Mansib Ahmed
CMPSC 497
11/2/2023

Cookie Defect Detection System Using Deep Learning

Objective: To design, test and develop a deep learning network to classify good crackers versus broken crackers

Materials:

- MATLAB
- Alexnet
- Built-in webcam
- cookies/crackers

Code:

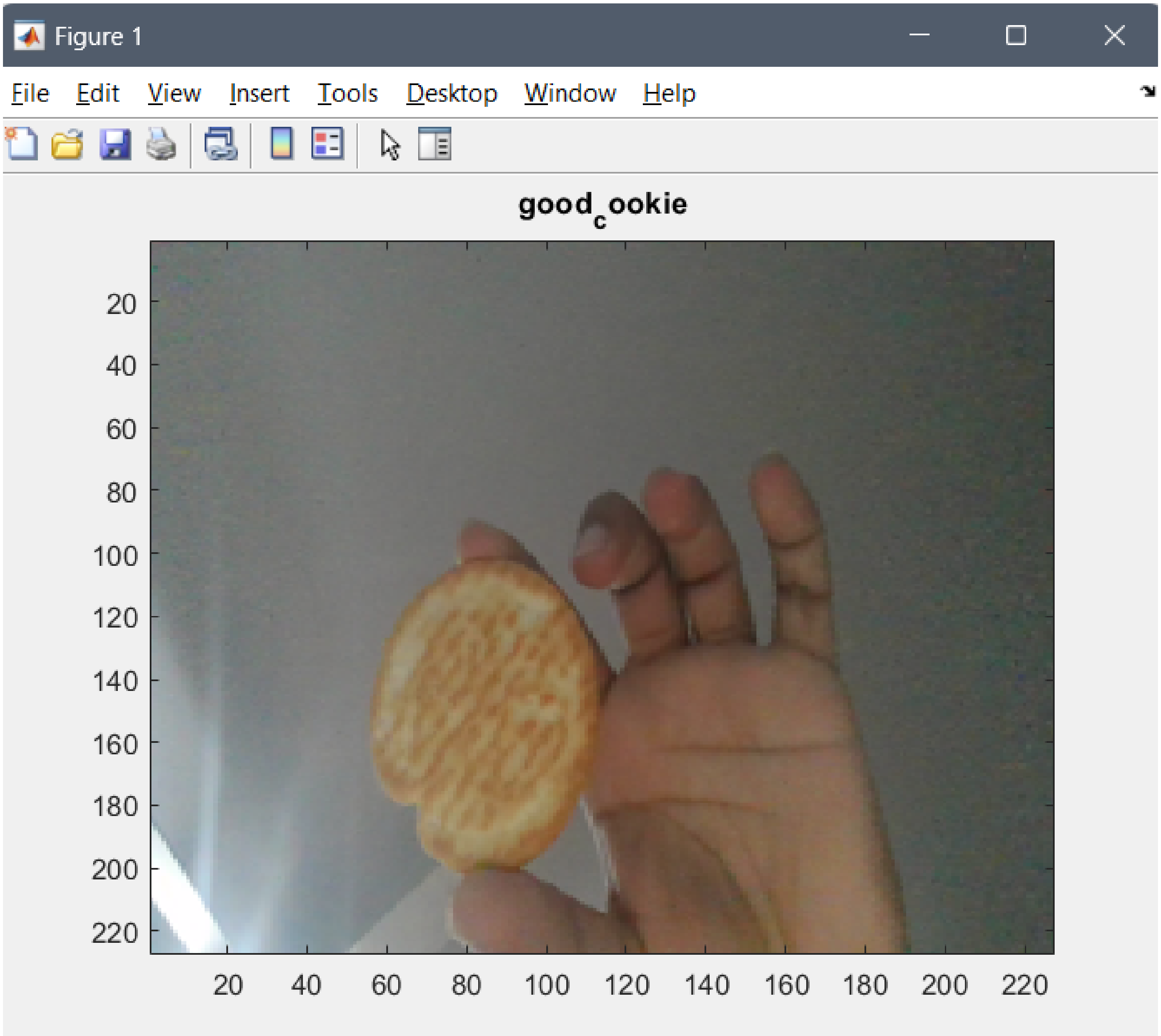
```
training_data = "C:/Users/mansi/OneDrive/Desktop/CMPSC 497/training_data";

good_cookie = "C:/Users/mansi/OneDrive/Desktop/CMPSC 497/training_data/good_cookie";

broken_cookie = "C:/Users/mansi/OneDrive/Desktop/CMPSC 497/training_data/broken_cookie";

allImages = imageDatastore(training_data, 'IncludeSubfolders', true, 'LabelSource', 'foldernames');

% Split data into training (80%)and test (20%) sets

[trainingImages, testImages] = splitEachLabel(allImages, 0.8, 'randomize');

alex = alexnet;

% Review Network Architecture

layers = alex.Layers;

% Modify Pre-trained Network
% AlexNet was trained to recognize 1000 classes, we need to modify it to

% recognize just 2 classes.

layers(23) = fullyConnectedLayer(2); % change this based on # of classes

layers(25) = classificationLayer

% Perform Transfer Learning (can be adjusted)

opts = trainingOptions('sgdm', 'InitialLearnRate', 0.001, ...

'MaxEpochs', 5, 'MiniBatchSize', 10);

% Set custom read function (this code is available in link)

trainingImages.ReadFcn = @readFunctionTrain; % resize

% Train the Network (may take 5 to 15+ minutes)

% Create a new network built on Alexnet w new layers

myNet = trainNetwork(trainingImages, layers, opts);

% Test Network Performance on Test Images

testImages.ReadFcn = @readFunctionTrain; % resize

predictedLabels = classify(myNet, testImages); % test

accuracy = mean(predictedLabels == testImages.Labels)
```

Results:

```
layers =

  25×1 Layer array with layers:

     1   'data'    Image Input                   227×227×3 images with 'zerocenter'
normalization
     2   'conv1'   2-D Convolution               96 11×11×3 convolutions with
stride [4  4] and padding [0  0  0  0]
     3   'relu1'   ReLU                          ReLU
     4   'norm1'   Cross Channel Normalization   cross channel normalization with 5
```

```
channels per element
     5   'pool1'   2-D Max Pooling               3×3 max pooling with stride [2  2]
and padding [0  0  0  0]
     6   'conv2'   2-D Grouped Convolution       2 groups of 128 5×5×48
convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'   ReLU                          ReLU
     8   'norm2'   Cross Channel Normalization   cross channel normalization with 5
channels per element
     9   'pool2'   2-D Max Pooling               3×3 max pooling with stride [2  2]
and padding [0  0  0  0]
    10   'conv3'   2-D Convolution               384 3×3×256 convolutions with
stride [1  1] and padding [1  1  1  1]
    11   'relu3'   ReLU                          ReLU
    12   'conv4'   2-D Grouped Convolution       2 groups of 192 3×3×192
convolutions with stride [1  1] and padding [1  1  1  1]
    13   'relu4'   ReLU                          ReLU
    14   'conv5'   2-D Grouped Convolution       2 groups of 128 3×3×192
convolutions with stride [1  1] and padding [1  1  1  1]
    15   'relu5'   ReLU                          ReLU
    16   'pool5'   2-D Max Pooling               3×3 max pooling with stride [2  2]
and padding [0  0  0  0]
    17   'fc6'     Fully Connected               4096 fully connected layer
    18   'relu6'   ReLU                          ReLU
    19   'drop6'   Dropout                       50% dropout
    20   'fc7'     Fully Connected               4096 fully connected layer
    21   'relu7'   ReLU                          ReLU
    22   'drop7'   Dropout                       50% dropout
    23   ''        Fully Connected               2 fully connected layer
    24   'prob'    Softmax                       softmax
    25   ''        Classification Output         crossentropyex
Training on single CPU.
Initializing input data normalization.
|===================================================================================
======|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base
Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate
|
|===================================================================================
======|
|       1 |           1 |       00:00:02 |       50.00% |       2.0135 |
0.0010 |
|       5 |           5 |       00:00:12 |       90.00% |       0.2555 |
0.0010 |
|===================================================================================
======|
Training finished: Max epochs completed.

accuracy =

    0.7500
```

Success:



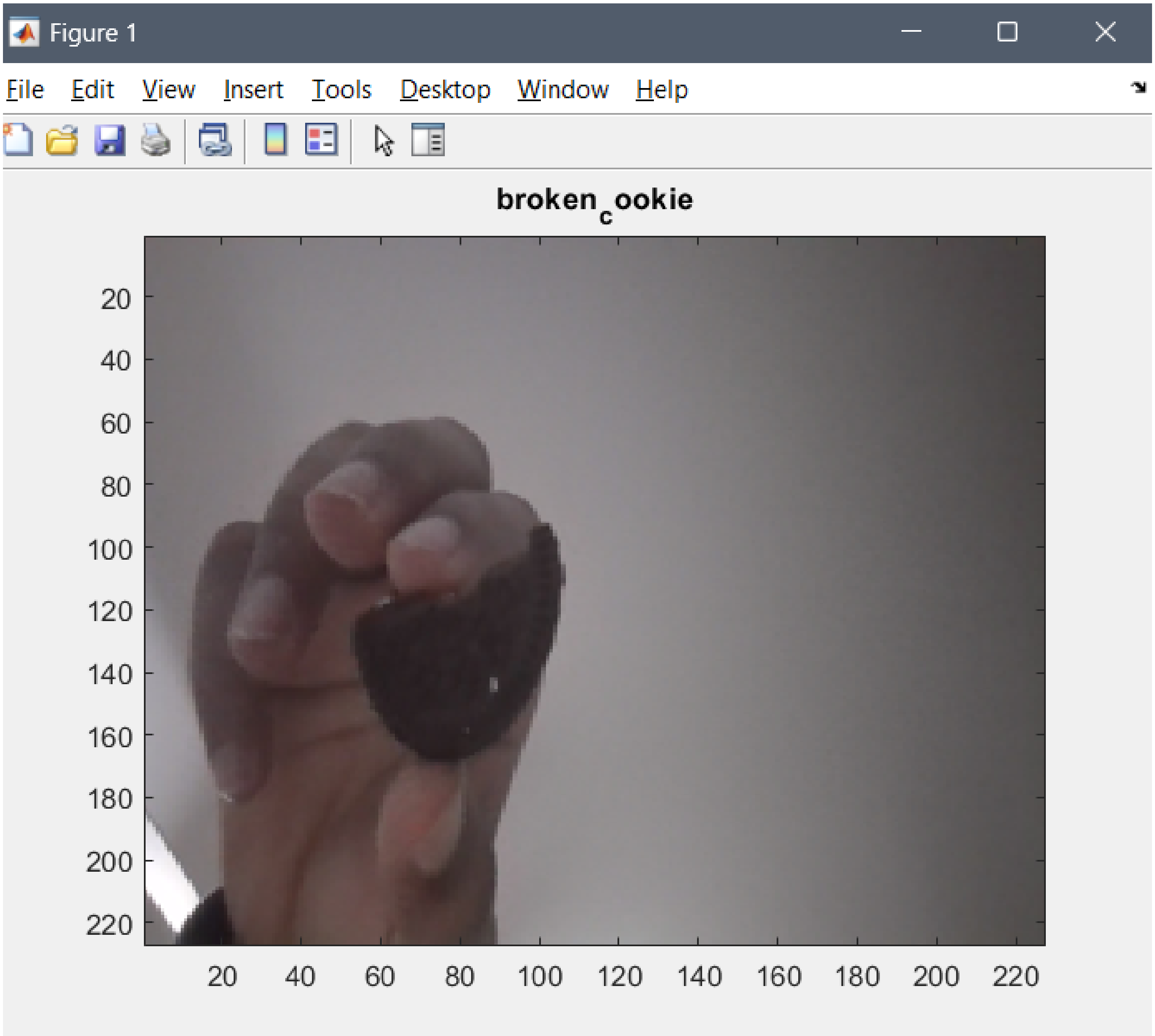


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
broken$_{c}$ookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

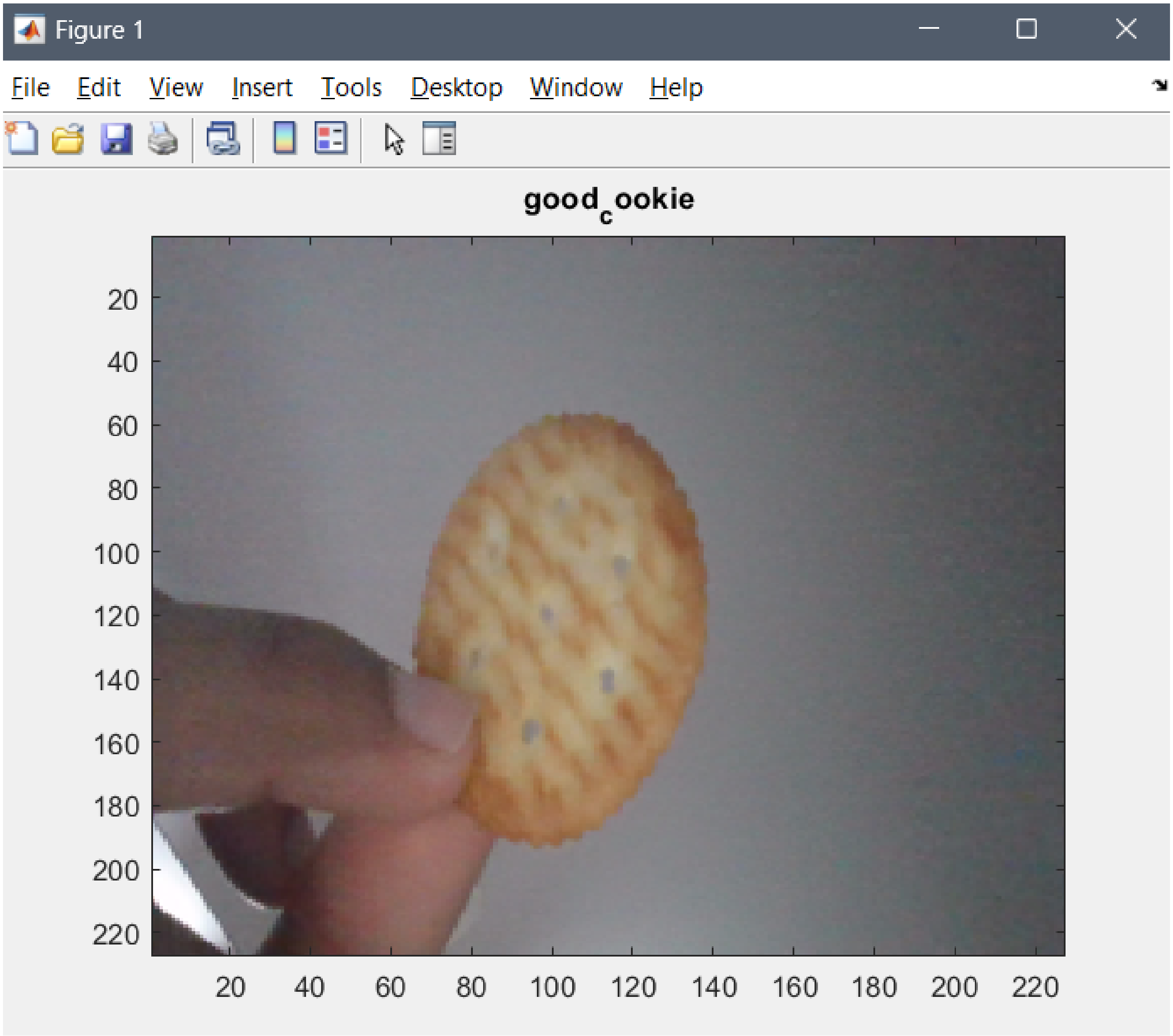
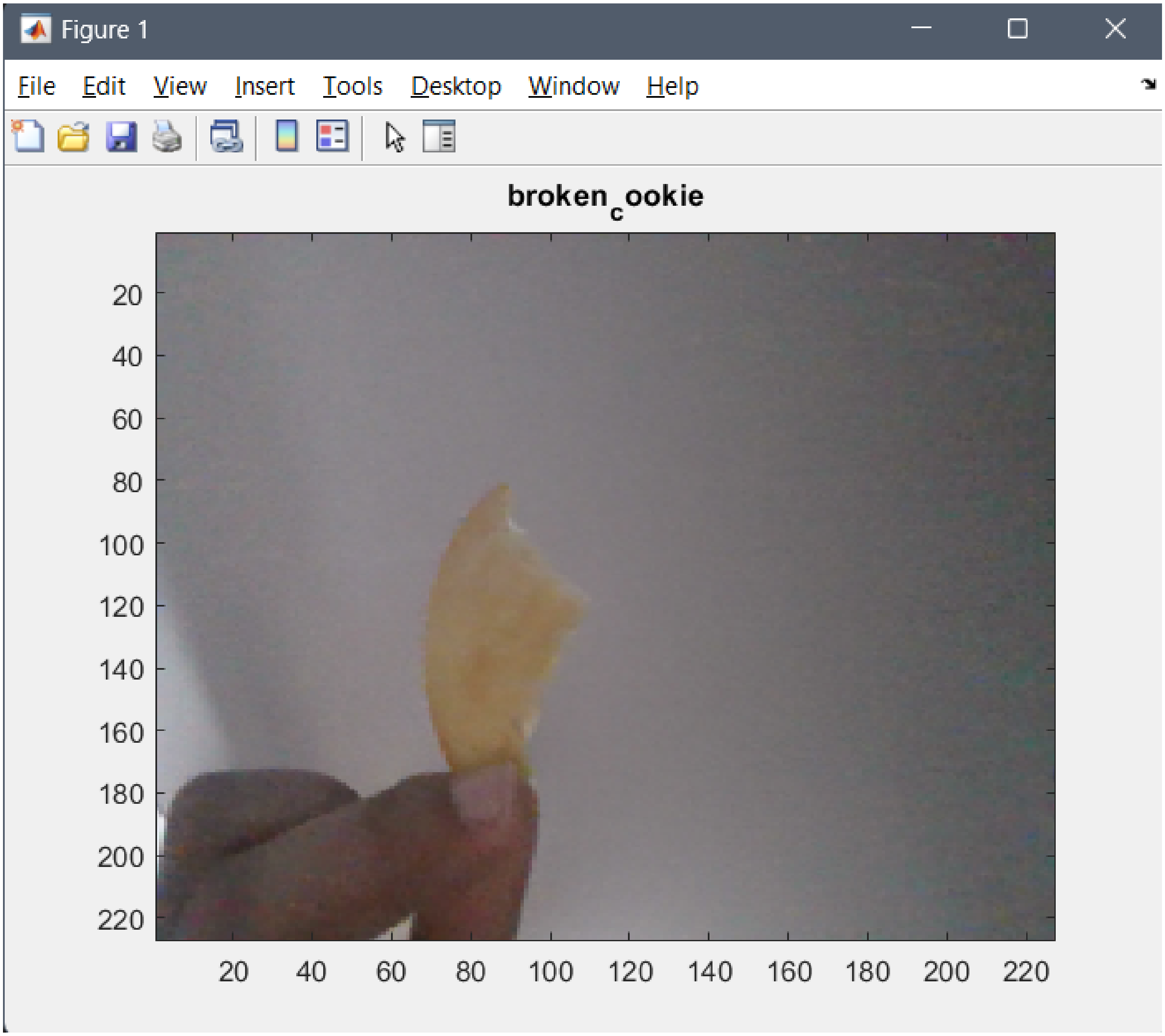


Figure 1
File Edit View Insert Tools Desktop Window Help
good_cookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

Figure 1
File Edit View Insert Tools Desktop Window Help
broken$_c$ookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

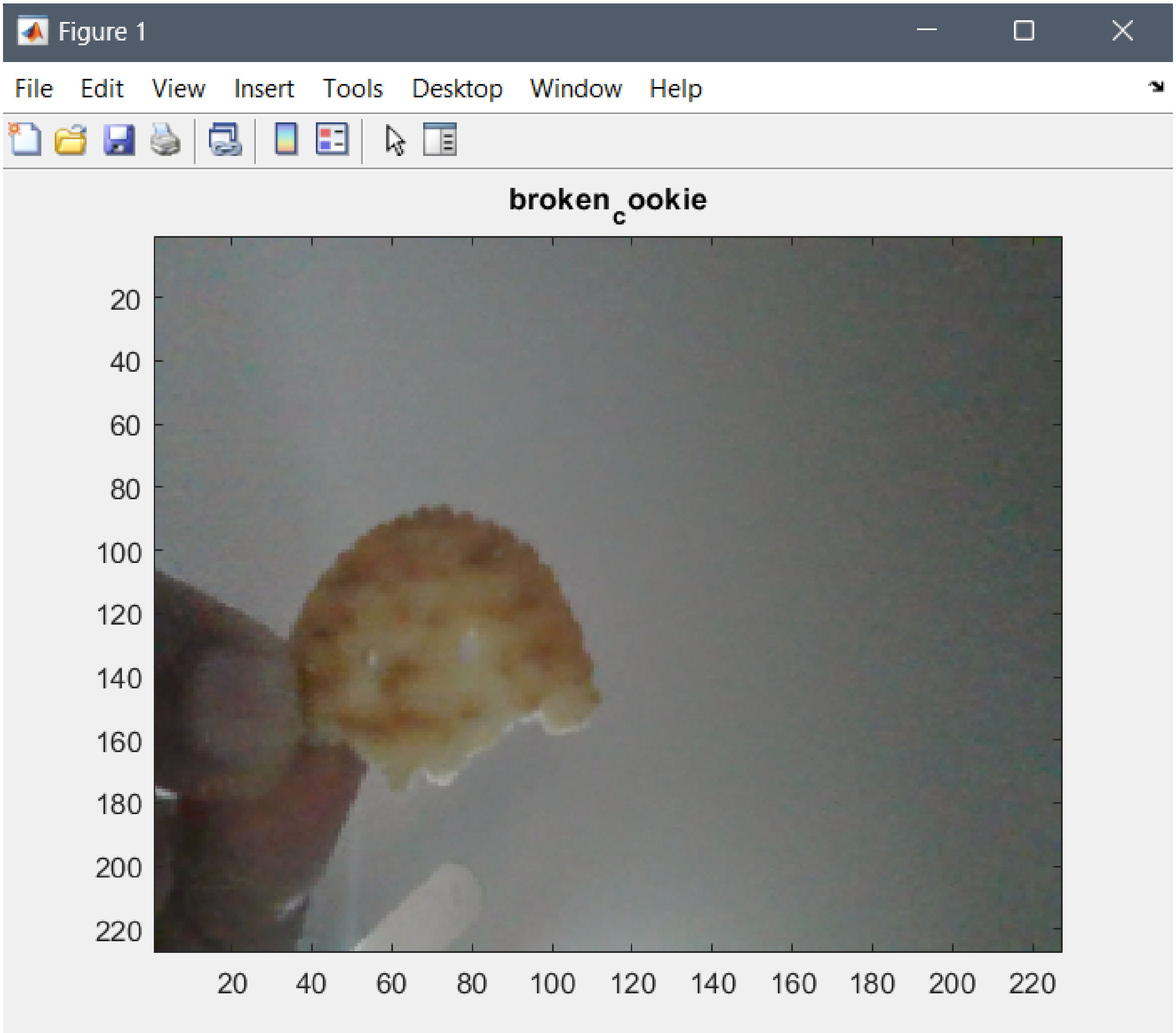


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
broken$_c$ookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

Failure:

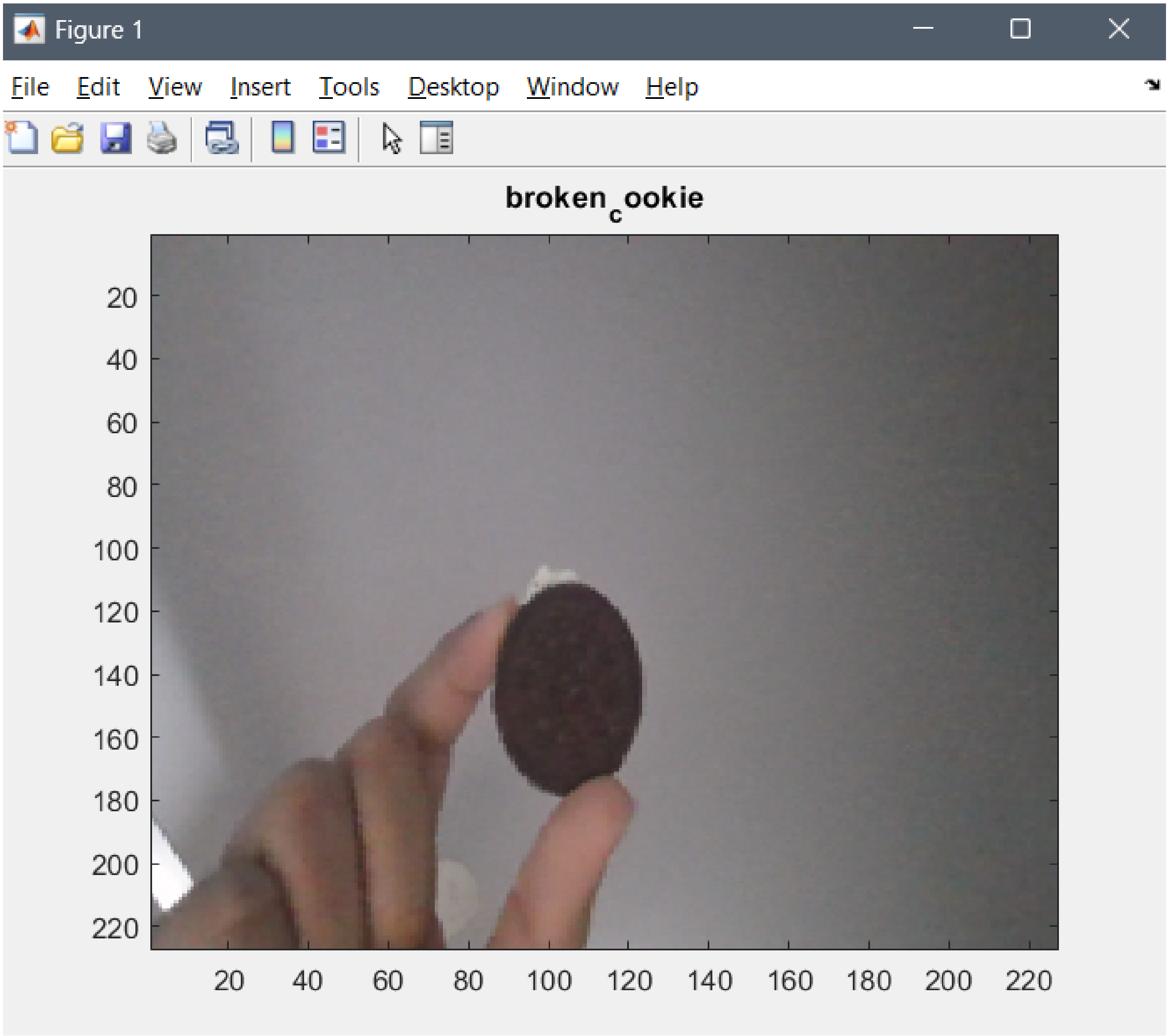


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
broken_cookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

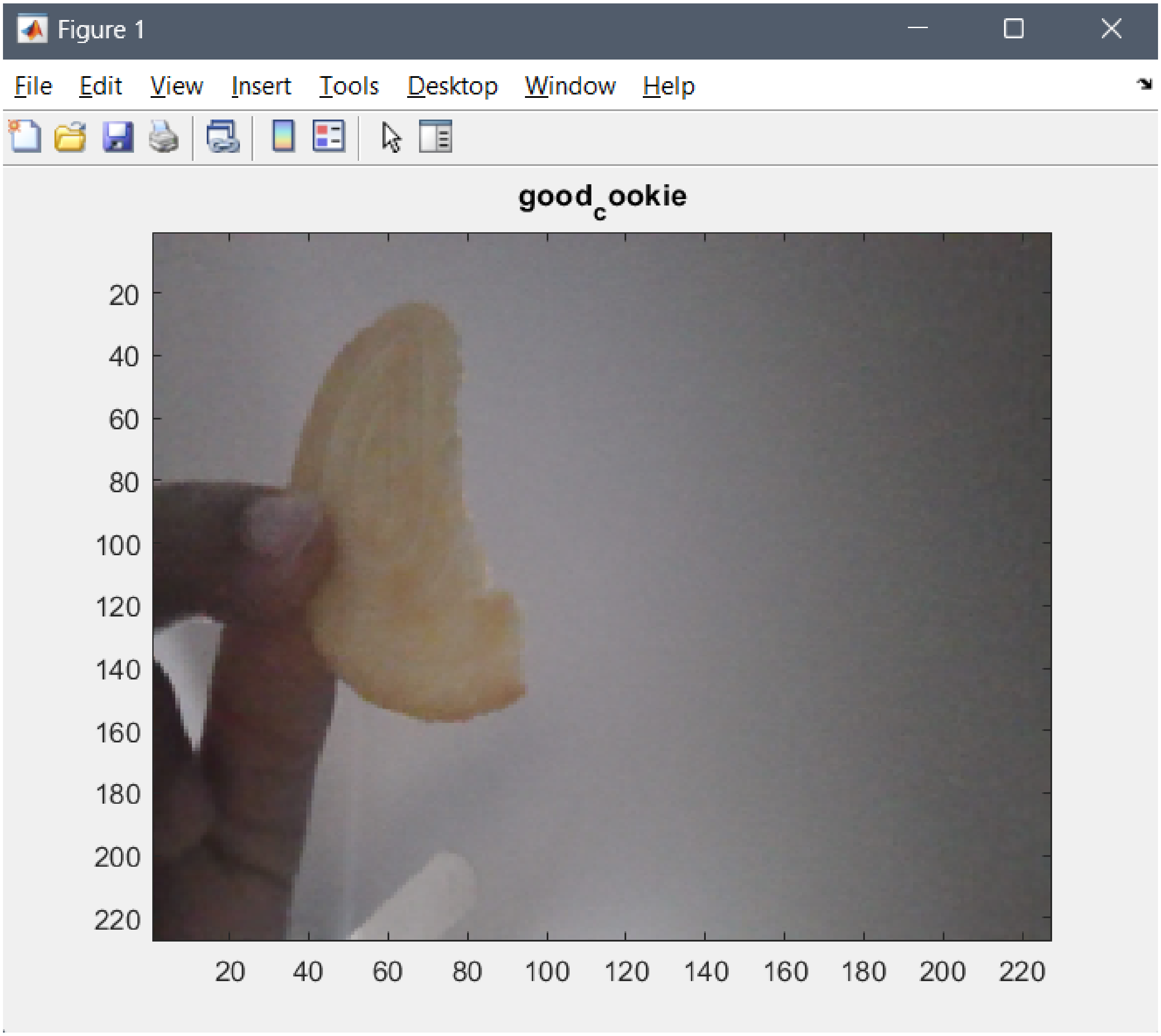


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
good$_c$ookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

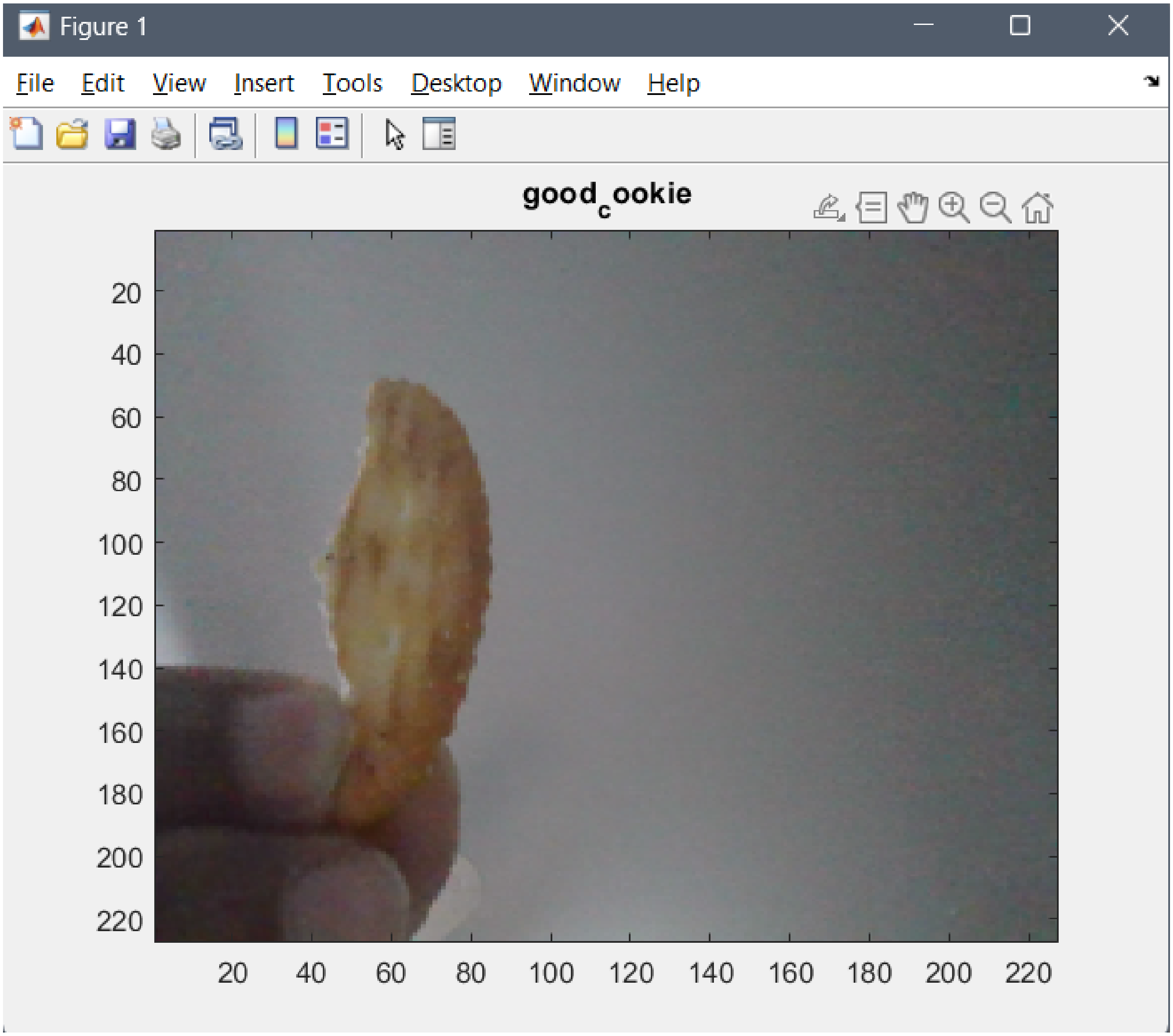


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
good$_{c}$ookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220

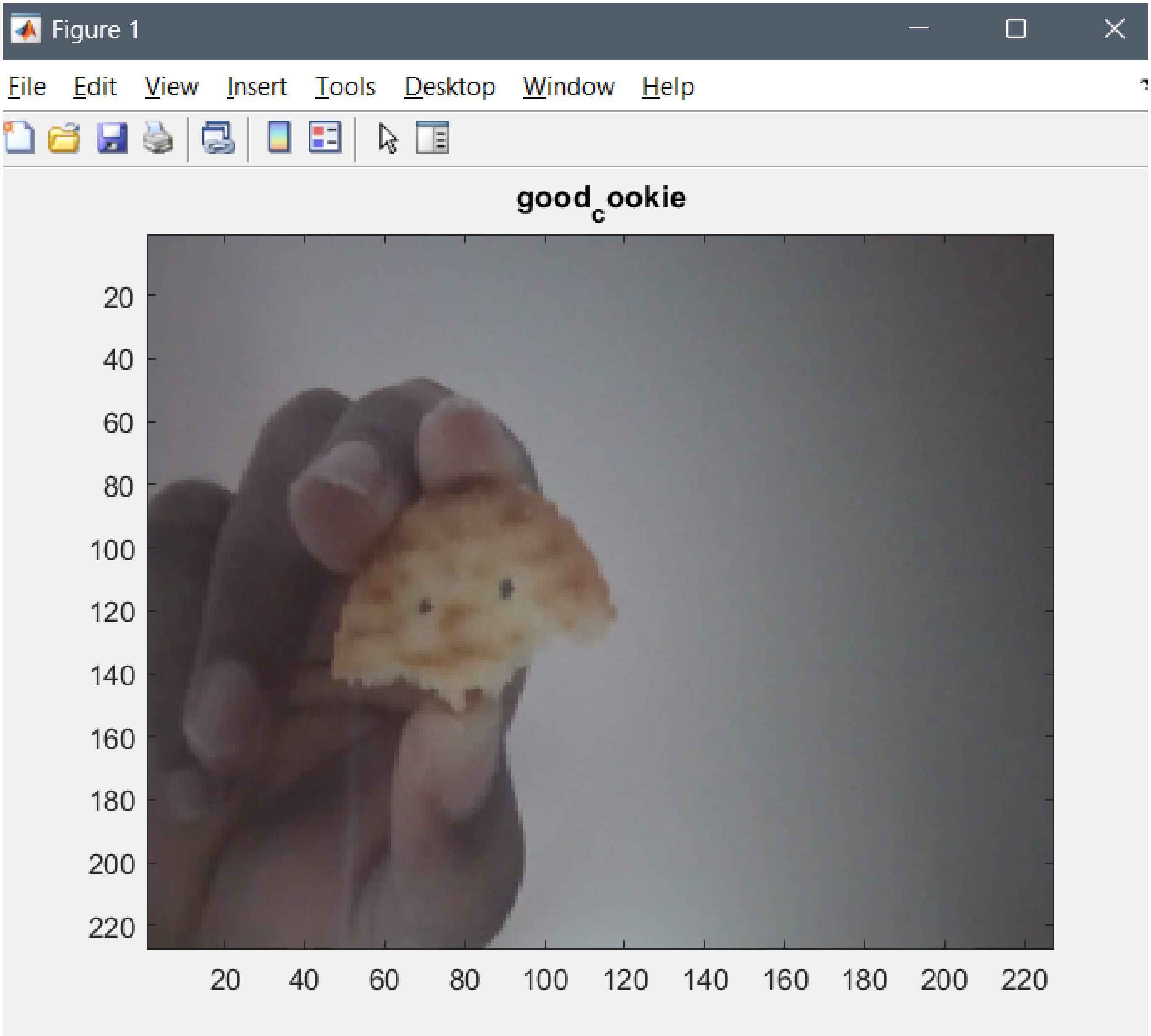
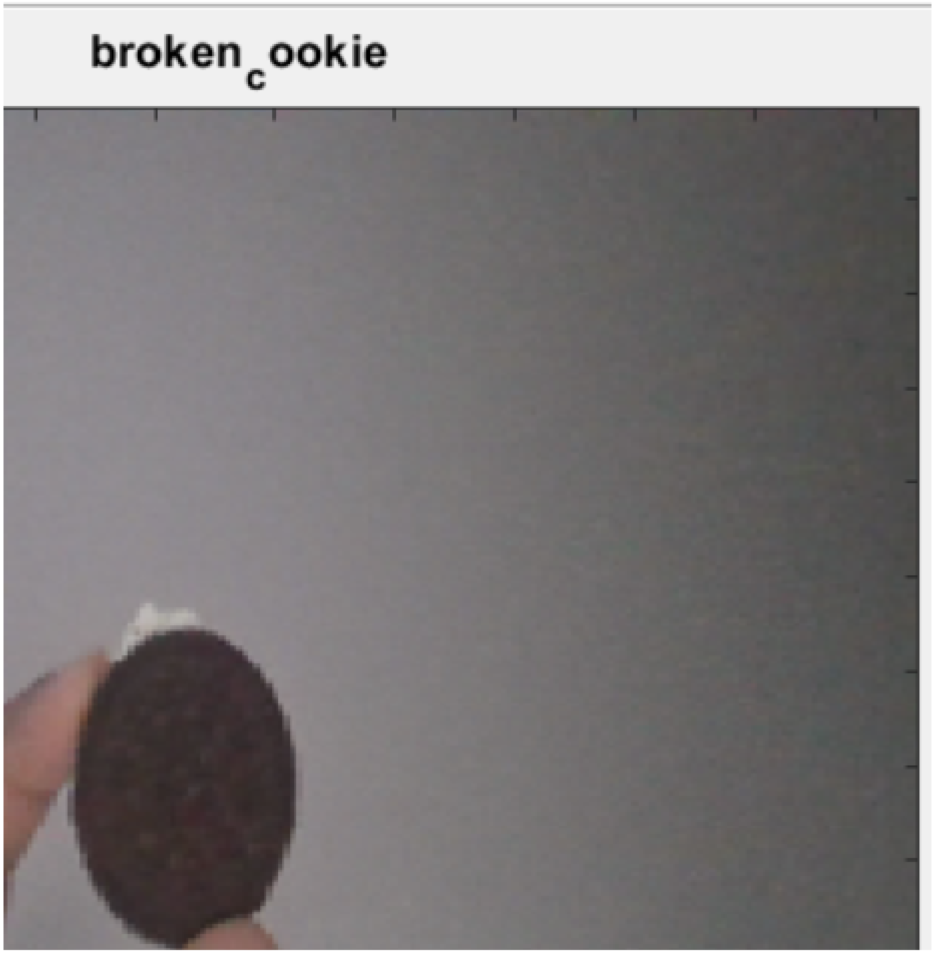


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
good$_c$ookie
20
40
60
80
100
120
140
160
180
200
220
20
40
60
80
100
120
140
160
180
200
220



Conclusion:

Overall, this cookie defect detection system was successful. Transfer learning was used to re-train alexnet's 23rd and 25th layers in order to classify the good cookie versus bad cookie. 20 images of different types of cookies were used as training data for this deep learning network with 10 good and 10 bad cookies. 80% of the data was used to train the network while the remaining 20% was used to test it. Ultimately, the accuracy of the classification of the test data came out to be 75%.