Ifactory_good_vs_bad_final

September 20, 2019

```
In [1]: import cv2                  # working with, mainly resizing, images
        import numpy as np          # dealing with arrays
        import os                   # dealing with directories
        from random import shuffle # mixing up or currently ordered data that might lead our n
        from tqdm import tqdm      # a nice pretty percentage bar for tasks. Thanks to viewer
```

```
In [2]: !pip install tqdm
```

```
Requirement already satisfied: tqdm in c:\users\mahmo\anaconda3\lib\site-packages (4.19.5)


WARNING: You are using pip version 19.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
In [14]: #TRAIN_DIR = 'train_n'
         GOOD_DIR = 'dataset/good'
         BAD_DIR = 'dataset/bad'
         TEST_DIR = 'dataset/final_test'
         IMG_SIZE = 512
         LR = 0.001
```

```
In [15]: MODEL_NAME = 'Ifact-{}-{}.model'.format(LR, '2conv-basic') # just so we remember whic
```

```
In [16]: def create_train_data():
             training_data = []

             for img in tqdm(os.listdir(GOOD_DIR)):
                 label = [1.0,0.0] # 'good'
                 path = os.path.join(GOOD_DIR,img)
                 img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                 img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))/255
                 training_data.append([np.array(img),np.array(label)])

             for img in tqdm(os.listdir(BAD_DIR)):
                 label = [0.0,1.0] #'bad'
                 path = os.path.join(BAD_DIR,img)
                 img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                 img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))/255
                 training_data.append([np.array(img),np.array(label)])

             shuffle(training_data)
             np.save('train_data.npy', training_data)

             return training_data

In [17]: def process_test_data():
             testing_data = []
             cnt = 0
             for img in tqdm(os.listdir(TEST_DIR)):
                 path = os.path.join(TEST_DIR,img)
                 img_num = img.split('.')[0]
                 print(img_num)
                 img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                 img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
                 testing_data.append([np.array(img), img_num])
                 cnt += 1

             shuffle(testing_data)
             np.save('test_data.npy', testing_data)

In [18]: train_data = create_train_data()
         test_data = process_test_data()
         # If you have already created the dataset:
         #train_data = np.load('train_data.npy')
```

```
100%|| 66/66 [00:00<00:00, 295.36it/s]
100%|| 55/55 [00:00<00:00, 294.86it/s]
  0%|

IMG_0001
IMG_0002
IMG_0003
IMG_0004
IMG_0005
IMG_0006
IMG_0007
IMG_0008
IMG_0009
IMG_0010
IMG_0011
IMG_0012
IMG_0013
IMG_0014
IMG_0015
IMG_0016
IMG_0017
```

```
IMG_0018
IMG_0019
IMG_0020


  7%|

IMG_0021
IMG_0022
IMG_0023
IMG_0024
IMG_0025
IMG_0026
IMG_0027
IMG_0028
IMG_0029
IMG_0030
IMG_0031
IMG_0032
IMG_0033
IMG_0034
IMG_0035
IMG_0036
IMG_0037
IMG_0038
IMG_0039
IMG_0040
IMG_0041


 14%|

IMG_0042
IMG_0043
IMG_0044
IMG_0045
IMG_0046
IMG_0047
IMG_0048
IMG_0049
IMG_0050
IMG_0051
IMG_0052
IMG_0053
IMG_0054
IMG_0055
IMG_0056
IMG_0057
IMG_0058
```

```
IMG_0059
IMG_0060
IMG_0061
IMG_0062
IMG_0063

 21%|

IMG_0064
IMG_0065
IMG_0066
IMG_0067
IMG_0068
IMG_0069
IMG_0070
IMG_0071
IMG_0072
IMG_0073
IMG_0074
IMG_0075
IMG_0076
IMG_0077
IMG_0078
IMG_0079
IMG_0080
IMG_0081
IMG_0082
IMG_0083
IMG_0084
IMG_0085

 28%|

IMG_0086
IMG_0087
IMG_0088
IMG_0089
IMG_0090
IMG_0091
IMG_0092
IMG_0093
IMG_0094
IMG_0095
IMG_0096
IMG_0097
IMG_0098
IMG_0099
```

```
IMG_0100
IMG_0101
IMG_0102
IMG_0103
IMG_0104
IMG_0105
IMG_0106
IMG_0107


 36%|

IMG_0108
IMG_0109
IMG_0110
IMG_0111
IMG_0112
IMG_0113
IMG_0114
IMG_0115
IMG_0116
IMG_0117
IMG_0118
IMG_0119
IMG_0120
IMG_0121
IMG_0122
IMG_0123
IMG_0124
IMG_0125
IMG_0126
IMG_0127
IMG_0128
IMG_0129


 43%|

IMG_0130
IMG_0131
IMG_0132
IMG_0133
IMG_0134
IMG_0135
IMG_0136
IMG_0137
IMG_0138
IMG_0139
IMG_0140
```

```
IMG_0141
IMG_0142
IMG_0143
IMG_0144
IMG_0145
IMG_0146
IMG_0147
IMG_0148
IMG_0149
IMG_0150
IMG_0151
IMG_0152
IMG_0153

 51%|

IMG_0154
IMG_0155
IMG_0156
IMG_0157
IMG_0158
IMG_0159
IMG_0160
IMG_0161
IMG_0162
IMG_0163
IMG_0164
IMG_0165
IMG_0166
IMG_0167
IMG_0168
IMG_0169
IMG_0170
IMG_0171
IMG_0172
IMG_0173
IMG_0174
IMG_0175
IMG_0176
IMG_0177

 59%|                                                                                | 177/300

IMG_0178
IMG_0179
IMG_0180
IMG_0181
```

```
IMG_0182
IMG_0183
IMG_0184
IMG_0185
IMG_0186
IMG_0187
IMG_0188
IMG_0189
IMG_0190
IMG_0191
IMG_0192
IMG_0193
IMG_0194
IMG_0196
IMG_0197
IMG_0198
IMG_0199
IMG_0200
IMG_0201
IMG_0202


 67%|                                                                       | 201/300 [00:00<00:00, 2

IMG_0203
IMG_0204
IMG_0205
IMG_0206
IMG_0207
IMG_0208
IMG_0209
IMG_0210
IMG_0211
IMG_0212
IMG_0213
IMG_0214
IMG_0215
IMG_0216
IMG_0217
IMG_0218
IMG_0219
IMG_0220
IMG_0221
IMG_0222
IMG_0223
IMG_0224
IMG_0225
```

```
 75%|                                                 | 224/300 [00:01<00:00, 216.38it/s]

IMG_0226
IMG_0227
IMG_0228
IMG_0229
IMG_0230
IMG_0231
IMG_0232
IMG_0233
IMG_0234
IMG_0235
IMG_0236
IMG_0237
IMG_0238
IMG_0239
IMG_0240
IMG_0242
IMG_0243
IMG_0244
IMG_0245
IMG_0246
IMG_0247
IMG_0248


 82%|                                   | 246/300 [00:01<00:00, 215.80it/s]

IMG_0249
IMG_0250
IMG_0251
IMG_0252
IMG_0253
IMG_0254
IMG_0255
IMG_0256
IMG_0257
IMG_0258
IMG_0259
IMG_0260
IMG_0261
IMG_0262
IMG_0263
IMG_0264
IMG_0265
IMG_0266
IMG_0267
IMG_0268
```

```
IMG_0269
IMG_0270
IMG_0271
```

```
 90%|                    | 269/300 [00:01<00:00, 216.12it/s]
```

```
IMG_0272
IMG_0273
IMG_0274
IMG_0275
IMG_0276
IMG_0277
IMG_0278
IMG_0279
IMG_0280
IMG_0281
IMG_0282
IMG_0283
IMG_0284
IMG_0285
IMG_0286
IMG_0287
IMG_0288
IMG_0289
IMG_0290
IMG_0291
IMG_0292
IMG_0293
IMG_0294
```

```
 97%|     | 292/300 [00:01<00:00, 216.88it/s]
```

```
IMG_0295
IMG_0296
IMG_0297
IMG_0298
IMG_0299
IMG_0300
IMG_0301
IMG_0302
```

```
100%|| 300/300 [00:01<00:00, 216.87it/s]
```

```
In [19]: !pip install tflearn
```

```
Requirement already satisfied: tflearn in c:\users\mahmo\anaconda3\lib\site-packages (0.3.2)
Requirement already satisfied: six in c:\users\mahmo\anaconda3\lib\site-packages (from tflearn)
Requirement already satisfied: Pillow in c:\users\mahmo\anaconda3\lib\site-packages (from tflea
Requirement already satisfied: numpy in c:\users\mahmo\appdata\roaming\python\python36\site-pac


WARNING: You are using pip version 19.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
In [20]: !pip install tensorflow_cpu
```

```
Collecting tensorflow_cpu


  ERROR: Could not find a version that satisfies the requirement tensorflow_cpu (from versions
ERROR: No matching distribution found for tensorflow_cpu
WARNING: You are using pip version 19.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
In [21]: !pip install --upgrade tflearn
```

```
Requirement already up-to-date: tflearn in c:\users\mahmo\anaconda3\lib\site-packages (0.3.2)
Requirement already satisfied, skipping upgrade: six in c:\users\mahmo\anaconda3\lib\site-packa
Requirement already satisfied, skipping upgrade: numpy in c:\users\mahmo\appdata\roaming\pythor
Requirement already satisfied, skipping upgrade: Pillow in c:\users\mahmo\anaconda3\lib\site-pa


WARNING: You are using pip version 19.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
In [22]: krl = 3
         drp_out_per = 0.2
         epoc_n = 7
         bsize = 20
```

```
In [25]: import tensorflow as tf

         import tflearn
         from tflearn.layers.conv import conv_2d, max_pool_2d
         from tflearn.layers.core import input_data, dropout, fully_connected
         from tflearn.layers.estimator import regression

         tf.reset_default_graph()

         convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')
         convnet = conv_2d(convnet, 32, krl, activation='relu')
         convnet = max_pool_2d(convnet, krl)
         #convnet = dropout(convnet, drp_out_per)

         convnet = conv_2d(convnet, 64, krl, activation='relu')
         convnet = max_pool_2d(convnet, krl)
         #convnet = dropout(convnet, drp_out_per)

         convnet = conv_2d(convnet, 128, krl, activation='relu')
         convnet = max_pool_2d(convnet, krl)
         #convnet = dropout(convnet, drp_out_per)

         convnet = conv_2d(convnet, 256, krl, activation='relu')
         convnet = max_pool_2d(convnet, krl)
         #convnet = dropout(convnet, drp_out_per)

         convnet = conv_2d(convnet, 512, krl, activation='relu')
         convnet = max_pool_2d(convnet, krl)
         #convnet = dropout(convnet, drp_out_per)

         convnet = fully_connected(convnet, 1024, activation='relu')
         convnet = dropout(convnet, drp_out_per)

         convnet = fully_connected(convnet, 2, activation='softmax')
         convnet = regression(convnet, optimizer='adam', learning_rate=LR, loss='categorical_c

         model = tflearn.DNN(convnet, tensorboard_dir='log')
```

```
curses is not supported on this machine (please install/reinstall curses for an optimal experie
```

```
In [24]: !pip install tflearn
```

```
Requirement already satisfied: tflearn in c:\users\mahmo\anaconda3\lib\site-packages (0.3.2)
Requirement already satisfied: Pillow in c:\users\mahmo\anaconda3\lib\site-packages (from tflea
Requirement already satisfied: numpy in c:\users\mahmo\appdata\roaming\python\python36\site-pac
Requirement already satisfied: six in c:\users\mahmo\anaconda3\lib\site-packages (from tflearn


WARNING: You are using pip version 19.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
In [26]: if os.path.exists('{}.meta'.format(MODEL_NAME)):
             model.load(MODEL_NAME)
             print('model loaded!')
```

```
In [27]: train = train_data[:-30]
         test = train_data[-30:]
         X = np.array([i[0] for i in train]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
         Y = [i[1] for i in train]

         test_x = np.array([i[0] for i in test]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
         test_y = [i[1] for i in test]
```

```
In [28]: model.fit({'input': X}, {'targets': Y}, n_epoch=epoc_n, batch_size=bsize ,
                    validation_set=({'input': test_x}, {'targets': test_y}),
                    snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 34  | total loss: 0.69253 | time: 11.346s
| Adam | epoch: 007 | loss: 0.69253 - acc: 0.5324 -- iter: 80/91
Training Step: 35  | total loss: 0.68691 | time: 14.389s
| Adam | epoch: 007 | loss: 0.68691 - acc: 0.5675 | val_loss: 0.71188 - val_acc: 0.4000 -- ite
--
```

```
In [29]: model.predict(test_x)
```

```
Out[29]: array([[0.5432391 , 0.45676085],
                [0.54299414, 0.45700592],
                [0.54130334, 0.45869666],
                [0.54111207, 0.45888793],
                [0.54257685, 0.4574231 ],
                [0.5400276 , 0.45997232],
                [0.5441604 , 0.45583966],
                [0.5436983 , 0.4563017 ],
                [0.54349273, 0.45650724],
                [0.5442844 , 0.45571557],
                [0.5432509 , 0.45674908],
                [0.54375905, 0.45624098],
                [0.5413831 , 0.4586169 ],
                [0.54261434, 0.4573857 ],
                [0.54786164, 0.45213836],
                [0.54534686, 0.45465317],
                [0.54492515, 0.45507488],
                [0.5445456 , 0.4554544 ],
                [0.54251045, 0.45748955],
                [0.54300195, 0.45699805],
                [0.5426757 , 0.45732427],
                [0.54540294, 0.454597  ],
                [0.54463756, 0.45536244],
                [0.543896  , 0.45610398],
                [0.54633313, 0.45366684],
                [0.5421926 , 0.4578074 ],
                [0.54133576, 0.45866424],
                [0.5400458 , 0.45995417],
                [0.54458714, 0.45541286],
                [0.5459029 , 0.45409715]], dtype=float32)
```

```
In [30]: print(test_x[2][1])
```

```
[[0.74509804]
 [0.70980392]
 [0.63921569]
 [0.56862745]
 [0.49019608]
 [0.41960784]
 [0.34901961]
 [0.29803922]
 [0.25490196]
 [0.23921569]
 [0.24705882]
 [0.26666667]
 [0.29019608]
 [0.30196078]
 [0.30196078]
 [0.37254902]
 [0.52156863]
 [0.61176471]
 [0.64313725]
 [0.68627451]
 [0.7372549 ]
 [0.78431373]
 [0.82352941]
 [0.84313725]
 [0.84313725]
 [0.83137255]
 [0.80784314]
 [0.8       ]
 [0.8       ]
 [0.80784314]
 [0.82745098]
 [0.82745098]
 [0.81176471]
 [0.80392157]
 [0.80784314]
 [0.80784314]
 [0.81176471]
 [0.81176471]
 [0.81568627]
 [0.81568627]
 [0.81960784]
 [0.82352941]
 [0.82352941]
```

```
[0.82745098]
[0.82745098]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83137255]
[0.82745098]
[0.82745098]
[0.82745098]
[0.82745098]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82745098]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83137255]
[0.83529412]
[0.83529412]
[0.83529412]
[0.83529412]
[0.83137255]
[0.83137255]
[0.83137255]
[0.82745098]
[0.82745098]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
```

```
[0.82352941]
[0.82352941]
[0.82352941]
[0.82352941]
[0.81960784]
[0.81568627]
[0.81568627]
[0.81568627]
[0.81568627]
[0.81568627]
[0.81568627]
[0.81176471]
[0.81176471]
[0.81176471]
[0.81176471]
[0.81176471]
[0.81176471]
[0.80784314]
[0.80784314]
[0.80784314]
[0.80784314]
[0.81176471]
[0.81176471]
[0.81176471]
[0.81176471]
[0.81176471]
[0.81176471]
[0.80784314]
[0.80784314]
[0.80784314]
[0.80784314]
[0.80784314]
[0.80784314]
[0.80392157]
[0.80392157]
[0.80392157]
[0.8       ]
[0.79607843]
[0.79215686]
[0.79215686]
[0.79215686]
[0.79215686]
[0.79215686]
[0.79607843]
[0.79607843]
[0.79607843]
[0.79607843]
[0.8       ]
```

```
[0.8       ]
[0.8       ]
[0.8       ]
[0.8       ]
[0.79607843]
[0.78823529]
[0.78431373]
[0.78431373]
[0.78431373]
[0.78039216]
[0.78039216]
[0.78039216]
[0.78039216]
[0.78039216]
[0.78039216]
[0.78431373]
[0.78431373]
[0.78431373]
[0.78431373]
[0.78431373]
[0.72156863]
[0.59215686]
[0.45490196]
[0.30588235]
[0.20392157]
[0.15294118]
[0.14509804]
[0.18823529]
[0.20392157]
[0.2       ]
[0.19607843]
[0.18431373]
[0.18823529]
[0.2       ]
[0.20392157]
[0.2       ]
[0.19215686]
[0.18039216]
[0.19215686]
[0.22352941]
[0.24705882]
[0.2627451 ]
[0.2627451 ]
[0.24313725]
[0.22745098]
[0.20784314]
[0.2       ]
[0.20392157]
```

```
[0.21568627]
[0.22745098]
[0.23921569]
[0.24313725]
[0.23137255]
[0.20392157]
[0.18823529]
[0.19215686]
[0.19607843]
[0.20392157]
[0.21176471]
[0.21960784]
[0.22745098]
[0.23529412]
[0.23529412]
[0.23529412]
[0.22745098]
[0.21568627]
[0.20392157]
[0.19215686]
[0.18823529]
[0.18823529]
[0.19607843]
[0.21176471]
[0.21960784]
[0.22745098]
[0.21960784]
[0.2       ]
[0.18039216]
[0.14901961]
[0.1254902 ]
[0.09803922]
[0.07843137]
[0.05490196]
[0.03921569]
[0.02352941]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01960784]
[0.01960784]
[0.01960784]
[0.01960784]
[0.01960784]
```

```
[0.01960784]
[0.01960784]
[0.01960784]
[0.01960784]
[0.01960784]
[0.02352941]
[0.02352941]
[0.01960784]
[0.01960784]
[0.01960784]
[0.01960784]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01568627]
[0.01960784]
[0.01960784]
[0.01960784]
[0.01960784]
[0.02352941]
[0.02352941]
[0.02745098]
[0.03137255]
[0.03529412]
[0.03529412]
[0.03529412]
[0.03529412]
[0.03921569]
[0.03921569]
[0.04313725]
[0.04705882]
[0.04705882]
[0.05098039]
[0.05490196]
[0.05490196]
[0.05490196]
[0.08627451]
[0.14901961]
[0.18823529]
[0.20784314]
[0.25098039]
[0.3254902 ]
[0.38039216]
[0.41568627]
[0.40392157]
[0.34901961]
[0.29803922]
[0.25490196]
```

```
[0.23529412]
[0.23921569]
[0.24313725]
[0.23921569]
[0.24705882]
[0.2627451 ]
[0.2627451 ]
[0.24705882]
[0.25882353]
[0.29803922]
[0.35686275]
[0.43529412]
[0.47058824]
[0.4627451 ]
[0.41568627]
[0.31764706]
[0.26666667]
[0.25882353]
[0.30196078]
[0.39607843]
[0.48627451]
[0.57647059]
[0.62745098]
[0.63529412]
[0.63921569]
[0.63921569]
[0.63529412]
[0.62745098]
[0.62352941]
[0.61960784]
[0.62352941]
[0.62745098]
[0.63137255]
[0.63529412]
[0.63921569]
[0.63529412]
[0.63529412]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
```

```
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.63921569]
[0.64313725]
[0.64705882]
[0.64705882]
[0.64705882]
[0.64705882]
[0.65098039]
[0.65098039]
[0.65098039]
[0.65098039]
[0.65098039]
[0.65490196]
[0.65490196]
[0.65490196]
[0.65490196]
[0.65490196]
[0.65490196]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65490196]
[0.65490196]
[0.65490196]
[0.65490196]
[0.65882353]
[0.65882353]
[0.65882353]
[0.6627451 ]
[0.6627451 ]
[0.6627451 ]
[0.6627451 ]
[0.6627451 ]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
```

```
[0.65882353]
[0.65882353]
[0.65882353]
[0.65882353]
[0.6627451 ]
[0.66666667]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.66666667]
[0.66666667]
[0.66666667]
[0.66666667]
[0.66666667]
[0.66666667]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.6745098 ]
```

```
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.67843137]
[0.67843137]
[0.67843137]
[0.67843137]
[0.67843137]
[0.68235294]
[0.68235294]
[0.68235294]
[0.68235294]
[0.68235294]
[0.67843137]
[0.6745098 ]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.67058824]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.6745098 ]
[0.67843137]
```

```
 [0.67843137]
 [0.67843137]
 [0.67843137]
 [0.67843137]
 [0.67843137]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]
 [0.68235294]]
```

```
In [18]: from PIL import Image
```

```
In [31]: def process_test_data():
             testing_data = []
             for img in tqdm(os.listdir(TEST_DIR)):
                 path = os.path.join(TEST_DIR,img)
                 img_num = img.split('.')[0]
                 img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
                 img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
```

```
                testing_data.append([np.array(img), img_num])

            shuffle(testing_data)
            np.save('test_data.npy', testing_data)
```

```
In [32]: prediction = model.predict(test_x)
         for i in range(len(test_x)):
             print(prediction[i][1])
```

```
0.45676085
0.45700592
0.45869666
0.45888793
0.4574231
0.45997232
0.45583966
0.4563017
0.45650724
0.45571557
0.45674908
0.45624098
0.4586169
0.4573857
0.45213836
0.45465317
0.45507488
0.4554544
0.45748955
0.45699805
0.45732427
0.454597
0.45536244
0.45610398
0.45366684
0.4578074
0.45866424
0.45995417
0.45541286
0.45409715
```

===== TESTING THE MODEL ===== TESTING THE MODEL ===== TESTING THE MODEL ===== TESTING THE MODEL =====

```
In [33]: X_DIR = 'X'
         import numpy as np
         from PIL import Image
         import matplotlib.pyplot as plt
         %matplotlib inline
         def create_X_data(var):
             training_data = []
             label = [0.0,1.0] #'bad'
             img = cv2.imread(var,cv2.IMREAD_GRAYSCALE)
             img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))/255
             training_data.append([np.array(img),np.array(label)])

             shuffle(training_data)
             np.save('train_data.npy', training_data)


             #im = image_path = cv2.imread('dataset\\good\\IMG_9002.JPG') #np.array(Image.open


             return training_data
```

0.1 Testing part of the code

```
In [35]: pic = 'D:\\_ashraf\\igood300\\IMG_9001.JPG'
         X_data = create_X_data(pic)
         print(len(X_data))
         X_1 = []
         X_1 = np.array([i[0] for i in X_data]).reshape(-1,IMG_SIZE,IMG_SIZE,1)
         Y = [i[1] for i in X_data]

         #print(yhat)
```

```
1
```

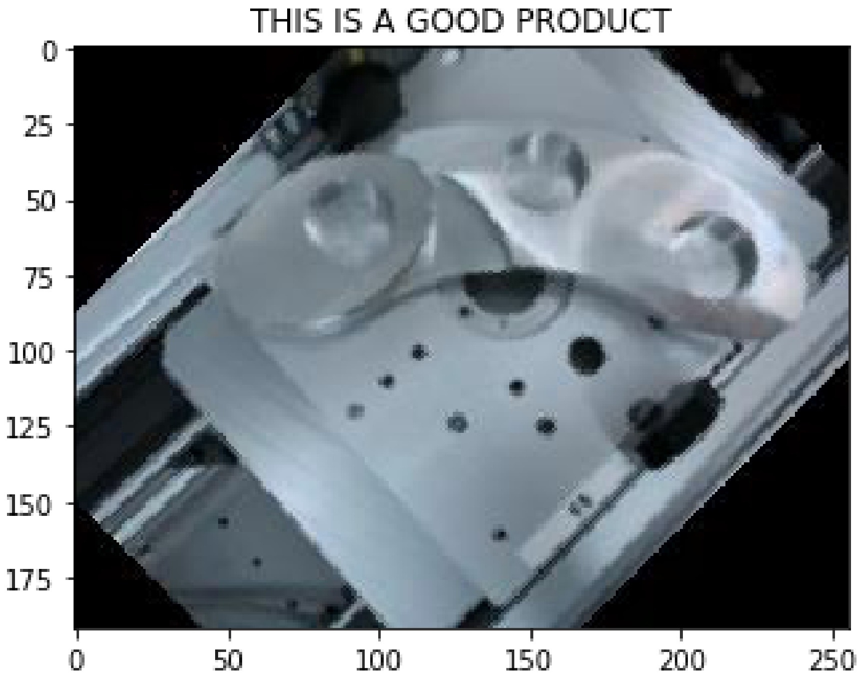
0.2 Reading image from webcamera

```
In [ ]: import cv2
        if cap:
            cap.release()

        cap=cv2.VideoCapture(0)
        if cap.isOpened():
            ret, frame=cap.read()
          # print(ret)
          # print(frame)
        else:
            ret=False
        imge1=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        plt.imshow(imge1)
        plt.title("color image ")
        plt.xticks([])
        plt.yticks([])
        plt.show()

        # save image
        status = cv2.imwrite('testImage.png',imge1)

        #print("Image written to file-system : ",status)
        cap.release()
        #del(cap)
```

```
In [72]: #cam.release()
```

```
In [38]: proba = model.predict(X_1)
         #print(proba)
         #print( proba.argmax(axis=-1))
         idxs = np.argsort(proba)[::-1][:2]
         #print(idxs)
         y = ""
         if idxs[0][1]==1:
             y="This is a Bad Product"
         else:
             y="This is a Good Product"

         image_path = cv2.imread(pic)
         #cv2.imshow('image',image_path)
         #cv2.waitKey(0)
         #cv2.destroyAllWindows()



         #im = image_path = cv2.imread('dataset\\good\\IMG_9002.JPG') #np.array(Image.open(ima

         plt.imshow(image_path)
         plt.title(y.upper())
         plt.show()


         # Sending SMS to a phone number

         import time
         from sinchsms import SinchSMS

         number = '+15193001412'
         #message = message1.upper()
         message = y


         client = SinchSMS('ce00e956-0ec9-47f7-9000-e68a6926a964', 'mN1Udit8NkeRdcpHSOb3mQ==')
```

```
print("Sending '%s' to %s" % (message, number))
response = client.send_message(number, message)
message_id = response['messageId']

response = client.check_status(message_id)
while response['status'] != 'Successful':
    print(response['status'])
    time.sleep(1)
    response = client.check_status(message_id)
    print(response['status'])
```



```
Sending 'This is a Good Product' to +15193001412
```

```
In [37]: !pip install sinchsms
```

```
Collecting sinchsms
  Using cached https://files.pythonhosted.org/packages/36/4c/47099a633d0ec855344962871b85b1f40
Building wheels for collected packages: sinchsms
  Building wheel for sinchsms (setup.py): started
  Building wheel for sinchsms (setup.py): finished with status 'done'
  Stored in directory: C:\Users\mahmo\AppData\Local\pip\Cache\wheels\f2\ab\b2\2fc205820f124ae0
Successfully built sinchsms
Installing collected packages: sinchsms
Successfully installed sinchsms-1.0.4
```

```
WARNING: You are using pip version 19.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.
```

```
In [ ]: import time
        from sinchsms import SinchSMS

        number = '+15193001412'
        #message = message1.upper()
        message = y


        client = SinchSMS('ce00e956-0ec9-47f7-9000-e68a6926a964', 'mN1Udit8NkeRdcpHSOb3mQ==')

        print("Sending '%s' to %s" % (message, number))
        response = client.send_message(number, message)
        message_id = response['messageId']

        response = client.check_status(message_id)
        while response['status'] != 'Successful':
            print(response['status'])
            time.sleep(1)
            response = client.check_status(message_id)
            print(response['status'])
```