FIND ME!

Finding items by using room cameras

Final Project Book

School of Computer Science – Deep Learning

Supervisor: Mr. Yoram Segal

Group Members:

Yoel Borochov, 316305721 – Model training, darknet, backend, Dataset.

Tal Cohen Harari, 307939470 – Dataset, Model training, Writing reports.

Yechiel Raveh, 206336885 – UI-React, build the model – Darknet, DL model.

Asher Noy, 301290375 – Backend – Django, Connecting the DL model to the backend.

Raphael Ben-Hamo, 315606996 – Camera service.

Group: 613

Date: 14/07/21

Version: 1.1

Table of Contents

Table of Figures

List of concepts

- **YOLO V4:**
  YOLO or You Only Look Once, is a popular real-time object detection algorithm. YOLO combines what was once a multi-step process, using a single neural network to perform both classification and prediction of bounding boxes for detected objects.

- **Darknet:**
  Darknet is Open-Source Neural Networks in C and CUDA Darknet is mainly for Object Detection, and have different architecture, features than other deep learning frameworks.

- **Object detection:**
  Object detection is a computer vision task that involves predicting both where the objects are in the image and what type of objects were detected

- **Model:**
  Deep learning models are built using neural networks which are then processed in hidden layers using weights that are adjusted during training.

- **Layers:**
  A layer is the highest-level building block in deep learning.
  A layer is a container that usually receives weighted input, transforms it with a set of mostly non-linear functions and then passes these values as output to the next layer.

- **Convolution:**
  A convolution is the simple application of a filter to an input that results in an activation.
  Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input.

- **Neural network:**
  A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. There are different types of neural networks but they always consist of the same components: neurons, synapses, weights, biases, and functions.

- **Augmentation:**
  The performance of deep learning neural networks often improves with the amount of data available. Data augmentation is a technique to artificially create new training data from existing training data.

- **Loss function:**
  Machines learn by means of a loss function.
  It's a method of evaluating how well specific algorithm models the given data.
  If predictions deviates too much from actual results, loss function would cough up a very large number.

- **Dense layer:**
  Fully connected - Dense layer, also called fully connected layer, refers to the layer whose inside neurons connect to every neuron in the preceding layer.

- **Fully connected:**
  A fully connected neural network consists of a series of fully connected layers that connect every neuron in one layer to every neuron in the other layer.

- **Optimizer:**
  Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate to reduce the losses.
  Optimizers help to get results faster

- **Segmentation:**
  is a type of labeling where each pixel in an image is labeled with given
  concepts. Here, whole images are divided into pixel groupings which can then be labeled and classified, with the goal of simplifying an image or changing how an image is presented to the model, to make it easier to analyze." [1]

Abstract

Finding objects in different spaces is familiar to each of us, we start turning every nook and cranny to find a room, but unfortunately, many of the companies that tried to find a solution in the past did not find an effective solution that utilizes the resources and technologies available today. More convenient and easy.

FIND MY is a platform that will help solve these problems and reduce the costly expensive chips that other companies use, the solution will be much more efficient without comparison to previous solutions at all.

Every solution is offered today based on **GPS** technology that is inserted into our objects in one way or another, from the introduction of chips in the production phase to chips that can be integrated independently afterwards as Apple recently launched the **AIRTag**, [2]

FIND MY gives you the ability to search for objects easily and simply by selecting the item
on your iPhone or computer,
Our platform is trained in a variety of objects and knows how to handle many different positions
That objects are placed to make object search as efficient as possible.

FIND MY technology:

We collected large data of handbags and halves handbags through the **google** library "**open image**" [3]
and through photos we collected ourselves,
we had to collect unique data for find me because at home the objects are probably not
always clear and visible and need a very smart model who can handle all kinds Difficult locations
of objects in space.

Therefore, we chose to perform **detection** using a model of **yolov4 with 162 layers**,
What is special about this **model** is that it works with the **"two stage models"** method, [4]
Which means instead of taking the image and performing a whole detection we first look for parts
of the image that may have objects in them, and only then do we perform **detection**
for those parts and look for objects.

In the first stage of the model, after entering the image, we run **convolution networks** called **Backbone**, [5]
they know how to extract the same suspicious areas in the image and only then do we run the detection itself.

Another reason we chose him because he knows to deal with many objects in the same image,
this is derived from the previous reason, because he knows how to separate objects in the image
and only then send them for detection,

The big advantage of using yolov4's network is that the network also consists of a platform called **darknet54**, [6]

The darknet allows us to fully process pieces of scattered information from the previous Backbone network and passes all parameters to the **dense network**,

There we do detection using the Neck which we will expand on its mode of operation later in the chapter which deals with how the model works.

One of the main reasons we used **yolov4** is because **SPP** is used,

SPP gives us a solution to 2 main problems:

1. Maintain the output vector size at a fixed length regardless of the input size.

2. His abilities to bind various features that came out of the network that preceded him.

The **FIND MY** application will run on several technologies:

1. **Camera Service** - Each space will have a connected camera
   Broadcast to our camera service once checked a picture from space.
2. **AI-DL service** - is responsible for receiving the image from the service production of the various Objects in the room, and managing the registration of the last location where each object is located Seen in DB.
3. **User interface** - The user interface of the FIND MY platform will be available to every user Interested in connecting to our service.

The Problem

There are companies that have tried to solve this problem using GPS technology that comes in a small chip that for every item we want to find we need to attach the chip to it.

This solution still leaves us in trouble because it has many disadvantages for example:

1. The chip is expensive and needs to be attached to each object.
2. After a certain time you need to replace the batteries in each chip.
3. The location is not accurate but relative.
4. There are objects for which there is no way to attach the chip.

And the biggest problem with this solution is that in order to find our objects we need to change our lifestyle and remember to attach a chip to each object,
This solution is both expensive and also takes a lot of time and ultimately does not give us a quick and easy answer.

We conducted a poll among different factions in the population and the results were clear,
everyone suffers from a search for objects,
But what was interesting was that among people aged 40-60 we found that at this stage of life they become paranoid from losing an object, they are so anxious not to find an object at a given time that they instruct the rest of the family not to touch their belongings.
Also another very common problem is the loss of car / home keys and the loss of the phone in shopping malls,
In shopping malls we are usually required to put our belongings aside in order to measure clothes or collect products into a cart and many times we forget to take our belongings afterwards.

Also visually impaired people will not have to feel helpless because of one disability problem or another.

Who among us does not know the phenomenon of searching for an object in the house and does not find it, do not remember where we placed it?

For example:

- Come turn on the air conditioner and do not find the remote,
- Looking for the bag before leaving the house and not remembering where we put it,
  And more full of everyday examples.

Obviously the problem would not have arisen if we did not put every object in place but for many of us it is a nice dream that in reality does not always happen,
a search at home can take a very long time, and sometimes we encounter despair and give up, this problem accompanies us every day everywhere, our objects are many and varied And it's hard to remember where we put everything, and sometimes we did not move the object so it is out of place,

another serious problem we found is, there are cases where we are sure that an object is around the house or work / office ..., and in the end it turns out it is not there, the time we spend On the search for objects is huge and completely unnecessary,
Here we realized that there is a need for help finding objects, making it easier for people to find what they are looking for, making it easier for all types of people, children, men and old women alike, helping the visually impaired to find easily without despair, There is an amazing company "be my eyes" that deals with the field of accessing content for the visually impaired that we are interested in putting our app into them clearly for the visually impaired. [7]

The understanding is that in 2021 with the technology at our disposal we must find a solution to this perpetual problem.

To this end, FIND MY convened the best minds in Israel to make a difference.

Related Work

For a long time companies have been trying to find solutions to the problem of losing items in the home, There are several solutions in all kinds:
At first there were sound-based solutions, it was mainly to find keys they would put a small model in a keychain and the little device would listen to the sounds as soon as he heard a clapping or whistling sound the device would beep and that way they would know where the keys were.

The device could be used not only with keys also with every item in the house, the device has many mistakenly detecting the sound of a clap or whistle and was beeping many times just because, because of its low quality these devices were quickly thrown into the trash and were not popular with consumers.
Over the years the electronics became smaller and cheaper and the battery life improved, they became using radio frequency-based devices, there was a device equipped with a small receiver and there was a remote that when pressed it makes the other device beep, As a result, the devices had to listen all the time. The battery life was every three months , but using a radio signal remove the false alarms that were with the previous sound-based device.

Today there are newer devices that are GPS and Bluetooth based on weak power, they have a unique ID for each and every user and you have to put them on anything you want to find, in case you want to find the object you enter the app on the phone and it shows you exact location of the object.
Example of trackers that work this way are Chipolo, ON!Track (by Hilti), Airtag by Apple, [2], [8] Tick (by Milwaukee Electric Tool Corporation), Tile, TrackTag (by Bosch), TrackR and PROTAG. [9]

The Solution

The FIND MY platform differs from all the solutions offered to date in several ways:

- The lifestyle remains as usual without the need to attach chips to objects.
- It is much easier to search for an item and get a photo of the room with an exact location where it is, without having to follow the GPS which is inaccurate.
The solution is significantly cheaper than all the options available today.

So how is the solution going to work:
User side:
The user will install a camera in the spaces of the house in different rooms until the area he wants to cover is covered,
Then all he has to do is log in to our app, select the item he wants to search for, and the system will return an answer,
If the system recognizes the object that the user has entered, the system will return an image with the item in a square marked in color with information about which room the item is in.
If the system does not recognize the item, the system will return it to where it was last seen on the camera.

Our technology:
We have built a smart technological system using artificial intelligence that knows how to classify images and do detection for images.
The camera service will send an image once a minute to the AI system that knows how to do detection with deep networks, after the system has managed to classify the object it will send an update to DB that will save the location of the object where it was last seen with the appropriate category classification,
Each time the user searches for an object we will retrieve the information according to the type of object requested the last location we will see it. [10]
General action description:

1. The customer will install cameras around the house and connect them to our system so that FIND MY can scan his house.
2. Data will be collected about all objects captured on the customer's camera.
3. Our system will detect images collected from the customer's cameras,
4. The information will be stored in a DB service that knows where the objects are located within the image and will be displayed to our user by a request / query from the database.

Architecture

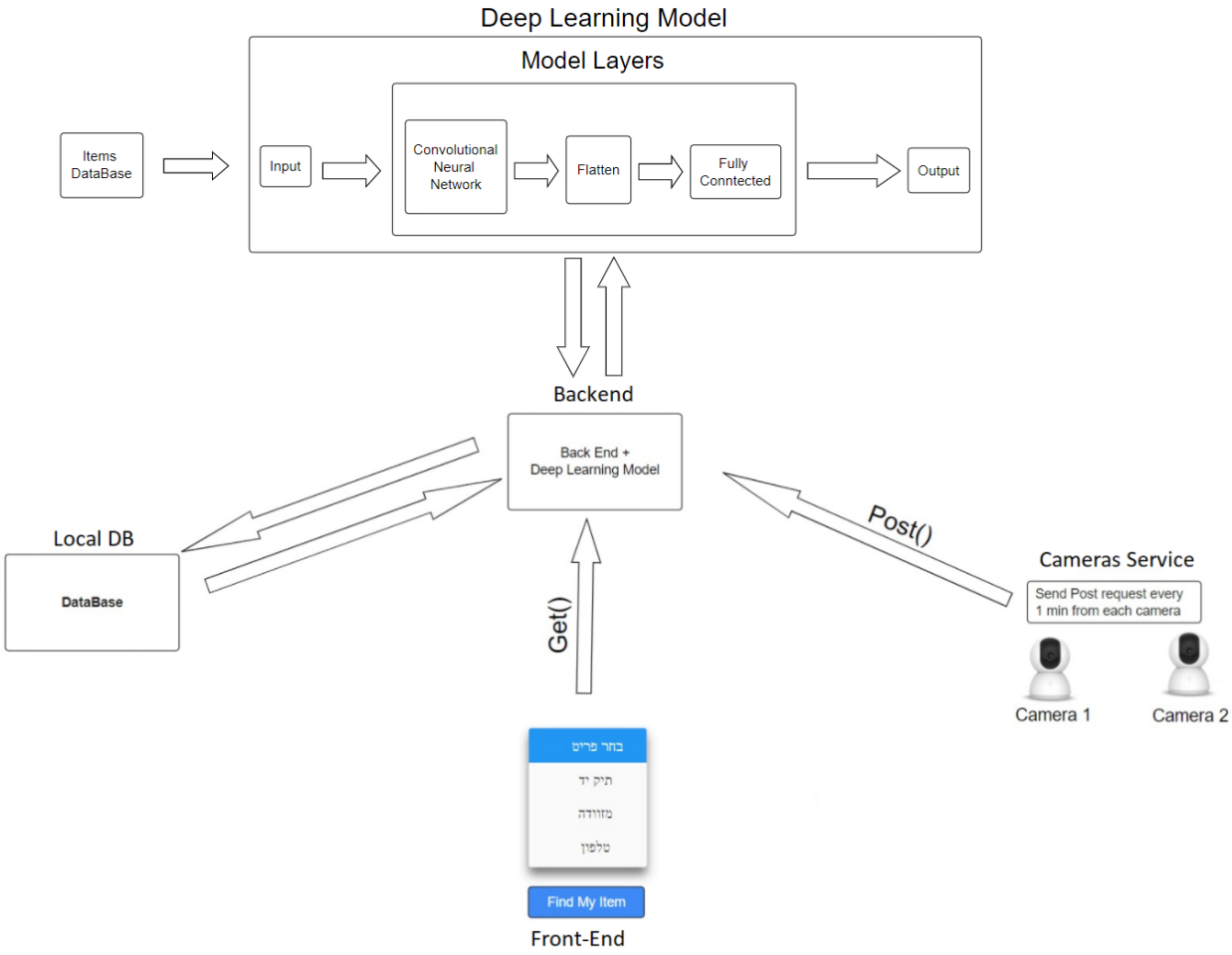


Figure 1: FIND ME Architecture

In figure 1 we see multiple components:

- **The front-end:**

- This is a part where a user enters the system site and is presented with a list of predefined objects that the system knows how to find.
- The user selects one of the options and clicks on the "Find my item" button.
  The request is sent to the backend server and the server looks for an image with such an item in Local DB and returns a response to the user by an image where the item was recently found.

- **The back-end:**
- The Backend server receives images every minute of what the camera service is
- and tries using the artificial intelligence model to find items in the image that are predefined for it and saves them in a folder sorted by camera number.
- In addition, the server handles user requests that come from the frontend and it searches for the item that the user requested and if it finds the item, it sends the user the image with the location of the item.
- **The camera service:**
- Cameras are installed in the house, the cameras every minute capture the room and send the image to the backend server.

- **The local DB:**
- The images that come out of the model of artificial intelligence are stored in a folder that serves as a database and each folder is divided according to the camera number, and within each folder of a camera number there is a folder for each item in the image.

- **The Deep Learning Model:**

  Model has 162 Layers, at first there is the INPUT layer that gets an image,

  Then the model continues with 44 Convolution layers to extract features in the image.

  We then FLATTEN the image and flatten it to prepare the image for FULLY CONNECTED layers.

The main flow of "FIND ME" platform:

We use DARKNET, technology and YOLO model, we use this model and changed the last layers of the model. [6], [11]

We trained the model on our own data that we downloaded from a site called "Open Images Dataset v6", [3]

We downloaded it using the code who downloaded all the data from the above site.

All this is the data for the files that the model will know to identify what is a portfolio and what is not a portfolio.

The model will indeed be able to identify using our data and our layers what is a portfolio and what is not a portfolio, but it has not been able to identify half-portfolios.

But of course we were not satisfied with that and we wanted to bring added value to our project and create something that never existed, we wanted to bring the model to higher levels, we wanted to teach the model to know how to identify halves of bags. In reality it is placed under a pillow or under a garment or anything else, in other words it is hidden.

We searched a lot on the internet for half-bag data but could not find it, but we did not give up and decided to take a lead and build our own data, we took about 300 pictures of half-bags ourselves and added them to the training group.

Then again we trained and taught the model about all our data both downloaded and photographed ourselves, and we saw wonders , the model did manage to detect half-bags!

FIND ME Modules

- **Client side (Front-end):** This is the FIND ME website, it's has one page for the client to search items from the camera.

  The client side run as a web application also for computer and also for mobile.

- **Server side (Back-end):** Server side is written in Django and it is responsible of getting posts requests and manage the data store

- The service contains two post services, update detection and receive last detection service

- When a camera is connected to the system, the camera sends a picture to the service every few seconds, as part of the post requests, an unique camera ID is also sent.

- The service receives the image and sends it to the model, if an object is detected, the system saves an image received from the model.

- **Model:** The model contains a function that receives an image

- The image is sent into the yolo v4 model.

- If the model detects an object, it produces an image that marks it and sends this image to the service component

- the data is maintained under the file system
- each folder under the main root represents camera id
- each camera id folder contains classes folder, a folder for each class

Database Relationships

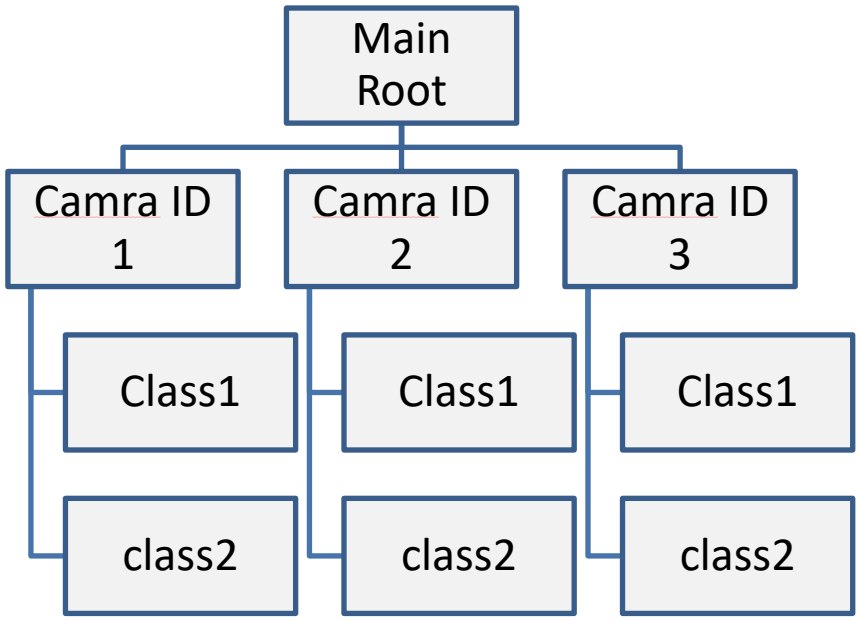


Figure 2: Database relationships

Project Runtime

Model Runtime:

Train the detection model without half handbag – 13.5 hours

Train the detection model with half handbag – 13.7 hours

Database Runtime:

Insert image from camera service – every minute photos from the camera service are sent to the model and insert to the database.

Pulling images from the database to the user – this happen every time the user ask from the UI.

Academic Justification

First of all, we collected all the pictures of the DB from the site of Open Images [3].

The date came categorized.

We trained our model using DARKNET technology which loaded our layer file and our data.

We added to our own data training photos that we took of half of the bags and not just of the bags themselves that we downloaded from the site.

The model learns to identify inside the image where there is a case and perform DETECTION on it.

Second, after the model is properly mastered we have created a FRONT-END service where the user can choose which item he wants to find from the given list of items, he does so by clicking the "Find My Item" button.

Third, our BACKEND side written in NODE JS connects the various platforms (between the client side and the AI) and in order to mediate a smooth transition of the data from the client straight to artificial intelligence.

Our BACKEND service receives images at the request of a POST from the CAMERA SERVICE service that takes every minute an image from the room where it is located, and performs DETECTION on it and saves the result in data in a measure and finds the item that defines it in the system as an item.

All BACKEND services communicate with DB to store and retrieve data.

As you can see, we used different tools and development languages in the different parts of

project.

YoloV4 Explanation

A thorough explanation of how YOLOv4 works

The Realtime object detection space remains hot and moves ever forward with the publication of YOLO v4. Relative to inference speed, YOLOv4 outperforms other object detection models by a significant margin. We have recently been amazed at the performance of YOLOv4 on custom object detection tasks and have published tutorials on how to train YOLOv4 in Darknet and how to train YOLOv4 in PyTorch.

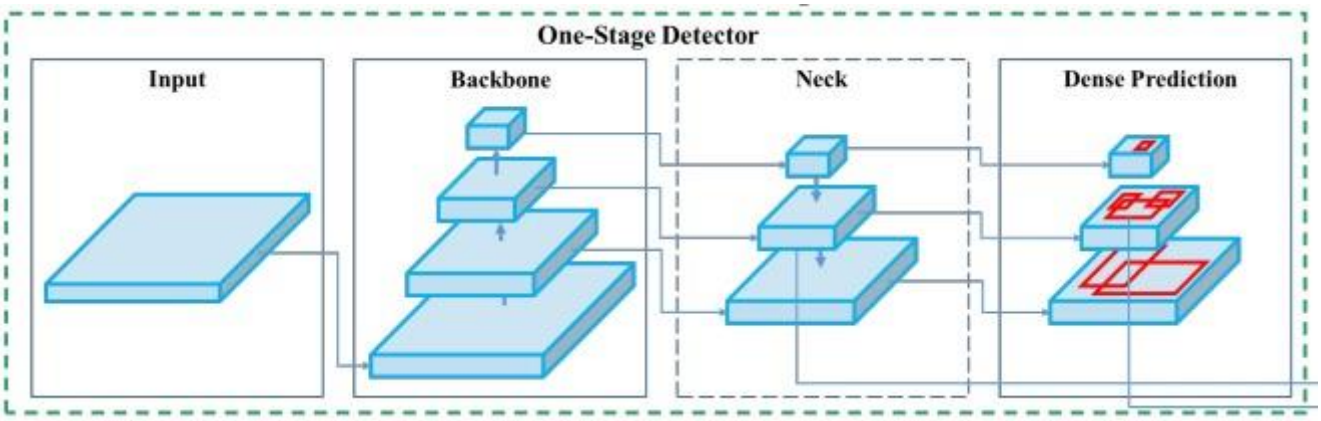


Figure 3: One-Stage Detector

Input - the input layer receives an image as array.

Backbone - This is a neural network consisting mainly of convolutional layers.

The main purpose of this network is to extract the features so that it is a key step in object recognition performance.

The backbone architecture consists of few parts:

1. Bag of freebies -

We do all kinds of manipulations on the data to improve performance

Identify objects, such as image rotation, image cropping, image brightness, and darkness

Image, etc. Noise on the image to increase performance.

2. bag of special -

This is a bag of special methods from the set of methods that increase the cost

Inference in small quantity , But can significantly improve the

The accuracy of object identification.

Neck - The essential function of the Neck is to collect features from different stages.

The Neck usually consists of several bottom-up tracks and a number of top-down tracks.

What is the problem caused by CNN and FC?

The FC network requires a fixed size so we need to get a fixed size image, when we identify objects we do not necessarily have a fixed size image.

This problem forces us to correct the images, this method can remove some of the object we want to identify and therefore reduce the accuracy of our model.

The second problem caused by CNN is that the size of the sliding window is fixed.

I SPP running? We can get a filter capable of detecting circular geometric shapes, this filter will produce a feature map that highlights these shapes while maintaining the position of the shape in the image.

Dense prediction - literally, the FC layers that help us identify the desired object.

The Loss function – YOLO V4 use IOU loss function, The definition of IOU loss is very simple, that is, the
difference between 1 and the intersection ratio between the predicted box A and the real box B.

$$LOSS = 1 - IoU + \frac{\rho^2(b,b^{gt})}{c^2} + \alpha v -$$
$$\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{obj}\left[\hat{C}_i \log(C_i) + \left(1-\hat{C}_i\right)\log\left(1-C_i\right)\right] -$$
$$\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B} I_{ij}^{noobj}\left[\hat{C}_i \log(C_i) + \left(1-\hat{C}_i\right)\log(1-C_i)\right] -$$
$$\sum_{i=0}^{S^2} I_{ij}^{obj}\sum_{c\in classes}\left[\hat{p}_i(c)\log(p_i(c)) + \left(1-\hat{p}_i(c)\right)\log(1-p_i(c))\right]$$

Figure 4: loss function formula

Object Detection Models

All of the YOLO models are object detection models. Object detection models are trained to look at an image and search for a subset of object classes. When found, these object classes are enclosed in a bounding box and their class is identified. Object detection models are typically trained and evaluated on the COCO dataset which contains a broad range of 80 object classes. From there, it is assumed that object detection models will generalize to new object detection tasks if they are exposed to new training data. Here is an example of me using YOLOv4 to detect cells in the bloodstream.

Software Requirements Specifications

Use Cases

- The customer wants to find an item that get lost.
- The customer wants to know where is each item in the house.
- The customer has a memory problem.
- The customer wants a summary of each item where it was in the house in order to analyze the behavior of people in the house.

Class Diagram

The main classes in the platform are:

- DL and Rest Service – this is the service that responsible on the deep learning logic and takes the pictures from the Backend service.
- Frontend – an React app that show to the user the UI/UX in order to use the software and see where each item is.
- Camera service – this service responsible to take a snapshot each X seconds (by configuration and send it to the Rest service inside the DL Service.

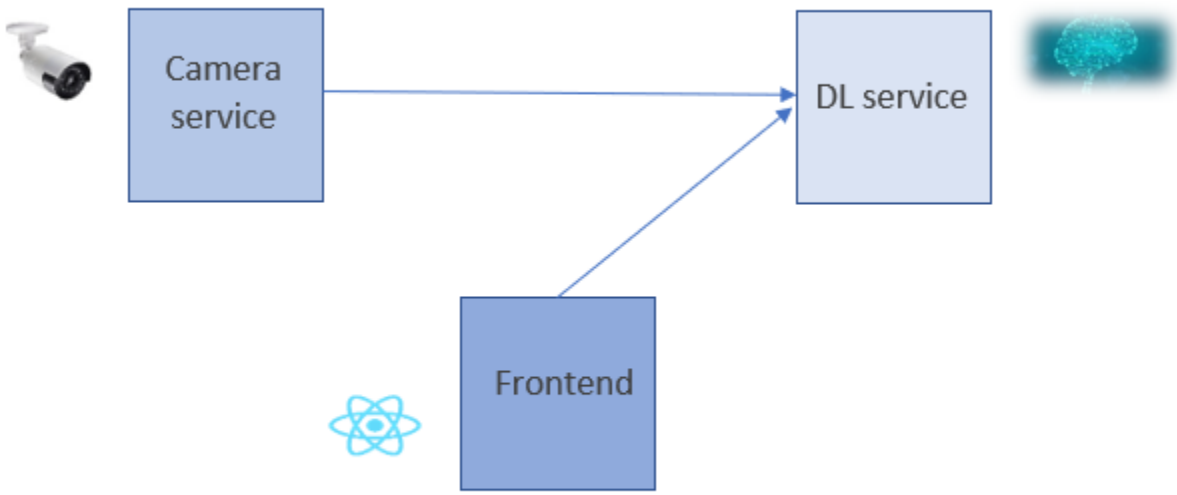


Figure 5: Service diagram

Sequence Diagrams

Flow of the main use case

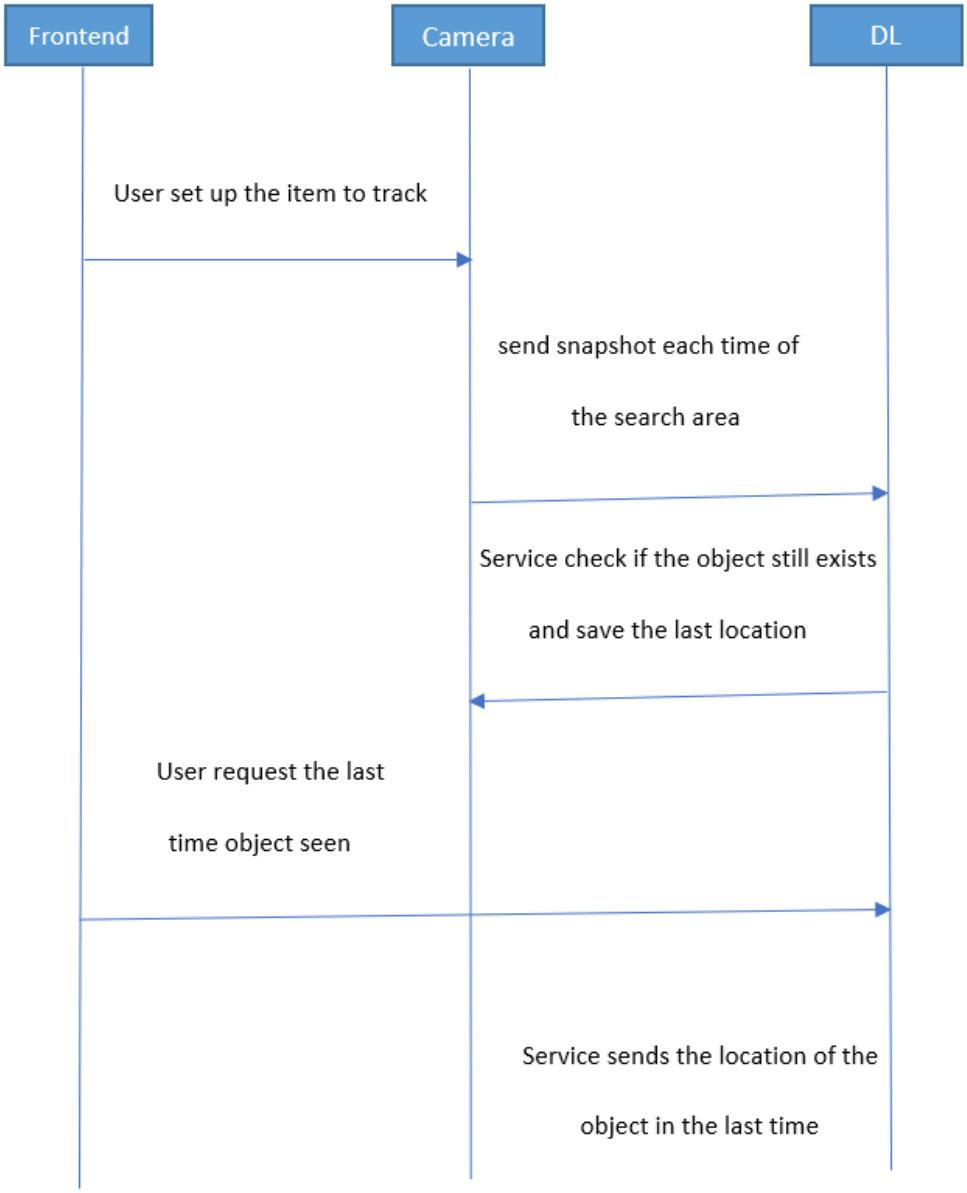


Figure 6: Sequence diagram – flow of the main use case

Dataset

Our database consists of two types of data.

One type is of handbags that are seen in their entirety, and the other type consists of half-bags.

The first type of data of the "perfect handbags" we downloaded from the Open Image website

(Link:

https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=segmentation&r=false&c=%2Fm%2F080hkjn

),

The data came categorized with Lables in advance.

The size of the data we downloaded is:

Handbag dataset:

- Train Dataset: 1500 pics
- Test Dataset: 55 pics
- Validation Dataset: 70 pics

As mentioned, it contains only data of handbags, not only handbags that see only handbags in the picture, but it consists of pictures of both handbags alone with a white background and with handbags on people or that are in all sorts of places with one background or another.

The second type of handbag halves is Data that we took ourselves picture by picture.

We added this data to all the data we downloaded, manually classified it with software and enlarged it, now we have data consisting of half bags and whole bags.

The data size of the half of the bags is:

Half handbag dataset:

- Train Dataset: 85 pics
- Test Dataset: 10 pics
- Validation Dataset: 10 pics

We trained our YOLO model on our data with our layer file, and we ended up with a file of weights trained on our data.

There was no need to do any processing on our data.

Design Requirements

- Cloud service based on an application written in react
- The server side manages, update and reception services for object detection and classification in the surveillance area
- Agile and minimalist information management of file system
- Yolo v3 model that recognizes and classifies the object at extremely fast rates
- All of these provide a stable system for locating objects in the surveillance area

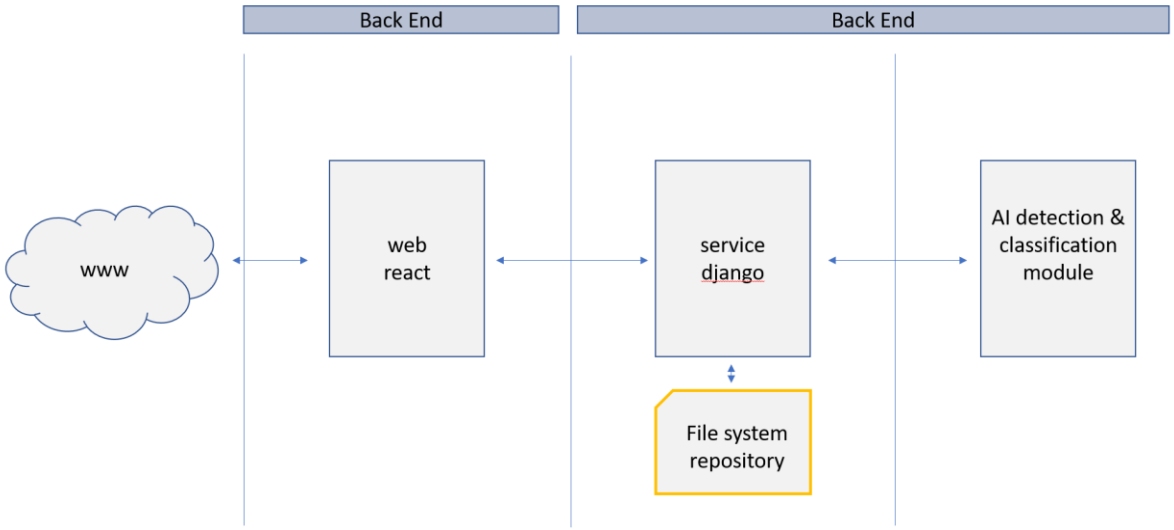


Figure 7: multi-tier architecture

Technical Specifications

Client side – Web interface

- Version: 1.0
- Language: JavaScript
    - Framework: React v17.0.2
    - Development environment: VS Code
- Hardware requirements: Only a web browser is required
- Operation system: Windows 7+

Server side – Camera service

- Version: 1.0
- Language: JavaScript
    - Runtime: NodeJS
    - Development environment: VS Code
- Hardware requirements:
    - Cameras: 2 X Camera Connected by USB
- Operation system: Windows 7+

Server side – Backend service

- Version: 1.0
- Language: Python
    - Framework: Django
    - Development environment: VS Code
- Hardware requirements:
    - CPU: 4
    - Memory: 4 GB
    - Disk: 10 GB
- Operation system: Windows 7+

Deep Learning Side – Train service

- Darknet
- Language: C
    - Framework: YOLOV4
    - Development environment: Visual Studio
- Software requirements:
    - CUDA
- Hardware requirements:
    - CPU: 4
    - Memory: 16 GB
    - Disk: 40GB
    - Graphic Card: NVIDIA GeForce GTX 1060 6GB
- Operation system: Windows 7+

Methodologies and Development Tools

Let's talk about technologies.

The Client side which is the UI/UX website was developed using ReactJS library that based on the component's architecture.

The DL service written with Python and uses Darknet framework in order to run the YOLOv4 model.

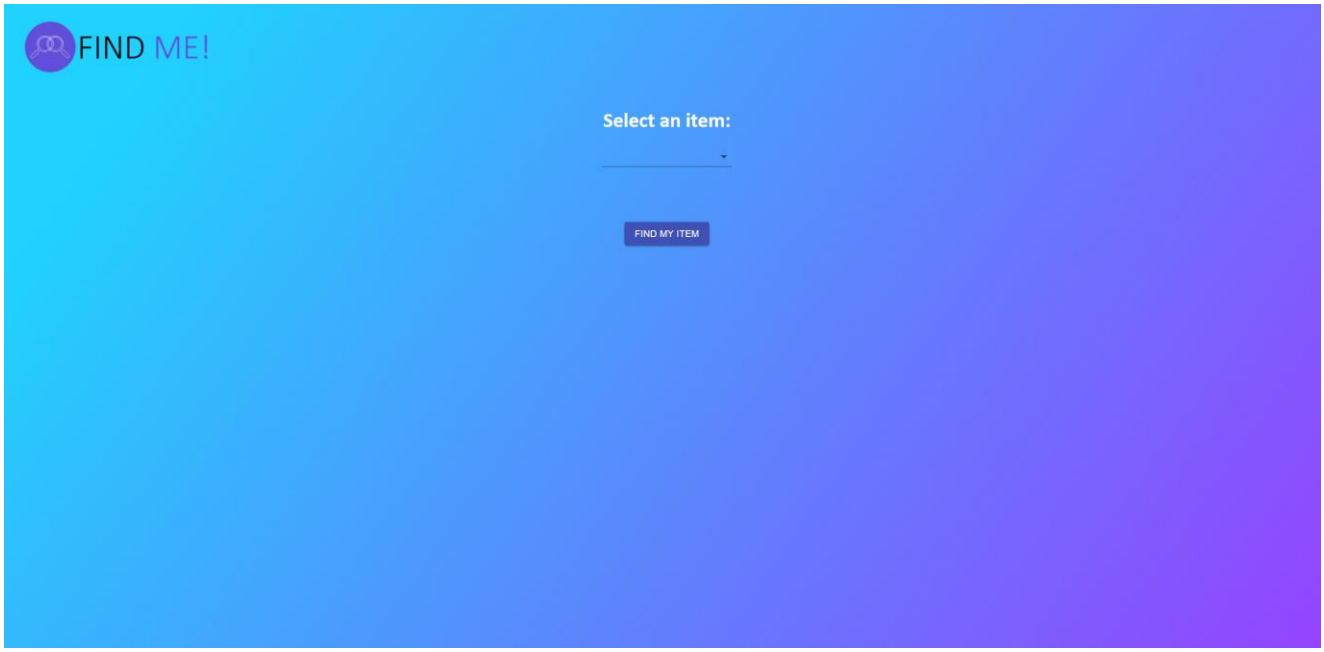
The Camera service written with Node JS which is JavaScript for desktop and compile the code scripts using

V8 engine that chrome browser runs, by now this is the best engine out there.

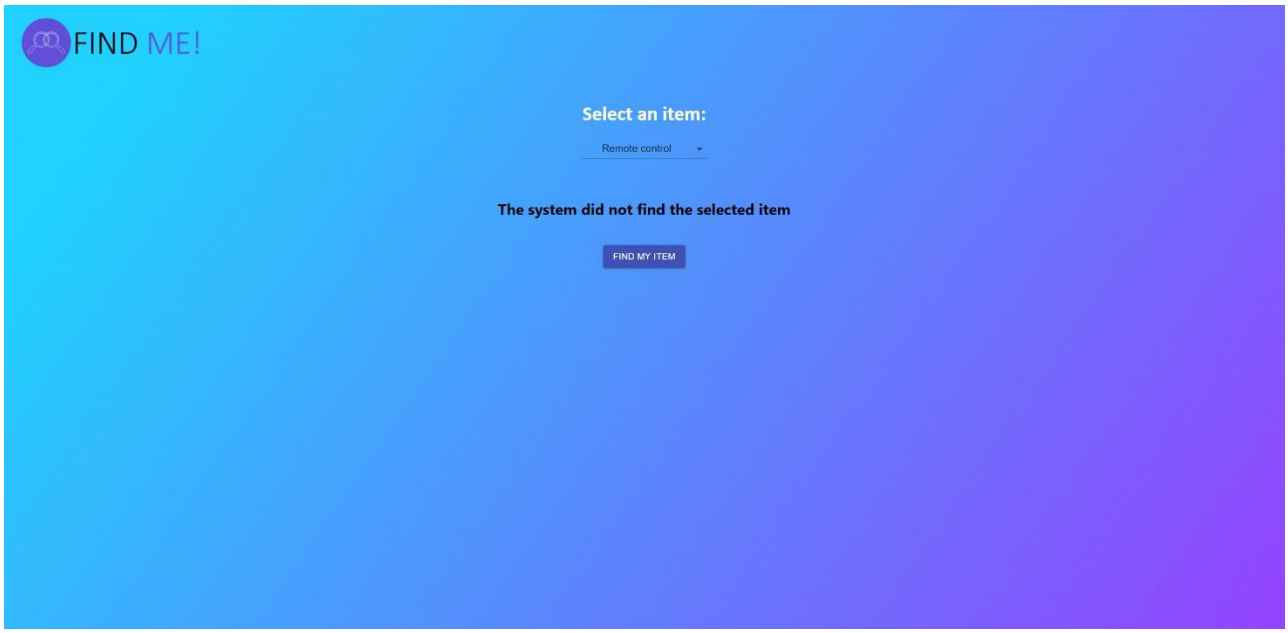
This service used express library in order to develop the Rest API and use a NPM library for taking snapshots from the cameras.

System Overview

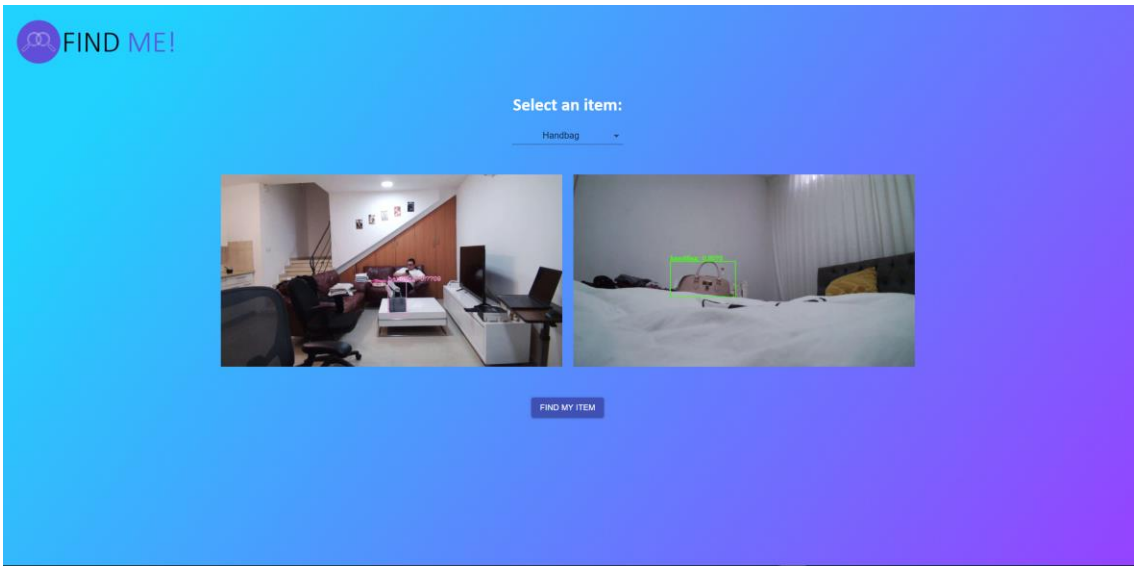
Item choose page: In this page the client choose an item from a list that he want the machine will tell him where he left it.



No item found Page: If there isn't any pictures with the selected item it will show the client error message.



The item images page: If there is any pictures of the selected item it will show him the pictures from all the camera's he found.



Project Outcomes

The system and initial goals

The platform will monitor the connected AWS accounts, provide recommendations to perform cost

This system is using the newest technologies out there, from the Client side and the Reach library to the YOLOv4 that is one of the best models of object detection in the world and have a better performance compare to more and more models.

These technologies allows us to use this system as a SaaS system which is stands for "Software as a service" that is web based platform that run in the cloud or server for example.

Numerical Data

Dataset
Handbag dataset:

- Train Dataset: **1500**
- Test Dataset: **55**
- Validation Dataset: **70**

Half handbag dataset:

- Train Dataset: **85**
- Test Dataset: **10**
- Validation Dataset: **10**

Model Layers:

- Num of layers: 162
- Convolutional Neural Network: 44
- Parameters: 60 million

Platform and SaaS advantages

The are couple of advantages when using this system as a SaaS system:

- You can use it whenever you want and see what's going on in your home office and more.
- You can fix and upload new version of the system very easily and quickly.
- All you need to have is a camera and you right to go!

Expansion options

The platform is written in a highly modular approach, which allows easy expansion of:

- Additional places to make them forget from losing items.
- Additional features.

Description of the activities we did in the project

detection or segmentation

At first, we tried to identify objects by segmentation that knows how to identify objects by pixels,

While working we realized that this method is both unnecessary for us and less good,

For several reasons:

1. Run times - takes longer to perform segmentation over detection because it is another step after detection,

Once you find the object in the image then also run a model that marks each pixel in the model.

2. It is more convenient for the end user to get an image in which the object is clearly visible in its original colors with the marking of a square around it,

This way he also recognizes the object he was looking for easily and sees that the color is the same as the item he was looking for.

What data should we work with

Once a decision was made to work with detection, we were looking for a good dataset that would be possible to work on and train the model.

1. **coco** - The first dataset we tried to work on is coco and after several attempts to run the code on this data we encountered some difficulties.

We were unable to match the image labels to the method that Coco's datasets use.

In all the examples we found using coco data we did not see the possibility of changing the model easily in the methods we are familiar with,

So, after a few failed attempts and a fair amount of colab notebooks we moved on to the second option we were left with.

2. **"open image"** of google,

The open image data is indeed very impressive and divided into categories,

But then we got stuck in a new problem, you cannot just download one category from the Google data.

Downloading everything was not an option because it is a huge and inconceivable amount of data and certainly it will not be possible to train on such an amount on our computers.

After searching the internet, we found a code running in colab that knows how to connect to the open image dataset and extract from there only the desired category.

detection in colab

After the data problem was resolved we tried to run detection in colab.

We encountered countless runtime errors, it turns out that the function that was supposed to receive the data wanted to receive the data tagging differently.

Finally, after all the necessary adjustments we were able to run detection in colab and indeed the model was able to identify a handbag in the picture,

We already after a long time in colab realized that we are very limited in working with colab because we must interface with 3 different platforms that run simultaneously, frontend, backend, camera service.

But that was our little problem, the model did not recognize us well enough all the pictures we put for it, we could not change the model as we wanted, every time we ran a training it took full time and sometimes stopped in the middle, and we realized it was time to install all the required libraries on the computer Directly complex as it may be,

darknet on desktop

This is where the project really started,

We have opened countless tutorials on YouTube and learned how to install darknet on your computer,

The main problem that accompanied us all the time is the incompatibility with the GPU versions that are installed on our computer.

The darknet in most of the world turns out to be running on Linux or mac, adapting the project to Windows is almost impossible.

After countless versions of NVIDIA to GPU we were able to install the darknet on the computer in good time.

Burning the CPU

A big mistake we made was that the first time we ran the workout on our local computer we did not notice that the darknet alone does not know how to run on the GPU.

The result was the burning of Intel's I7 CPU.

But from this stage everything ran smoothly for us, in the end we used the model of yolov4 which is the most advanced on the market as of April 2021.

By the darknet we were able to run a computer workout on half bags because the model did not recognize objects that were not clearly visible,

Things we succeeded in:

- Collect our own data and tag it by code.
- Run the project in the local environment and not in colab.
- Full synchronization between the various platforms we have built.
- Change the model until you get a better result.
- A DL app that uses all the products of deep learning.

we did not succeed:

Filter a search request from the user of an object by color.

What in the future?

Extending the model to many categories of objects.

Implementation of the app not only for the use of locating objects but also for security that will identify where a baby has gone, drowning in the pool, burglary of houses and more.

Bibliography

[1] N. Fletcher, "Classification Vs Detection Vs Segmentation Models: The Differences Between Them And When To Use Each," [Online]. Available: https://www.clarifai.com/blog/classification-vs-detection-vs-segmentation-models-the-differences-between-them-and-how-each-impact-your-results.

[2] Apple, "Lose your knack for losing things.," 2020. [Online].

[3] google, "Open Images Dataset V6," [Online]. Available: https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=segmentation&r=false&c=%2Fm%2F080hkjn.

[4] I. S. Aakash K. Shetty, "Object Detection Models," *International Conference for Convergence in Technology (I2CT),* 2021.

[5] C.-Y. W. H.-Y. M. L. Alexey Bochkovskiy, "Optimal Speed and Accuracy of Object Detection," *arXiv,* 2020.

[6] AlexeyAB, "Darknet is an open source neural network framework written in C and CUDA. It is fast, easy to install, and supports CPU and GPU computation.," [Online]. Available: https://github.com/pjreddie/darknet.

[7] be my eyes, "Bringing sight to blind and low-vision people," 2018. [Online].

[8] chipolo, "The finder with a reminder," [Online]. Available: https://chipolo.net/en/.

[9] the trackr, "features," 2019. [Online]. Available: https://www.thetrackr.com/.

[10] YR, "where-is-my-stuff," 2021. [Online]. Available: https://yrw.co.il/where-is-my-stuff/.

[11] @ersheng-ai, "pytorch-YOLOv4," [Online]. Available: https://github.com/Tianxiaomo/pytorch-YOLOv4.