ELE 492: Image Processing
Assoc. Prof. Seniha Esen Yuksel
Department of Electrical and Electronics Engineering
Hacettepe University

HW-4

We pledge that we have not received or given any aid in this homework. All the work presented below is our own work.

Furkan ÇÖZELİ 21427837

Yaşar Oğuzhan TOY 21428465

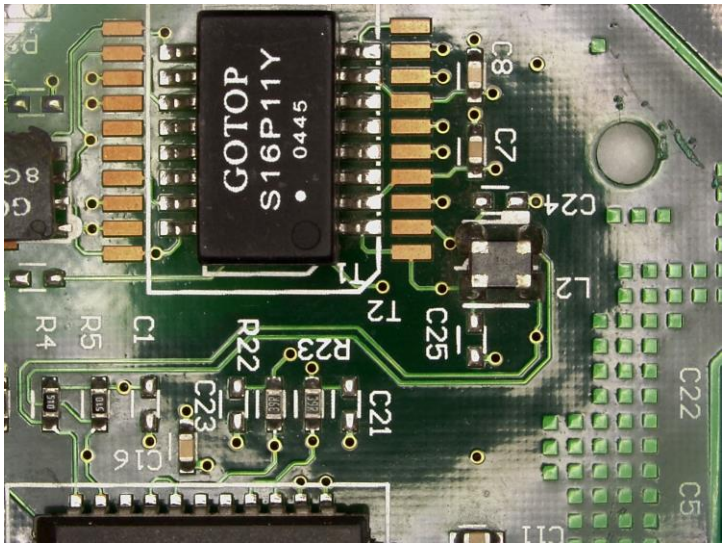
1) Research:

Our homework is to detect the components on the pcb. For this, we need to train the Yolo V3 model. First, we started researching. Then we watched videos about it. As a result of these researches, we learned that we can do it using the Darknet that open source neural network framework. We needed a powerful computer for training. We solved this with google colaboratory.

Now it's time to set the data.

2) Data organization :

We decided to detect the resistors on the pcb. We needed a lot of images for the training. We used all datasets for this. Microscope images were used for training.



We wrote python code to pull all images and information. First, all csv files were accessed

one by one. Csv Files. Contain information such as image name, component name and component location. The pandas library was used to read csv files.

```
>>> data = pd.read_csv("s1_front.csv")
```

As each csv file is found and read, the path of the corresponding image is created. Then the images are saved elsewhere and the txt files we need are created. We created data consisting of 1379 images in total. 1241 images were used for training and 138 images were used for testing. Datasets S11, S14, S15 and S26 were reserved for testing after training.

3-) Darknet:

First, we make GPU and CUDNN adjustments for the fast training, then we make necessary arrangements to use the model with opencv.

Makefile - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
GPU=1
CUDNN=1
CUDNN_HALF=0
OPENCV=1
AVX=0
OPENMP=0
LIBSO=0
ZED_CAMERA=0
ZED_CAMERA_v2_8=0

# set GPU=1 and CUDNN=1 to speedup on GPU
# set CUDNN_HALF=1 to further speedup 3 x times (Mixed-precision on Tensor Cores) GPU: Volta, Xavier, Turing and higher
# set AVX=1 and OPENMP=1 to speedup on CPU (if error occurs then set AVX=0)
# set ZED_CAMERA=1 to enable ZED SDK 3.0 and above
# set ZED_CAMERA_v2_8=1 to enable ZED SDK 2.X
```

Then the we prepare the config file require for training. The batch value is set to 32 and the subdivisions value set to 8. I set my max_batches = 4000, steps = 3200, 3600, I changed the classes = 1 in the three YOLO layers and filters = 18 in the three convolutional layers before the YOLO layers.

yolov3_custom.cfg - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
[net]
# Testing
#batch=1
#subdivisions=1
# Training
batch=32
subdivisions=8
width=416
height=416
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.001
burn_in=1000
max_batches = 4000
policy=steps
steps=3200,3600
scales=.1,.1
```

yolov3_custom.cfg - Not Defteri

Dosya Düzen Biçim Görünüm Yardım

```
[convolutional]
size=1
stride=1
pad=1
filters=21
activation=linear


[yolo]
mask = 3,4,5
anchors = 10,13,  16,30,  33,23,  30,61,
classes=2
num=9
jitter=.3
ignore_thresh = .7
truth_thresh = 1
random=1
```

Everything is ready to train the model.

4-) Train of Dataset:

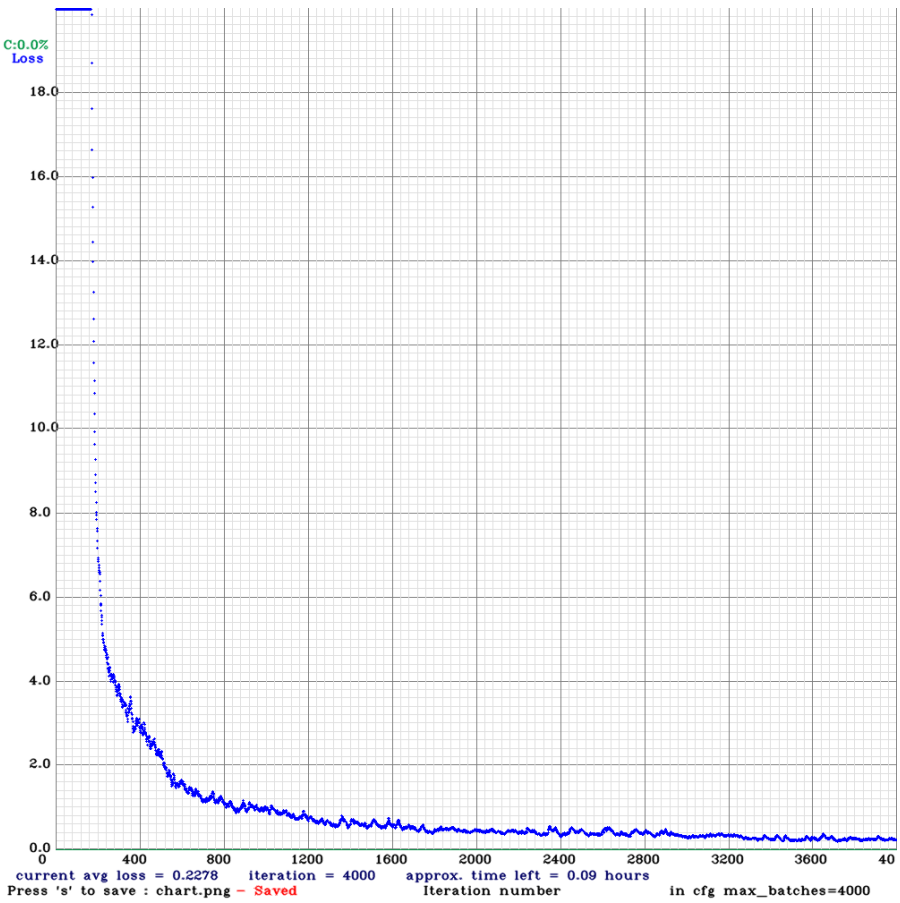
Now we can start our training with the command below.

```
!darknet/darknet detector train images/labelled_data.data darknet/cfg/yolov3_custom.cfg images_weight/darknet53.conv.74 -dont_show
```

New weights saving to our drive to every 100 iterations. Because occur any error or mistake during training , we can lose all the weight. In this way, we do not lose our weight and we can continue to train our model whenever we want. We can use the following command to continue the training.

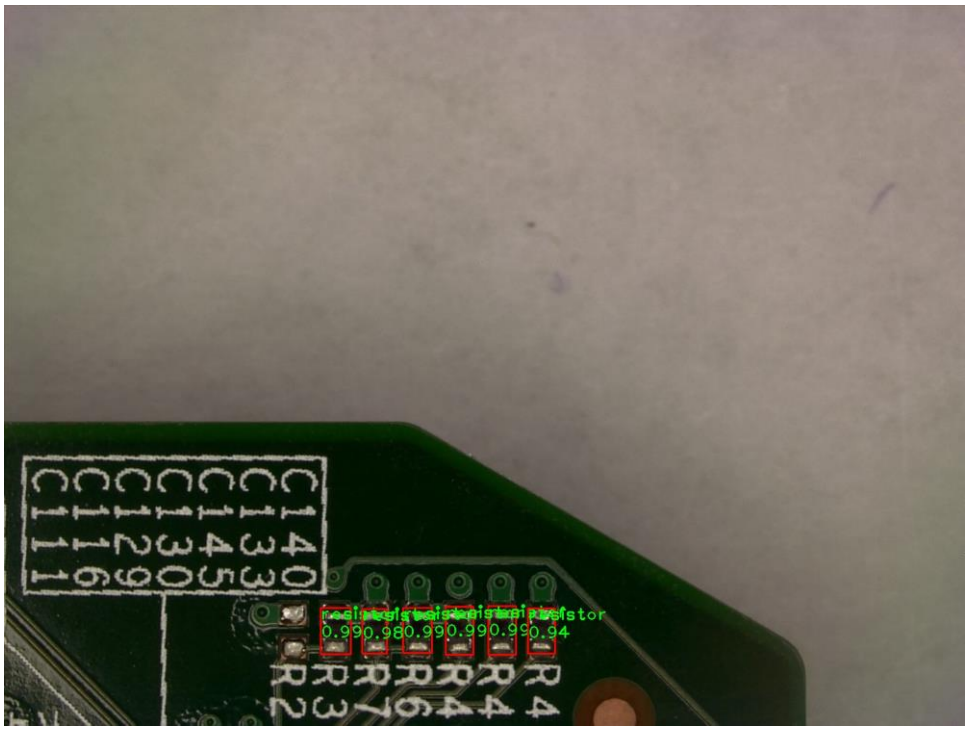
```
!darknet/darknet detector train images/labelled_data.data darknet/cfg/yolov3_custom.cfg backup/yolov3_custom_last.weights -dont_show
```

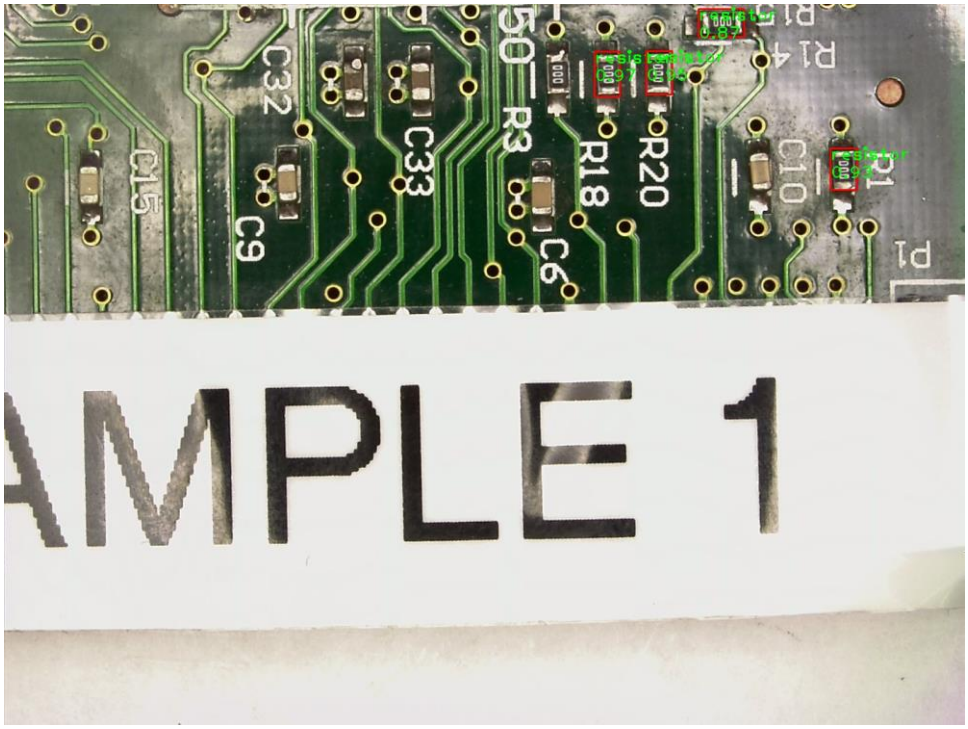
We can track how much loss is during the training. There is a link between this amount of loss and the success of the training. low loss provide high success. we are follows this los with the graph. This graph is updated at the end of every 100 iterations. Easier to follow with graphics.



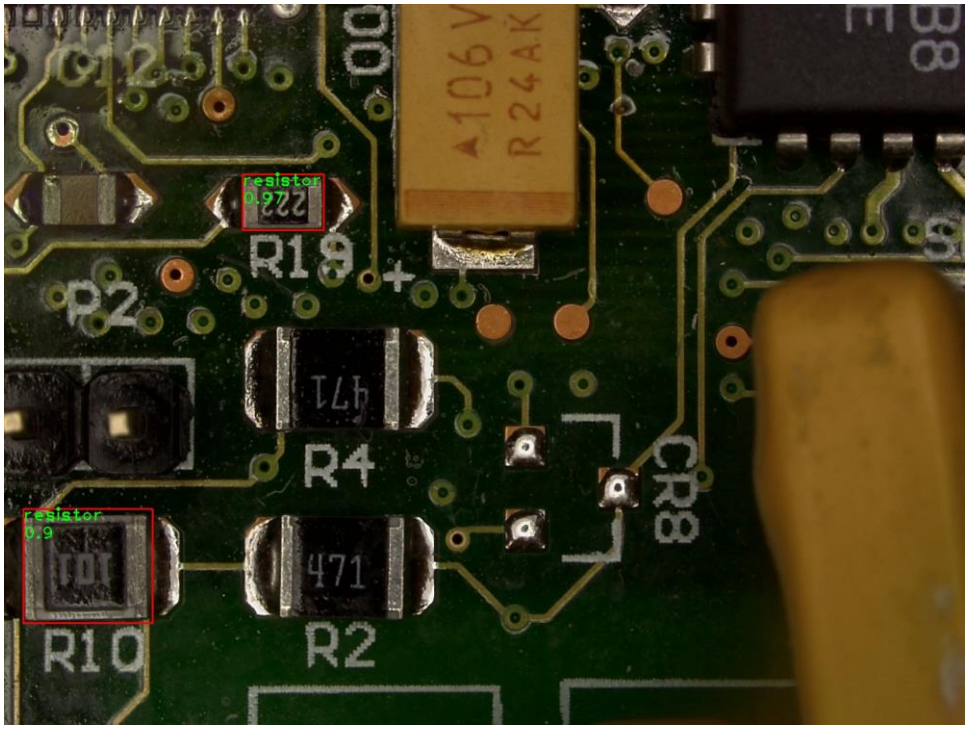
5-) Test

First, it was tested on images selected as tests within the training.



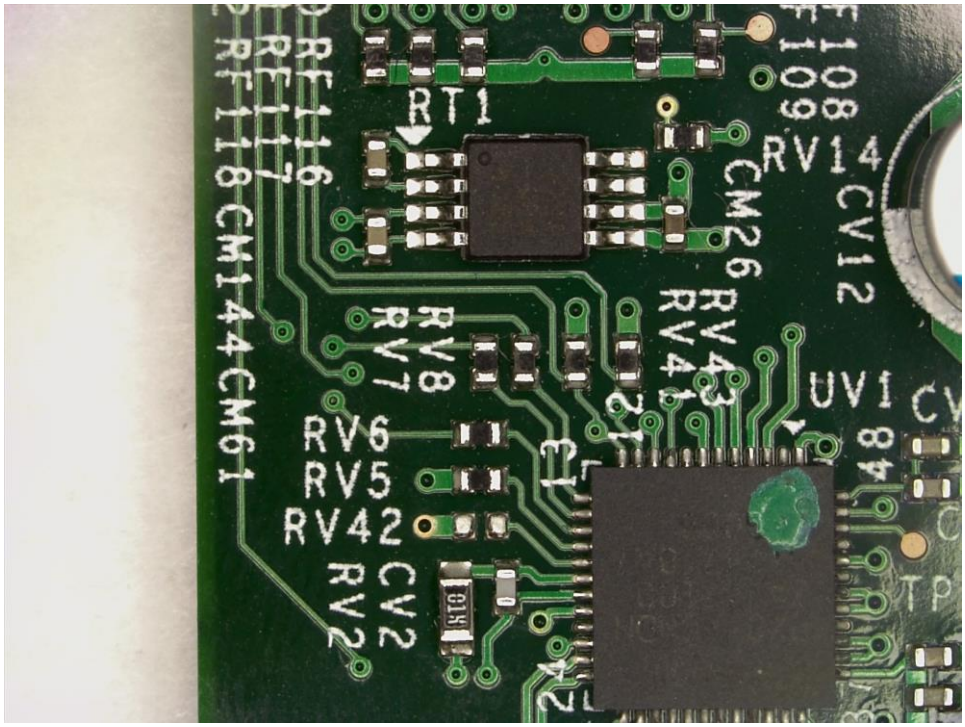


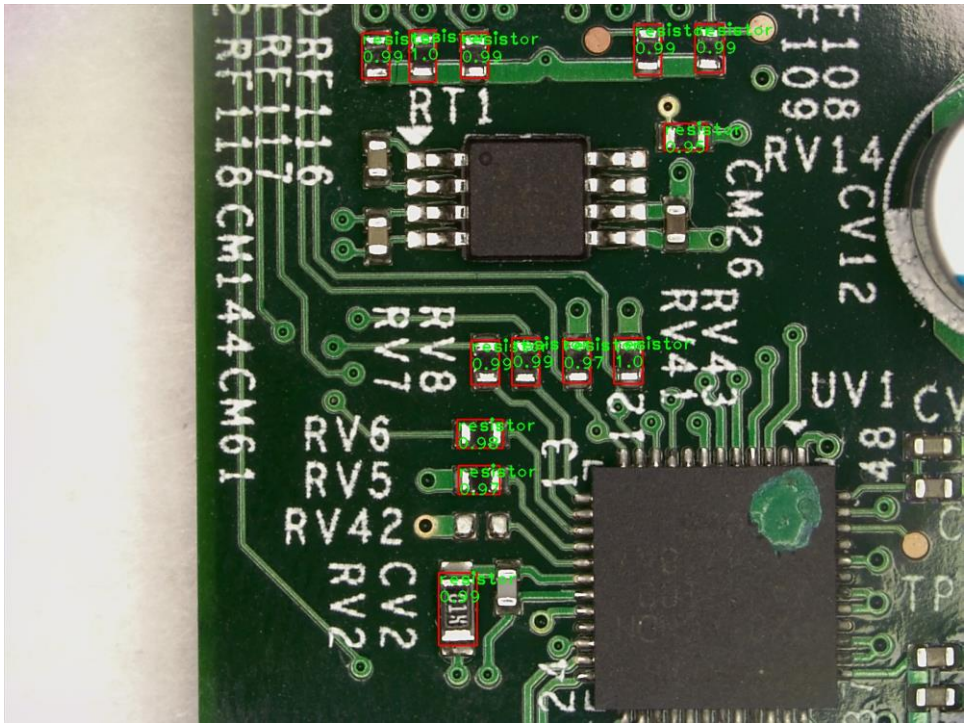
Here it is missing 1 resistor.



In this image, it is missing 2 resistors.

Then we tested it with an image that we chose from the s11 dataset.





It successfully detected all resistors. You can see the information of this pcb in the 1st of the images below. This information was taken from the csv file. In the second image, the location information and size information of the resistors are given according to the result obtained as a result of the test. When these 2 informations are compared, we see that the values are very close to each other.

34	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":782,"y":559,"width":42,"height":78}	resistors
35	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":1155,"y":34,"width":42,"height":78}	resistors
36	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":1053,"y":36,"width":42,"height":78}	resistors
37	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":848,"y":557,"width":42,"height":78}	resistors
38	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":1022,"y":557,"width":42,"height":78}	resistors
39	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":601,"y":48,"width":42,"height":78}	resistors
40	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":677,"y":48,"width":42,"height":78}	resistors
41	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":763,"y":50,"width":42,"height":78}	resistors
42	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":934,"y":559,"width":42,"height":78}	resistors
43	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":756,"y":772,"width":78,"height":44}	resistors
44	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":1098,"y":201,"width":78,"height":44}	resistors
45	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":756,"y":696,"width":78,"height":44}	resistors
46	s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif	{"name":"rect","x":724,"y":955,"width":64,"height":115}	resistors

```
675 44 45 85
1016 557 48 76
1151 33 48 81
778 561 47 74
596 48 46 76
724 949 64 121
846 557 46 79
1050 34 45 84
761 48 46 75
755 693 76 48
931 559 46 73
757 770 69 48
1100 199 71 45

Out[4]: True
```

We did our last test with an image we took. This is a led lamp and it has resistors on it. It successfully detected 4 out of 6 resistors.

Appendix :

Dataset Code:

```
import os
import numpy as np
import cv2
```

```
import pandas as pd
import glob
import re

def getLocation(string):
    """
    This function is for pulling numbers from a string.
    """
    p = '[\d]+[.,\d]+|[\d]*[.][\d]+|[\d]+'
    string = string.replace(',',' ')
    string = string.replace(':',' ')
    location = []
    if re.search(p, string) is not None:
        for catch in re.finditer(p, string):
            location.append(int(catch[0]))
    return location



def listOfName(data):
    """
    This function pulls all image names in data. It takes each name once.
    """
    imageNames = data["image_name"]
    names = []
    for imageName in imageNames:
        if not (imageName in names):
            names.append(imageName)

    return names

def ComponentLocationsForImages(data,names,component):
```

```
    """
    In this function, data about the images containing the selected
    component is obtained.
    """
    components = data["component_type"]
    imageNames = data["image_name"]
    location = data["component_location"]

    finalLocation = []
    for name in names:
        arrayLoca = []
        counter = 0
        for imageName in imageNames:
            if name == imageName and component == components[counter]:
                loca = getLocation(location[counter])
                arrayLoca.append(loca)

            counter += 1
        finalLocation.append(arrayLoca)

    return finalLocation

def listOfImageAndInfo(data,component):
    """
    In this function, the images containing the selected component
    and the array containing the data are obtained.
    """
    names = listOfName(data)
    locations = ComponentLocationsForImages(data,names,component)
    numberOfComponents = []
    for location in locations:
```

```
        numberOfComponents.append(len(location))

    size = len(names)
    i = 0
    finalInfo = []
    while i < size:
        if numberOfComponents[i] != 0:
            info = [names[i],locations[i]]
            finalInfo.append(info)

        i += 1
    return finalInfo

def createPath(folder, i, frontOrBack):
    """
    In this function, the path of the image is edited.
    """
    if '_1x_' in folder:
        x = 1
    elif '_1.5x_' in folder:
        x = 1.5
    elif '_2x_' in folder:
        x = 2
    if 'x_20' in folder:
        y = 20
    elif 'x_40' in folder:
        y = 40
    elif 'x_60' in folder:
        y = 60
    if 'p1_' in folder:
        p = 1
```

```
    elif 'p2_' in folder:
        p = 2
    elif 'p3_' in folder:
        p = 3
    elif 'p4_' in folder:
        p = 4
    elif 'p5_' in folder:
        p = 5

    pathImg = "D:/indir/s{}/Microscope/img/" + frontOrBack +
"/{}x/s{}_p{}_{}x_{}_ring/TileScan_001/".format(i,x,i,p,x,y)

    return pathImg



i = 1
counter = 1
while i < 32:
    path = "D:/indir/s{}/Microscope/annotation".format(i)
    folders = os.listdir(path)
    if i == 11 or i == 14 or i == 15 or i == 26:
        i += 1
        continue
    if "front" in folders:
        pathCSV = "D:/indir/s{}/Microscope/annotation/front".format(i)
        folders2 = glob.glob(pathCSV + "/*.csv")

        for folder in folders2:

            pathImg = createPath(folder, i, "front")
            pcbcsv = pd.read_csv(folder)
```

```
        component = "resistors"
        infos = listOfImageAndInfo(pcbcsv,component)

        j, size = 0, len(infos)
        while j < size:
            pathImgcik = pathImg + infos[j][0]
            img = cv2.imread(pathImgcik)
            row, col, ch = img.shape
            cv2.imwrite("D:/data 4/images/{}.jpg".format(counter), img)
            f = open("D:/data 4/text/{}.txt".format(counter), "x")
            size2, k = len(infos[j][1]), 0
            while k < size2:
                centerX = (infos[j][1][k][0] + infos[j][1][k][2]/2.0)/col
                centerY = (infos[j][1][k][1] + infos[j][1][k][3]/2.0)/row
                rateX = infos[j][1][k][2]/float(col)
                rateY = infos[j][1][k][3]/float(row)
                f.write("0 {} {} {} {}\n".format(centerX, centerY, rateX, rateY))
                k += 1
            f.close()
            j += 1

            counter += 1

    if "back" in folders:
        pathCSV = "D:/indir/s{}/Microscope/annotation/back".format(i)
        folders2 = glob.glob(pathCSV + "/*.csv")

        for folder in folders2:
            pathImg = createPath(folder, i, "back")
            pcbcsv = pd.read_csv(folder)
            component = "resistors"
```

```
        infos = listOfImageAndInfo(pcbcsv,component)

        j, size = 0, len(infos)
        while j < size:
            pathImgcik = pathImg + infos[j][0]
            img = cv2.imread(pathImgcik)
            row, col, ch = img.shape
            cv2.imwrite("D:/data 4/images/{}.jpg".format(counter), img)
            f = open("D:/data 4/text/{}.txt".format(counter), "x")
            size2, k = len(infos[j][1]), 0
            while k < size2:
                centerX = (infos[j][1][k][0] + infos[j][1][k][2]/2.0)/col
                centerY = (infos[j][1][k][1] + infos[j][1][k][3]/2.0)/row
                rateX = infos[j][1][k][2]/float(col)
                rateY = infos[j][1][k][3]/float(row)
                f.write("0 {} {} {} {}\n".format(centerX, centerY, rateX, rateY))
                k += 1
            f.close()
            j += 1

            counter += 1
    i += 1
```

Create train.txt and test.txt

```
i = 1
f = open("D:/data 4/Train YOLO/images/train.txt", "w")
f2 = open("D:/data 4/Train YOLO/images/test.txt", "w")
while i <= 1379:
    if i % 10 == 4:
        f2.write("images/{}.jpg\n".format(i))
```

```
    else:
        f.write("images/{}.jpg\n".format(i))

    i += 1
f.close()
f2.close()
```

Test Code:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import random
net = cv2.dnn.readNetFromDarknet("C:/Users/furkan/Downloads/yolov3_custom (3).cfg","C:/Users/furkan/Downloads/yolov3_custom_final (3).weights")
classes = ['resistor']
img = cv2.imread("D:/indir/s11/Microscope/img/front/1.5x/s11_p1_1.5x_40_ring/TileScan_001/s11_p1_1.5x_40_ring_TileScan_001--Stage03.tif")
height,width,_ = img.shape
height,width,_ = img.shape
blob = cv2.dnn.blobFromImage(img, 1/255,(416,416),(0,0,0),swapRB = True, crop= False)

net.setInput(blob)

output_layers_name = net.getUnconnectedOutLayersNames()

layerOutputs = net.forward(output_layers_name)

boxes =[]
confidences = []
class_ids = []
```

```
for output in layerOutputs:
    for detection in output:
        score = detection[5:]
        class_id = np.argmax(score)
        confidence = score[class_id]
        if confidence > 0.7:
            center_x = int(detection[0] * width)
            center_y = int(detection[1] * height)
            w = int(detection[2] * width)
            h = int(detection[3]* height)
            x = int(center_x - w/2)
            y = int(center_y - h/2)
            boxes.append([x,y,w,h])
            confidences.append((float(confidence)))
            class_ids.append(class_id)


indexes = cv2.dnn.NMSBoxes(boxes,confidences,.8,.4)


font = cv2.FONT_HERSHEY_PLAIN
color = (0,0,255)
if  len(indexes)>0:
    for i in indexes.flatten():
        x,y,w,h = boxes[i]
        label = str(classes[class_ids[i]])
        confidence = str(round(confidences[i],2))

        print(x,y,w,h)
        cv2.rectangle(img,(x,y),(x+w,y+h),color,2)
        cv2.putText(img,label, (x,y+20),font,2,(0,255,0),2)
```

```
        cv2.putText(img,confidence, (x,y+50),font,2,(0,255,0),2)

cv2.imwrite("result.jpg",img)
cv2.imshow('img',img)
cv2.waitKey()
cv2.destroyAllWindows()
```