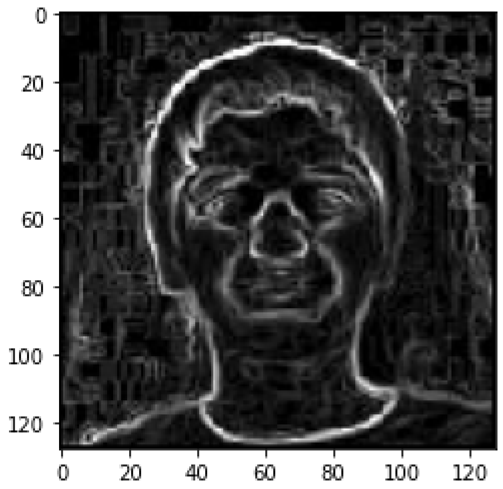
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-41-2188fe66f14c> in <module>
      9 #validation_generator.reset()
     10 
---> 11 Y_pred = model.predict_grenerator(validation_generator, 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 [42]:

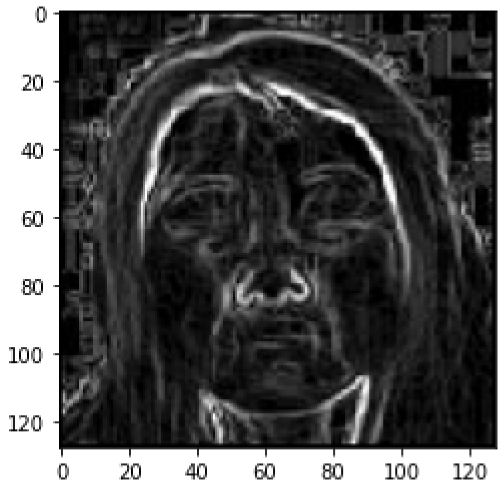
```
# ------------------------------------------
# Displaying 12 images with the prediction |
# -----------------------------------------

# 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)
```



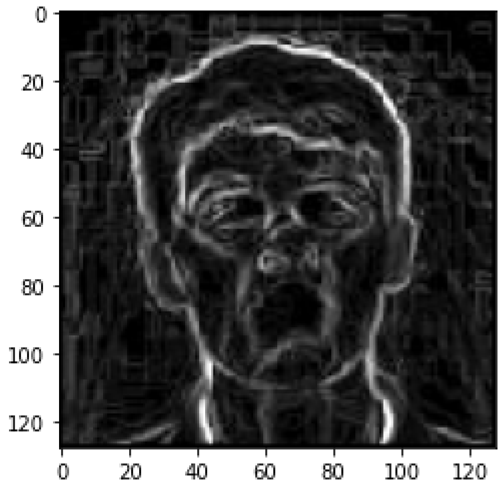
```
Actual value:  0        Predicted value:  3
```



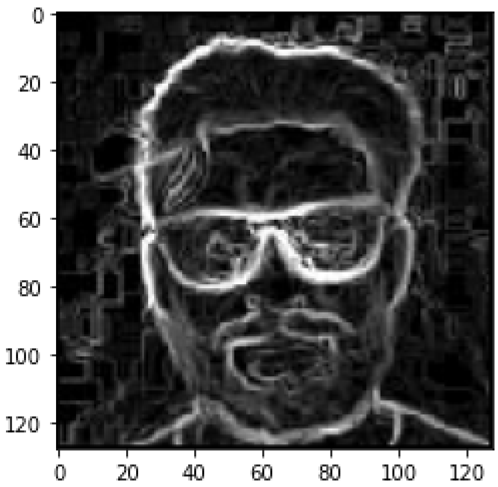
```
Actual value:  1        Predicted value:  3
```



```
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 [43]:

```
# 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_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 126, 126, 32)      896
_________________________________________________________________
batch_normalization_6 (Batch (None, 126, 126, 32)      128
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 61, 61, 64)        18496
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 30, 30, 64)        0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 28, 28, 128)       73856
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 14, 14, 128)       0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 12, 12, 128)       147584
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 6, 6, 128)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 4608)              0
_________________________________________________________________
dropout_6 (Dropout)          (None, 4608)              0
```

```
_________________________________________________________________
dense_18 (Dense)             (None, 32)                147488    
_________________________________________________________________
dense_19 (Dense)             (None, 64)                2112      
_________________________________________________________________
dense_20 (Dense)             (None, 5)                 325       
=================================================================
Total params: 390,885
Trainable params: 390,821
Non-trainable params: 64
_________________________________________________________________
```

In [44]:

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

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

In [45]:

```
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)
```

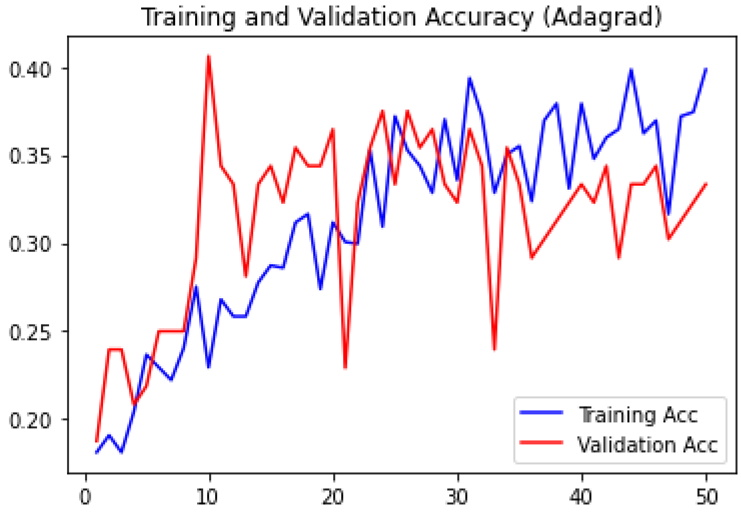
```
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
WARNING:tensorflow:sample_weight modes were coerced from
  ...
    to
  ['...']
Train for 13 steps, validate for 3 steps
Epoch 1/50
13/13 [==============================] - 10s 735ms/step - loss: 1.6587 - acc: 0.1812 -
val_loss: 1.6102 - val_acc: 0.1875
Epoch 2/50
13/13 [==============================] - 8s 637ms/step - loss: 1.6243 - acc: 0.1908 - v
al_loss: 1.6072 - val_acc: 0.2396
Epoch 3/50
13/13 [==============================] - 8s 622ms/step - loss: 1.6170 - acc: 0.1812 - v
al_loss: 1.6088 - val_acc: 0.2396
Epoch 4/50
13/13 [==============================] - 8s 629ms/step - loss: 1.6121 - acc: 0.2043 - v
al_loss: 1.6082 - val_acc: 0.2083
Epoch 5/50
13/13 [==============================] - 8s 632ms/step - loss: 1.6026 - acc: 0.2367 - v
al_loss: 1.6083 - val_acc: 0.2188
Epoch 6/50
13/13 [==============================] - 8s 641ms/step - loss: 1.5933 - acc: 0.2295 - v
al_loss: 1.6021 - val_acc: 0.2500
Epoch 7/50
13/13 [==============================] - 8s 630ms/step - loss: 1.6088 - acc: 0.2222 - v
al_loss: 1.6031 - val_acc: 0.2500
Epoch 8/50
13/13 [==============================] - 8s 631ms/step - loss: 1.5829 - acc: 0.2404 - v
al_loss: 1.5954 - val_acc: 0.2500
Epoch 9/50
13/13 [==============================] - 8s 643ms/step - loss: 1.5669 - acc: 0.2754 - v
al_loss: 1.5909 - val_acc: 0.2917
Epoch 10/50
13/13 [==============================] - 8s 620ms/step - loss: 1.5844 - acc: 0.2295 - v
al_loss: 1.5898 - val_acc: 0.4062
Epoch 11/50
13/13 [==============================] - 8s 629ms/step - loss: 1.5741 - acc: 0.2681 - v
al_loss: 1.5905 - val_acc: 0.3438
Epoch 12/50
13/13 [==============================] - 8s 620ms/step - loss: 1.5474 - acc: 0.2585 - v
al_loss: 1.5770 - val_acc: 0.3333
Epoch 13/50
13/13 [==============================] - 8s 622ms/step - loss: 1.5401 - acc: 0.2585 - v
al_loss: 1.5737 - val_acc: 0.2812
Epoch 14/50
```

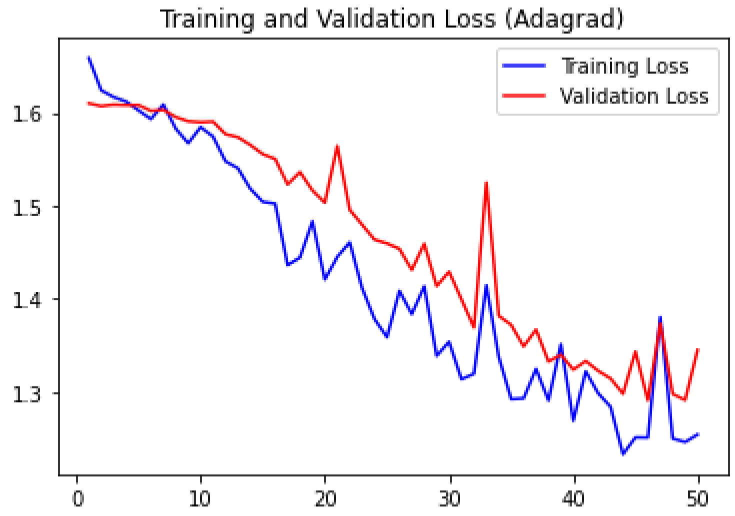
```
13/13 [==============================] - 8s 624ms/step - loss: 1.5180 - acc: 0.2778 - v
al_loss: 1.5654 - val_acc: 0.3333
Epoch 15/50
13/13 [==============================] - 8s 646ms/step - loss: 1.5039 - acc: 0.2874 - v
al_loss: 1.5555 - val_acc: 0.3438
Epoch 16/50
13/13 [==============================] - 8s 630ms/step - loss: 1.5023 - acc: 0.2861 - v
al_loss: 1.5503 - val_acc: 0.3229
Epoch 17/50
13/13 [==============================] - 8s 624ms/step - loss: 1.4358 - acc: 0.3116 - v
al_loss: 1.5230 - val_acc: 0.3542
Epoch 18/50
13/13 [==============================] - 8s 613ms/step - loss: 1.4442 - acc: 0.3164 - v
al_loss: 1.5362 - val_acc: 0.3438
Epoch 19/50
13/13 [==============================] - 8s 622ms/step - loss: 1.4832 - acc: 0.2740 - v
al_loss: 1.5165 - val_acc: 0.3438
Epoch 20/50
13/13 [==============================] - 8s 622ms/step - loss: 1.4201 - acc: 0.3116 - v
al_loss: 1.5033 - val_acc: 0.3646
Epoch 21/50
13/13 [==============================] - 8s 623ms/step - loss: 1.4451 - acc: 0.3005 - v
al_loss: 1.5642 - val_acc: 0.2292
Epoch 22/50
13/13 [==============================] - 8s 621ms/step - loss: 1.4601 - acc: 0.2995 - v
al_loss: 1.4957 - val_acc: 0.3229
Epoch 23/50
13/13 [==============================] - 8s 617ms/step - loss: 1.4115 - acc: 0.3527 - v
al_loss: 1.4796 - val_acc: 0.3542
Epoch 24/50
13/13 [==============================] - 8s 617ms/step - loss: 1.3783 - acc: 0.3092 - v
al_loss: 1.4636 - val_acc: 0.3750
Epoch 25/50
13/13 [==============================] - 8s 622ms/step - loss: 1.3589 - acc: 0.3720 - v
al_loss: 1.4597 - val_acc: 0.3333
Epoch 26/50
13/13 [==============================] - 8s 627ms/step - loss: 1.4076 - acc: 0.3527 - v
al_loss: 1.4537 - val_acc: 0.3750
Epoch 27/50
13/13 [==============================] - 8s 616ms/step - loss: 1.3837 - acc: 0.3438 - v
al_loss: 1.4312 - val_acc: 0.3542
Epoch 28/50
13/13 [==============================] - 8s 615ms/step - loss: 1.4139 - acc: 0.3285 - v
al_loss: 1.4591 - val_acc: 0.3646
Epoch 29/50
13/13 [==============================] - 8s 621ms/step - loss: 1.3388 - acc: 0.3702 - v
al_loss: 1.4138 - val_acc: 0.3333
Epoch 30/50
13/13 [==============================] - 8s 632ms/step - loss: 1.3536 - acc: 0.3357 - v
al_loss: 1.4293 - val_acc: 0.3229
Epoch 31/50
13/13 [==============================] - 8s 619ms/step - loss: 1.3133 - acc: 0.3937 - v
al_loss: 1.3996 - val_acc: 0.3646
Epoch 32/50
13/13 [==============================] - 8s 616ms/step - loss: 1.3198 - acc: 0.3720 - v
al_loss: 1.3693 - val_acc: 0.3438
Epoch 33/50
13/13 [==============================] - 8s 619ms/step - loss: 1.4143 - acc: 0.3285 - v
al_loss: 1.5249 - val_acc: 0.2396
```

```
Epoch 34/50
13/13 [==============================] - 8s 618ms/step - loss: 1.3387 - acc: 0.3502 - v
al_loss: 1.3819 - val_acc: 0.3542
Epoch 35/50
13/13 [==============================] - 8s 615ms/step - loss: 1.2919 - acc: 0.3551 - v
al_loss: 1.3721 - val_acc: 0.3333
Epoch 36/50
13/13 [==============================] - 8s 612ms/step - loss: 1.2926 - acc: 0.3237 - v
al_loss: 1.3489 - val_acc: 0.2917
Epoch 37/50
13/13 [==============================] - 8s 621ms/step - loss: 1.3257 - acc: 0.3696 - v
al_loss: 1.3671 - val_acc: 0.3021
Epoch 38/50
13/13 [==============================] - 8s 629ms/step - loss: 1.2905 - acc: 0.3792 - v
al_loss: 1.3327 - val_acc: 0.3125
Epoch 39/50
13/13 [==============================] - 8s 635ms/step - loss: 1.3506 - acc: 0.3309 - v
al_loss: 1.3401 - val_acc: 0.3229
Epoch 40/50
13/13 [==============================] - 8s 621ms/step - loss: 1.2688 - acc: 0.3792 - v
al_loss: 1.3240 - val_acc: 0.3333
Epoch 41/50
13/13 [==============================] - 8s 616ms/step - loss: 1.3215 - acc: 0.3478 - v
al_loss: 1.3333 - val_acc: 0.3229
Epoch 42/50
13/13 [==============================] - 8s 619ms/step - loss: 1.2983 - acc: 0.3599 - v
al_loss: 1.3228 - val_acc: 0.3438
Epoch 43/50
13/13 [==============================] - 8s 622ms/step - loss: 1.2835 - acc: 0.3647 - v
al_loss: 1.3145 - val_acc: 0.2917
Epoch 44/50
13/13 [==============================] - 8s 626ms/step - loss: 1.2330 - acc: 0.3986 - v
al_loss: 1.2981 - val_acc: 0.3333
Epoch 45/50
13/13 [==============================] - 8s 619ms/step - loss: 1.2511 - acc: 0.3623 - v
al_loss: 1.3435 - val_acc: 0.3333
Epoch 46/50
13/13 [==============================] - 8s 647ms/step - loss: 1.2506 - acc: 0.3696 - v
al_loss: 1.2910 - val_acc: 0.3438
Epoch 47/50
13/13 [==============================] - 9s 662ms/step - loss: 1.3793 - acc: 0.3164 - v
al_loss: 1.3737 - val_acc: 0.3021
Epoch 48/50
13/13 [==============================] - 8s 644ms/step - loss: 1.2509 - acc: 0.3720 - v
al_loss: 1.2979 - val_acc: 0.3125
Epoch 49/50
13/13 [==============================] - 8s 653ms/step - loss: 1.2456 - acc: 0.3744 - v
al_loss: 1.2913 - val_acc: 0.3229
Epoch 50/50
13/13 [==============================] - 8s 645ms/step - loss: 1.2540 - acc: 0.3986 - v
al_loss: 1.3451 - val_acc: 0.3333
```

In [46]:

```
# ---------------------------------------------
# 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()
```




Training and Validation Loss (Adagrad)
Training Loss
Validation Loss
1.6
1.5
1.4
1.3
0
10
20
30
40
50

```
<Figure size 432x288 with 0 Axes>
```

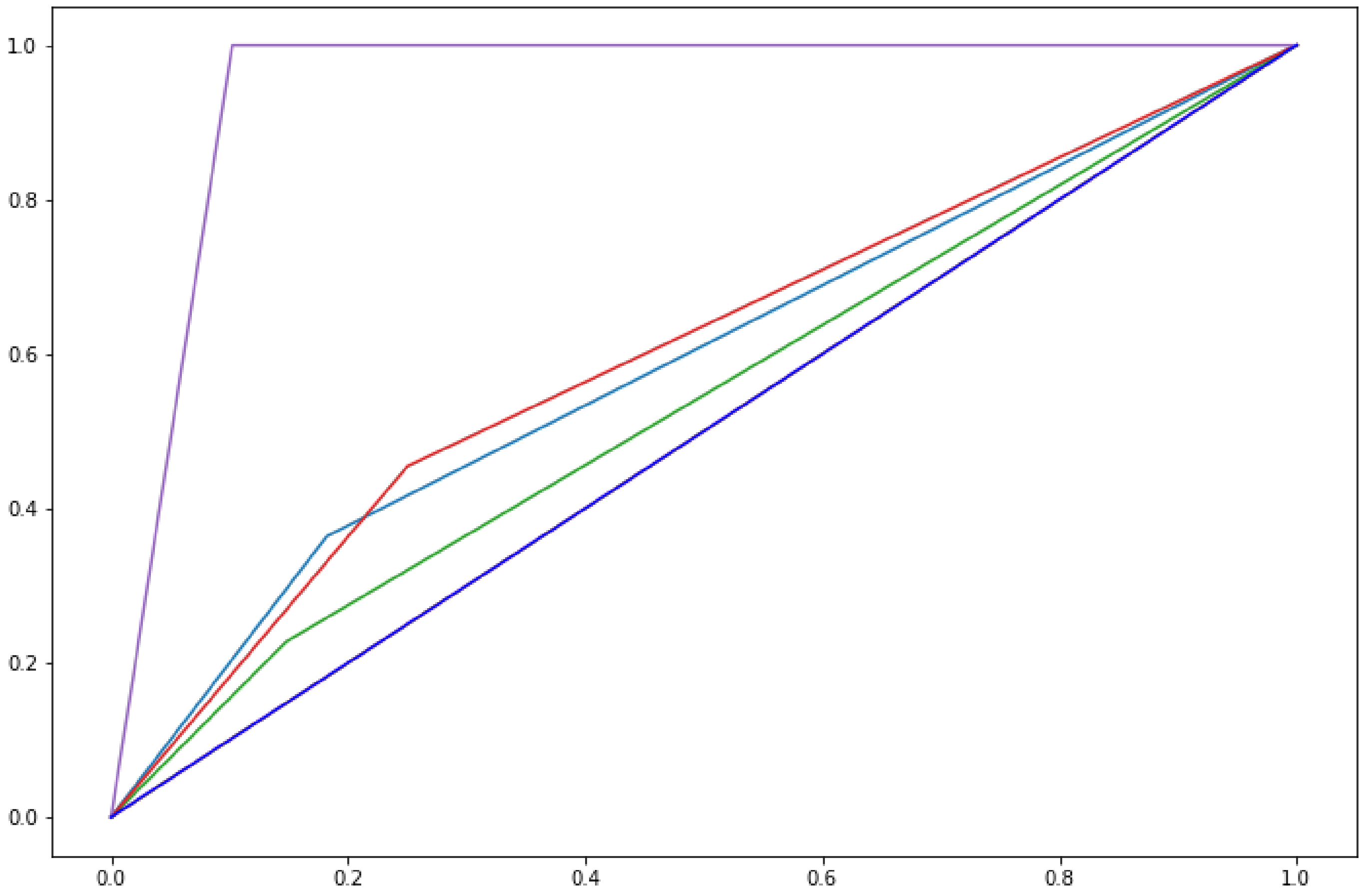
In [47]:

```
# ---------------
# 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 106ms/step
```

Out[47]:

```
0.6363636363636364
```



In [48]:

```
# ------------------------
# 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_name
```

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

   1_Neutral       0.33      0.36      0.35        22
   2_Smiling       0.20      0.05      0.07        22
    3_Sleepy       0.28      0.23      0.25        22
  4_Surprise       0.31      0.45      0.37        22
5_Sunglasses       0.71      1.00      0.83        22

    accuracy                           0.42       110
   macro avg       0.37      0.42      0.37       110
weighted avg       0.37      0.42      0.37       110
```

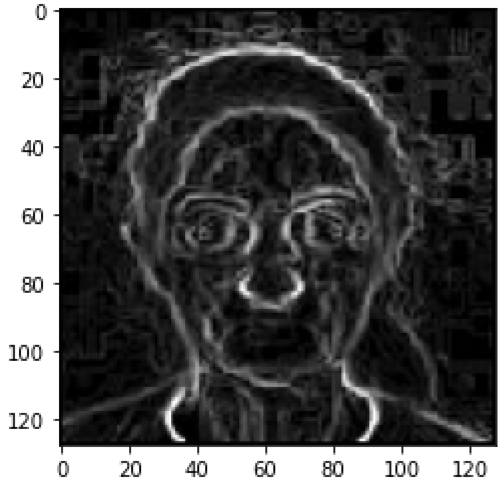
In [49]:

```
# ------------------
# 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))
```

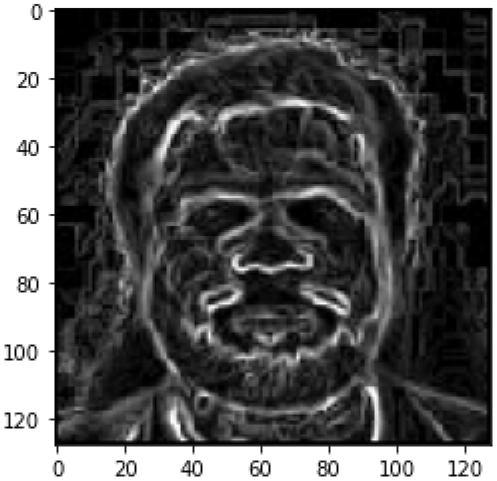
```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 13 batches). You may need to use the repeat() function when building your dataset.
Confusion Matrix fo Model with Adagrad
[[ 8  1  5  6  2]
 [ 4  1  4 10  3]
 [ 6  2  5  6  3]
 [ 6  1  4 10  1]
 [ 0  0  0  0 22]]
```

In [50]:

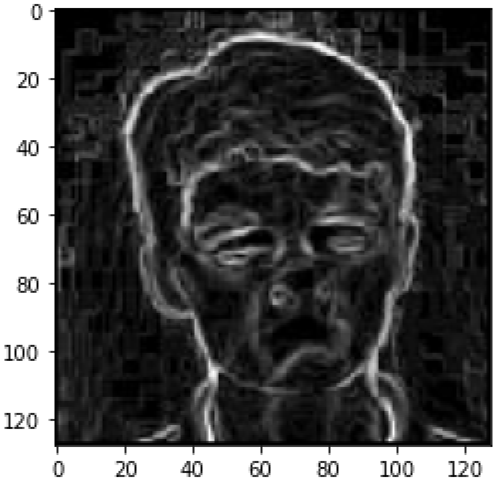
```
# -----------------------------------------
# Displaying 12 Images with the Prediction |
# -----------------------------------------

# 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)
```



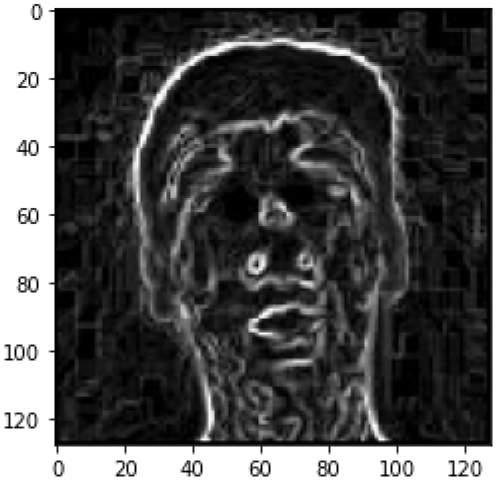
```
Actual value:  0        Predicted value:  3
```



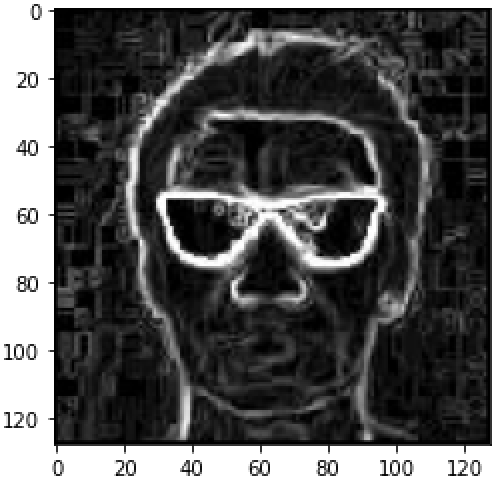
```
Actual value:  1        Predicted value:  2
```



```
Actual value:  2        Predicted value:  1
```



```
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 [105]:

```
# 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(32, (3,3), activation='relu'))
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.Flatten())
model3.add(layers.Dropout(.5))
model3.add(layers.Dense(128, activation='relu',
                       kernel_initializer = 'glorot_normal',
                       bias_initializer = 'zeros'))
model3.add(layers.Dense(512, activation='relu'))
model3.add(layers.Dense(5, activation='softmax'))
model3.summary()
```

```
Model: "sequential_15"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_59 (Conv2D)           (None, 126, 126, 32)      896       
_________________________________________________________________
batch_normalization_15 (Batc (None, 126, 126, 32)      128       
_________________________________________________________________
max_pooling2d_59 (MaxPooling (None, 63, 63, 32)        0         
_________________________________________________________________
conv2d_60 (Conv2D)           (None, 61, 61, 32)        9248      
_________________________________________________________________
max_pooling2d_60 (MaxPooling (None, 30, 30, 32)        0         
_________________________________________________________________
conv2d_61 (Conv2D)           (None, 28, 28, 64)        18496     
_________________________________________________________________
max_pooling2d_61 (MaxPooling (None, 14, 14, 64)        0         
_________________________________________________________________
conv2d_62 (Conv2D)           (None, 12, 12, 128)       73856     
_________________________________________________________________
max_pooling2d_62 (MaxPooling (None, 6, 6, 128)         0         
_________________________________________________________________
flatten_15 (Flatten)         (None, 4608)              0         
_________________________________________________________________
dropout_15 (Dropout)         (None, 4608)              0         
```

```
_________________________________________________________________
dense_45 (Dense)             (None, 128)               589952
_________________________________________________________________
dense_46 (Dense)             (None, 512)               66048
_________________________________________________________________
dense_47 (Dense)             (None, 5)                 2565
=================================================================
Total params: 761,189
Trainable params: 761,125
Non-trainable params: 64
_________________________________________________________________
```

In [106]:

```
rmsprop = optimizers.RMSprop(lr=0.0001, rho=0.9)

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

In [107]:

```
# 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 [==============================] - 8s 620ms/step - loss: 1.6533 - acc: 0.2029 - v
al_loss: 1.6117 - val_acc: 0.1042
Epoch 2/50
13/13 [==============================] - 7s 562ms/step - loss: 1.6196 - acc: 0.2126 - v
al_loss: 1.6145 - val_acc: 0.0833
Epoch 3/50
13/13 [==============================] - 7s 522ms/step - loss: 1.6118 - acc: 0.2126 - v
al_loss: 1.6103 - val_acc: 0.2188
Epoch 4/50
13/13 [==============================] - 8s 624ms/step - loss: 1.5871 - acc: 0.2754 - v
al_loss: 1.6090 - val_acc: 0.1562
Epoch 5/50
13/13 [==============================] - 7s 564ms/step - loss: 1.5933 - acc: 0.2391 - v
al_loss: 1.6063 - val_acc: 0.2396
Epoch 6/50
13/13 [==============================] - 7s 508ms/step - loss: 1.6091 - acc: 0.2512 - v
al_loss: 1.6068 - val_acc: 0.2188
Epoch 7/50
13/13 [==============================] - 7s 556ms/step - loss: 1.5772 - acc: 0.2778 - v
al_loss: 1.6055 - val_acc: 0.2812
Epoch 8/50
13/13 [==============================] - 8s 583ms/step - loss: 1.5976 - acc: 0.2198 - v
al_loss: 1.5974 - val_acc: 0.2604
Epoch 9/50
13/13 [==============================] - 7s 570ms/step - loss: 1.5718 - acc: 0.2729 - v
al_loss: 1.6009 - val_acc: 0.2604
Epoch 10/50
13/13 [==============================] - 8s 602ms/step - loss: 1.5799 - acc: 0.2440 - v
al_loss: 1.6030 - val_acc: 0.2083
Epoch 11/50
13/13 [==============================] - 8s 577ms/step - loss: 1.5607 - acc: 0.2512 - v
al_loss: 1.5892 - val_acc: 0.3021
Epoch 12/50
13/13 [==============================] - 7s 528ms/step - loss: 1.5529 - acc: 0.2633 - v
al_loss: 1.5966 - val_acc: 0.1875
Epoch 13/50
13/13 [==============================] - 7s 534ms/step - loss: 1.5479 - acc: 0.2596 - v
al_loss: 1.5764 - val_acc: 0.3333
Epoch 14/50
```

```
13/13 [==============================] - 7s 559ms/step - loss: 1.5244 - acc: 0.2802 - v
al_loss: 1.5732 - val_acc: 0.2604
Epoch 15/50
13/13 [==============================] - 8s 636ms/step - loss: 1.4980 - acc: 0.3068 - v
al_loss: 1.5650 - val_acc: 0.2604
Epoch 16/50
13/13 [==============================] - 7s 531ms/step - loss: 1.4873 - acc: 0.2971 - v
al_loss: 1.5545 - val_acc: 0.2812
Epoch 17/50
13/13 [==============================] - 7s 503ms/step - loss: 1.4681 - acc: 0.3382 - v
al_loss: 1.5390 - val_acc: 0.3542
Epoch 18/50
13/13 [==============================] - 7s 513ms/step - loss: 1.4704 - acc: 0.3140 - v
al_loss: 1.5404 - val_acc: 0.3333
Epoch 19/50
13/13 [==============================] - 7s 525ms/step - loss: 1.4563 - acc: 0.3406 - v
al_loss: 1.5100 - val_acc: 0.3333
Epoch 20/50
13/13 [==============================] - 7s 520ms/step - loss: 1.4140 - acc: 0.3599 - v
al_loss: 1.4983 - val_acc: 0.3125
Epoch 21/50
13/13 [==============================] - 7s 521ms/step - loss: 1.4367 - acc: 0.3357 - v
al_loss: 1.4944 - val_acc: 0.3438
Epoch 22/50
13/13 [==============================] - 7s 516ms/step - loss: 1.4081 - acc: 0.3019 - v
al_loss: 1.4783 - val_acc: 0.2812
Epoch 23/50
13/13 [==============================] - 7s 511ms/step - loss: 1.4322 - acc: 0.2971 - v
al_loss: 1.4690 - val_acc: 0.3333
Epoch 24/50
13/13 [==============================] - 7s 513ms/step - loss: 1.4012 - acc: 0.3333 - v
al_loss: 1.4688 - val_acc: 0.3229
Epoch 25/50
13/13 [==============================] - 7s 518ms/step - loss: 1.4000 - acc: 0.3213 - v
al_loss: 1.4346 - val_acc: 0.3542
Epoch 26/50
13/13 [==============================] - 7s 516ms/step - loss: 1.3838 - acc: 0.3261 - v
al_loss: 1.4274 - val_acc: 0.3125
Epoch 27/50
13/13 [==============================] - 7s 526ms/step - loss: 1.3571 - acc: 0.3502 - v
al_loss: 1.4085 - val_acc: 0.3229
Epoch 28/50
13/13 [==============================] - 7s 518ms/step - loss: 1.3357 - acc: 0.3647 - v
al_loss: 1.3871 - val_acc: 0.3333
Epoch 29/50
13/13 [==============================] - 7s 509ms/step - loss: 1.3573 - acc: 0.3454 - v
al_loss: 1.3822 - val_acc: 0.3542
Epoch 30/50
13/13 [==============================] - 7s 535ms/step - loss: 1.3726 - acc: 0.3237 - v
al_loss: 1.3719 - val_acc: 0.4167
Epoch 31/50
13/13 [==============================] - 7s 521ms/step - loss: 1.3628 - acc: 0.3502 - v
al_loss: 1.3650 - val_acc: 0.3646
Epoch 32/50
13/13 [==============================] - 7s 511ms/step - loss: 1.3394 - acc: 0.3164 - v
al_loss: 1.3621 - val_acc: 0.3646
Epoch 33/50
13/13 [==============================] - 7s 516ms/step - loss: 1.3659 - acc: 0.3454 - v
al_loss: 1.3504 - val_acc: 0.3333
```

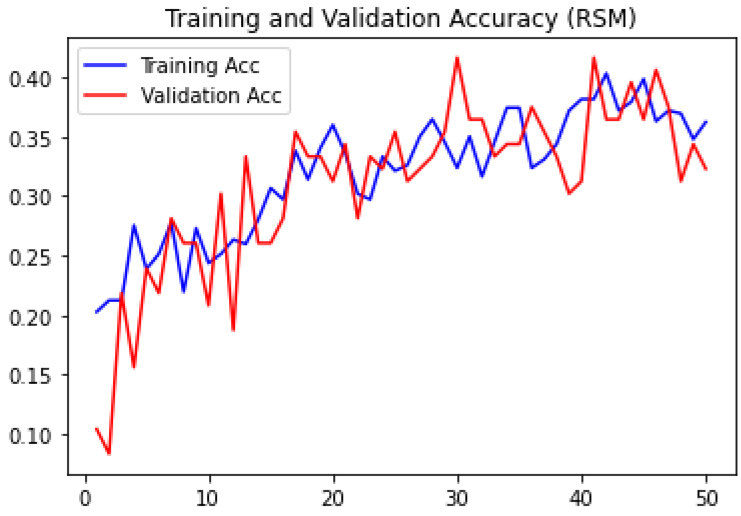
```
Epoch 34/50
13/13 [==============================] - 7s 520ms/step - loss: 1.2949 - acc: 0.3744 - v
al_loss: 1.3823 - val_acc: 0.3438
Epoch 35/50
13/13 [==============================] - 7s 512ms/step - loss: 1.3017 - acc: 0.3744 - v
al_loss: 1.3491 - val_acc: 0.3438
Epoch 36/50
13/13 [==============================] - 7s 522ms/step - loss: 1.3347 - acc: 0.3237 - v
al_loss: 1.3238 - val_acc: 0.3750
Epoch 37/50
13/13 [==============================] - 7s 521ms/step - loss: 1.3076 - acc: 0.3309 - v
al_loss: 1.3481 - val_acc: 0.3542
Epoch 38/50
13/13 [==============================] - 7s 511ms/step - loss: 1.3343 - acc: 0.3438 - v
al_loss: 1.3119 - val_acc: 0.3333
Epoch 39/50
13/13 [==============================] - 7s 521ms/step - loss: 1.2763 - acc: 0.3720 - v
al_loss: 1.4215 - val_acc: 0.3021
Epoch 40/50
13/13 [==============================] - 7s 532ms/step - loss: 1.2863 - acc: 0.3816 - v
al_loss: 1.4007 - val_acc: 0.3125
Epoch 41/50
13/13 [==============================] - 7s 509ms/step - loss: 1.2868 - acc: 0.3816 - v
al_loss: 1.3115 - val_acc: 0.4167
Epoch 42/50
13/13 [==============================] - 7s 525ms/step - loss: 1.2610 - acc: 0.4034 - v
al_loss: 1.3183 - val_acc: 0.3646
Epoch 43/50
13/13 [==============================] - 7s 513ms/step - loss: 1.2892 - acc: 0.3720 - v
al_loss: 1.2852 - val_acc: 0.3646
Epoch 44/50
13/13 [==============================] - 7s 519ms/step - loss: 1.2773 - acc: 0.3792 - v
al_loss: 1.2859 - val_acc: 0.3958
Epoch 45/50
13/13 [==============================] - 7s 518ms/step - loss: 1.2477 - acc: 0.3986 - v
al_loss: 1.2816 - val_acc: 0.3646
Epoch 46/50
13/13 [==============================] - 7s 514ms/step - loss: 1.2922 - acc: 0.3630 - v
al_loss: 1.2831 - val_acc: 0.4062
Epoch 47/50
13/13 [==============================] - 7s 513ms/step - loss: 1.2781 - acc: 0.3720 - v
al_loss: 1.3427 - val_acc: 0.3750
Epoch 48/50
13/13 [==============================] - 7s 516ms/step - loss: 1.2732 - acc: 0.3696 - v
al_loss: 1.2840 - val_acc: 0.3125
Epoch 49/50
13/13 [==============================] - 7s 518ms/step - loss: 1.2994 - acc: 0.3478 - v
al_loss: 1.2736 - val_acc: 0.3438
Epoch 50/50
13/13 [==============================] - 7s 528ms/step - loss: 1.2848 - acc: 0.3623 - v
al_loss: 1.3084 - val_acc: 0.3229
```

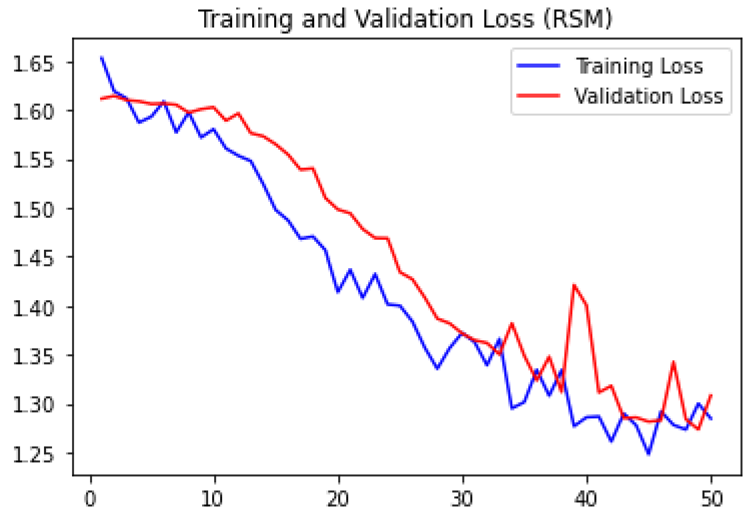
In [ ]:

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

In [108]:

```
# ---------------------------------------------
# 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()
```




Training and Validation Loss (RSM)
Training Loss
Validation Loss
1.65
1.60
1.55
1.50
1.45
1.40
1.35
1.30
1.25
0
10
20
30
40
50

```
<Figure size 432x288 with 0 Axes>
```

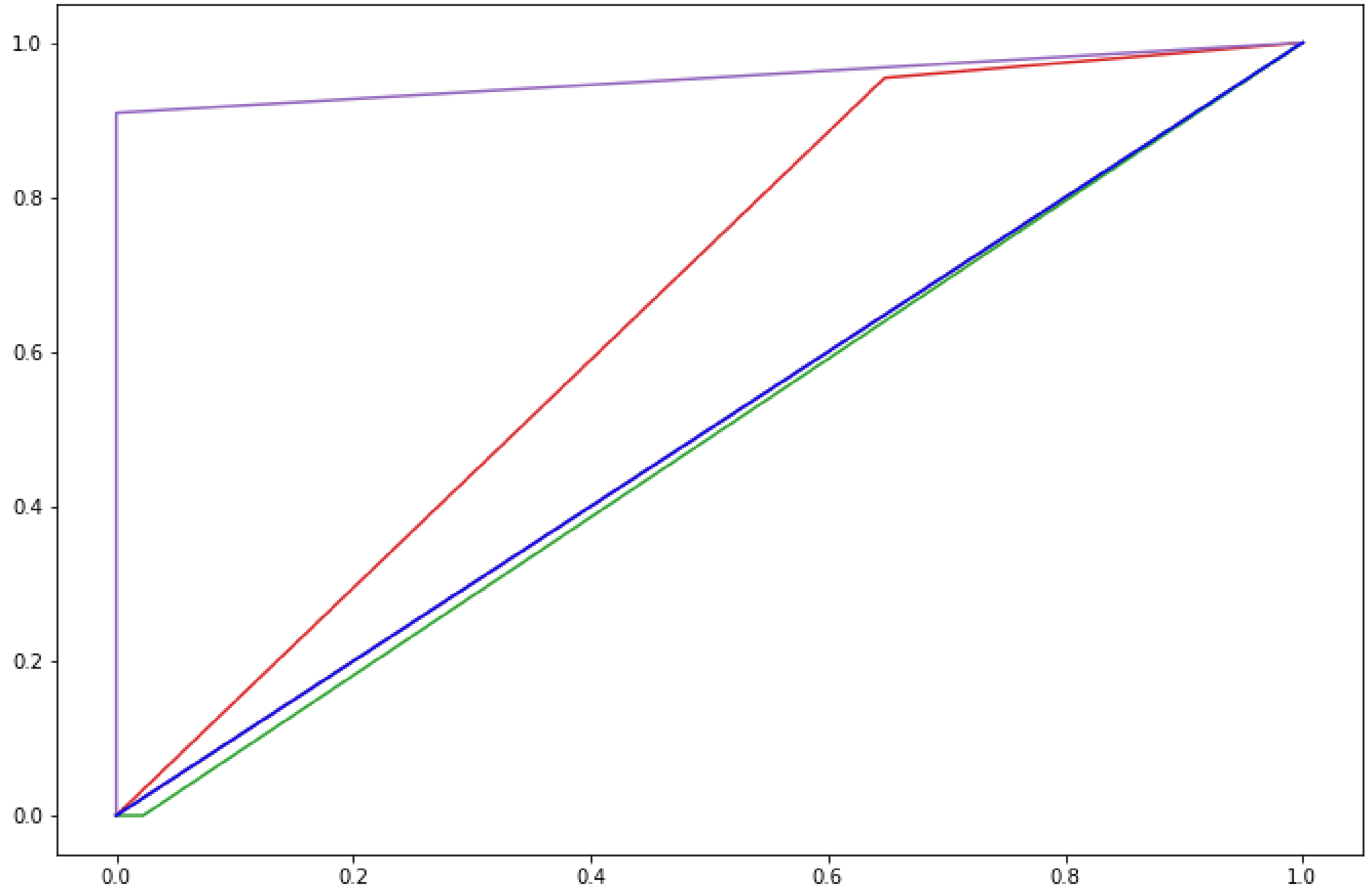
In [109]:

```
# ----------------
# 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 83ms/step
```

Out[109]:

```
0.6193181818181819
```



In [110]:

```
# -----------------------
# 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_name
```

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

   1_Neutral       0.20      0.05      0.07        22
   2_Smiling       0.20      0.05      0.07        22
    3_Sleepy       0.00      0.00      0.00        22
  4_Surprise       0.27      0.95      0.42        22
5_Sunglasses       1.00      0.91      0.95        22

    accuracy                           0.39       110
   macro avg       0.33      0.39      0.30       110
weighted avg       0.33      0.39      0.30       110
```

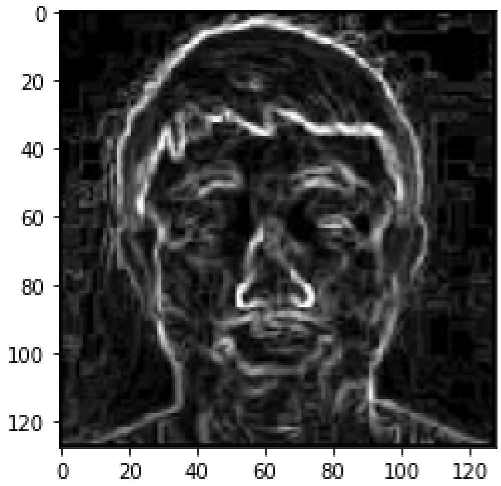
In [111]:

```
# ------------------
# 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))
```

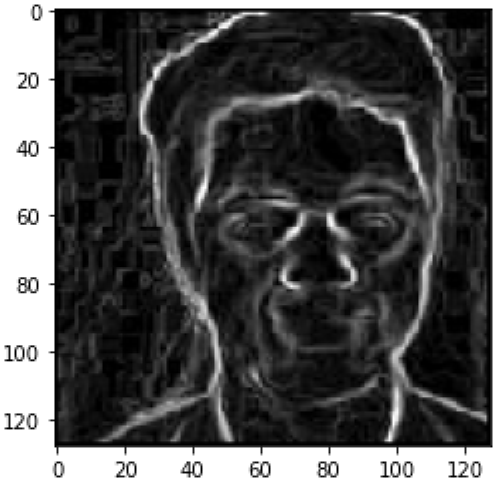
```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 13 batches). You may need to use the repeat() function when building your dataset.
Confusion Matrix for Model with RSM
[[ 1  1  0 20  0]
 [ 1  1  0 20  0]
 [ 3  2  0 17  0]
 [ 0  1  0 21  0]
 [ 0  0  2  0 20]]
```

In [112]:

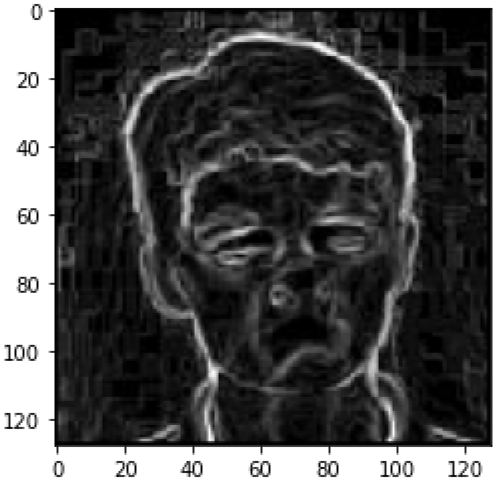
```
# -----------------------------------------
# Displaying 12 Images with the Prediction |
# -----------------------------------------

# 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)
```



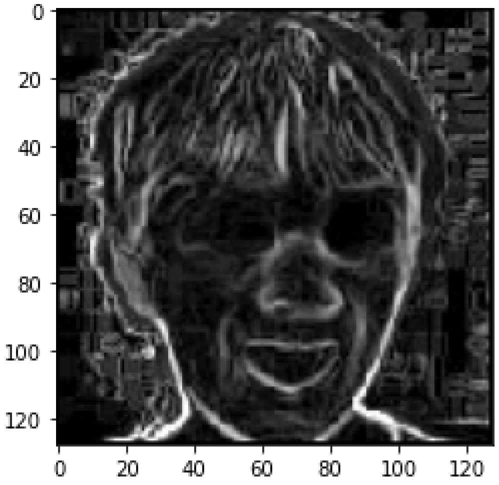
```
Actual value:  0        Predicted value:  3
```



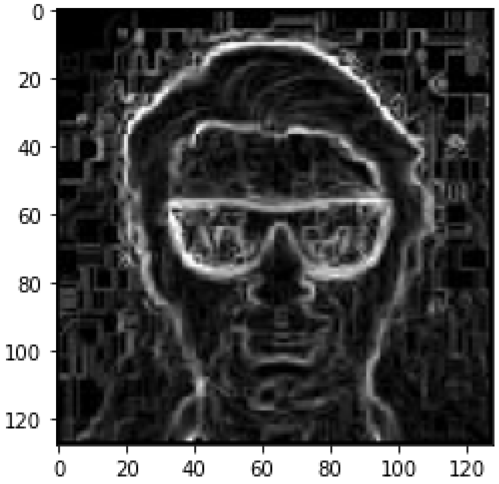
```
Actual value:  1        Predicted value:  3
```



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



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



```
Actual value:  4        Predicted value:  2
Total number of images for "testing":
Found 110 images belonging to 5 classes.
```

In [59]:

```
# 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_8"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_31 (Conv2D)           (None, 126, 126, 32)      896       
_________________________________________________________________
batch_normalization_8 (Batch (None, 126, 126, 32)      128       
_________________________________________________________________
max_pooling2d_31 (MaxPooling (None, 63, 63, 32)        0         
_________________________________________________________________
conv2d_32 (Conv2D)           (None, 61, 61, 64)        18496     
_________________________________________________________________
max_pooling2d_32 (MaxPooling (None, 30, 30, 64)        0         
_________________________________________________________________
conv2d_33 (Conv2D)           (None, 28, 28, 128)       73856     
_________________________________________________________________
max_pooling2d_33 (MaxPooling (None, 14, 14, 128)       0         
_________________________________________________________________
conv2d_34 (Conv2D)           (None, 12, 12, 128)       147584    
_________________________________________________________________
max_pooling2d_34 (MaxPooling (None, 6, 6, 128)         0         
_________________________________________________________________
flatten_8 (Flatten)          (None, 4608)              0         
_________________________________________________________________
dropout_8 (Dropout)          (None, 4608)              0         
```

```
_________________________________________________________________
dense_24 (Dense)             (None, 128)               589952
_________________________________________________________________
dense_25 (Dense)             (None, 512)               66048
_________________________________________________________________
dense_26 (Dense)             (None, 5)                 2565
=================================================================
Total params: 899,525
Trainable params: 899,461
Non-trainable params: 64
_________________________________________________________________
```

In [60]:
```python
# 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 [61]:
```
# 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 [==============================] - 9s 715ms/step - loss: 1.6929 - acc: 0.1884 - v
al_loss: 1.6110 - val_acc: 0.1250
Epoch 2/30
13/13 [==============================] - 8s 624ms/step - loss: 1.6204 - acc: 0.2101 - v
al_loss: 1.6098 - val_acc: 0.2292
Epoch 3/30
13/13 [==============================] - 8s 619ms/step - loss: 1.6207 - acc: 0.1981 - v
al_loss: 1.6086 - val_acc: 0.2292
Epoch 4/30
13/13 [==============================] - 8s 627ms/step - loss: 1.6138 - acc: 0.2005 - v
al_loss: 1.6075 - val_acc: 0.2396
Epoch 5/30
13/13 [==============================] - 9s 660ms/step - loss: 1.6052 - acc: 0.2295 - v
al_loss: 1.6071 - val_acc: 0.2708
Epoch 6/30
13/13 [==============================] - 9s 709ms/step - loss: 1.6091 - acc: 0.2077 - v
al_loss: 1.6072 - val_acc: 0.2917
Epoch 7/30
13/13 [==============================] - 9s 681ms/step - loss: 1.6123 - acc: 0.2077 - v
al_loss: 1.6065 - val_acc: 0.2604
Epoch 8/30
13/13 [==============================] - 8s 649ms/step - loss: 1.5980 - acc: 0.2115 - v
al_loss: 1.6062 - val_acc: 0.2500
Epoch 9/30
13/13 [==============================] - 9s 669ms/step - loss: 1.6123 - acc: 0.2319 - v
al_loss: 1.6051 - val_acc: 0.2292
Epoch 10/30
13/13 [==============================] - 9s 719ms/step - loss: 1.5984 - acc: 0.2560 - v
al_loss: 1.6040 - val_acc: 0.2292
Epoch 11/30
13/13 [==============================] - 8s 637ms/step - loss: 1.6084 - acc: 0.2101 - v
al_loss: 1.6042 - val_acc: 0.2396
Epoch 12/30
13/13 [==============================] - 9s 660ms/step - loss: 1.6083 - acc: 0.2319 - v
al_loss: 1.6038 - val_acc: 0.2396
Epoch 13/30
13/13 [==============================] - 9s 689ms/step - loss: 1.5959 - acc: 0.2512 - v
al_loss: 1.6030 - val_acc: 0.2396
Epoch 14/30
```

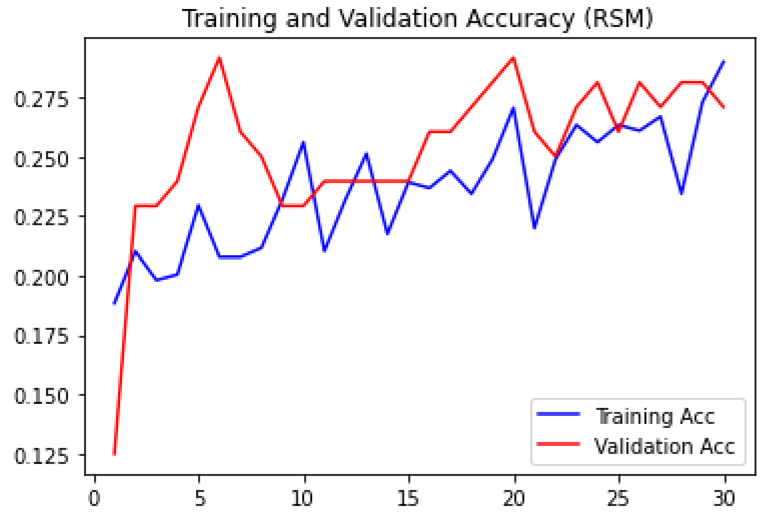
```
13/13 [==============================] - 9s 712ms/step - loss: 1.6028 - acc: 0.2174 - v
al_loss: 1.6031 - val_acc: 0.2396
Epoch 15/30
13/13 [==============================] - 8s 653ms/step - loss: 1.5943 - acc: 0.2391 - v
al_loss: 1.6021 - val_acc: 0.2396
Epoch 16/30
13/13 [==============================] - 8s 654ms/step - loss: 1.5855 - acc: 0.2367 - v
al_loss: 1.6024 - val_acc: 0.2604
Epoch 17/30
13/13 [==============================] - 8s 641ms/step - loss: 1.5952 - acc: 0.2440 - v
al_loss: 1.6022 - val_acc: 0.2604
Epoch 18/30
13/13 [==============================] - 8s 644ms/step - loss: 1.5888 - acc: 0.2343 - v
al_loss: 1.6021 - val_acc: 0.2708
Epoch 19/30
13/13 [==============================] - 9s 670ms/step - loss: 1.5920 - acc: 0.2488 - v
al_loss: 1.6009 - val_acc: 0.2812
Epoch 20/30
13/13 [==============================] - 9s 699ms/step - loss: 1.5891 - acc: 0.2705 - v
al_loss: 1.6013 - val_acc: 0.2917
Epoch 21/30
13/13 [==============================] - 9s 677ms/step - loss: 1.5943 - acc: 0.2198 - v
al_loss: 1.6006 - val_acc: 0.2604
Epoch 22/30
13/13 [==============================] - 9s 703ms/step - loss: 1.5853 - acc: 0.2488 - v
al_loss: 1.6004 - val_acc: 0.2500
Epoch 23/30
13/13 [==============================] - 9s 672ms/step - loss: 1.5824 - acc: 0.2633 - v
al_loss: 1.6005 - val_acc: 0.2708
Epoch 24/30
13/13 [==============================] - 9s 684ms/step - loss: 1.5855 - acc: 0.2560 - v
al_loss: 1.5991 - val_acc: 0.2812
Epoch 25/30
13/13 [==============================] - 9s 684ms/step - loss: 1.5910 - acc: 0.2633 - v
al_loss: 1.5982 - val_acc: 0.2604
Epoch 26/30
13/13 [==============================] - 9s 693ms/step - loss: 1.5873 - acc: 0.2609 - v
al_loss: 1.5982 - val_acc: 0.2812
Epoch 27/30
13/13 [==============================] - 9s 703ms/step - loss: 1.5797 - acc: 0.2668 - v
al_loss: 1.5965 - val_acc: 0.2708
Epoch 28/30
13/13 [==============================] - 8s 633ms/step - loss: 1.5879 - acc: 0.2343 - v
al_loss: 1.5945 - val_acc: 0.2812
Epoch 29/30
13/13 [==============================] - 9s 698ms/step - loss: 1.5731 - acc: 0.2729 - v
al_loss: 1.5944 - val_acc: 0.2812
Epoch 30/30
13/13 [==============================] - 9s 678ms/step - loss: 1.5781 - acc: 0.2899 - v
al_loss: 1.5925 - val_acc: 0.2708
```

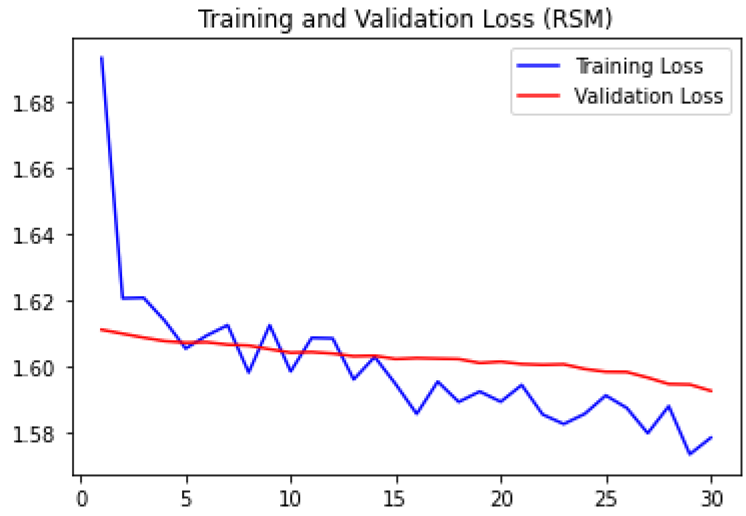
In [ ]:

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

In [62]:

```
# ----------------------------------------------
# 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()
```




Training and Validation Loss (RSM)
Training Loss
Validation Loss
1.68
1.66
1.64
1.62
1.60
1.58
0
5
10
15
20
25
30

```
<Figure size 432x288 with 0 Axes>
```

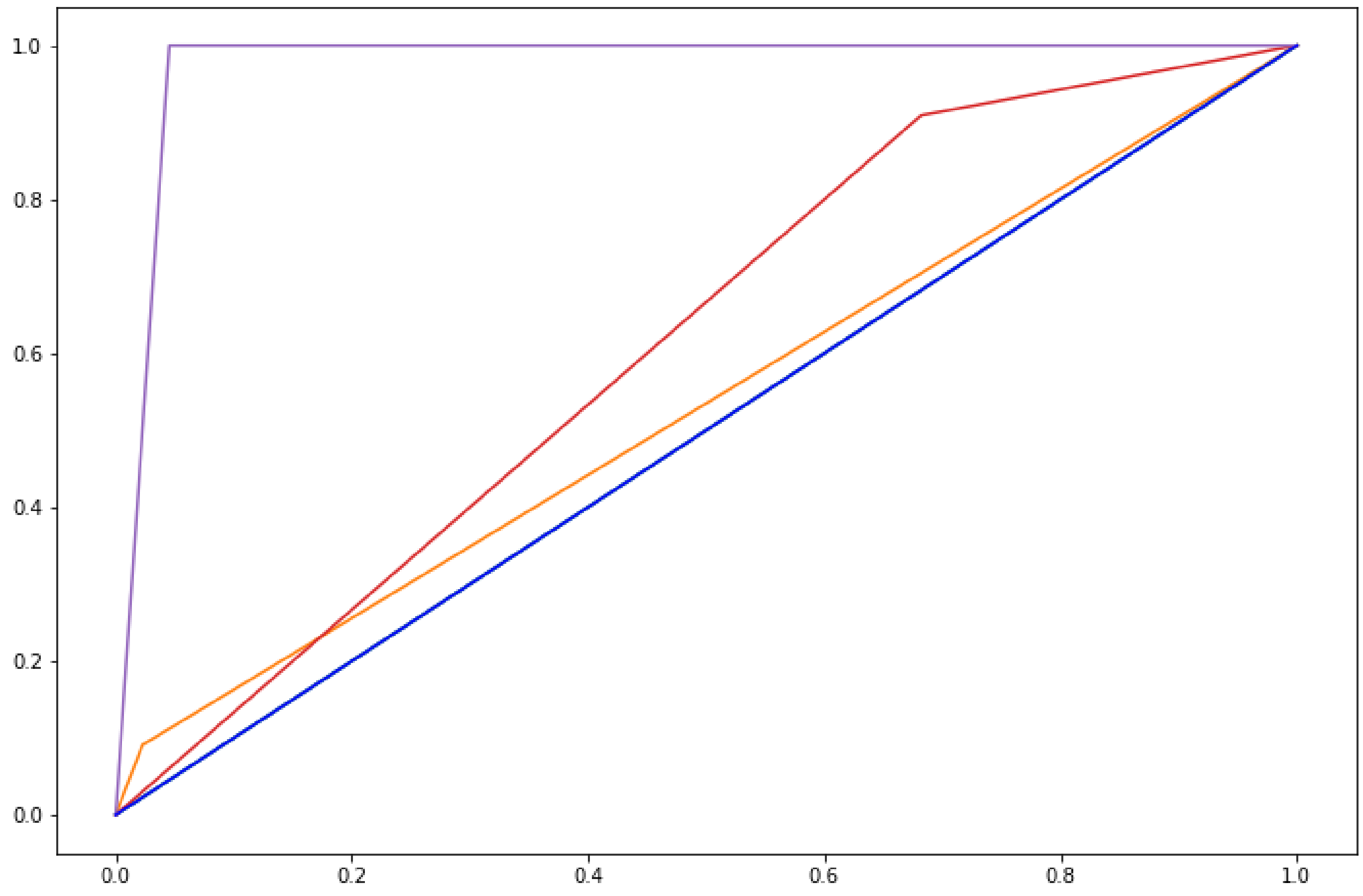
In [63]:

```
# ---------------
# 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 94ms/step
```

Out[63]:

```
0.625
```



In [64]:

```
# -----------------------
# 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_name
```

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

   1_Neutral       0.00      0.00      0.00        22
   2_Smiling       0.50      0.09      0.15        22
    3_Sleepy       0.00      0.00      0.00        22
  4_Surprise       0.25      0.91      0.39        22
5_Sunglasses       0.85      1.00      0.92        22

    accuracy                           0.40       110
   macro avg       0.32      0.40      0.29       110
weighted avg       0.32      0.40      0.29       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))
```

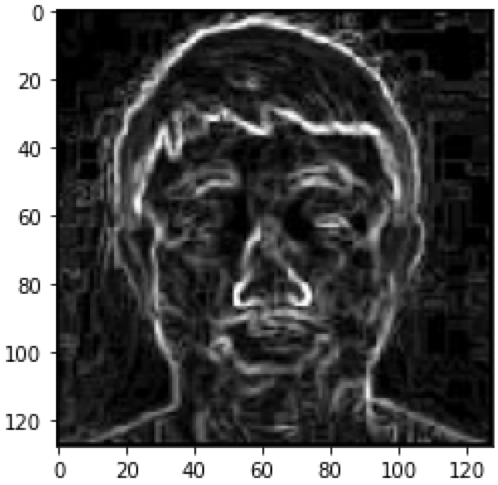
In [65]:

```
# ------------------
# 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))
```

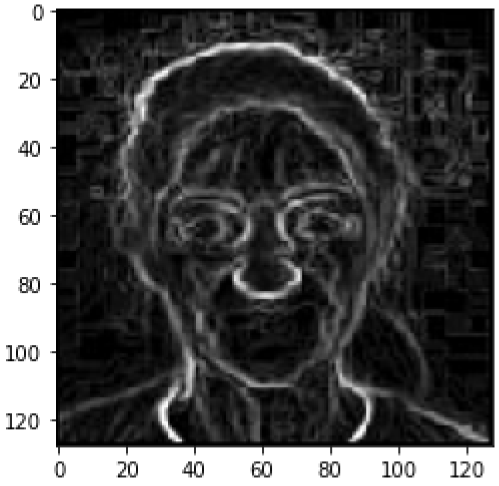
```
WARNING:tensorflow:Your input ran out of data; interrupting training. Make sure that your
dataset or generator can generate at least `steps_per_epoch * epochs` batches (in this ca
se, 13 batches). You may need to use the repeat() function when building your dataset.
Confusion Matrix for Model with RSM
[[ 0  1  0 20  1]
 [ 0  2  0 19  1]
 [ 0  0  0 21  1]
 [ 0  1  0 20  1]
 [ 0  0  0  0 22]]
```

In [66]:

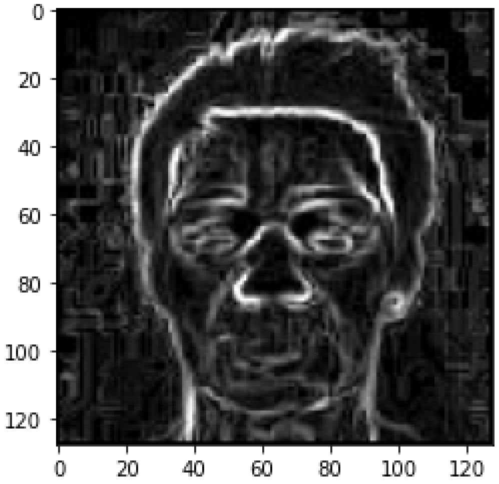
```
# ------------------------------------------
# Displaying 12 Images with the Prediction |
# -----------------------------------------

# 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)
```



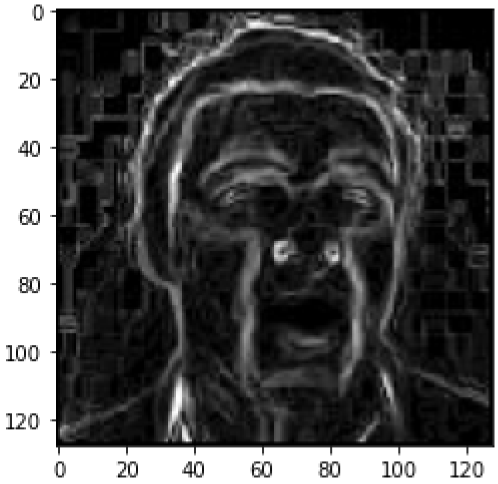
```
Actual value:  0        Predicted value:  3
```



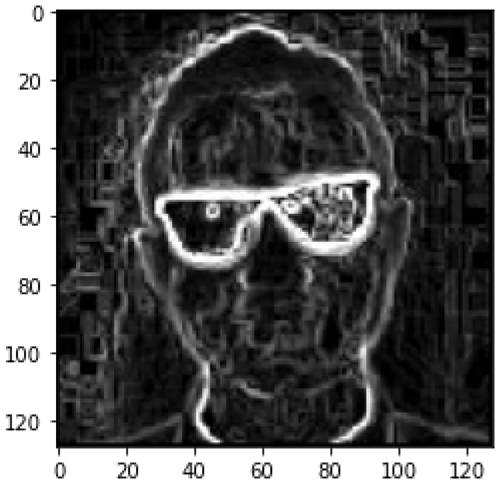
```
Actual value:  1        Predicted value:  1
```



```
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 [ ]:
```