Katherine Banis

CMPSC 497

Lab #9: Cookie/Cracker Defect Detection System

Date: 11/14/2024

**Objective:** Develop a computer vision system using deep learning and transfer learning with AlexNet to classify good crackers and broken crackers.

**Materials:** MATLAB, AlexNet, phone camera, crackers

**Procedure:**

1. Data Collection: 20 photos of good cookies and 20 photos of bad cookies placed in two subfolders “good\_cookies” and “bad\_cookies” in parent folder: “cookies”.
2. Pre-testing with AlexNet: Test 2 “good\_cookies” and 2 “bad\_cookies” images using original AlexNet.
3. Data Preparation: Split dataset into training (80%) and testing (20%) sets.
4. Network Modification: Modify AlexNet’s layers 23 and 25 to accommodate 2 classes.
5. Training: Train modified network with 10 epochs and 10 minibatch size.
6. Real-World Testing: Test trained network on web camera.

**Test Cases:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Good Cookies** | **A chocolate cookie on a white surface  Description automatically generated** | **A chocolate cookie on a white surface  Description automatically generated** | **A chocolate cookie on a white surface  Description automatically generated** | **A chocolate cookie on a white surface  Description automatically generated** | **A chocolate cookie on a white surface  Description automatically generated** |
| **Bad Cookies** | **A cookie with a bite taken out of it  Description automatically generated** | **A cookie with a white frosting on it  Description automatically generated** | **A half eaten cookie on a white surface  Description automatically generated** | **A piece of food on a white surface  Description automatically generated** | **A half eaten chocolate cookie  Description automatically generated** |

**Results:**

|  |  |
| --- | --- |
| **A collage of cookies  Description automatically generated** | A collage of cookies with a bite taken out  Description automatically generated |
| Pre-test before AlexNet modification | Test after AlexNet modification |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Epoch | Iteration | Time Elapsed (hh:mm:ss) | Mini-batch Accuracy | Mini-batch Loss | Base Learning Rate |
| 1  10 | 1  30 | 00:00:02  00:00:33 | 70.00%  80.00% | 1.0209  0.3886 | 0.0010  0.0010 |

Training finished: Max epochs completed.  
Validation Accuracy: 87.50%A screenshot of a graph

Description automatically generated

**Conclusion:**

In conclusion, the results indicates that the modified AlexNet successfully differentiated between good and bad cookies with an accuracy of 87.5%. I had to increase the epoch value from 5 to 10 to get a higher accuracy percentage as my first result was 50%. Pre-testing with the original AlexNet did not work very well as it classified the bad cookies as a screw and a bottlecap, and the good cookies as a trilobite and a puck. Comparing this assignment to the previous cookie defect assignment, I think my results were pretty similar.

**MATLAB Script:**

% Lab #9: Cookie/Cracker Defect Detection System

% Set the directories for your dataset

cookies = '/Users/katherinebanis/Desktop/cookies';

% Load image datastore

allCookies = imageDatastore(cookies, ...

'IncludeSubfolders', true, ...

'LabelSource', 'foldernames', ...

'FileExtensions', {'.jpg', '.jpeg', '.png'}, ...

'ReadFcn', @readAndPreprocessImage);

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

[trainingCookies, testCookies] = splitEachLabel(allCookies, 0.8, 'randomized');

% Load pre-trained AlexNet

alex = alexnet;

% Specify paths for good and bad cookies

good = fullfile(cookies, 'good\_cookies');

bad = fullfile(cookies, 'bad\_cookies');

% Create datastores for good and bad cookies

goodCookies = imageDatastore(good, 'IncludeSubfolders', false, 'LabelSource', 'foldernames');

badCookies = imageDatastore(bad, 'IncludeSubfolders', false, 'LabelSource', 'foldernames');

% Select first 2 images from each folder

numImages = 2;

goodImages = cell(numImages, 1);

badImages = cell(numImages, 1);

% Read the first 2 images from goodCookies datastore

for i = 1:numImages

if hasdata(goodCookies)

goodImages{i} = read(goodCookies);

end

end

% Read the first 2 images from badCookies datastore

for i = 1:numImages

if hasdata(badCookies)

badImages{i} = read(badCookies);

end

end

% Combine images and create labels

testImages = [goodImages; badImages];

trueLabels = ["good\_cookies", "good\_cookies", "bad\_cookies", "bad\_cookies"];

% Pre-test with unmodified AlexNet on the selected images

figure;

for i = 1:length(testImages)

img = imresize(testImages{i}, [227, 227]);

predictedLabel = classify(alex, img);

subplot(2, 2, i);

imshow(img);

title(sprintf('True: %s, Predicted: %s', trueLabels(i), string(predictedLabel)));

end

% Modify AlexNet layers for transfer learning

layers = alex.Layers;

layers(23) = fullyConnectedLayer(2, 'Name', 'fc8'); % 2 classes: good and broken

layers(25) = classificationLayer('Name', 'output');

% Set training options

opts = trainingOptions('sgdm', ...

'InitialLearnRate', 0.001, ...

'MaxEpochs', 10, ...

'MiniBatchSize', 10, ...

'Plots', 'training-progress', ...

'Shuffle', 'every-epoch', ...

'Verbose', true);

% Train the modified network

myNet = trainNetwork(trainingCookies, layers, opts);

% Test the network

predictedLabels = classify(myNet, testCookies);

accuracy = mean(predictedLabels == testCookies.Labels);

% Display the accuracy

fprintf('Validation Accuracy: %.2f%%\n', accuracy \* 100);

% Display sample predictions from the test set

figure;

for i = 1:4

% Read the image file directly

imgFile = testCookies.Files{i};

im = imread(imgFile);

% Resize for display (optional)

im = imresize(im, [227 227]);

% Get the true label and predicted label

trueLabel = testCookies.Labels(i);

predictedLabel = predictedLabels(i);

% Display the image with title

subplot(2, 2, i);

imshow(im);

title(sprintf('True: %s, Predicted: %s', string(trueLabel), string(predictedLabel)));

end

function data = readAndPreprocessImage(filename)

data = imread(filename);

if size(data, 3) == 1

data = cat(3, data, data, data);

end

data = imresize(data, [227 227]);

end