Machine Learning Model Powered Shoplifting And Crime Detection For Shops, Airports, BookStores

BERAT BERKAY ERKEN
191180758

GÖKAY DİNDAR
181180024

MOTIVATION

In most retail stores nowadays, there are clear signs that state that shoplifters will be prosecuted and that the shop is monitored with cameras.

Yet, despite these anti-theft measures,billions of potential profits are lost each year due to shoplifting. our time. To make shop’s profit high we decided to decrease the number of stolen goods.

Dataset

Kaggle dataset of UCF Crime. The dataset contains images extracted from every video from the UCF Crime Dataset. Every 10th frame is extracted from each full-length video and combined for every video in that class. All the images are of size 64*64 and in .png format.
The dataset has a total of 14 Classes :
1.Abuse 2. Arrest 3. Arson 4. Assault 5. Burglary 6. Explosion 7. Fighting 8. Normal Videos
9.Road Accidents 10.Robbery 11.Shooting
12.Shoplifting 13. Stealing 14. Vandalis

Kaggle dataset of yolo-coco. This is ready to use data with weights and configuration along with coco names to detect objects with YOLO algorithm.
80 names of objects (labels) that can be Detected on the image.

Our Algorithm is Different

Convolutional neural networks are very good at picking up on patterns in the input image, such as lines, gradients, circles, or even eyes and faces. It is this property that makes convolutional neural networks so powerful for computer vision. Unlike earlier computer vision algorithms, convolutional neural networks can operate directly on a raw image and do not need any preprocessing.!

YOLOv3 (You Only Look Once, Version 3) is a real-time object detection algorithm that identifies specific objects in videos, live feeds, or images. The YOLO machine learning algorithm uses features learned by a convolutional neural network to detect an object. Yolo version3 uses 53 convolution layers.

The YOLO algorithm is named “you only look once” because its prediction uses 1×1 convolutions; this means that the size of the prediction map is exactly the size of the feature map before it.

YOLO V3 Uses DARKNET53 Backbone Consist of 53 Convolutional Layers

Backbone	Top-1	Top-5	Ops	BFLOP/s	FPS
Darknet-19	74.1	91.8	7.29	1246	**171**
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Comparison of backbones. Accuracy, billions of operations (Ops), billion floating-point operations per second (BFLOP/s), and frames per second (FPS) for various networks – Source: YOLOv3 Paper

Sample From Our Trained Layers With COCO Dataset

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
1x	Convolutional	32	1 x 1	
	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
2x	Convolutional	64	1 x 1	
	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
8x	Convolutional	128	1 x 1	
	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
8x	Convolutional	256	1 x 1	
	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
4x	Convolutional	512	1 x 1	
	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

```
[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=32
size=1
stride=1
pad=1
activation=leaky
```

Compare Rcnn - YOLO

RCNN :

It works by initially applying the selective search algorithm to find region proposals. The proposals are then extracted and warped to a standard size so they can be processed by a neural network.

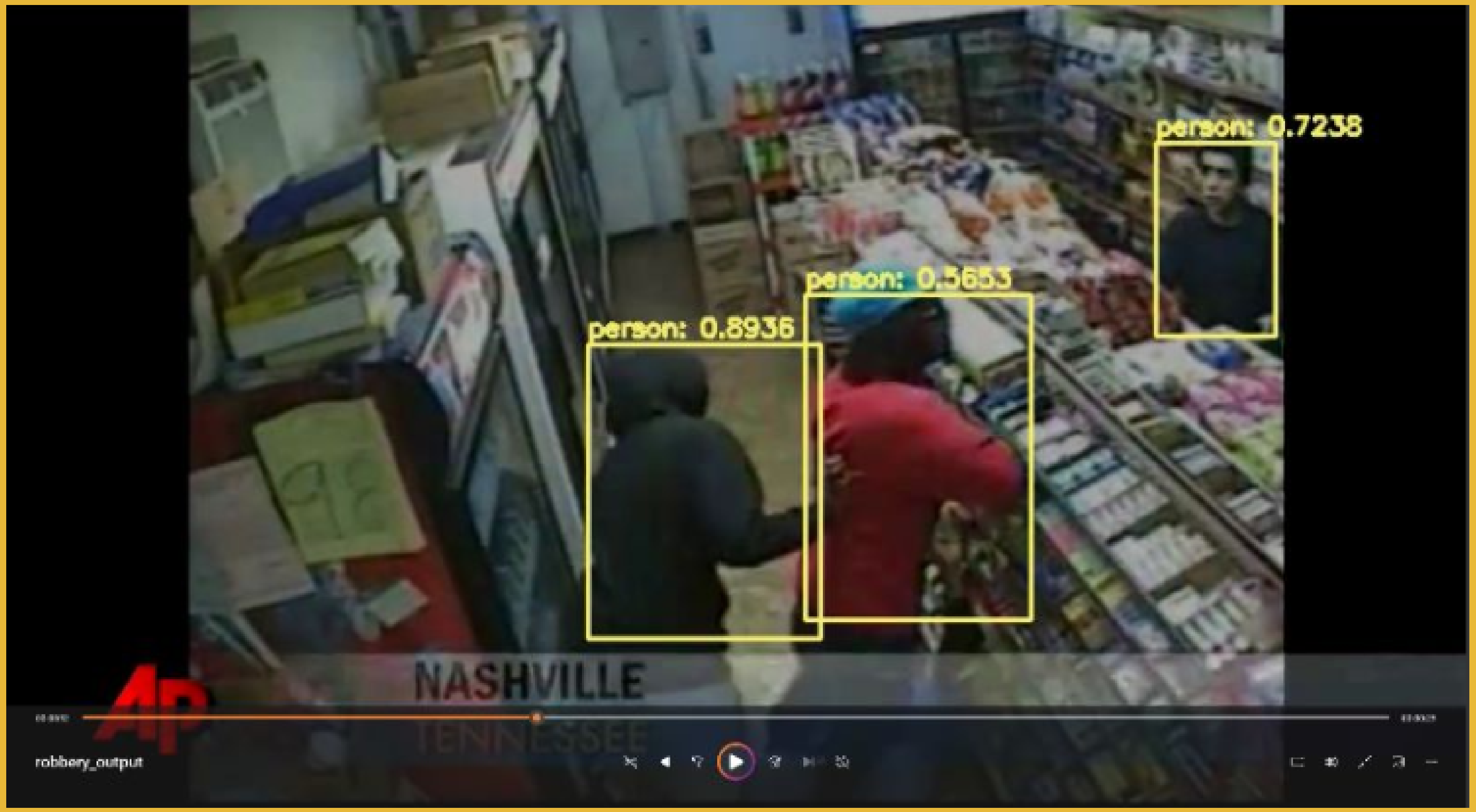
YOLO:

You only look once (YOLO) marks a break with the previous approach of repurposing object classification networks for object detection. Yolo breaks new ground by using a single fully connected layer to predict the locations of objects in an image, essentially requiring only a single iteration to find the objects of interest. This results in a massive acceleration at inference time compared to previous architectures like RNN.

In Progress

Object Detection With Yolo v3

POSE Detection and Body Joints at Second.



ResNet50

Resnet is an acronym for residual neural networks. This model is an improved version of convolutional neural networks (CNN).

Resnet50 is a 50-layer network trained on the ImageNet dataset. ImageNet is an image database of more than 14 million images from more than 20,000 categories created for image recognition competitions. Instead of using 2 (3x3) convolutions, the Resnet model uses convolution layers (1x1), (3x3), (1x1).

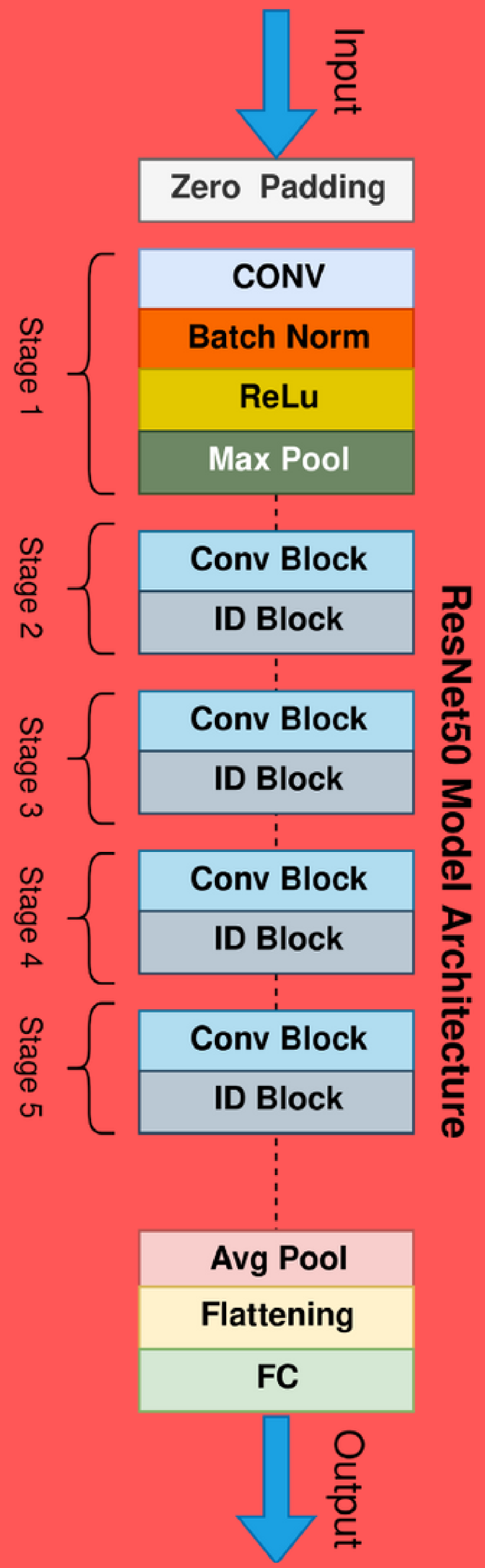
ResNet50

While the Resnet50 architecture is based on the Resnet34 model, there is one major difference. In this case, the building block was modified into a bottleneck design due to concerns over the time taken to train the layers. This used a stack of 3 layers instead of the earlier 2. Therefore, each of the 2-layer blocks in Resnet34 was replaced with a 3-layer bottleneck block, forming the Resnet 50 architecture. This has much higher accuracy than the 34-layer ResNet model. The 50-layer ResNet achieves a performance of 3.8 bn FLOPS.

ResNet50

ResNet50 Comparing

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 64\\3\times3, 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 64\\3\times3, 64\\1\times1, 256\end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 128\\3\times3, 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times4$	$\begin{bmatrix}1\times1, 128\\3\times3, 128\\1\times1, 512\end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 256\\3\times3, 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times6$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times23$	$\begin{bmatrix}1\times1, 256\\3\times3, 256\\1\times1, 1024\end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times2$	$\begin{bmatrix}3\times3, 512\\3\times3, 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$	$\begin{bmatrix}1\times1, 512\\3\times3, 512\\1\times1, 2048\end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



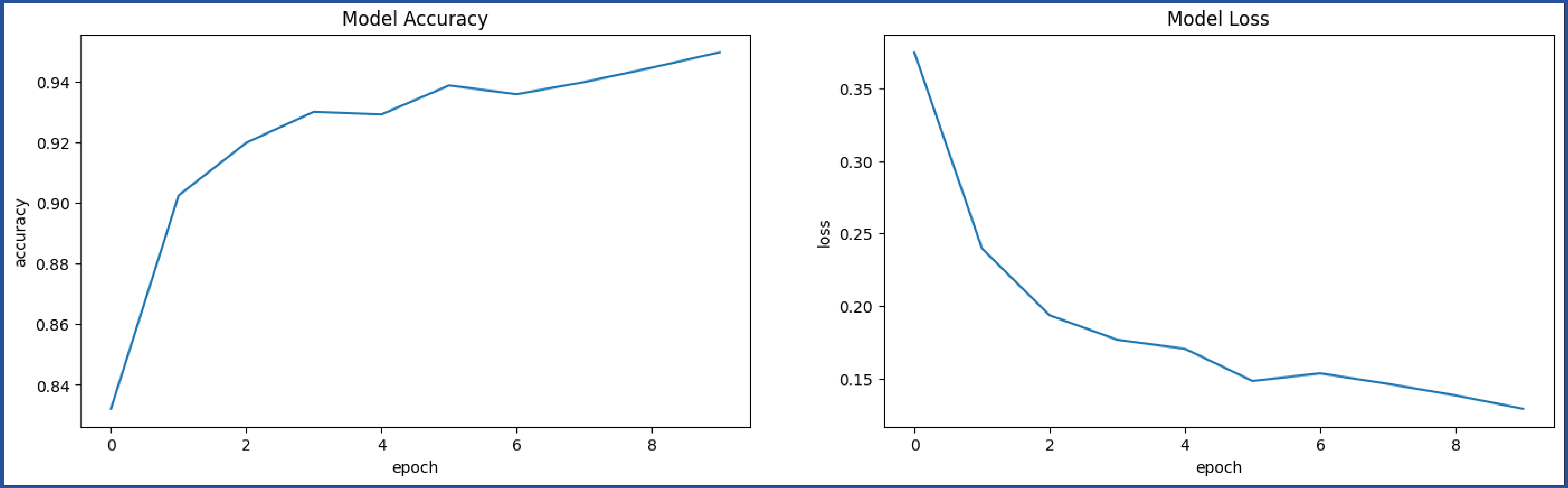
Final

Train Our Model with Resnet50 Weights for 96 minutes!

```
Epoch 1/10
114/114 [==============================] - 642s 6s/step - loss: 0.3751 - accuracy: 0.8321
Epoch 2/10
114/114 [==============================] - 625s 5s/step - loss: 0.2396 - accuracy: 0.9024
Epoch 3/10
114/114 [==============================] - 592s 5s/step - loss: 0.1935 - accuracy: 0.9197
Epoch 4/10
114/114 [==============================] - 569s 5s/step - loss: 0.1766 - accuracy: 0.9299
Epoch 5/10
114/114 [==============================] - 561s 5s/step - loss: 0.1704 - accuracy: 0.9290
Epoch 6/10
114/114 [==============================] - 547s 5s/step - loss: 0.1481 - accuracy: 0.9386
Epoch 7/10
114/114 [==============================] - 563s 5s/step - loss: 0.1534 - accuracy: 0.9357
Epoch 8/10
114/114 [==============================] - 590s 5s/step - loss: 0.1462 - accuracy: 0.9397
Epoch 9/10
114/114 [==============================] - 555s 5s/step - loss: 0.1382 - accuracy: 0.9445
Epoch 10/10
114/114 [==============================] - 526s 5s/step - loss: 0.1289 - accuracy: 0.9496
```

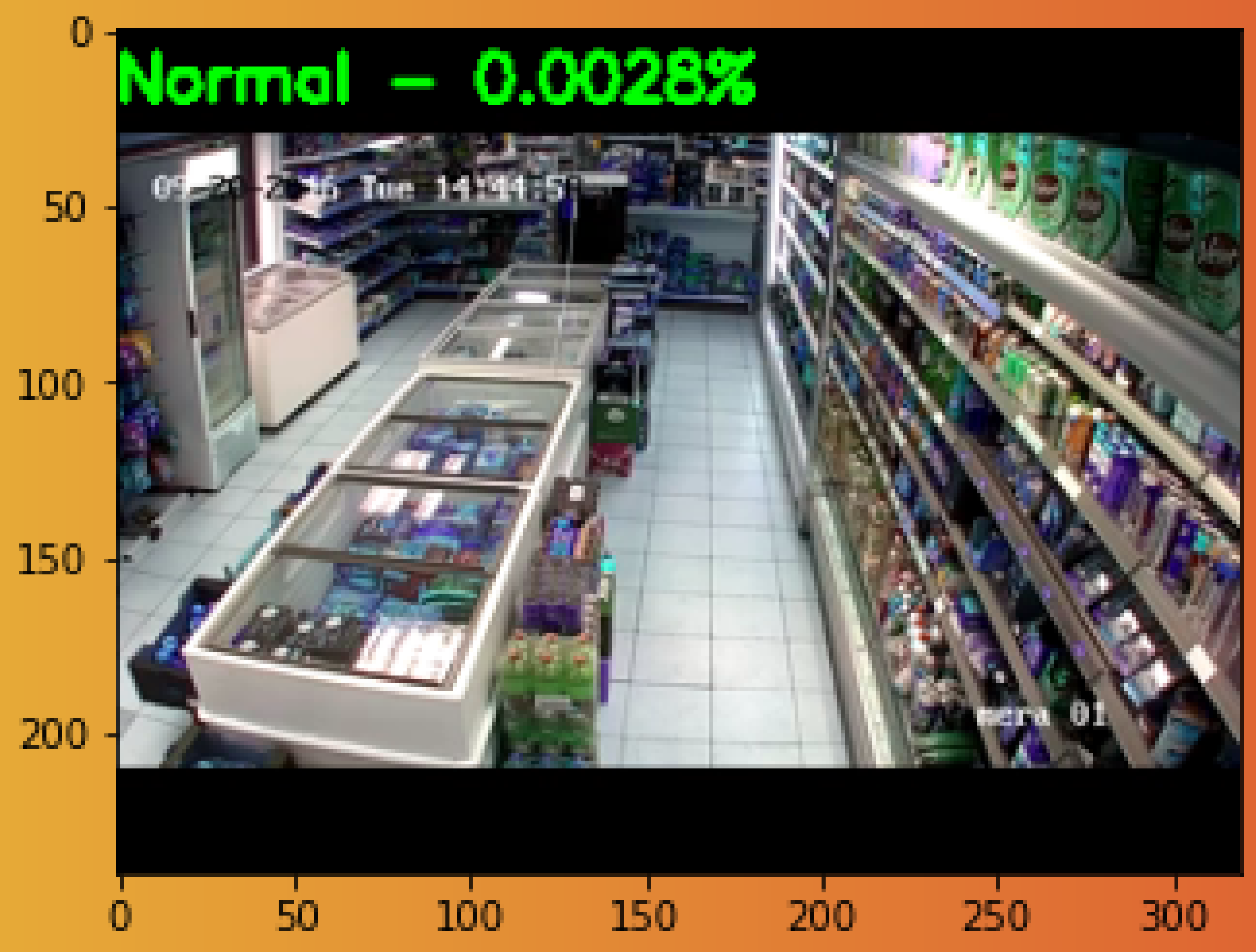
Final

Train Our Model 96 minutes!



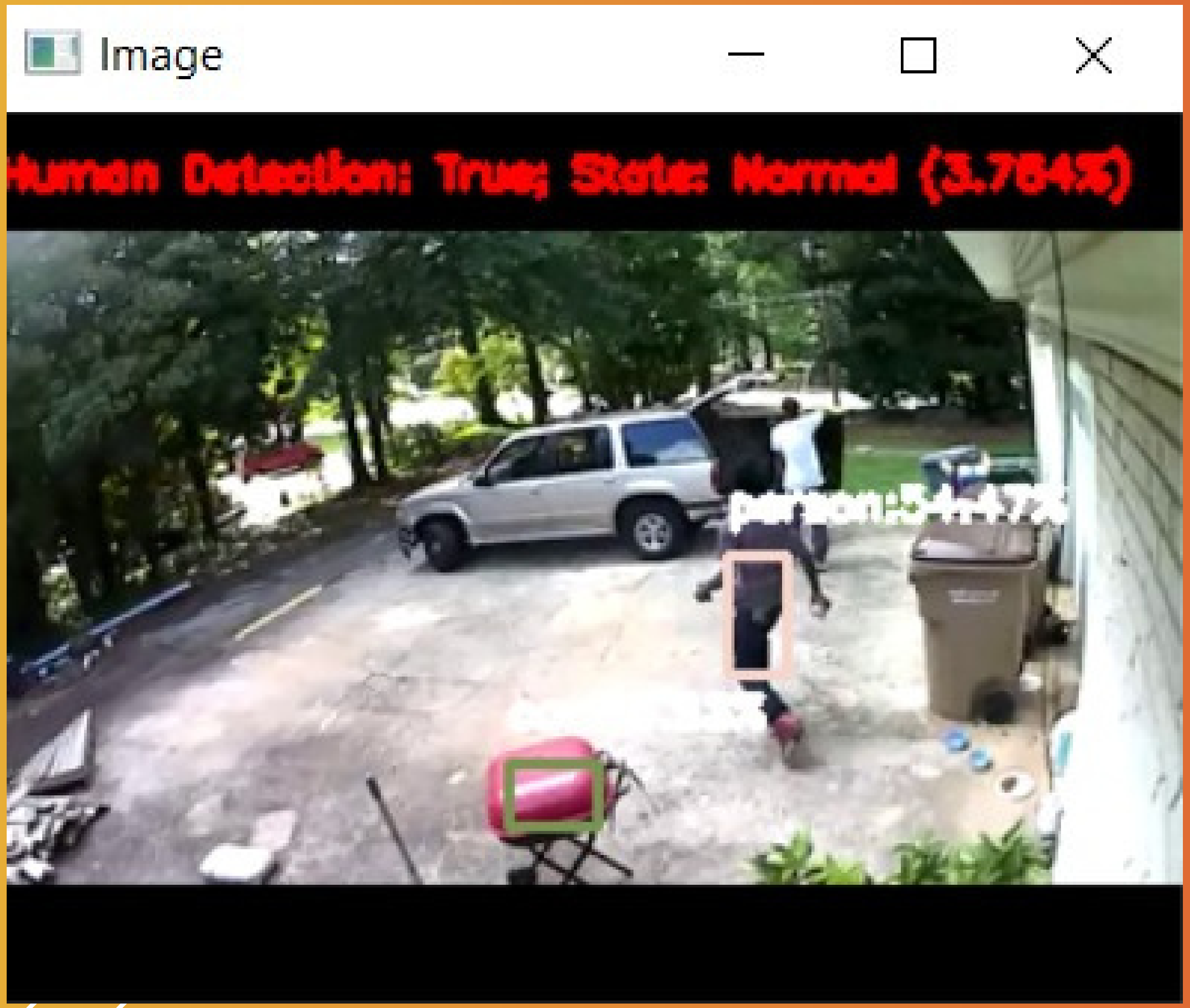
Final

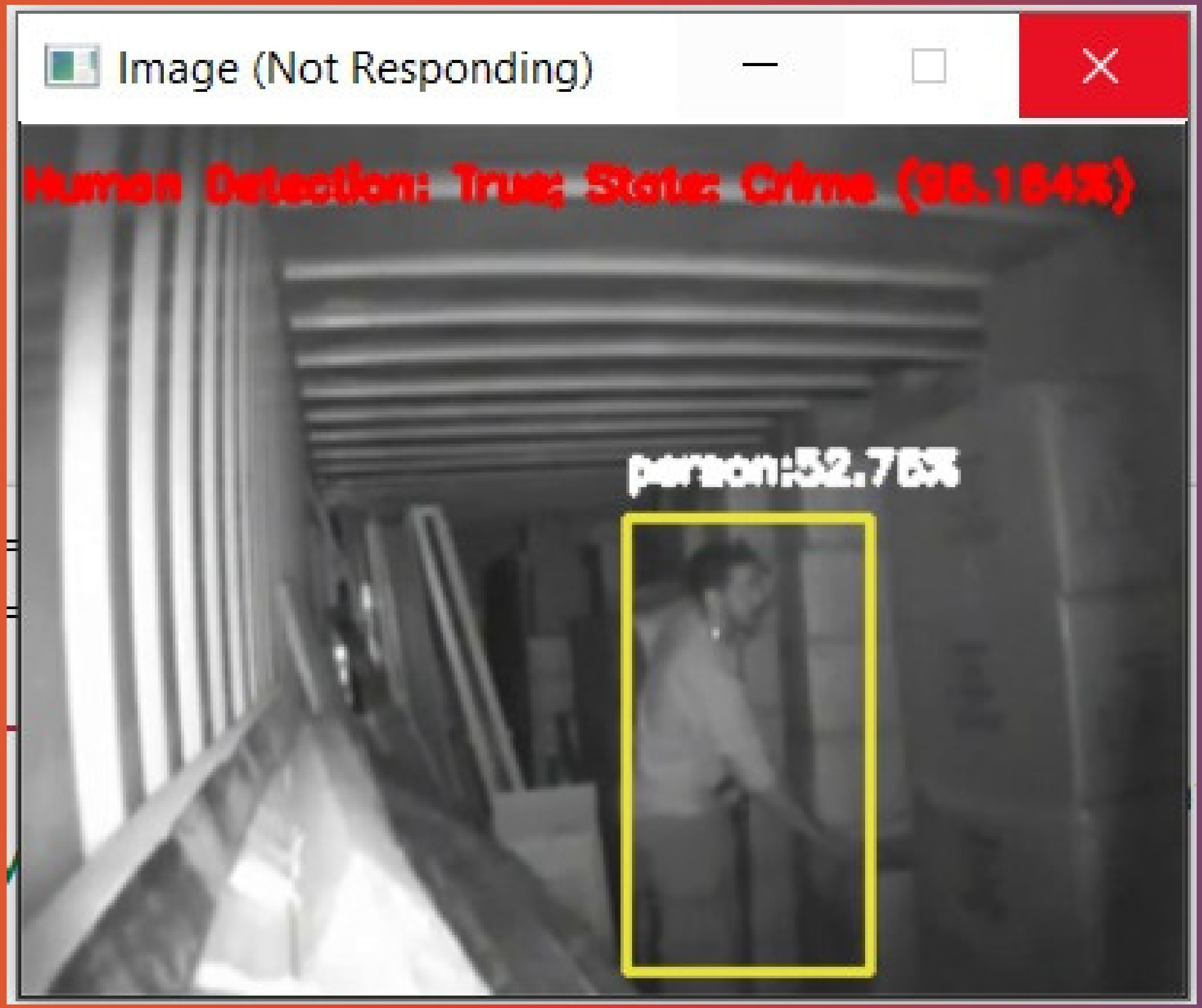
Crime Detection



Final

Crime Detection





File Edit Selection View Go Run Terminal Help YOLO + ResNet-50 Crime Detector.ipynb - Visual Studio Code

YOLO + ResNet-50 Crime Detector.ipynb

C: > Users > berat > Desktop > CENG476FINAL > YOLO + ResNet-50 Crime Detector.ipynb > M↓ YOLO+ResNet-50 based Crime Detection System:

+ Code + Markdown | Interrupt Clear Outputs of All Cells Go To Restart | Variables Outline ··· Python 3.9.1

```python
            if cv2.waitKey(1) & 0xFF==ord('q'):
                break

        else:
            break


    cap.release()
    cv2.destroyAllWindows()


# PART-2. Import the video files
import os
```

[1] 0.2s Python

```python
# PART-3: Import tensorflow and load model
import tensorflow as tf

model = tf.keras.models.load_model(f"C:/Users/berat/Desktop/CENG476FINAL/model/model_x-y-21.h5")
```

[2] 9.8s Python

```python
folder = "C:/Users/berat/Desktop/CENG476FINAL//videos/crime"
video_number=1

makePrediction(

    model,
    color="red",
    buffer_size=90,
    yolo_version=4,
    min_conf=0.1
)
```

[3] 0.3s Python

[ ] Python

SCREEN RECORDER

Search TUR 10:24 6.01.2023