

Trabajo Práctico N1

Reconocimiento de Objetos

Estudiante:

Scala, Tobias 55391

Profesores:

Perfumo, Lucas Alberto

Fortunatti, Nelso Ariel

Materia:

Mecatrónica Aplicada

Facultad:

Instituto Tecnológico de Buenos Aires, ITBA

Introducción

El reconocimiento de objetos es una colección de tareas relacionadas con la visión por computadora que involucran la identificación de objetos en fotografías digitales. Se trata de identificar y rastrear objetos presentes en imágenes y videos. La detección de objetos tiene múltiples aplicaciones como detección de rostros, detección de vehículos, conteo de peatones, autos autónomos, sistemas de seguridad, etc.

Para poder realizar un programa que realice dicha tarea, se ha tenido que instalar los paquetes requeridos vistos en clase. Y así poder reconocer objetos en una imagen haciendo uso de un modelo pre-entrenado y poder entrenar un nuevo modelo que reconozca un objeto nuevo que no pueda ser reconocido por los modelos pre-entrenados.

Reconocimiento de Objetos con Modelo Pre-entrenado

En primer lugar se realizó un programa en Python para reconocer objetos en una imagen haciendo uso de modelos pre-entrenados. El código utilizado es el siguiente:

```
1 from imageai.Detection import ObjectDetection
2
3 detector = ObjectDetection()
4 model_path = "models/yolo-tiny.h5"
5 input_path = "input/inputimg.jpg"
6 output_path = "output/outputimg.jpg"
7
8 detector.setModelTypeAsTinyYOLOv3()
9 detector.setModelPath(model_path)
10 detector.loadModel()
11 detection = detector.detectObjectsFromImage(input_image=input_path, output_image_path=
    output_path)
12 for eachItem in detection:
13     print(eachItem["name"] , " : " , eachItem["percentage_probability"])
```

En principio se probó este programa con la siguiente imagen tomada en Argentina:



Figura 1: Imagen de entrada tomada en Argentina.

Y se obtuvo a la salida la siguiente imagen:



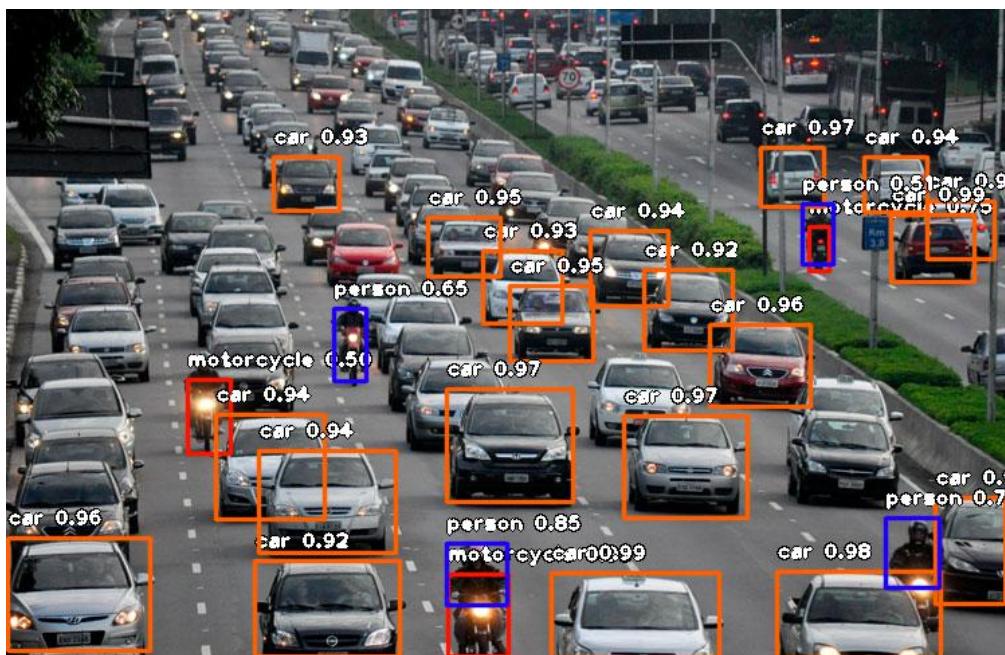
Figura 2: Imagen de salida con objetos reconocidos.

Como se puede ver, el modelo pre-entrenado funcionó bien reconociendo autos y personas.

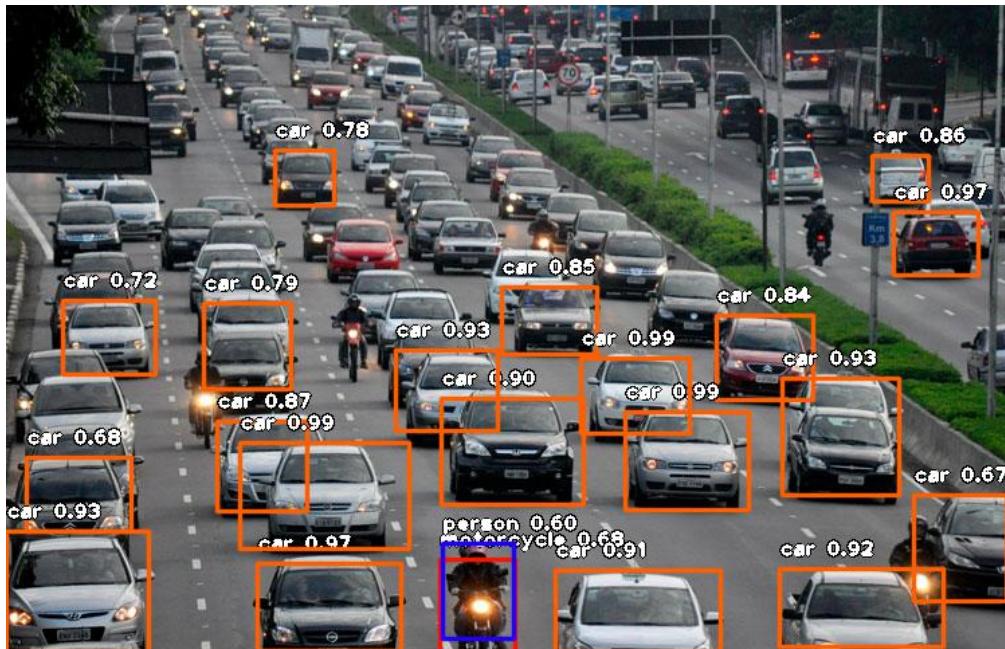
Luego de haber probado el programa, se prosiguió a comparar la eficiencia de 2 modelos pre-entrenados distintos los cuales son: YOLO y tinyYOLO. Dicha comparación se hizo con 4 imágenes distintas como se podrá ver en las siguientes imágenes:



Figura 3: Primera imagen a la entrada.



(a) Objetos reconocidos con YOLO



(b) Objetos reconocidos con tinyYOLO

Figura 4: Imágenes a la salida.

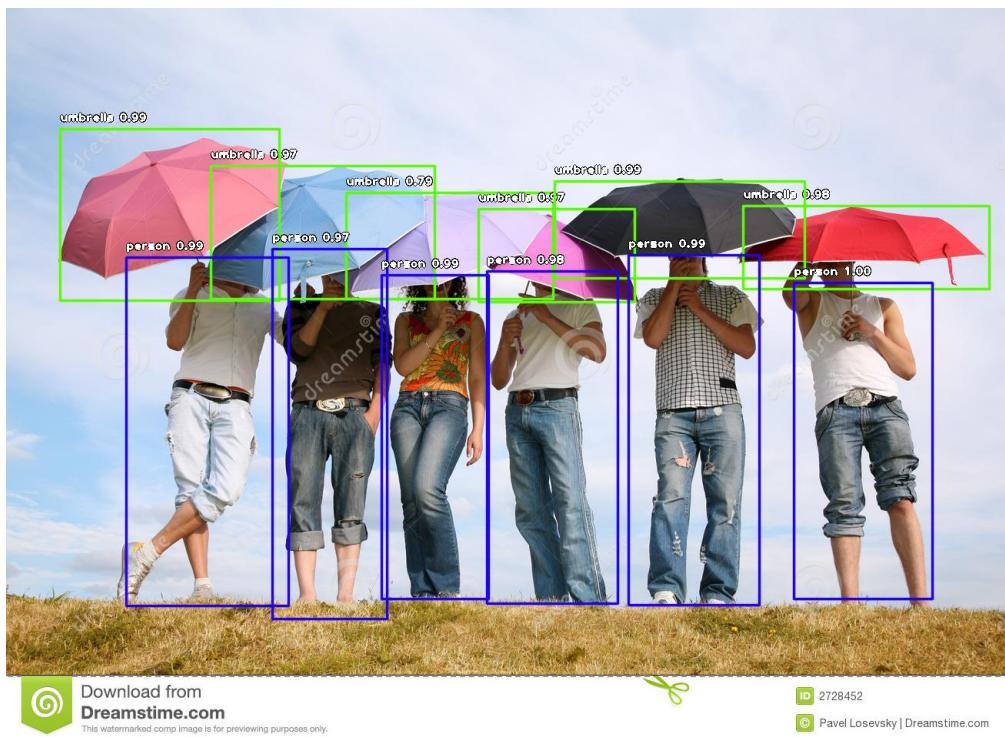


Download from
Dreamstime.com
This watermark comp image is for previewing purposes only.



ID 2728452
© Pavel Losevsky | Dreamstime.com

Figura 5: Segunda imagen a la entrada.



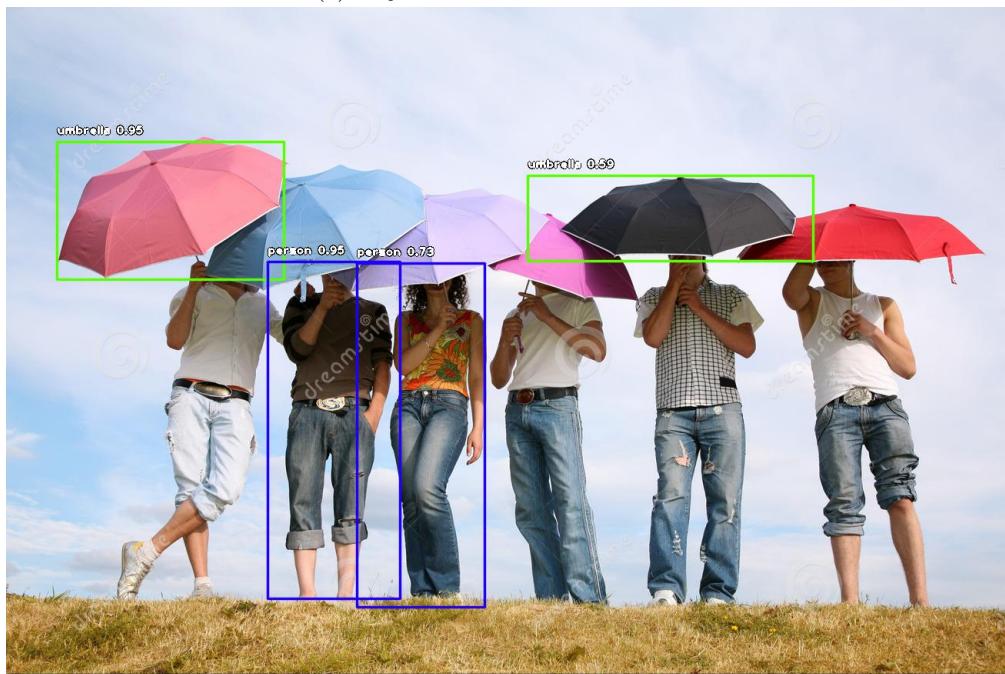
Download from
Dreamstime.com

This watermarked comp image is for previewing purposes only.

ID 2728452

Pavel Losevsky | Dreamstime.com

(a) Objetos reconocidos con YOLO



Download from
Dreamstime.com

This watermarked comp image is for previewing purposes only.

ID 2728452

Pavel Losevsky | Dreamstime.com

(b) Objetos reconocidos con tinyYOLO

Figura 6: Imágenes a la salida.



Figura 7: Tercera imagen a la entrada.



(a) Objetos reconocidos con YOLO

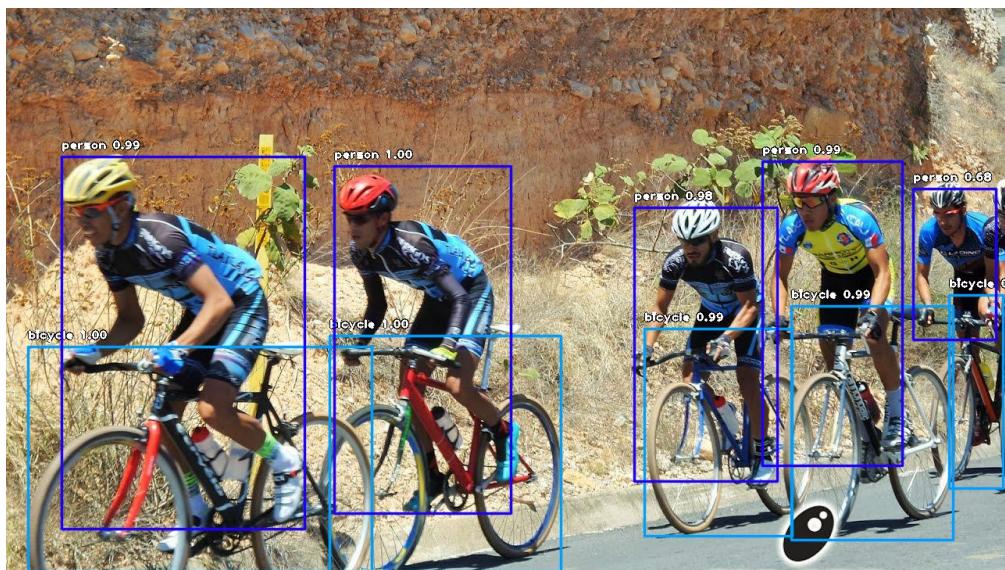


(b) Objetos reconocidos con tinyYOLO

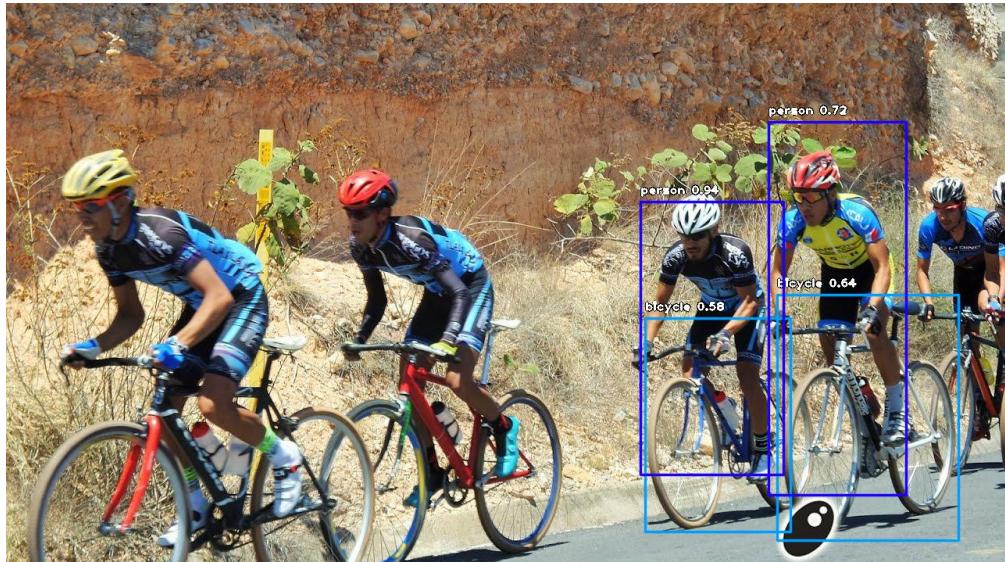
Figura 8: Imágenes a la salida.



Figura 9: Cuarta imagen a la entrada.



(a) Objetos reconocidos con YOLO



(b) Objetos reconocidos con tinyYOLO

Figura 10: Imágenes a la salida.

Se pudo observar una notable diferencia en el reconocimiento de objetos entre los modelos pre-entrenados YOLO y tinyYOLO. El modelo YOLO resultó ser mucho más preciso pero tiene la desventaja que ocupa mucha más memoria.

Entrenamiento de un Nuevo Modelo

Tras familiarizarnos con los modelos pre-entrenados, ahora podemos dar un paso adelante y entrenar un modelo que pueda reconocer un objeto que los modelos pre-entrenados no pueden.
El nuevo objeto a reconocer será el siguiente:



Figura 11: Nuevo objeto a reconocer. Ladrillo inteligente EV3

El nuevo objeto a reconocer es un ladrillo de lego perteneciente al kit "Lego Mindstorms EV3". Con el siguiente código se entrenó el modelo:

```
1 from imageai.Detection.Custom import DetectionModelTrainer  
2  
3 trainer = DetectionModelTrainer()  
4 trainer.setModelTypeAsYOLOv3()  
5 trainer.set_data_directory(data_directory="EV3brickset")  
6 trainer.setTrainConfig(object_names_array=["EV3brick"], batch_size=8, num_experiments=50,  
    train_from_pretrained_model="pretrained-yolov3.h5")  
7 trainer.trainModel()
```

El entrenamiento del modelo duró un total de 30 horas aproximadamente. Cada iteración tardó un promedio de 35 minutos.

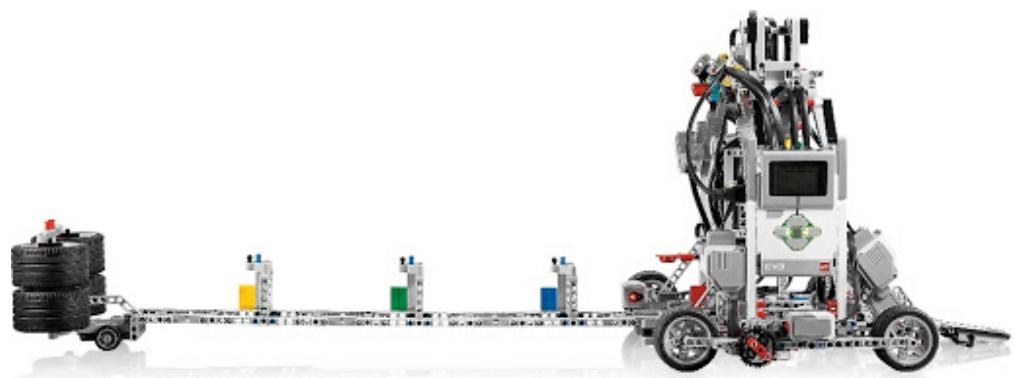
Se usaron 50 iteraciones para el entrenamiento. En la primera iteración se encontró una perdida del 92 %, luego fue bajando considerablemente hasta la iteración 8 donde la perdida fue de un 9.3 %. A partir de ahí, la perdida fue bajado un 1 % por cada 5 iteraciones aproximadamente. Finalmente, en la iteración 44, se obtuvo la menor perdida posible el cual fue de 2.5 %.

Para dicho entrenamiento se han utilizado 125 imágenes en total (100 imágenes para entrenar y 25 para validación) y se utilizó el modelo YOLOv3 como base.

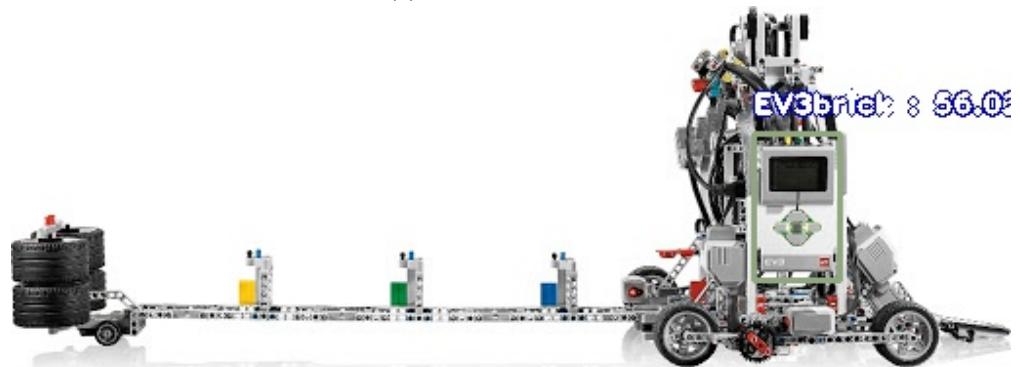
Se ha utilizado el siguiente código para probar el nuevo modelo:

```
1 from imageai.Detection.Custom import CustomObjectDetection  
2  
3 detector = CustomObjectDetection()  
4 detector.setModelTypeAsYOLOv3()  
5 detector.setModelPath("detection_model-ex-044--loss - 0002.569.h5")  
6 detector.setJsonPath("detection_config.json")  
7 detector.loadModel()  
8 detections = detector.detectObjectsFromImage(input_image="input/EV3bricktest1.jpg",  
    output_image_path="output/EV3bricktested1.jpg", minimum_percentage_probability=30)  
9 for detection in detections:  
10     print(detection["name"], " : ", detection["percentage_probability"], " : ", detection["  
        box_points"])
```

Dicho programa se probó con las siguientes imágenes:



(a) Imagen de entrada.

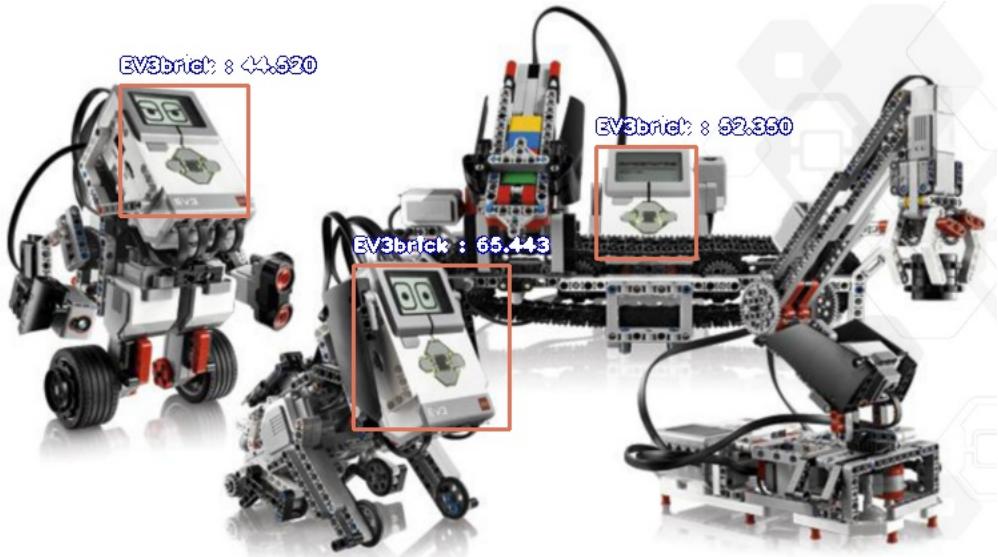


(b) Imagen de salida.

Figura 12: Primera imagen utilizada.

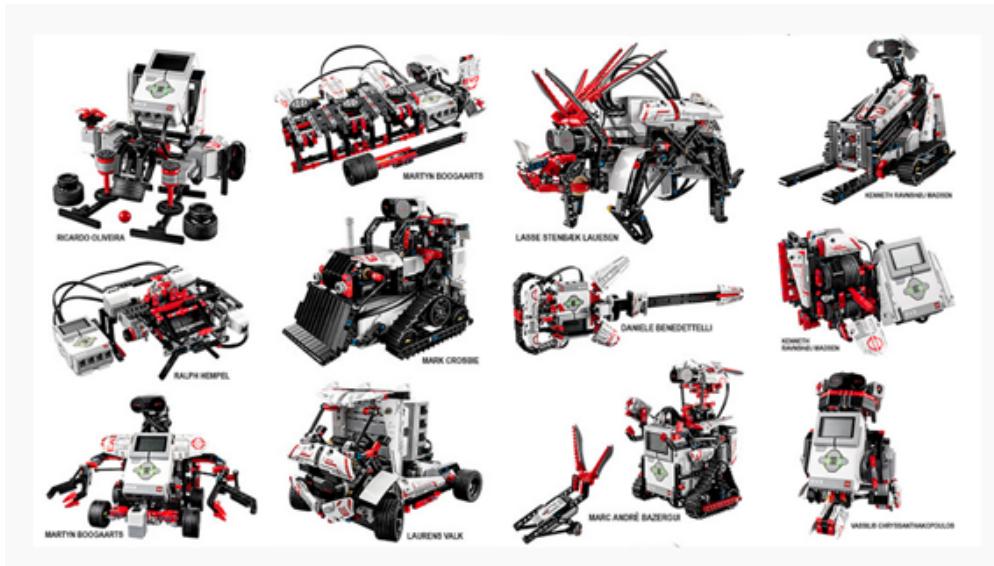


(a) Imagen de entrada.

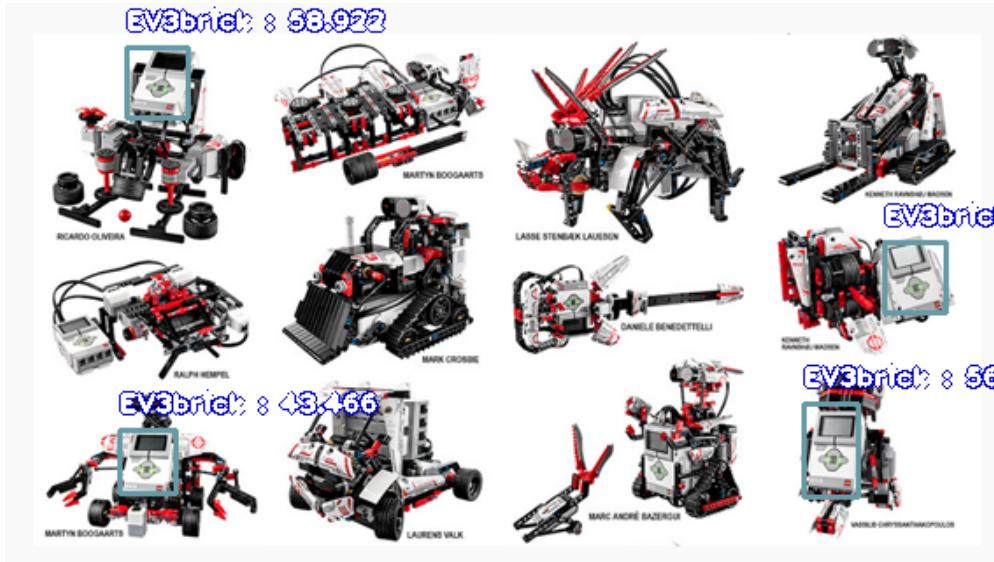


(b) Imagen de salida.

Figura 13: Segunda imagen utilizada.



(a) Imagen de entrada.



(b) Imagen de salida.

Figura 14: Tercera imagen utilizada.

Como se pudo observar, el modelo se entreno bien, puesto que pudo reconocer el objeto repetidas veces en las imágenes. Lamentablemente en la última imagen, el modelo no pudo reconocer todos los objetos.

Se podría mejorar el reconocimiento de este objeto utilizando más imágenes en el entrenamiento, ya que se recomienda un mínimo de 200 imágenes. También se lo podría mejorar con más iteraciones.