جـامـعـة حـلـوان
كـلـيـة الـحـاسـبـات والـذكـاء الإصـطـنـاعي
قـسـم عـلـوم الـحـاسـب

SMART GLASSES

رسالة مشروع تخرج مقدمة من:

[كريم سيد ابراهيم عبدالتواب (20180428)]

[محمد عاطف محمود محمود (20180514)]

[شهاب شريف محمد السيد (20180292)]

[كيرلس فايق ذكى غبريال (20180439)]

[معاذ إسماعيل محمد إسماعيل (20170549)]

[ميريت ميخائيل نجيب سامى (20180643)]

رسالة مقدمة ضمن متطلبات الحصول على درجة البكالوريوس في الحاسبات والذكاء الإصطناعي، بقسم علوم الحاسب، كلية الحاسبات والذكاء الإصطناعي، جامعة حلوان

تحت إشراف:

(د/هالة عبدالجليل)

يوليو / تموز 2022


SMART
GLASSES

CONTENTS

Abstract

One of the most important problems facing visual impaired and blind people is vision and recognition in our daily lives; for example, 3.5 million Egyptians are blind, and another 5% have visual problems. An individual's eyes are their primary sense organ. A quick glance about us reveals how visual the majority of the information in our surroundings is. Timetables in railway stations, signs showing the correct path or a possible hazard, detection stairs and currency are all examples of visual information that we encounter in our daily lives. Most of this information is inaccessible to the blind and visually handicapped, limiting their liberty.

The main idea in project is a Software of smart glass with multi option Like: object detection, currency detection, text detection and timestamp. we will talk details in later. Our software is helping to detect the environment around you depended on object detection then it once detects convert the result in sound can you hear that and it work in real time, this helps blind people can walk in street and detect three objects Like: person, chairs and stairs around them. In text detection you can shot any text then read and translate the typed text which is written in the English language and convert into Arabic then convert it in sound so the person will listen to the audio through a headphone , These kinds of inventions consider a solution to motivate blind students to complete their education also can help blind people and visually impaired to read anything and also in currency detection, can them detect any Egyptian currency , it can take shot this currency through the camera and detect the currency and convert the result in sound.

We use a pre-trained model “MobileNetV2 + SSD” will be used in the train, and we'll go through it in detail: MobileNetV2 is a powerful feature extractor for detecting and segmenting objects. SSD, on the other hand, is meant to function on top of MobileNetV2 and is not dependent on the base network. For the object identification element of the network, the MobileNetV2 + SSD combo employs a variation dubbed SSDLite, which uses depthwise separable layers instead of conventional convolutions, making it simpler to acquire real-time results. When used in conjunction with SSDLite, the models are around 35% quicker while maintaining the same accuracy as MobileNetV1. I just wanted to mention that MobileNet and SSD are a fantastic match.

We trained the model on two datasets

First: Egyptian currency

Second: objects dataset (person, chairs and stairs).

Achievement

Evaluation of Currency detection –> 95% accuracy

Evaluation of Object detection –> 79% accuracy

من أهم المشاكل التي تواجه ضعاف البصر والمكفوفين هي الرؤية والتعرف على الاشياء في حياتنا اليومية ؛ على سبيل المثال هناك 3.5 مليون مصري مكفوفين و 5٪ آخرون يعانون من مشاكل بصرية. عيون الفرد هي جهاز الإحساس الأساسي. تكشف نظرة سريعة عنا عن مدى وضوح غالبية المعلومات الموجودة في محيطنا. مثل الجداول الزمنية في محطات السكك الحديدية والعلامات التي توضح المسار الصحيح أو الخطر المحتمل وصعود ونزول السلالم والتعرف علي العملات كلها أمثلة على المعلومات المرئية التي نواجهها في حياتنا اليومية. معظم هذه المعلومات لا يمكن الوصول إليها للمكفوفين والمعاقين بصريًا ، مما يحد من حريته.

الفكرة الرئيسية في المشروع هي برنامج النظارة الذكية مع خيارات متعددة مثل: الكشف عن الأشياء ، والكشف عن العملات ، وقراءة وترجمة النص . سنتحدث في التفاصيل في وقت لاحق. يساعد برنامجنا في اكتشاف البيئة من حولك اعتمادًا على اكتشاف الأشياء ، ثم يكتشف بمجرد تحويل النتيجة إلى صوت و تسمع ذلك ويعمل في الوقت الفعلي من خلال سماعة الاذن ، وهذا يساعد المكفوفين على المشي في الشارع واكتشاف ثلاثة أشياء مثل: الأشخاص و الكراسي والسلالم من حولهم و قراءة النصوص ، يمكنك تصوير أي نص ثم قراءة وترجمة النص المكتوب باللغة الإنجليزية وتحويله إلى اللغة العربية ثم تحويله إلى صوت حتى يستمع الشخص إلى الصوت من خلال سماعة رأس ، وتعتبر هذه الأنواع من الاختراعات حلاً لتحفيز الطلاب المكفوفين وضعاف النظر على إكمال تعليمهم أيضًا يمكن أن يساعد المكفوفين وضعاف البصر على قراءة أي شيء وأيضًا في الكشف عن العملات الورقية المصرية ويمكن التقاط هذه العملة من خلال الكاميرا.

نستخدم نموذجًا مُدربًا مسبقًا "MobileNetV2 + SSD" سيتم استخدامه من خلال التدريب، وسنتناوله بالتفصيل MobileNetV2 :هو مستخرج ميزة قوي للكشف عن الكائنات وتقسيمها. من ناحية أخرى من المفترض أن يعمل SSD أعلى MobileNetV2 ولا يعتمد على الشبكة الأساسية. بالنسبة لعنصر تعريف الكائن في الشبكة ، فإن مجموعة MobileNetV2 + SSD تستخدم نوعًا مختلفًا يُدعى SSDLite ، والذي يستخدم طبقات قابلة للفصل في العمق بدلاً من التلافيف التقليدية ، مما يجعل الحصول على نتائج في الوقت الفعلي أسهل. عند استخدامها مع SSDLite ، تكون النماذج أسرع بحوالي 35٪ مع الحفاظ على نفس الدقة مثل MobileNetV1. أردت فقط أن أذكر أن MobileNet و SSD هما تطابقان رائعان.

قمنا بتدريب النموذج على مجموعتي بيانات

أولاً: العملات المصرية

ثانيًا: التعرف والكشف علي الاشياء (شخص ، كراسي ، سلالم).

وحصلنا على النتائج التالية

- الكشف على العملات المصرية بنسبة 95٪ - التعرف والكشف على الأشياء بنسبة 79٪

Chapter 1: Introduction

This project is designed to help to **Detect Object** in real-time, **Currency Detection** And **Text Detection.**
object detection at first the object detection is work in real time and the output is a sound coming out of a person, then the person gives a voice command to model to make currency detection and at once the camera take photo and detect it and the output is a sound coming out of a person, Or the person gives a voice command to model to make text detection and at once the camera take photo of text and detect it and the output is a sound coming out of a person,

---

1.1 Problem definition

In our lives, there are many people who are suffering from different diseases or handicaps, there are 3.5 million blind Egyptians is blind and another 5% have vision difficulties. The major sensory organ of a person is their eyes. One glimpse around us is enough to make us realize how visual most of the information in our environment is. Timetables in train stations, signs indicating the right way or potential danger, a billboard advertising a new product in the market, these are all the visual types of information we all come across in our daily life. Most of this information is inaccessible for the blind and the visually impaired, inhibiting their independence, since access to information signifies autonomy. It's very common for sighted individuals, strangers, friends or family, to be overly excited to help a visually impaired person. Very frequently, this behavior holds the assumption that the blind or low vision individual requires assistance, although this might not reflect reality. Blind people might perform a regular task slower but that doesn't mean they're incapable of completing it. Rushing to help the visually impaired without asking or being asked to do so, might make them feel helpless instead of independent. Moreover, not allowing a visually impaired individual perform a task by themselves, does not give them the room to learn how to do so independently.

So these people need some help to make their life easier and better without helping of another people.

1.2 Objective

These **"Smart Glasses"** are designed to help the blind people and visually impaired to read and translate the typed text which is written in the English language and Arabic language and convert into voice. These kinds of inventions consider a solution to motivate blind and visually impaired students to complete their education despite all their difficulties. Its main objective is to develop a new way of reading texts for blind people and facilitate their communication. The main task of the glasses is to **Detect Object** front of the camera and tell the person about objects. The person will listen to the audio through a headphone that's connected to the glasses. The second task of the glasses is to **Detect Text** image and convert it into audio text. Also, can take audio order by the microphone. The third task is to translate the whole text or some words. Also, it can **Detect Egyptian Currency**.

1.3 Related works

The We-WALK

The blind and visually impaired are now able to lead more independent lives than ever. The We-WALK Smart Cane is a great example of what is now possible. The We WALK looks like the cane that some blind and visually impaired people have used for decades to avoid obstacles while walking, but it incorporates a few modern twists.
With a standard cane, you can still run into obstacles that are not immediately underfoot, like poles, tree branches, and barriers. The We-WALK, however, detects objects above chest level and audibly alerts you if you're getting too close, which can save you from a painful fall.

The Back-Pack

The computer sees through an AI camera that can be embedded in a vest or fanny pack to view the wearer's surroundings. It then works with the laptop or computer in the backpack to notify the wearer of signs or obstacles via Bluetooth earbuds. The spatial camera, by Luxonis, can read signs, detect crosswalks, see changes in elevation, and detect potential obstacles. The Bluetooth earphones connect the wearer to the computer, alerting them of said obstacles or hazards. Also, the user can talk to the computer via the earphones. Because of the built-in GPS, the computer can even provide the wearer with location information.

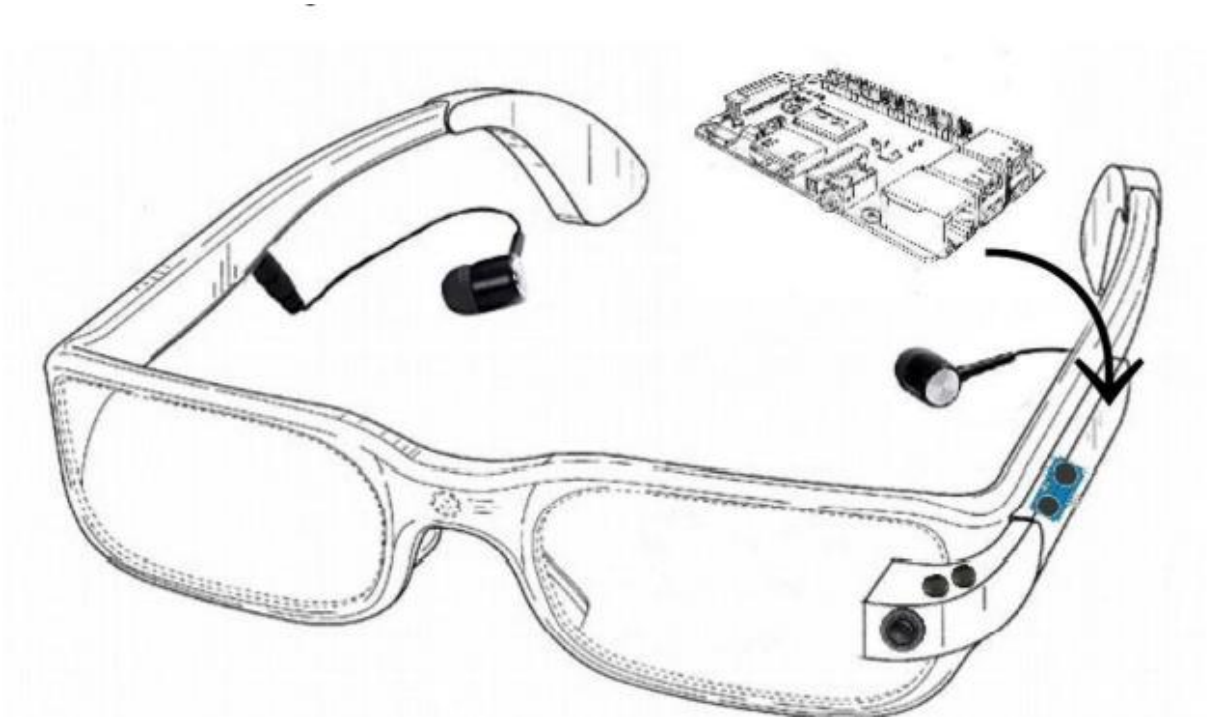
1.5 Extra Features

We can add some feature first, blind navigation with google maps but the cost of API is expensive for us. Second, making app to be easy to control the mobile phone by sound order like answer calls, read massages, search in google, make sound notes and use radio to listen to music.

Need to improve

However, these limitations can be improved in the future, for example, the glasses can support different languages and more Banknotes, and it also can be smaller and easy to wear. We can make a connection with a mobile by app or more smart devises.

Initial Design of Smart glasses and smaller.



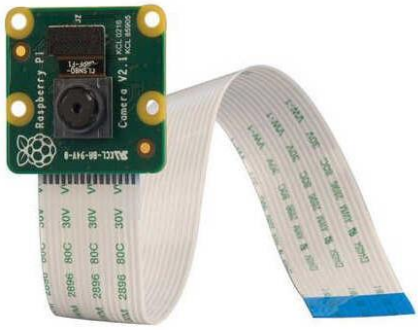
Architecture and Components:

Raspberry Pi Model B

Function:

Raspberry Pi 3 is used for many purposes such as education, coding, and building hardware projects. It is the main component of the project. It is used as a low-cost embedded system to control and connect all of the components together. It uses Raspbian as the operating system which can accomplish many important tasks.

Webcam (Pi camera)



Function:

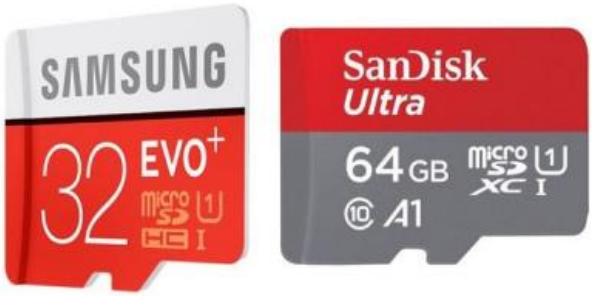
The Webcam will be used the eyes of the person who wears the Smart Glasses, The camera is going to capture a picture when the button is pressed, in order to detect and recognize the text or object from the image.

Headphones

Function:

The headphones will be used to help the user listen to the text that is been converted to audio after it is been captured by the camera or to listen to the translation of the text.

SD card (64GB)



Function:

Installed operating system + store any information.

Chapter 2: Technology & Tools

2.1 Software

2.1.1 Object Detection

Object detection is a computer vision technique, and this is a software system can detect, locate, and trace the object from a given image or video. The special probity about object detection is that it identifies the class of object (person, Chair , and stairs) and their location-specific coordinates in the given image. The location is pointed out by drawing a bounding box around the object. The bounding box may or may not accurately locate the position of the object. The ability to locate the object inside an image defines the accuracy of the algorithm used for detection. Face detection is one of the examples of object detection. These object detection algorithms might be pre-trained or can be trained from scratch. In most, we use pre-trained weights from pre-trained models and then fine-tune them as per our requirements and different use cases.

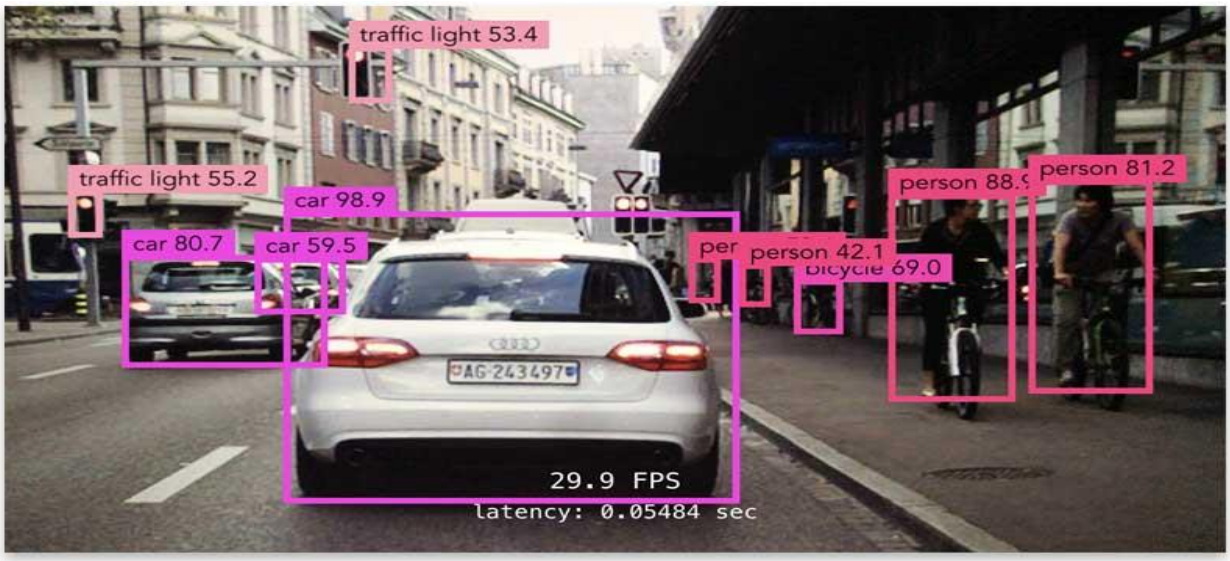
Two types of Object Detection:

One-stage object detection

Two-stage object detection

One-stage object detection:

Object detection is the computer vision technique for finding objects of an image:



This is more advanced than classification, tells us what the "main subject" of the image is whereas object detection can find multiple objects, classify them, and locate where they are in the image.

An object detection model predicts bounding boxes, one for each object found, as well as classification probabilities for each object.

It's common for object detection to predict a lot of bounding boxes. Each box also has a confidence score that says how likely the model thinks this box really contains an object. As a post-processing step, we filter out the boxes whose score falls below a sure threshold (also called non-maximum suppression)

One of the problems you'll encounter is that a training image can have anywhere from zero to dozens of objects in them, and the model may output more than one prediction, so how do you figure out which prediction should be compared to which ground-truth bounding box in the loss function?

When you work with mobile, you will be interested in one-stage object detectors.

The most common examples of one-stage object detectors are YOLO, SSD, SqueezeDet, and DetectNet.

Difference between YOLO and SSD

There are two types of deep neural networks here. Base network and detection network. SSD, RCNN, Faster RCNN are examples of detection networks. All YOLO networks are executed in the Darknet, which is an open-source ANN library written in C. The key difference between the two architectures is that the YOLO architecture utilizes 2 fully connected layers, whereas the SSD network uses convolutional layers of varying sizes.

You only Look Once (YOLO)

For YOLO, detection is a straightforward regression dilemma which takes an input image and learns the class possibilities with bounding box coordinates. YOLO divides every image into a grid of S x S and every grid predicts N bounding boxes and confidence. The confidence reflects the precision of the bounding box and whether the bounding box in fact contains an object despite the defined class. YOLO even forecasts the classification score for every box for each class. You can merge both the classes to work out the chance of every class being in attendance in a predicted box. So, total (SxSxN) boxes are forecasted. On the other hand, most of these boxes have lower confidence scores and if we set a doorstep say 30% confidence, we can get rid of most of them.

Single Shot Detector (SSD)

SSD attains a better balance between swiftness and precision. SSD runs a convolutional network on input image only one time and computes a feature map. Now, we run a small 3×3 sized convolutional kernel on this feature map to foresee the bounding boxes and categorization probability. SSD also uses anchor boxes at a variety of aspect ratio comparable to Faster-RCNN and learns the off-set to a certain extent than learning the box. In order to hold the scale, SSD predicts bounding boxes after multiple convolutional layers. Since every convolutional layer function at a diverse scale, it is able to detect objects of a mixture of scales.

SSD VS YOLO

SSD	YOLO
Single Shot Detector	You Only Look Once
Runs a convolutional network on input images at just one time and computes a feature map.	the open-source technique of object detection which will acknowledge objects in pictures and videos fleetly
SSD could be a higher choice as we tend to square measure able to run it on a video and therefore the truth trade-off is extremely modest.	YOLO is a better option when exactness is not too much of disquiet, but you want to go super quick
When the object size is tiny, the performance dips a touch	YOLO could be a higher choice even when the object size is small.
runs a convolutional network on input image just one time and computes a feature map	can be enforced for applications as well as artificial intelligence, self-driving cars, and cancer recognition approaches
In SSD the precision more than recall	In YOLO the recall more than precision

The problem with YOLO on mobile is that, while the actual detection portion of the neural network is simple and fast, the feature extractor (Darknet-19) uses regular convolutional layers. Running YOLO on an iPhone only gets you about 10 – 15 FPS. YOLO would be much faster if it was running on top of MobileNet instead of the Dark-net feature extractor.

SSD is designed to be independent of the base network, and so it can run on top of pretty much anything, including MobileNet. Even better, MobileNet+SSD uses a variant called SSDLite that uses depthwise separable layers instead of regular convolutions for the object detection portion of the network. With SSDLite on top of MobileNet, you can easily get truly real-time results (i.e., 30 FPS or more).

A two-stage detector, high-end model like Faster R-CNN first generates so-called region proposals areas of the image that potentially contain an object and then it makes a separate prediction for each of these regions. That works well, but it's also quite slow as it requires running the detection and classification portion of the model multiple times.

How these one-stage detectors work?

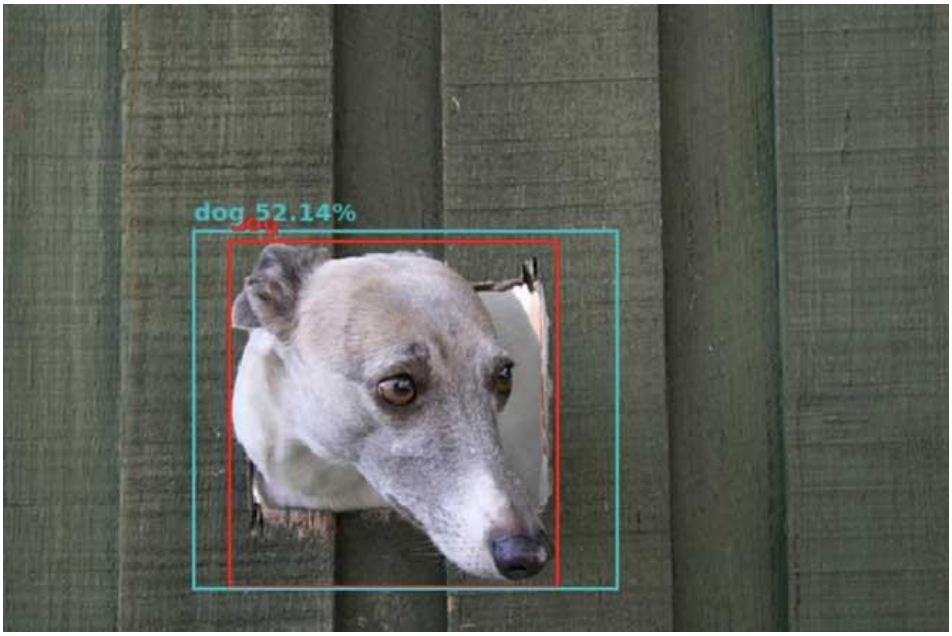
One-stage detectors Steps

The model has two outputs:

1- The probability distribution for the classification result
2- A bounding box regression.

The loss function for the model adds the regression loss for the bounding box to the cross-entropy loss for the classification, with the mean squared error (MSE)

Now you use SGD to optimize the model, with this combined loss, and it works well. Here is an example prediction:

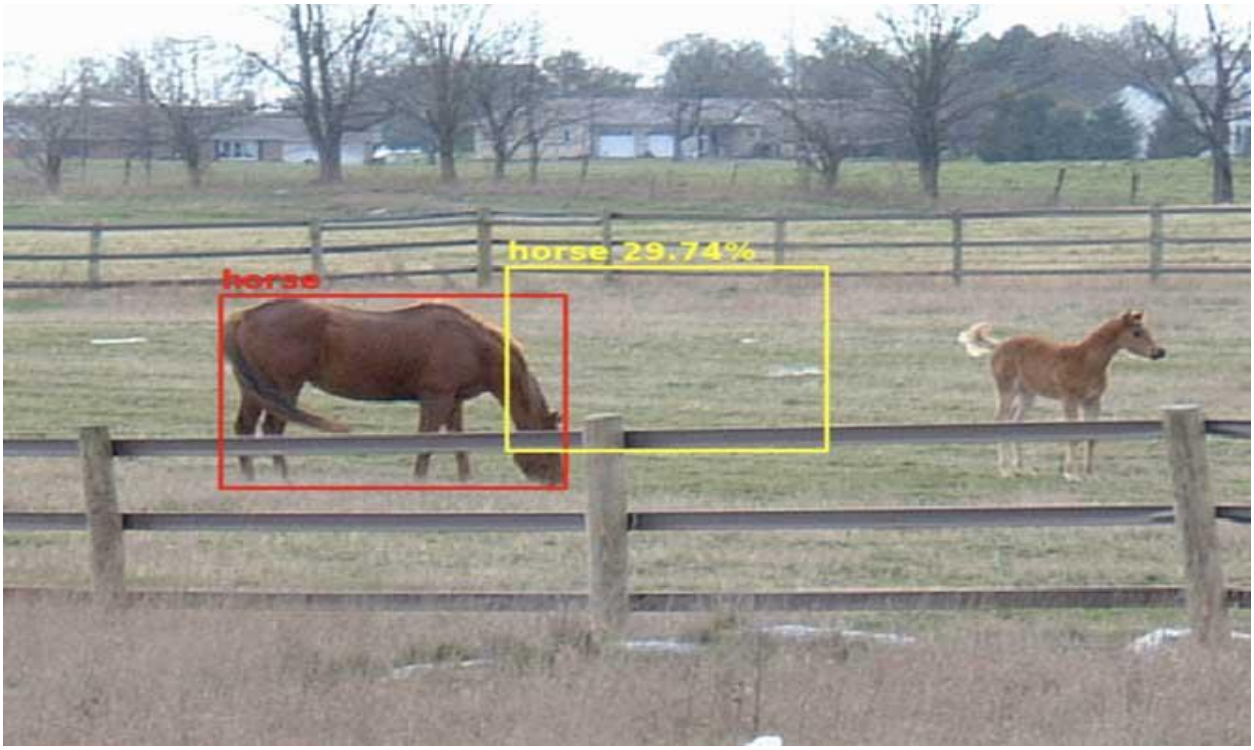


The model has correctly found the class for this object (dog) and its location in the image. The red box is the ground-truth while the cyan box is the prediction. It's not a perfect match, but close.

Note: The score of 52.14% given here is a combination of the class score, which was 82.16% dog, and a confidence score of how likely it is that the bounding box really contains an object, which was 63.47%.

To score the predicted box matches the ground-truth we can compute the IOU (intersection-over-union, also known as the Jaccard index) between the two bounding boxes. The IOU is a number between 0 and 1, with larger being better. Ideally, the predicted box and the ground-truth have an IOU of 100% but in practice anything over 50% is usually considered to be a correct prediction. For the above example the IOU is 74.9% and you can see the boxes are a good match.

Using a regression output to predict a single bounding box gives good results. However, just like classification doesn't work so well when there's more than one object of interest in the image, so fails this simple localization scheme:



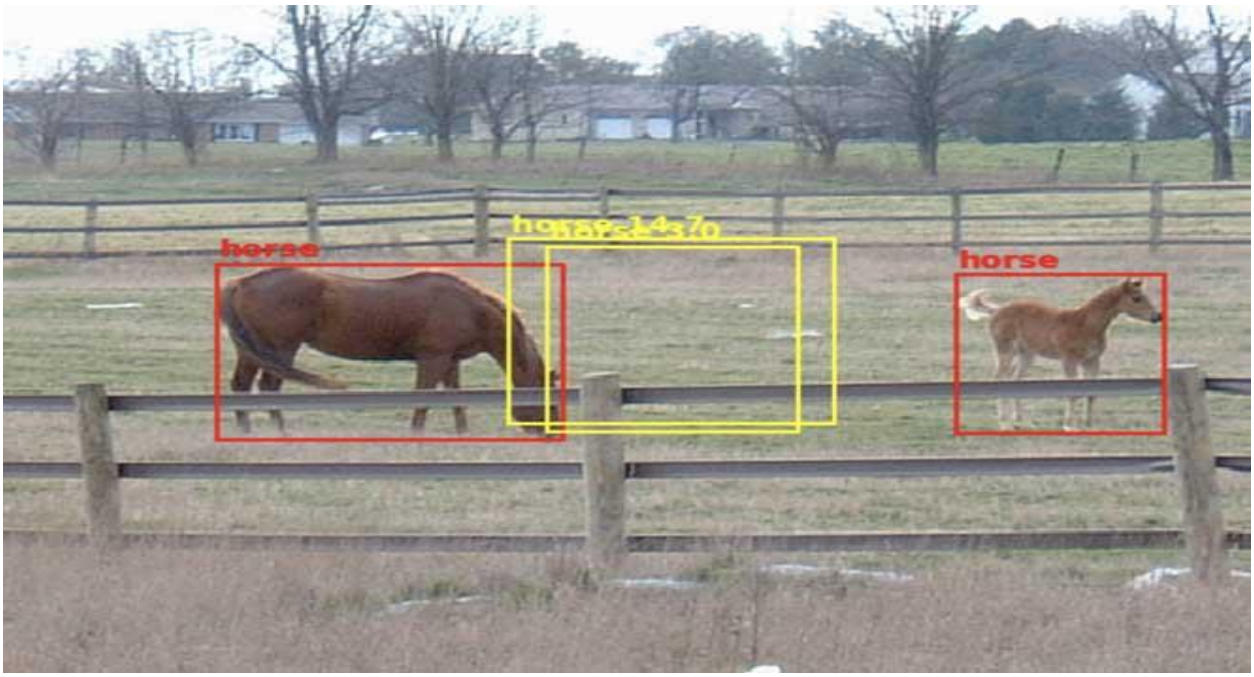
The model can predict only one bounding box and so it must choose one of the objects, but instead the box ends up somewhere in the middle. What happens here makes perfect sense: the model knows there are two objects, but it has only one

bounding box to give away, so it compromises and puts the predicted box in between the two horses. The size of the box is also halfway between the sizes of the two horses.

Note: You might expect that the model would now draw the box around both objects, but that doesn't happen because it has not been trained to do so. The ground-truth annotations in the dataset are always drawn around individual objects, not around groups of objects.

You think this is good when add more bounding box detectors by giving the model additional regression outputs. if the model can predict N bounding boxes, then it should be able find up to N objects.

Even with a model that has multiple of these detectors, we still get bounding boxes that all end up in the middle of the image like this:



The problem is that the model doesn't know which bounding box it should assign to which object, and to be safe it puts them both somewhere **in the middle.**

The model has no way to decide, I can put bounding box 1 around the horse on the left, and bounding box 2 around the horse on the right.

Each detector still tries to predict all the objects rather than just one of them. Even the model has N detectors, they don't work together as a team. A model with

multiple bounding box detectors still behaves exactly like the model that predicted only one bounding box.

We need make bounding box detectors specialize, so that each detector will try to predict only a single object, and different detectors will find different objects.

In a model that doesn't specialize, each detector is supposed to be able to handle every possible kind of object at any possible location in the image. This is simply too much to ask, the model is learned to predict boxes that are always in the center of the image because over the entire training set that minimizes the number of mistakes it makes.

On view of SGD, doing good result on average, but it's also not really a useful result in practice, and so we need to be smarter about how we train the model.

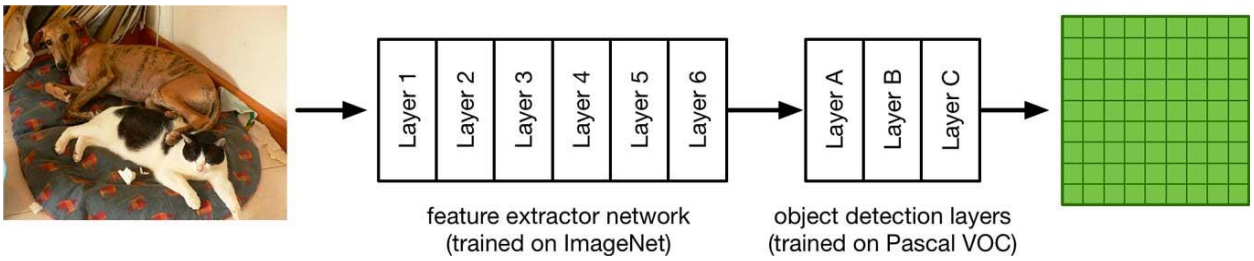
One-stage detectors such as YOLO, SSD, and DetectNet all solve this problem by assigning each bounding box detector to a specific position in the image. That way the detectors learn to specialize on objects in certain locations. For even better results, we can let detectors specialize on the shapes and sizes of objects.

Enter the grid

Using a fixed grid of detectors is the main idea that powers **one-stage detectors**, and what sets them apart from region proposal-based detectors such as **R-CNN.**

We consider the simplest architecture for this kind of model. It exists of a base network that acts as a feature extractor. Like most feature extractors, it's typically trained on ImageNet.

In the case of YOLO, the feature extractor takes in a 416×416-pixel image as input. SSD typically uses 300×300 images. These are larger than images for classification (typically something like 224×224) because we don't want small details to get lost.



The base network can be anything, such as Inception or ResNet or YOLO's DarkNet, but on mobile it makes sense to use a small, fast architecture such as SqueezeNet or MobileNet.

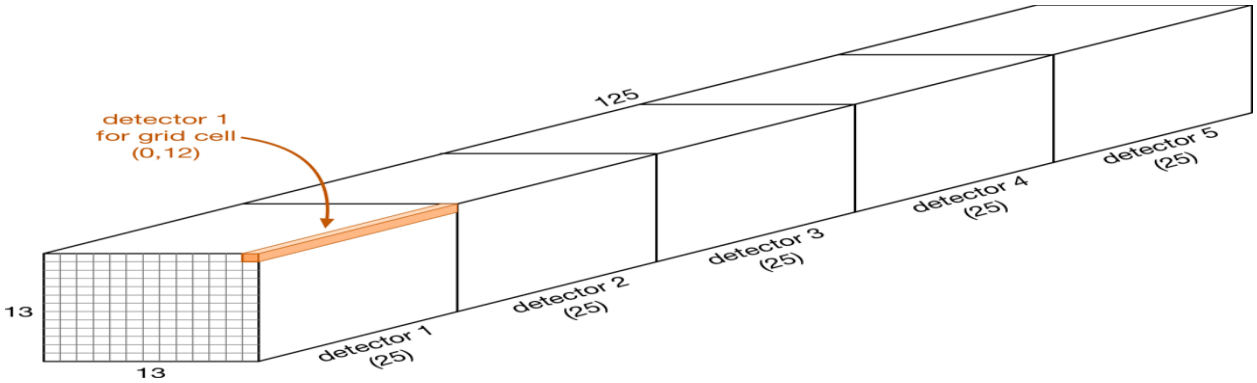
On top of the feature extractor are several additional convolutional layers. These are fine-tuned to learn how to predict bounding boxes and class probabilities for the objects inside these bounding boxes. This is the object detection part of the model..

When look at the architectures of YOLO or SSD, they're a little more complicated than this simple example network, the above model will suffice, and can be used to build a fast and fairly accurate object detector. (It is very similar to the "Tiny YOLO" architecture that I've written about before.)

The output of the final layer is a feature map (the green thing in the illustration above). For our example model this is a 13×13 feature map with 125 channels.

Note: The number 13 here comes from the input size of 416 pixels and the fact that the base network used here has five pooling layers (or convolution layers with stride 2), which together scale down the input by a factor of 32, and 416 / 32 = 13. If you want a finer grid, for example 19×19, then the input image should be 19 × 32 = 608 pixels wide and tall (or you can use a network with a smaller stride).

We interpret this feature map as being a grid of 13 by 13 cells. The number is odd so that there is a single cell in the center. Each cell in the grid has 5 independent object detectors, and each of these detectors predicts a single bounding box.



The key thing that the position of a detector is fixed: it can only detect objects located near that cell (in fact, the object's center must be inside the grid cell). This is what lets us avoid the problem from the previous section, where detectors had too much freedom. With this grid, a detector on the left-hand side of the image will never predict an object that is located on the right-hand side.

Each object detector produces 25 numbers:

20 numbers containing the class probabilities

4 bounding box coordinates (center x, center y, width, height)

1 confidence score

Since there a 5 detectors per cell, and 5 × 25 = 125, that's why we have 125 output channels.

Like regular classifier, the 20 numbers for the class probabilities have had a softmax applied to them. We can find the winning class by looking at the highest number. (Although it's also common to treat this as multi-label classification, in which case the 20 classes are independent, and we use sigmoid instead of soft-max)

The confidence score is a number between 0 and 1 (or 100%) and describes how the model thinks this predicted bounding box contains a real object. Note that this score only says something about whether this is an object but says nothing about what kind of object that's what the class probabilities are for.

This model always predicts the same number of bounding boxes: 13×13 cells times 5 detectors give 845 predictions. Obviously, most of these predictions will be no good at all, most images only contain a handful of objects at most, not over 800. The confidence score tells us which predicted boxes we can ignore.

Typically, we'll end up with a dozen or so predictions that the model thinks are good. Some of these will overlap this happens because nearby cells may all make a prediction for the same object, and sometimes a single cell will make multiple predictions

Largely overlapping predictions is common with object detection. The standard postprocessing technique is to apply non-maximum suppression (NMS) to remove such duplicates. In short, NMS keeps the predictions with the highest confidence scores and removes any other boxes that overlap these by more than a certain threshold (say 60%).

We'll only keep the 10 or so best predictions and discard the others. Ideally, we want to have only a single bounding box for each object in the image.

There is two types of data:

1- Pascal(VOC)
2- YOLO

The type of our data

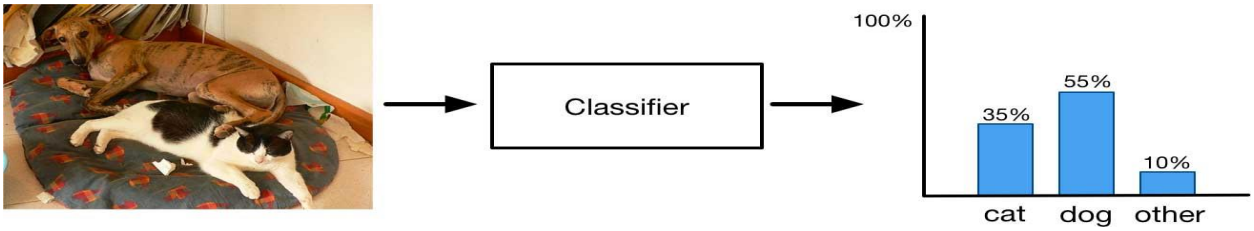
The Pascal VOC.

Object detection VS Image recognition:

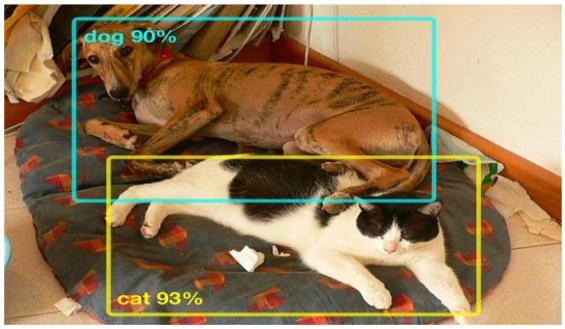
Object detection is commonly confused with image recognition, so before we proceed, it's important that we clarify the distinctions between them.
But there are important differences. Image recognition only outputs a class label for an identified object, and image segmentation creates a pixel-level understanding of a scene's elements. What separates object detection from these other tasks is its unique ability to locate objects within an image or video. This then allows us to count and then track those objects.

Object detection VS Classification:

A classifier takes an image as input and produces a single output, the probability on the classes. But this only gives you a summary of what is in the image, it doesn't work so well when the image has multiple objects of interest on the following image, a classifier might recognize that the image contains a certain number of cats and certain number of dogs but that's the best it can do.

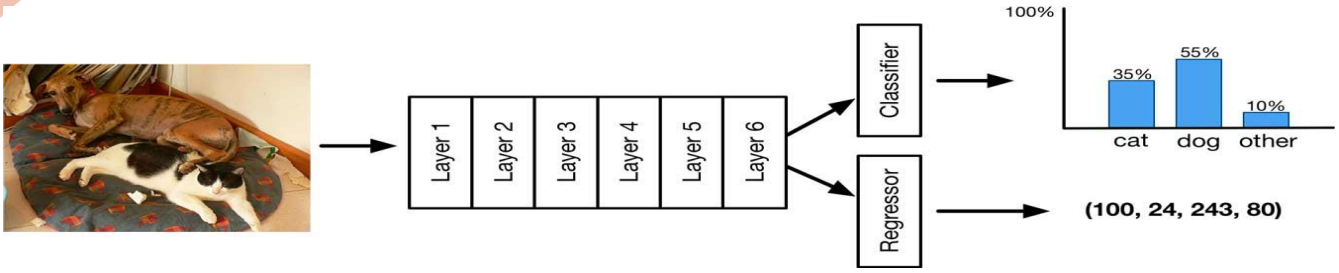


An object detection model, will tell you where the individual objects are by predicting a bounding box for each object:



Once it can now focus on classifying the objects inside the bounding box and ignore everything outside, the model is able to give much more confident score predictions for the individual objects.

If your dataset with bounding box annotations (so called ground truth boxes), it's easy to add a localization output to your model. Simply predict an additional 4 numbers, one for each corner of the bounding box.



2.1.2 Text Detection and Extraction using Open-CV and OCR

Open-CV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. Open-CV in python helps to process an image and apply various functions like resizing image, pixel manipulations, object detection, etc. In this article, we will learn how to use contours to detect the text in an image and save it to a text file.

We use Open-CV package to read an image and perform certain image processing techniques 10 seconds after opening the camera.

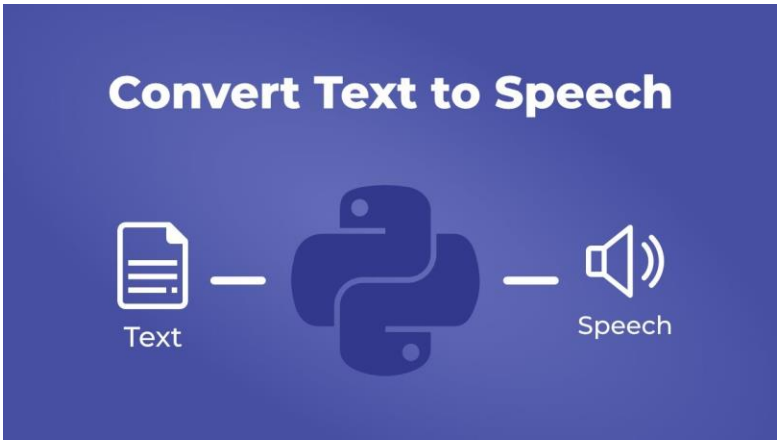
Optical Character Recognition (OCR)

OCR, or Optical Character Recognition, is a process of recognizing text inside images and converting it into an electronic form. These images could be of handwritten text, printed text like documents, receipts, name cards, etc., or even a natural scene photograph.

OCR has two parts to it. The first part is text detection where the textual part within the image is determined. This localization of text within the image is important for the second part of OCR, text recognition, where the text is extracted from the image. Using these techniques together is how you can extract text from any image.

After extracting the text from the image**,** we must translate this text, whatever the language, into Arabic using Google Translation.

The last step is to convert text into audio using gTTS (Google Text-to-Speech).



2.2 Hardware

2.2.1 Model of Glasses

2.2.2 Headphone

Chapter 3: Requirement & Analysis

3.1/3.2: Function and non-Function

Functional specification	Non-functional specification
Text detection: when take audio, The glasses will capture an image and convert the text in the image into audio text.	**Performance**: the glasses must be able to perform properly and extract the text from any image and work in real time
Object detection: work in real time and **Detect objects** in front the glasses	**Reliability**: the glasses must achieve high accuracy in identifying the text in the image.
Currency detection: detect **Egyptian Currency** in front the glasses	**Scalability**: The glasses are designed to take a clear picture for any text and object.
Speech to text: take the voice command and convert to text as a input	**Flexibility**: The glasses can be used by all people and in different places such as college, school, and hospital and even in the streets.
	Usability: The software are light, safe and easily wearable.

Chapter 4: coding description and technology

4.1.1: Object Detection (Real-Time)

1- Load The Model

We have **3** files as inputs, and they are: (Label Map - Pipe Line Config – Check Point).

2- Tensor Flow (tf) in object detection

We used this function to take one frame from real time stream to detect object in frame.

3- Open CV in object detection

We used Open-CV to read frames from the real-time video stream, and we put our video capture as **1** to capture from a mobile phone; **because** the quality of the phone is higher than the quality of the laptop camera.

4- Object detection Function

Real-time object detection displayed **3** detections and they are:

1- Detection Boxes
   - Put a box around the object

2- Detection Classes
   - Like (Person – Chair – Stairs).

3- Detection Scores
   - Example: the model detects the person by 80%.
   - And the minimum score to detect the object is 0.7

5- text to speech

We convert the class name (Text) of the object to voice so that the blind can hear, we used a library named win32com in our model

6- Object voice repetition

In each frame, if the model sees one object, the sound does not repeat until after the change of the frame and the appearance of a new object.

4.1.2: Text Detection

1- We used Open CV to take an image within 10 seconds and we used a library named time to count the 10 sec, and we saved the image named “test2.jpg”
2- Using Arabic Ocr library we can take the text in the image
3- If our text is in English we use google translate library to translate this text from English to Arabic
4- We used **gtts** (google text-to-speech) library to convert the text to voice (mp3) file, and play this mp3 using a library named play sound and remove mp3 file from a library named os as shown

4.2 Threads

What is the thread in python?

A thread is a simple procedure or job. One technique to introduce concurrency to your applications is to use threads. Even though your Python program uses several threads, if you look at the processes running on your operating system, you will only see a single entry for your script.

What is the multi thread (Threading) in python?

So that one longer-running process does not block all the others, the software establishes many threads with execution cycling among them. This works well for activities that can be split down into smaller subtasks, each of which may be assigned to a different thread to execute.

Some functions in thread uses our project

Start ()

- No parameters
- Uses to start thread

Join ()

- No parameters
- Uses to stop any threads running until Owen thread to finish then kill the Owen thread

Thread uses in our project

Main thread is speech recognition and
Create 4 threads

1. Time
2. Text Detection
3. Currency Detection
4. Object Detection

For object detection thread create your Owen flag turn **on & off**
and create your Owen thread function take the **stop** parameter (your Owen flag turn **on & off**)
and remainder threads create your Owen thread function no take parameter ()

```
8       stop_thread_objectDetection = True
9
10      speak = wincl.Dispatch("SAPI.SpVoice")
11      imageCam = ImageCam()
12
13  >   def thread_objectDetection(stop):···
16
17  >   def thread_currncydetection(self):···
22
23  >   def thread_textdetection(self):···
28
29  >   def thread_time(self):···
31
```

Explain the main thread (speech recognition)

Real time running Object Detection
In real time take the voice and cheek if Time, Text, Currency

Explain cheek if “Currency” for example:

Create the Currency thread object ➔ using **Start ()** method ➔ using **Join ()** method

```
28          if words == "currency":
29              thread.stop_thread_objectDetection = False
30              currncy = threading.Thread(target=thread.thread_currncydetection(self=Threading()), )
31              currncy.start()
32              currncy.join()
33              print("currncy detection")
34              thread.stop_thread_objectDetection = True
35              object = threading.Thread(target=thread.thread_objectDetection,
36                                        args=(lambda: thread.stop_thread_objectDetection,))
37              object.start()
38              continue
```

After the end of any thread

Create the object detection thread ➔ using **Start ()** method (Real-time thread).

4.3 problem of connecting each feature with other

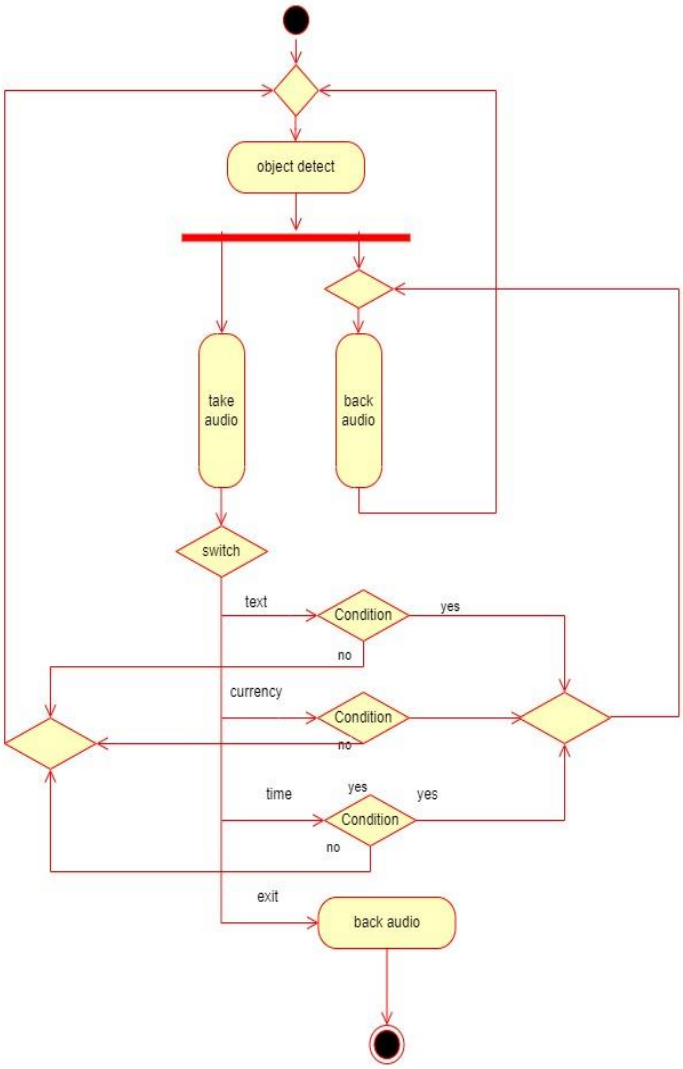
We want to connect all our features with each other with voice commands for example when we say “currency detection” the system must stop running object detection and open the class of currency and after it is ending its work the system must reopen object detection once again and so on for all features.

So we take the idea of threading to connect all of this features, we apply on feature for one thread and we control all the system by threading.

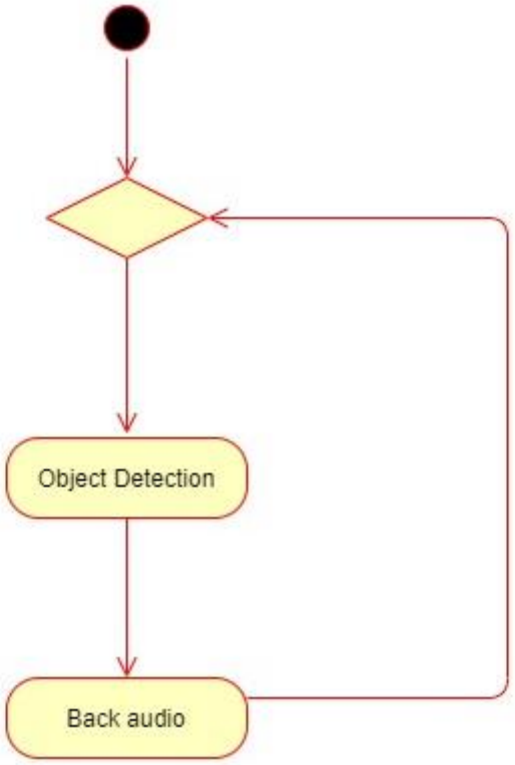
At the beginning the thread that it`s responsibility the object detection is running and all threads of all other features are idle, After we say “currency detection” the thread stop working and the thread that it`s responsibility the currency detection is running and after it finish its work it return idle again and the thread of object detection will reopen again.

Chapter 5: System Design

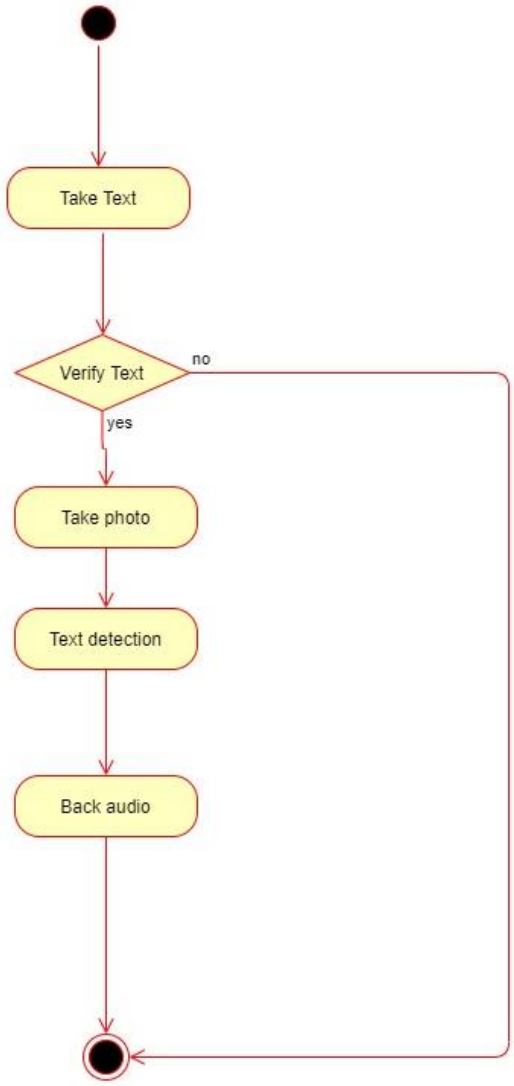
5.1 Activity diagram



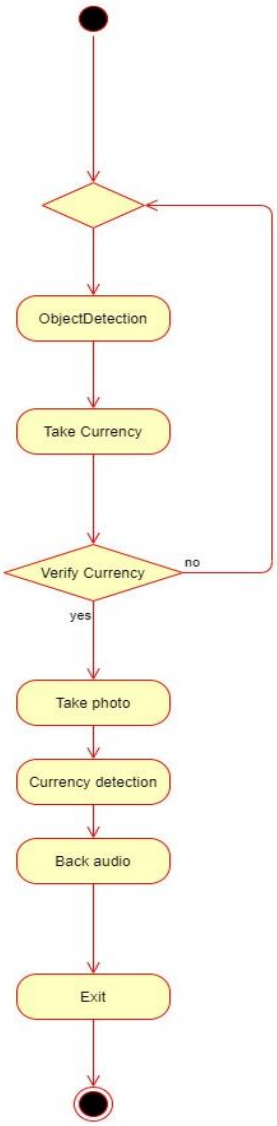
5.1.1 Object detection



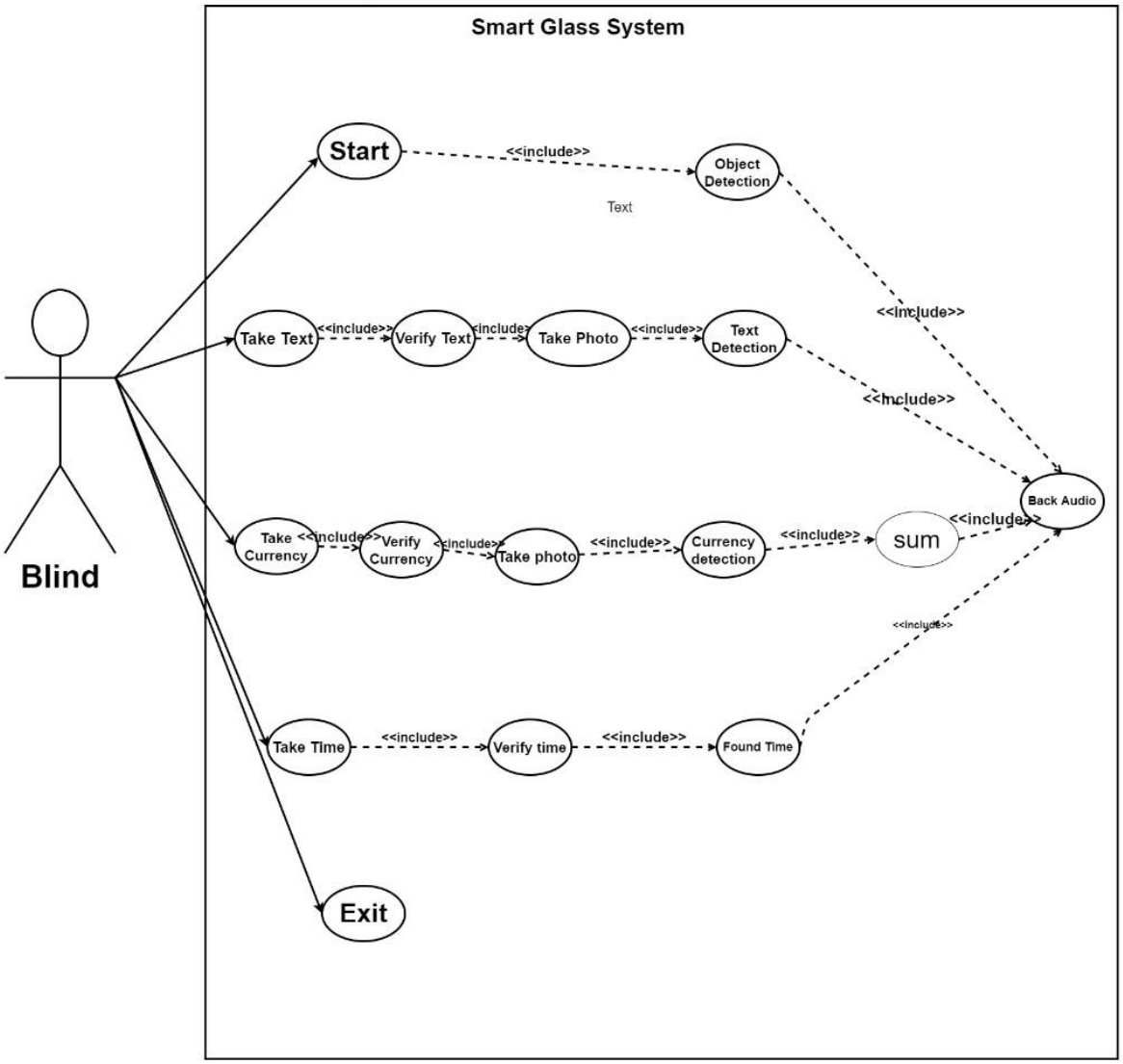
5.1.2 Text detection



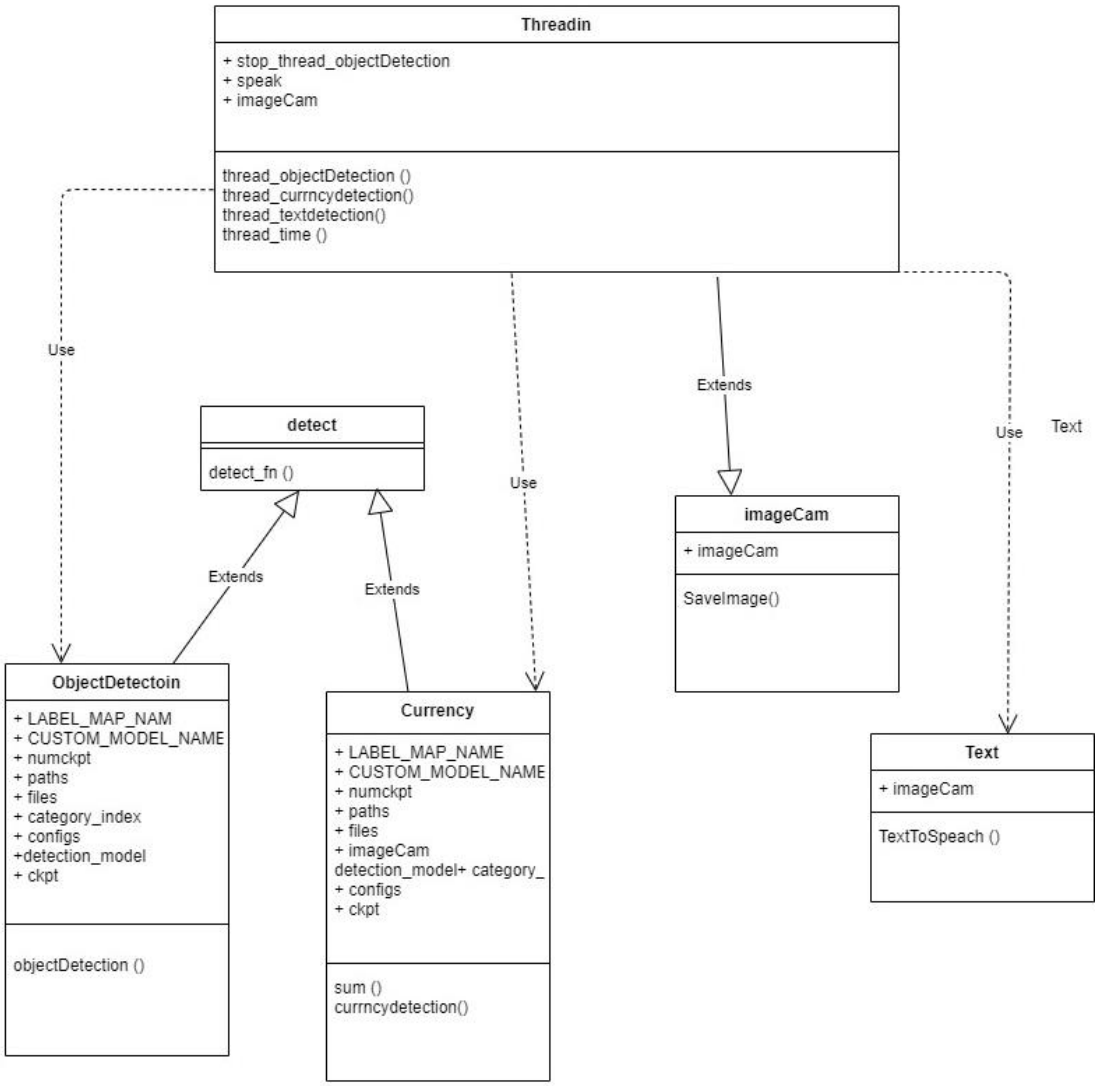
5.1.3 Currency detection



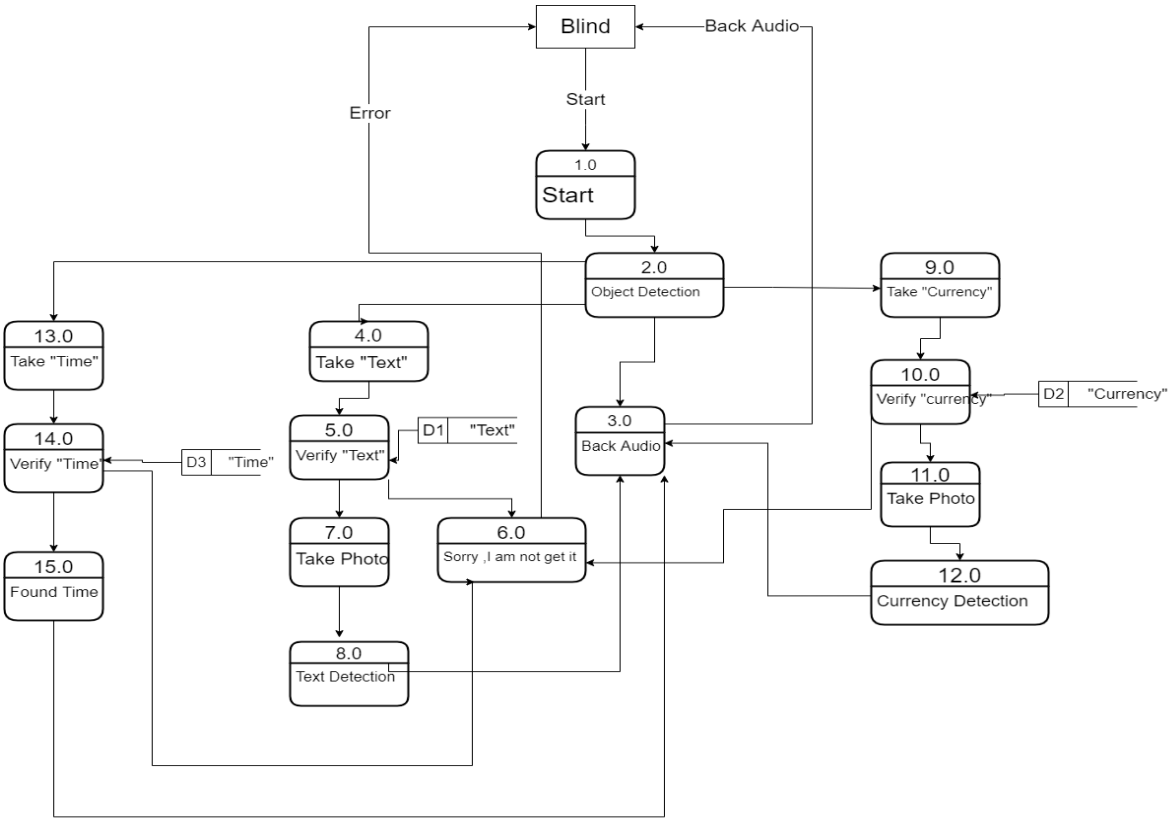
5.2 Use case


Smart Glass System
Start
<<include>>
Object Detection
Take Text
<<include>>
Verify Text
<<include>>
Take Photo
<<include>>
Text Detection
<<include>>
<<include>>
Back Audio
Take Currency
<<include>>
Verify Currency
<<include>>
Take photo
<<include>>
Currency detection
<<include>>
sum
<<include>>
Take Time
<<include>>
Verify time
<<include>>
Found Time
<<include>>
Exit
Blind

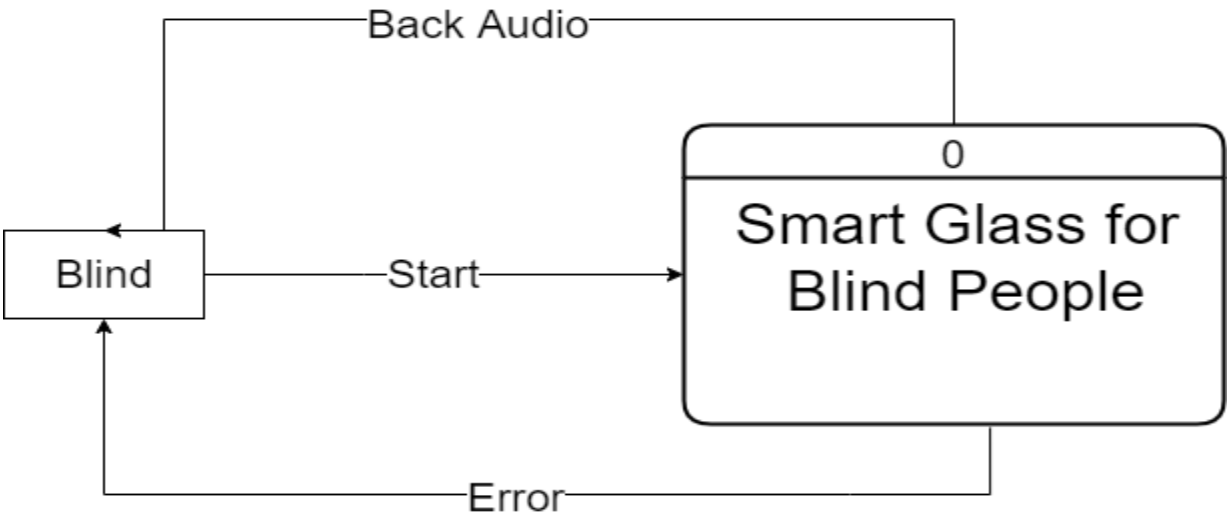
5.4 Class Diagram



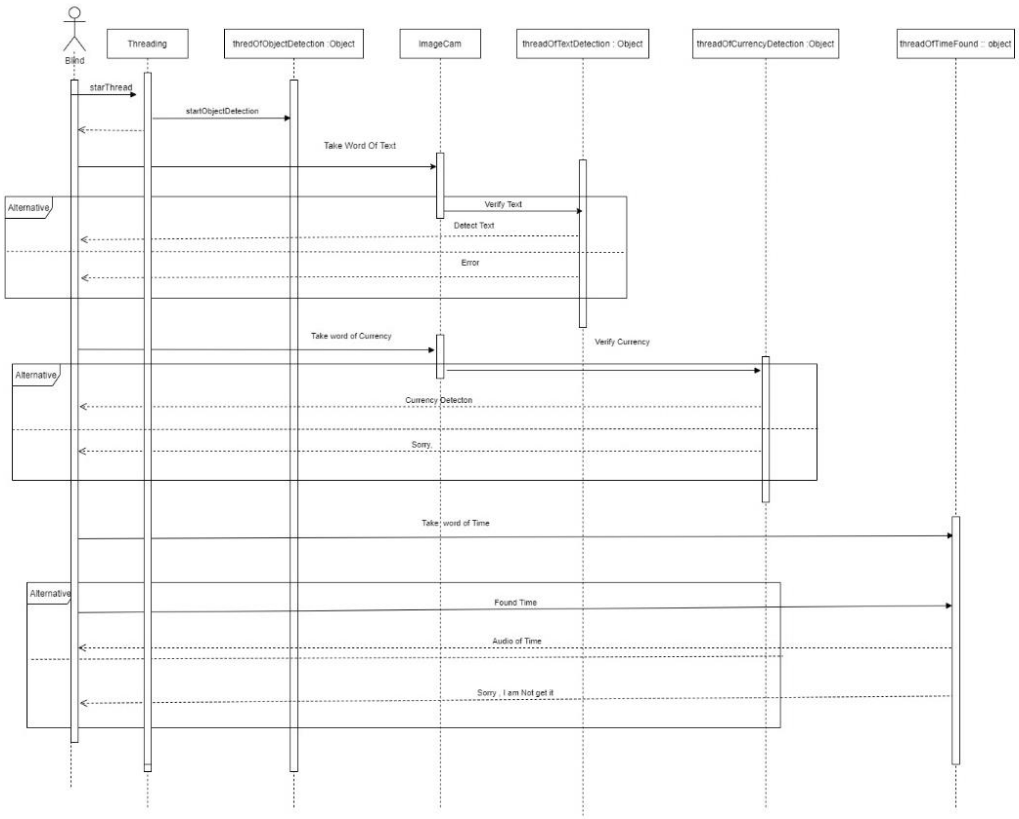
5.5 Data Flow diagram



5.6 Context diagram



5.7 Sequence diagram



Chapter 6: System implementation & Results

In this chapter we will support our paper with snapshots of implementation and snapshots of results.

6.1 Model

We Use MobileNetV2 + SSDLite with Core ML to train the data

MobileNetV2 + SSDLite with Core ML

MobileNetV2 + SSD is model will be used in train and will talk it in details:

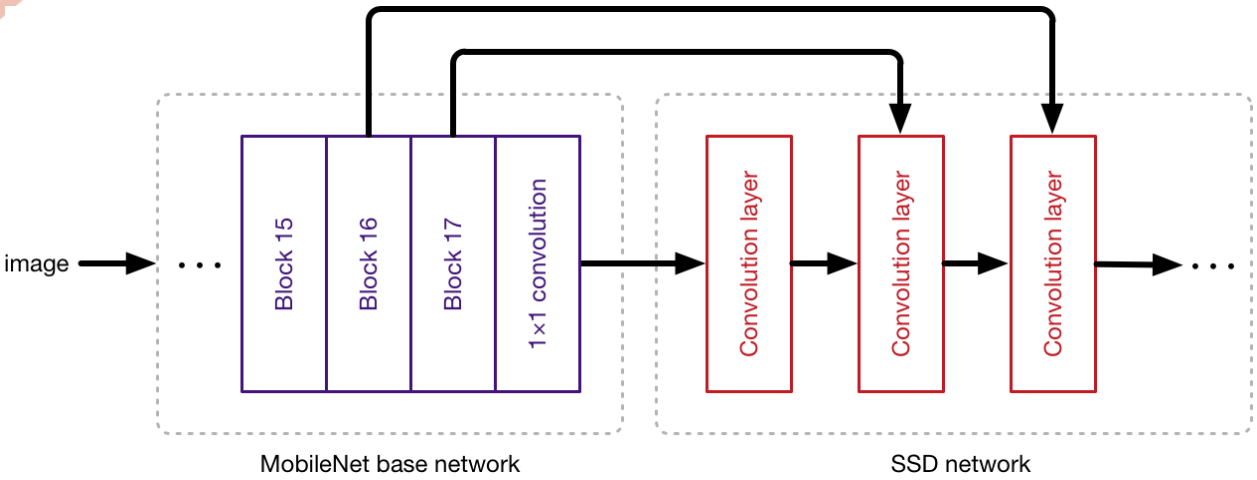
MobileNetV2 is a very effective feature extractor for object detection and segmentation. On the other side, SSD is designed to be independent of the base network, and so it can run on top of MobileNetV2. The MobileNetV2 + SSD combination uses a variant called SSDLite that uses depthwise separable layers instead of regular convolutions for the object detection portion of the network, which makes it easier to get real-time results. When paired with SSDLite the models are about 35% faster with the same accuracy than MobileNetV1.

I just wanted to point out here that MobileNet and SSD make a great combination.

Architecture

When paired with SSD the job of the MobileNetV2 network is to convert the pixels into features that describe what's in the image and pass them onto the next layers. In this case the next layers can be considered as the ones from SSD. Here not only the outputs of the last MobilenetV2 layer are forwarded to SSD, but also some previous layers in order to get high- and low-level features and SSD is an object detector that is fast enough it can be used on real-time video.

To use something like SSDLite with MobileNet, the last layers will look like this instead:



Not only do we take the output of the last base network layer but also the outputs of several previous layers, and we feed these outputs into the SSD layers. The job of the MobileNet layers is to convert the pixels from the input image into features that describe the contents of the image, and pass these along to the other layers. Hence, MobileNet is used here as a feature extractor for a second neural network.

In the case of classification, we're interested in the features that describe high-level concepts, such as (there is a face) and (there is fur), which the classifier layer then can use to draw a conclusion (this image contains a cat); While the classification score on the ImageNet dataset is useful to know, in practice you'll probably never use the pre-trained ImageNet classifier in your apps.

You'll either re-train the classifier on your own dataset or use the base network as a feature extractor for something like object detection (finding multiple objects in the same image) or image segmentation (making a class prediction for every pixel instead of a single prediction for the whole image) or some other exciting computer vision tasks.

In the case of object detection with SSD, we want to know not just these high-level features but also lower-level ones, which is why we also read from the previous layers. Since object detection is more complicated than classification, SSD adds many additional convolutional layers on top of the base network. So, it's important to have a feature extractor that is fast — and that's exactly what MobileNet V2 is.

The MobileNet V2 paper also shows that it's possible to run an advanced semantic segmentation model such as DeepLabv3 on top of MobileNet-extracted features.

SSD

SSD is a single-shot detection model. The core idea is to obtain the location and category of the target in a regression manner on the multi-scale feature map. However, there is a problem of inaccurate detection of small targets. Through analysis, the factors affecting the detection of small targets are mainly the resolution of the feature map and the global information and feature extraction capabilities. The residual block structure of MobileNetV2 can improve the high-resolution low-level feature expression of the feature map used for SSD detection. In addition, the original SSD framework uses the ReLU activation function, which is modified into a SeLU activation function. The network obtained by the activation function has self-normalization characteristics. Among them, ShaoHua. et al. compared SeLU with other activation functions, which proved its effectiveness and robustness, even surpassing batch normalization.Comparison of ReLU & SeLU: People know that the addition of a neuron to an activation function has the ability of non-linear representation, which is the biggest difference between **neural networks** and linear classifiers. Compared to the original activation function Sigmoid, ReLU has three differences:

1) unilateral suppression

2) wider excitation boundary

3) sparse activation.

ReLU can transfer gradients very well: after repeated back propagation, the gradient will not be greatly reduced, suitable for training deep neural networks. However, experiments have shown that the ReLU activation function is prone to training interruptions, and ReLU-enforced sparse processing reduces the effective parameter capacity of the model (when x < 0, the negative gradient is zeroed and will no longer be activated by any data. It is called neuron "necrosis"). One of the similarities between ReLU and Sigmod is that the results are all positive, reducing the ability to express features in 2017, the literature [introduced a new activation function SeLU (scaled exponential linear units), introducing the properties of self-normalization.

The following are the mathematical formulas for the SeLU and ReLU:

$$ReLU_{(z)} = \begin{cases} z & z > 0 \\ 0 & z \leq 0 \end{cases} \tag{3}$$

$$SeLU_{(z)} = \gamma \begin{cases} z & z > 0 \\ \alpha(exp(z) - 1) & z \leq 0 \end{cases} \tag{4}$$

$$where\ \gamma \approx 1.0507,\ \ \alpha \approx 1.673$$

SeLU mainly uses a function F to establish the mapping relationship of the neural network layer.

At the same time, the parameters are transformed to a fixed mean and variance to achieve the

normalized effect. Compared with ReLU,

it has the following advantages:

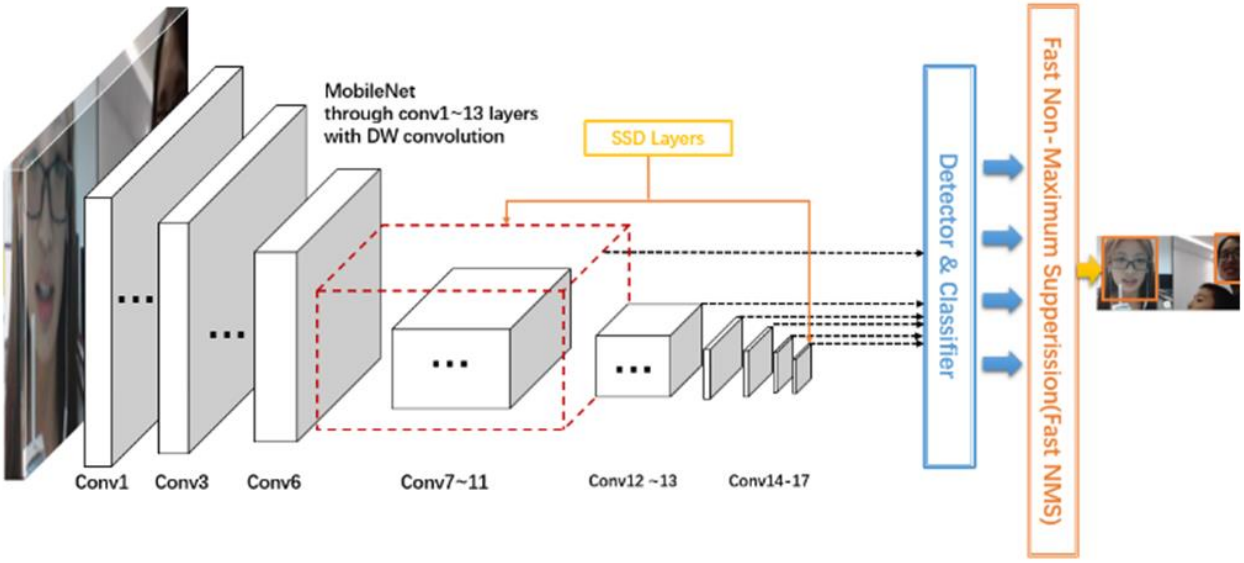
1) Strong convergence property, even if there is noise and interference in the data, it will converge faster after multi-layer forward and backward propagation.

2) Regularization effect, enhance Algorithm robustness.

3) In addition, for the excitation values that do not approximate the unit variance, the variance has upper bound and lower bound, so gradient disappearance and gradient explosion are almost impossible.

Model Structure:

the SSD detection framework based on MobileNet. Adding 8 convolution layers behind the conv13 of MobileNet. A total of 6 layers are extracted for detection, and the activation function is SeLU



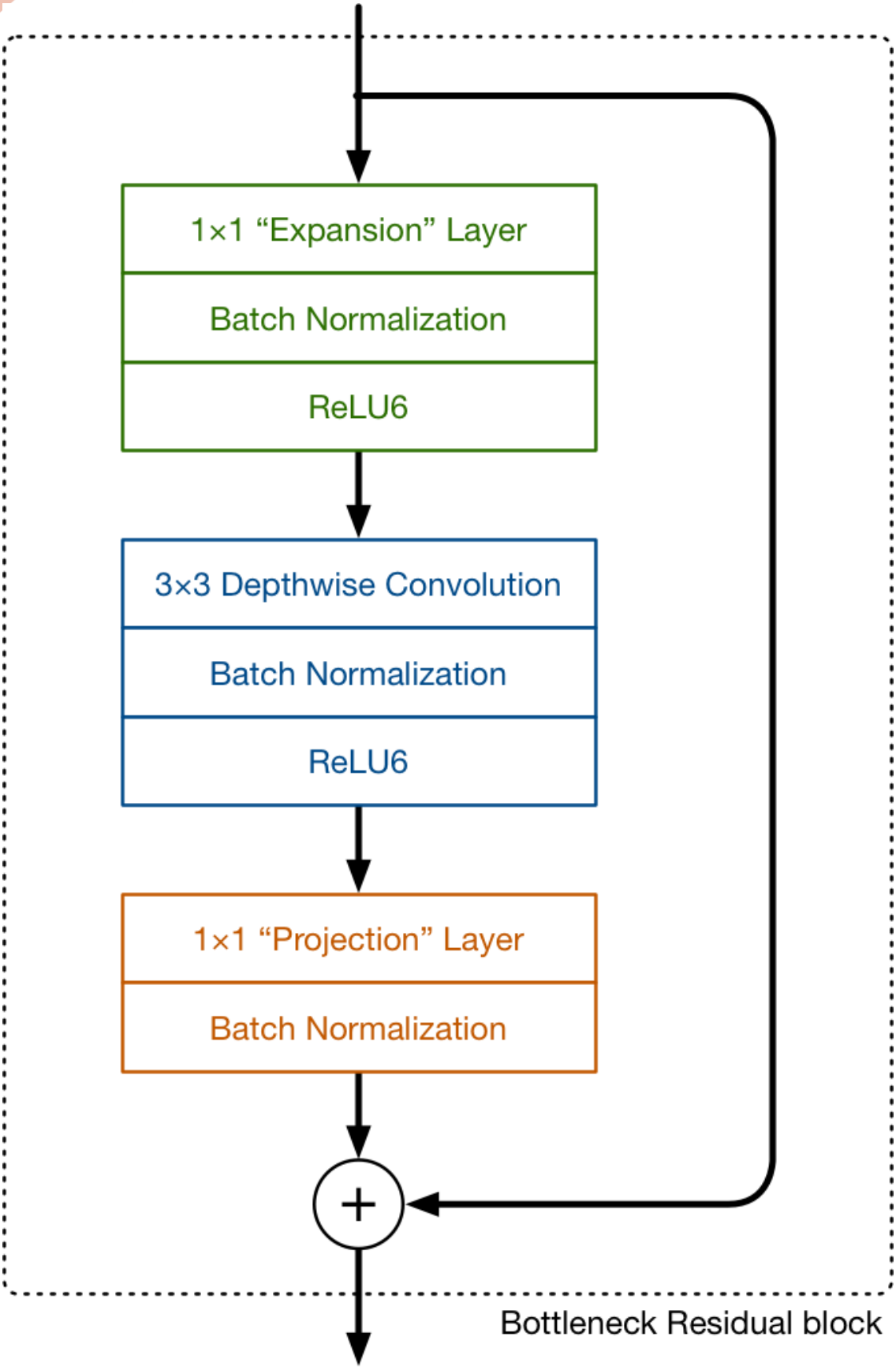
the left part is the feature extraction network MobileNet, which contains the output of the conv1-conv13 layer. After the conv13, 8 layers of convolution are added, and 4 layers and conv11 and conv13 of MobileNet are selected as the scale feature map of the SSD, and finally attributed to a prediction unit. The activation function in the middle uses the SeLU function.

MobileNetV2

MobileNetV2 that improves the state-of-the-art performance of mobile models on multiple tasks and benchmarks as well as across a spectrum of different model sizes. We also describe efficient ways of applying these mobile models to object detection in a novel framework we call **SSDLite**. Additionally, we demonstrate how to build mobile semantic segmentation models through a reduced form of DeepLabv3 which we call Mobile DeepLabv3.

The MobileNetV2 architecture is based on an inverted residual structure where the input and output of the residual block are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input an MobileNetV2 uses lightweight depthwise convolutions to filter features in the intermediate expansion layer. Additionally, we find that it is important to remove non-linearity's in the narrow layers in order to maintain representational power. We demonstrate that this improves performance and provide an intuition that led to this design. Finally, our approach allows decoupling of the input/output domains from the expressiveness of the transformation, which provides a convenient framework for further analysis. We measure our performance on ImageNet classification, COCO object detection, VOC image segmentation. We evaluate the trade-offs between accuracy, and number of operations measured by multiply-adds (MAdd), as well as the number of parameters.

MobileNet V2 still uses depthwise separable convolutions, but its main building block now looks like this:


1×1 “Expansion” Layer
Batch Normalization
ReLU6
3×3 Depthwise Convolution
Batch Normalization
ReLU6
1×1 “Projection” Layer
Batch Normalization
Bottleneck Residual block

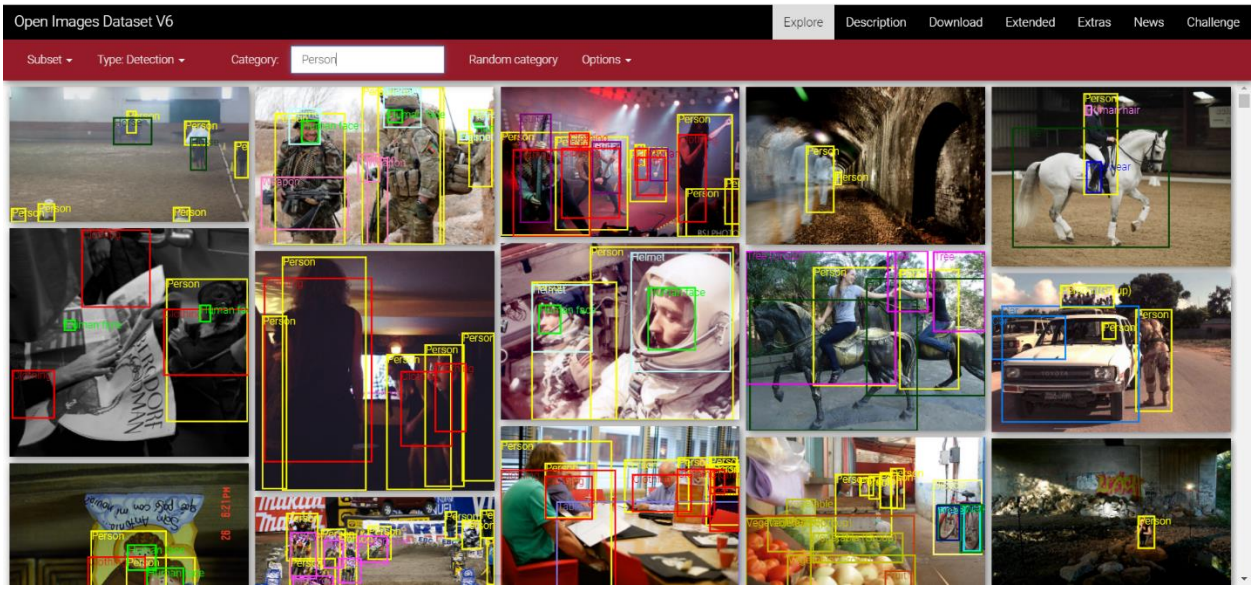
6.2 Data Acquisition & Image Tagging

We have two types of datasets:

First: dataset of images about 3 classes (person, chairs and stairs) for object detection

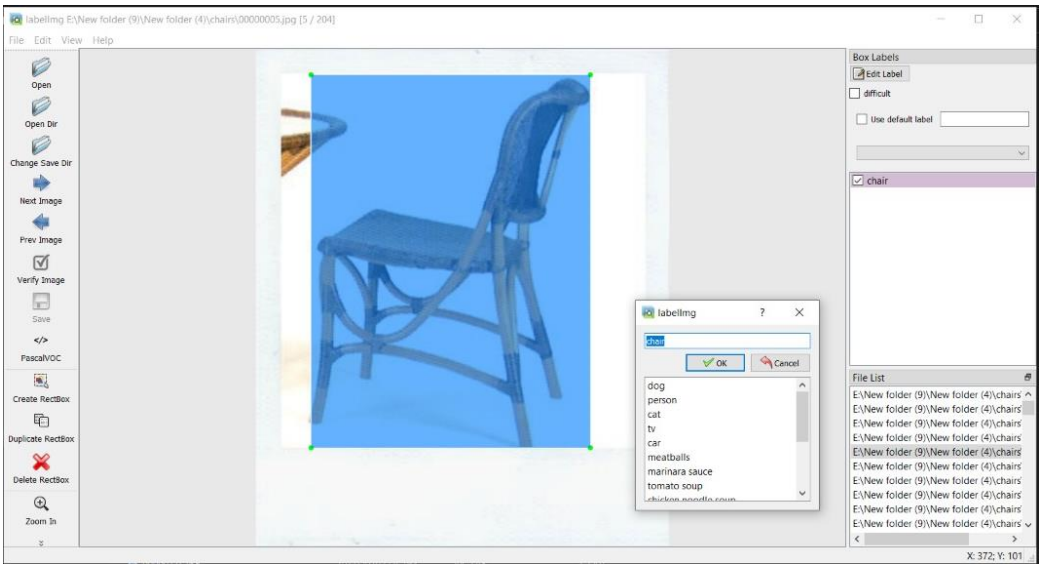
Second: dataset of Egyptian currency for currency detection

First: Collect images for object detection from storage.googleapis.com:



Dataset consists of 4k Images all are 227*227 about 3 Classes: Person, chairs and stairs.

Then make label for them:



After labeling, dataset is annotated using Pascal Visual Object Classes (VOC) as an XML file.

Annotations

The annotations describe what is in the image. In order words, the annotations provide the targets we need for training.

The annotations are in XML format, one per training image. The annotation file contains one or more <object> sections, with the name of the class, a bounding box given as xmin, xmax, ymin, ymax, and a few other attributes for each ground-truth object.

If an object is marked as difficult, we will ignore it. These are often objects that are very small. Such objects are also ignored by the official evaluation metric for the VOC challenge.

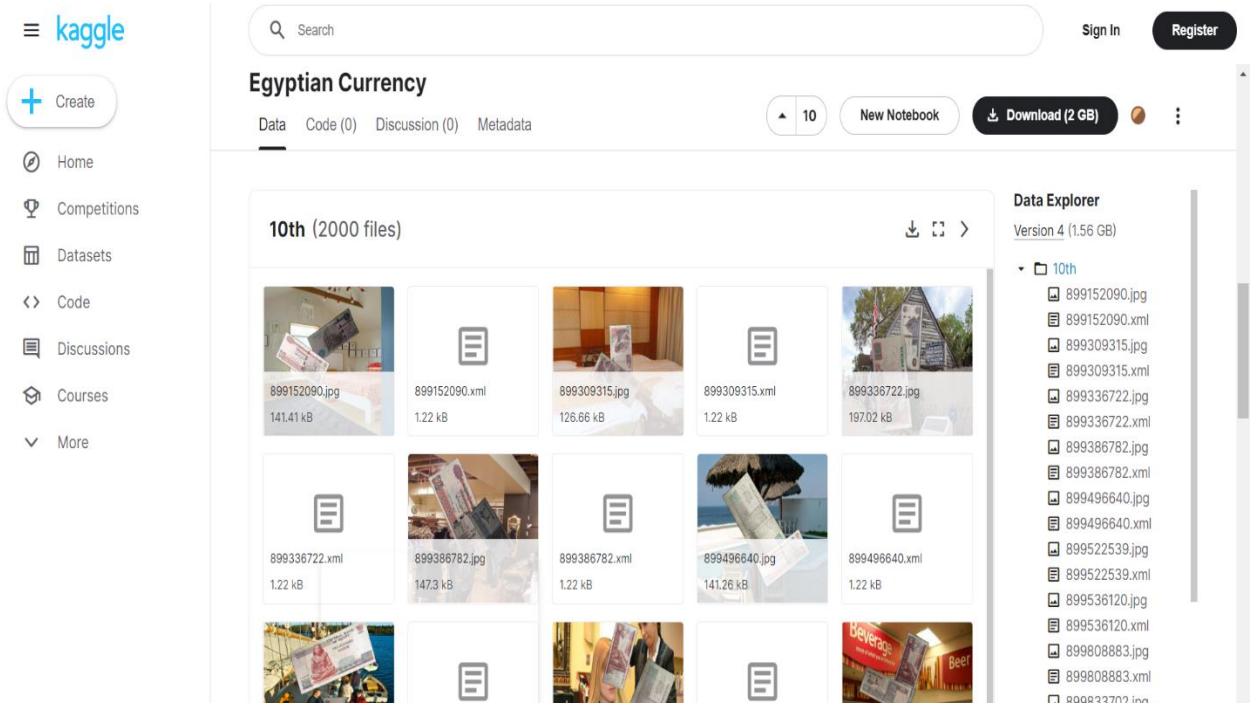
Here is an example annotations file, VOC2007/Annotations/674x898-1_-r570hx09xjcBi.xml:

674x898-1_-r5ZohX09lxJcBi.xml - Notepad

File Edit Format View Help

```
<annotation>
        <folder>stairs</folder>
        <filename>674x898-1_-r5ZohX09lxJcBi.jpg</filename>
        <path>C:\Users\karim\Desktop\stars\stairs\674x898-1_-r5ZohX09lxJcBi.jpg</path>
        <source>
                <database>Unknown</database>
        </source>
        <size>
                <width>674</width>
                <height>898</height>
                <depth>3</depth>
        </size>
        <segmented>0</segmented>
        <object>
                <name>upstair</name>
                <pose>Unspecified</pose>
                <truncated>0</truncated>
                <difficult>0</difficult>
                <bndbox>
                        <xmin>67</xmin>
                        <ymin>475</ymin>
                        <xmax>607</xmax>
                        <ymax>843</ymax>
                </bndbox>
        </object>
</annotation>
```

Second: in currency detection use data set of Egyptian currency prepared with its annotations from kaggle , and it consists of 10k Images all are 1080*1080 you can easily resize it before training.



Training

To simply start training the model, run the below code which will initiate the training pipeline in Tensor-Flow. Remember to provide the logging parameter so that the results of the model training can be observed in the tensor board

In our project two features have dataset.

6.2.1 Egyptian currency detection

Dataset consists of 10k Images all are 1080*1080 you can easily resize it before training. Dataset is annotated as an XML file. Each 1000 images are loaded to a separate file for more freedom while splitting the train/test data. The dataset were created randomly so no count for each class. Classes: 5, 10, 20, 50, 100, 200

The results shown above depict the training over 81000 steps or epochs with a batch size of 8. As observed, the loss profiles have tapered down as the model has extracted the key features and learned the representation of a tree. With the training completed, the model is saved to the directory and can be used for testing on new sets of images.

The screen contain 6 classes for currency detection.

Calculate (precision,recall,loss)

precision $$Precision = \frac{TP}{TP + FP}$$

recall $$Recall = \frac{TP}{TP + FN}$$

loss $$Loss = \frac{P+N}{FP+FN}$$

TP = True positive

TN = True negative

FP = False positive

FN = False negative

For example, in the testing for detect person:

$$Precision = \frac{TP}{total\ positive\ results}$$

$$Recall = \frac{TP}{total\ person\ cases}$$

$$Loss = \frac{P+N}{total\ negative\ results}$$

T prediction is correct even if person or not .

F prediction is incorrect.

P positive its person.

N negative its not person.

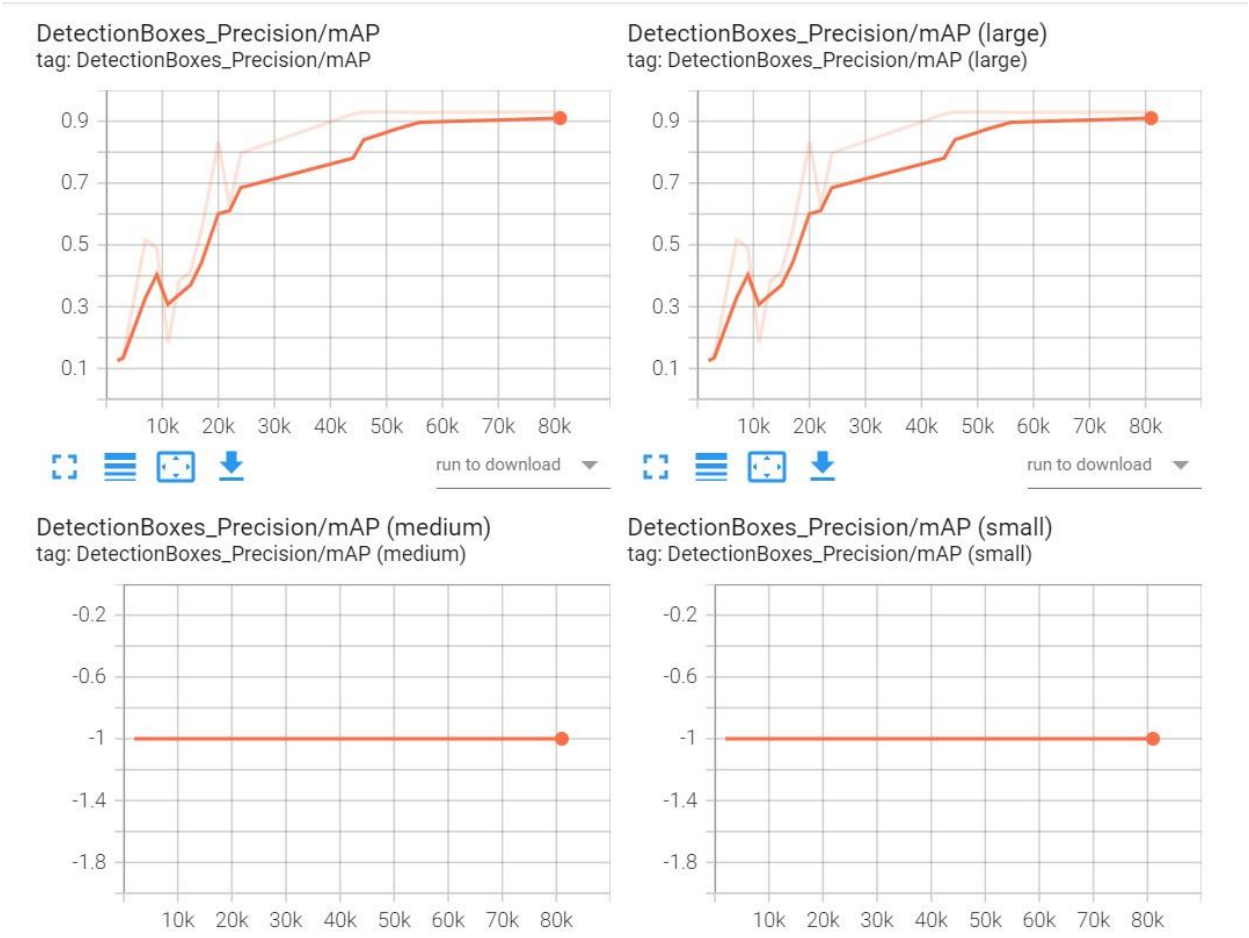
TP Expect that he was a person and already he is a person .

TN Expect that he wasn't a person and already he isn't a person.

FP Expect that he wasn't a person and already he is a person.

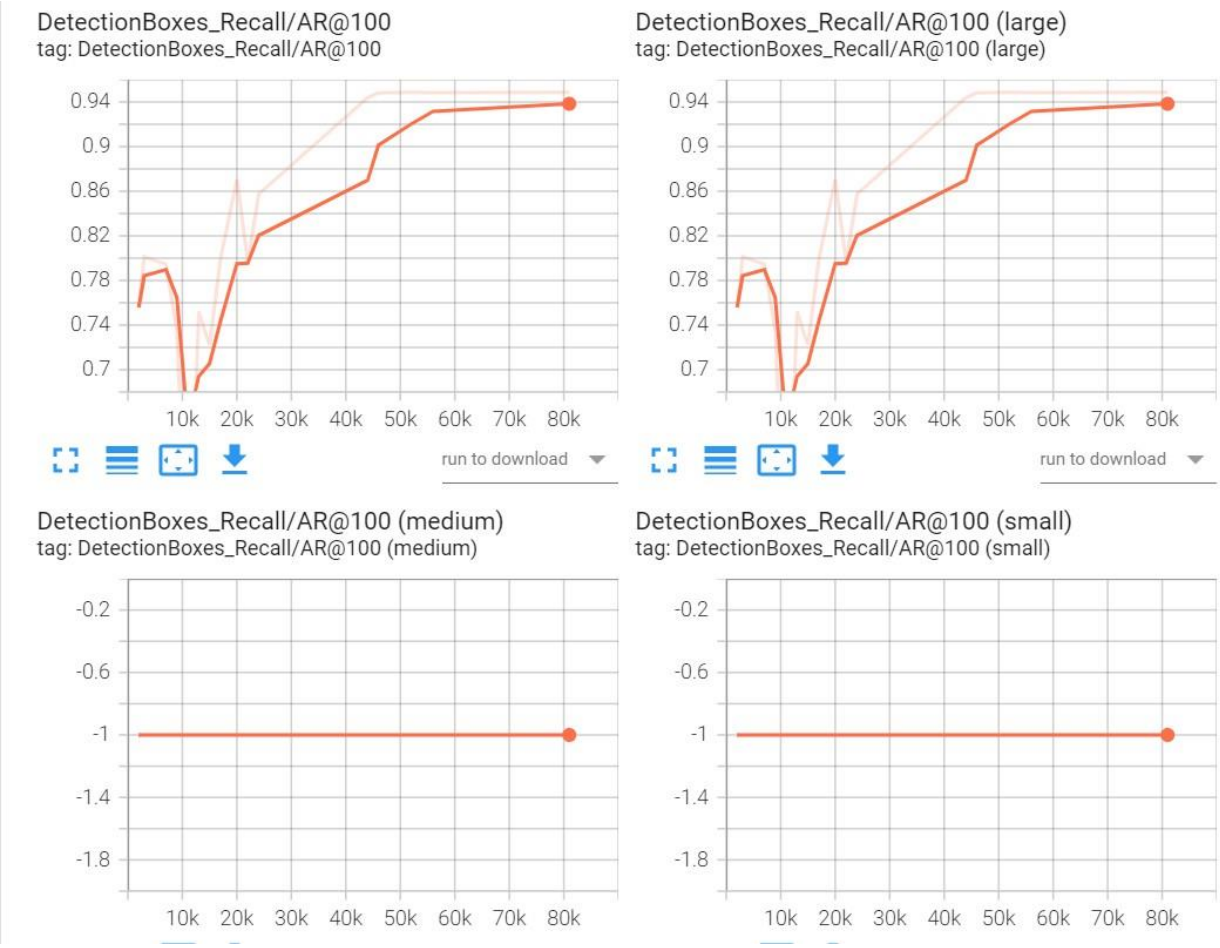
FN Expect that he was a person and already he isn't a person.

Precision is a measure of, "when your model guesses how often does it guess correctly?"



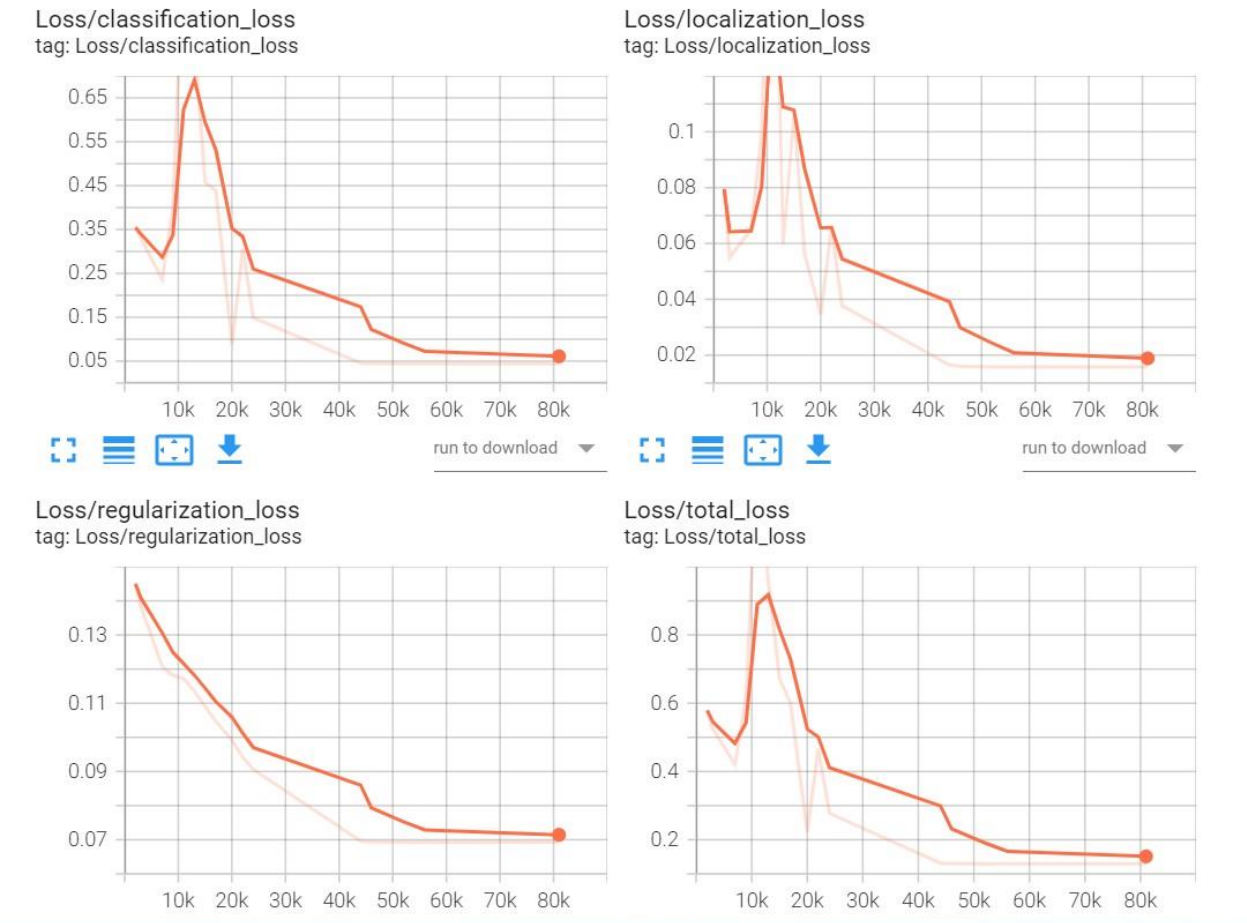
In this model guesses how often does it guess correctly From 0.12 to .9 then the guess correct rate stopped.

Recall is a measure of "has your model guessed every time that it should have guessed?" Consider an image that has 10 red blood cells. A model that finds only one of these ten but correctly labels is as "RBC" has perfect precision (as every guess it makes – one – is correct) but imperfect recall (only one of ten RBC cells has been found).

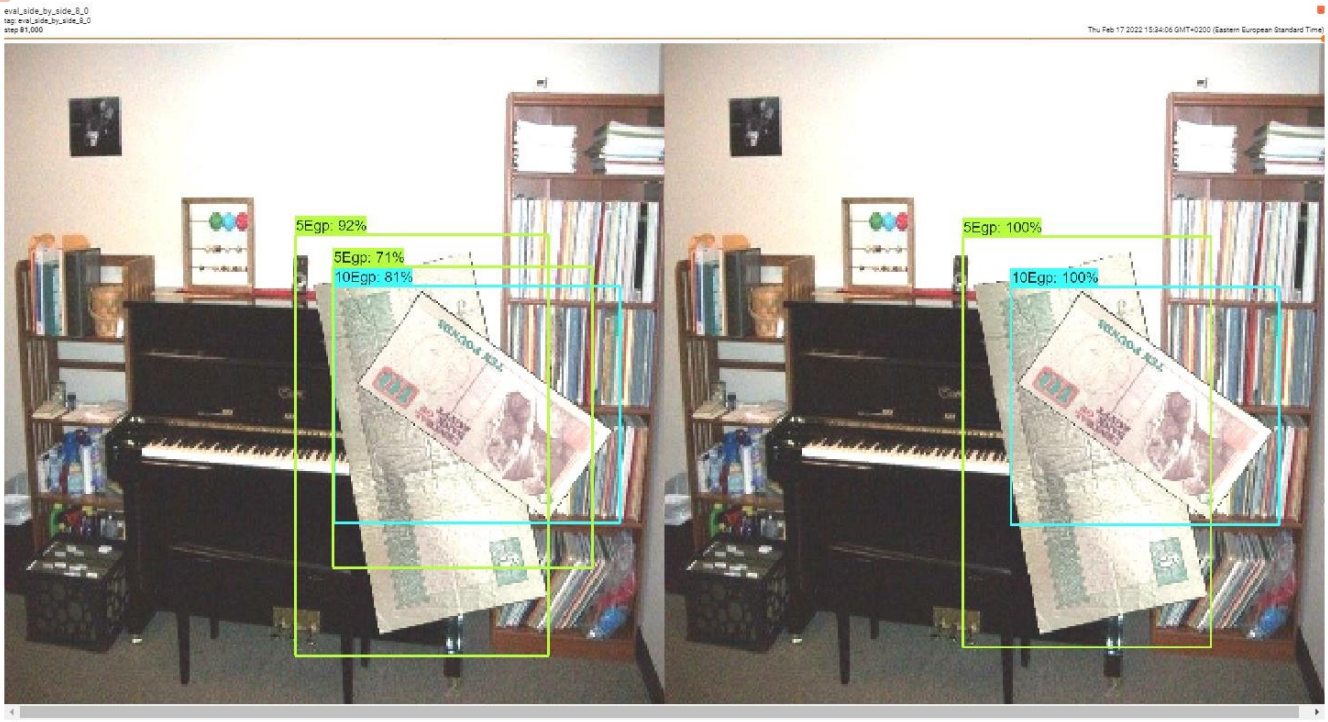


Recall is a measure of same model guessed every time that it should have guessed and start from 0.76 to 0.94 .

Loss functions is a crucial factor that affecting the detection precision in object detection task



In this model crucial factor that affecting the detection precision for total loss Start from 0.55 steps to 0.1 and stop losing.



There are samples for evaluation in currency detection.

You have it, the model is able to successfully infer the Egyptian currency present in the picture and performs quite well at it too. This concludes the development of object detection and inference pipeline for custom objects.

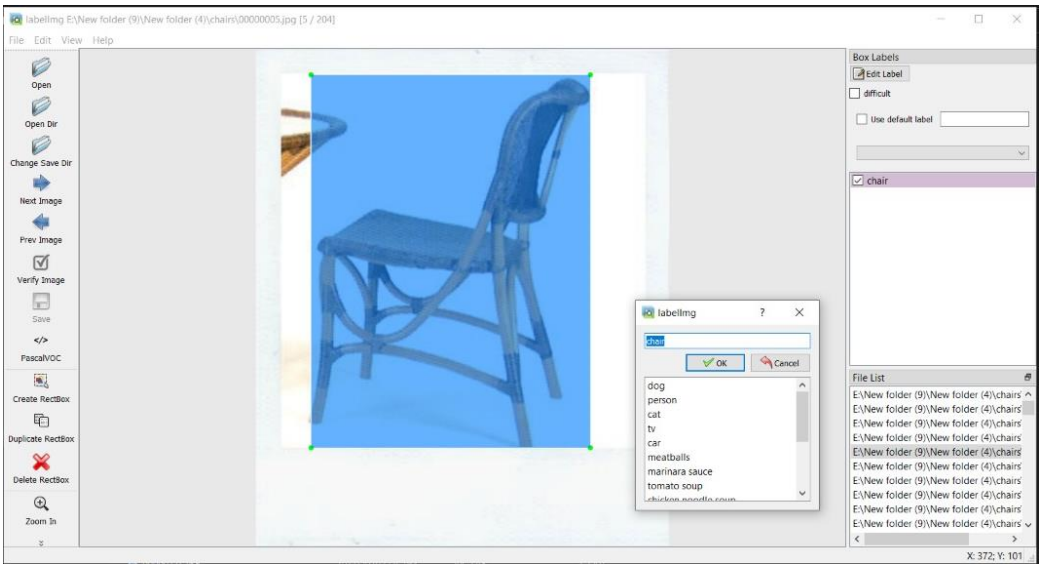
6.2.2 Object Detection

Dataset consists of 4k Images all are 227*227. Dataset is annotated as an XML file. We split the train/test data.

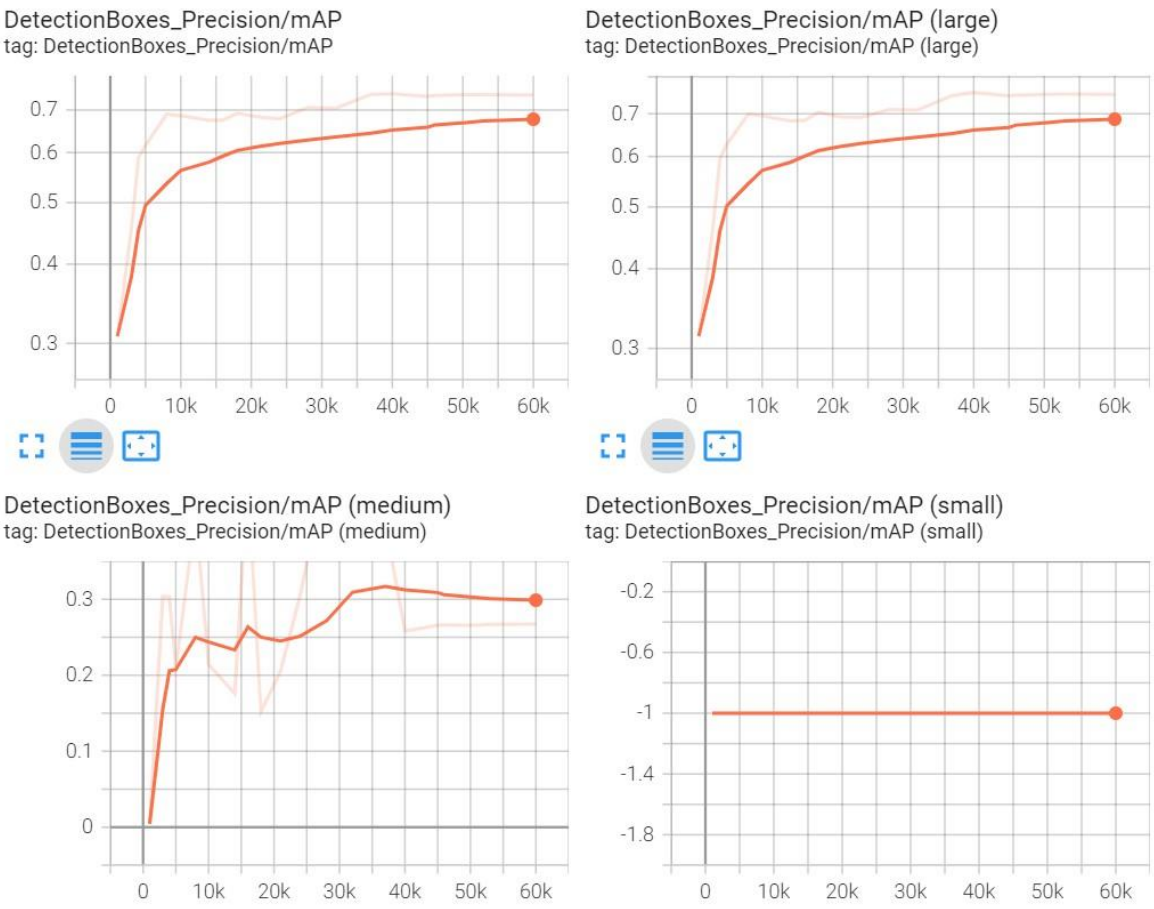
The journey of training an object detection model begins with data acquisition. Although there exist many ways of automating the process, in this specific training case, data acquisition was done manually to filter out noisy images and to ensure the quality of the training dataset. Once the data is acquired, the images in the dataset need to be tagged. Image tagging was completed using the LabelImg software, which is an open-source python-based implementation of a system that tags the bounding boxes and records them in an XML file. Below is an image taken during the tagging of the pictures the generated result of the image tagging

efforts is an XML file that encompasses the bounding boxes of the tagged images. A sample XML file is shown below for reference

3 Classes: person, chair, stairs

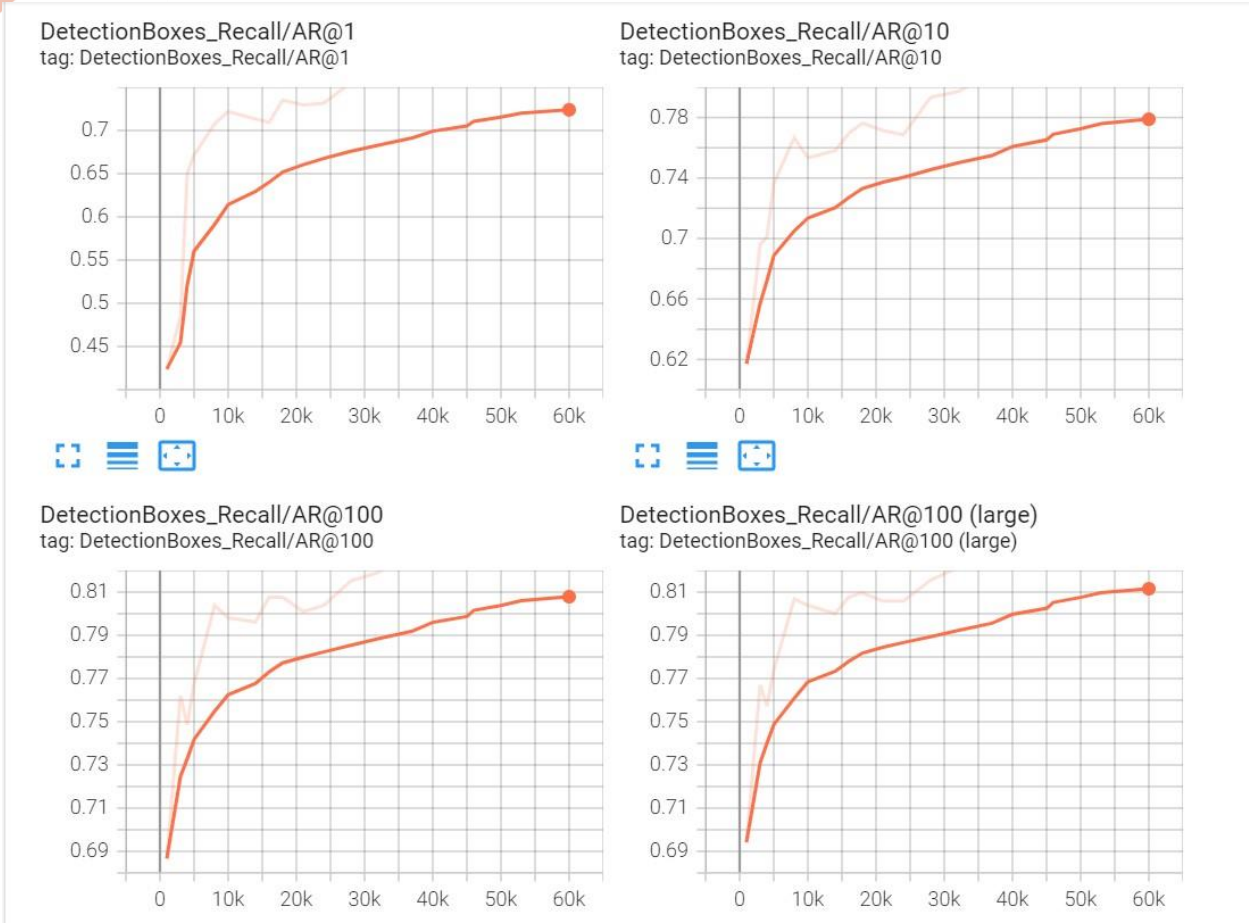


The results shown above depict the training over 61.000 steps or epochs with a batch size of 8. As observed, the loss profiles have tapered down as the model has extracted the key features and learned the representation of a tree. With the training completed, the model is saved to the directory and can be used for testing on new sets of images.

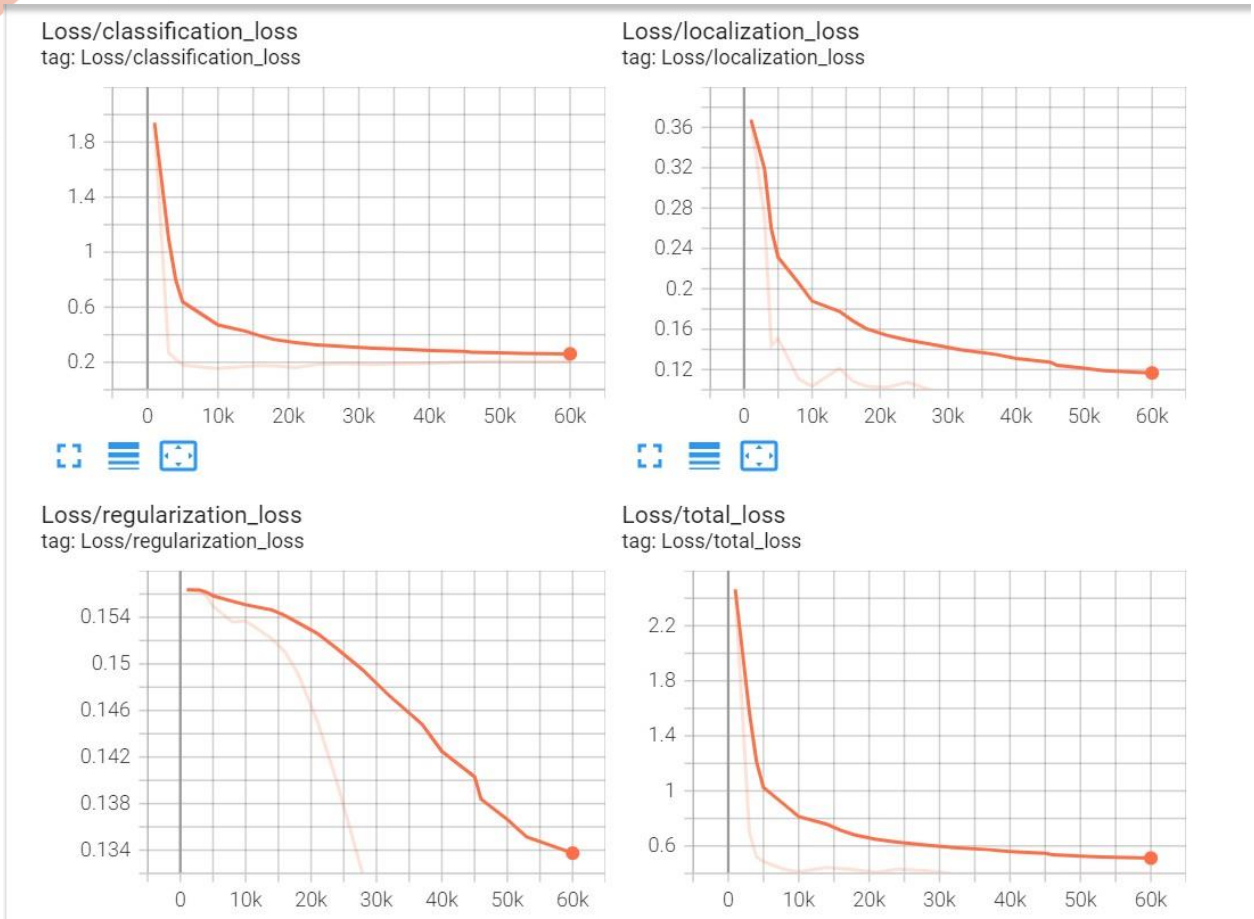


In this model guesses how often, it guesses correctly

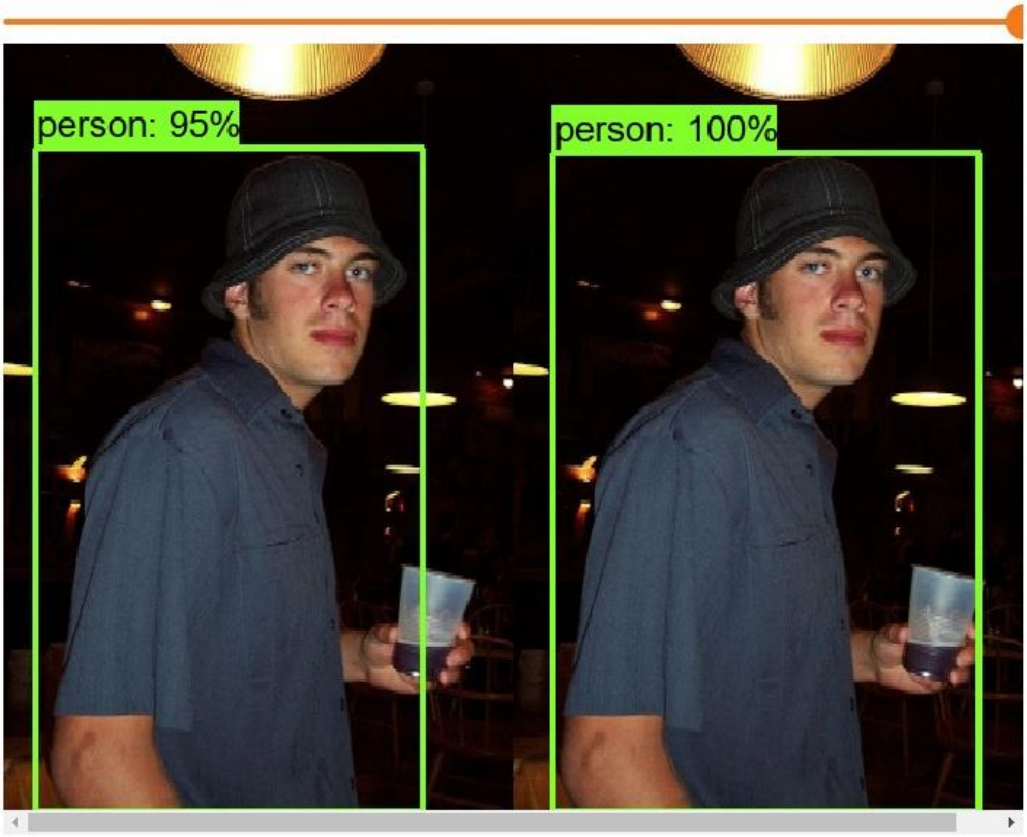
From 0.32 to .68 then the guess correct rate stopped.

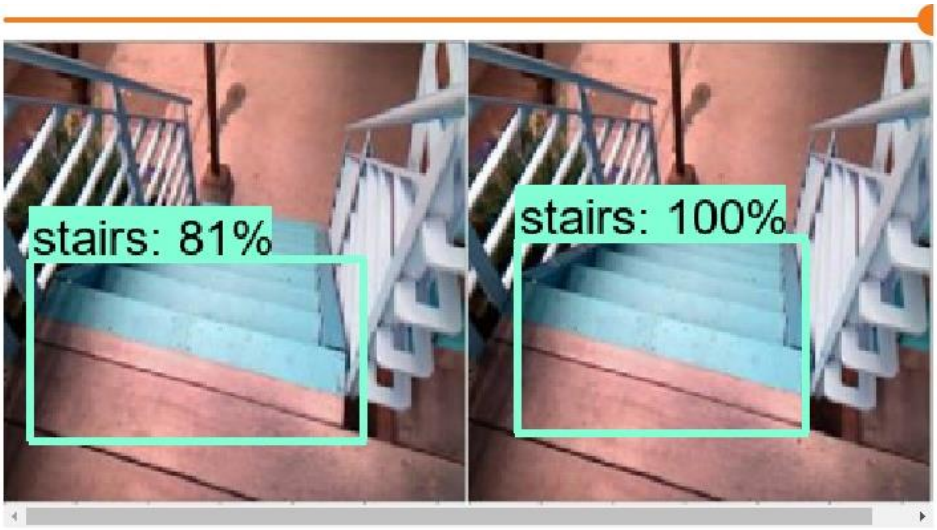


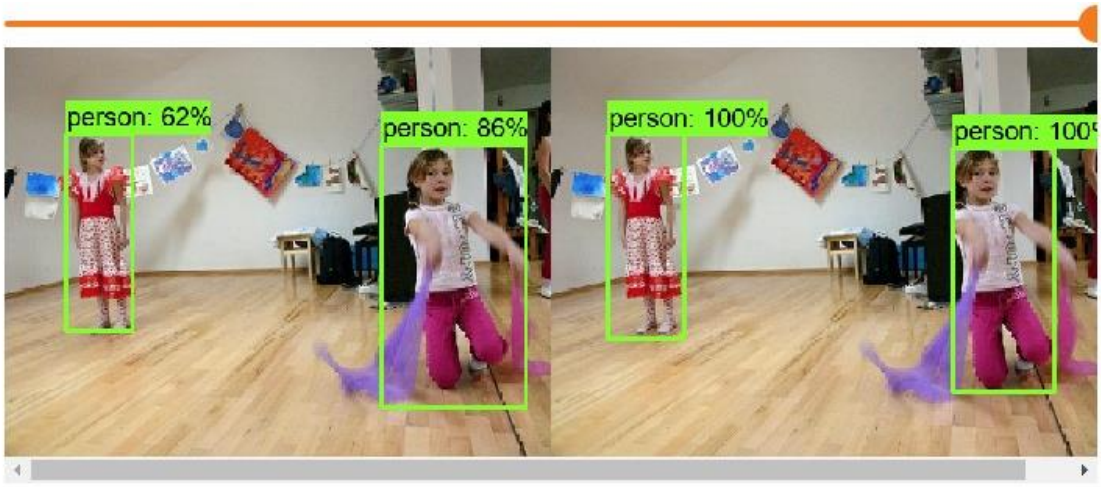
Recall is a measure of same model guessed every time that it should have guessed and start from 0.43 to 0.73.



In this model crucial factor that affecting the detection precision for total loss Start from 2.3 steps to 0.3 and stop losing.


person: 95%
person: 100%


stairs: 81%
stairs: 100%



There are samples for evaluation in object detection

You have it, the model is able to successfully infer the 3 classes (chair, stairs, person) present in the picture and performs quite well at it too. This concludes the development of object detection and inference pipeline for custom objects.

6.2 Evaluation Criterion

The target of detection model is selected, and people counting in different scenarios is applied. RMSE is selected as the evaluation criterion.

The calculation formula is:

$$RMSE(X,f) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(f(x^i) - y^{(i)})^2}$$

$y^{(i)}$ is the number of original people in each picture,

$f(x^i)$ is the number of people detected,

And m is the number of video frames detected. In addition, this article uses the frames per second (FPS)

To measure the target detection speed and uses 25 fps as the real-time consideration threshold

6.4 System Snapshots:

6.4.1 Main:

main.py

```python
import threading
import speech_recognition as sr
from Threading import Threading
from CurrencyDetection import *

if __name__ == "__main__":
    thread = Threading

    object = threading.Thread(target=thread.thread_objectDetection, args=(lambda: thread.stop_thread_objectDetection

    object.start()

    while (True):
        speak = wincl.Dispatch("SAPI.SpVoice")

        r = sr.Recognizer()  # initialize recognizer
        mic = sr.Microphone()
        print("speak")
        try:
            print("speak++++++++++")
            with mic as source:
                audio = r.listen(source)
                words = r.recognize_google(audio)
                print(words)
        except:
            print('error')
            continue

        if words == "currency":
            thread.stop_thread_objectDetection = False
            currncy = threading.Thread(target=thread.thread_currncydetection(self=Threading()), )
            currncy.start()
            currncy.join()
            print("currncy detection")
            thread.stop_thread_objectDetection = True
            object = threading.Thread(target=thread.thread_objectDetection,
                                      args=(lambda: thread.stop_thread_objectDetection,))
            object.start()
            continue

        if words == "text":
            thread.stop_thread_objectDetection = False
            text = threading.Thread(target=thread.thread_textdetection(self=Threading()),
                                    )
            text.start()
            text.join()
            print("Text detection")
            thread.stop_thread_objectDetection = True
            object = threading.Thread(target=thread.thread_objectDetection,
                                      args=(lambda: thread.stop_thread_objectDetection,))
            object.start()
            continue
```

```
        if words == "text":
            thread.stop_thread_objectDetection = False
            text = threading.Thread(target=thread.thread_textdetection(self=Threading()),
                                    )
            text.start()
            text.join()
            print("Text detection")
            thread.stop_thread_objectDetection = True
            object = threading.Thread(target=thread.thread_objectDetection,
                                      args=(lambda: thread.stop_thread_objectDetection,))
            object.start()
            continue
        if words == 'time':
            time = threading.Thread(target=thread.thread_time(self=Threading()),)
            time.start()
            time.join()
            continue
        if words == 'exit':
            speak.Speak('goodbye Baby')
            thread.stop_thread_objectDetection = False
            break

```

6.4.2 Object Detection:

objectDetection.py

```
import os
import win32com.client as wincl
import cv2
import numpy as np
import tensorflow as tf
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util

from Detection import Detect

class ObjectDetection:
    LABEL_MAP_NAME = 'label_map.pbtxt'
    CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
    numckpt = 'ckpt-51'

    paths = {
        'ANNOTATION_PATH': os.path.join('ObjectDetection/Tensorflow', 'workspace', 'annotations'),
        'IMAGE_PATH': os.path.join('ObjectDetection/Tensorflow', 'workspace', 'images'),
        'CHECKPOINT_PATH': os.path.join('ObjectDetection/Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME),
    }

    files = {
        'PIPELINE_CONFIG': os.path.join('ObjectDetection/Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'pipe
        'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
    }

```

objectDetection.py

```
28
29      category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
30
31      configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
32      detection_model = model_builder.build(model_config=configs['model'], is_training=False)
33
34      # Restore checkpoint
35      ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
36      ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], numckpt)).expect_partial()
37
38      detect=Detect()
39
40
41      def objectDetection(self,stop):
42          cap = cv2.VideoCapture(0)
43          speak = wincl.Dispatch("SAPI.SpVoice")
44          words = ""
45          print("in")
46          list = []
47          while True:
48              if stop:
49                  ret, frame = cap.read()
50                  image_np = np.array(frame)
51                  listt = []
52                  listt = list
53                  list = []
54                  input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
55                  detections = self.detect.detect_fn(input_tensor,self.detection_model)
56                  # detections = self.detect fn(input tensor)
```

objectDetection.py

```
55                  detections = self.detect.detect_fn(input_tensor,self.detection_model)
56                  # detections = self.detect_fn(input_tensor)
57                  num_detections = int(detections.pop('num_detections'))
58                  detections = {key: value[0, :num_detections].numpy()
59                                 for key, value in detections.items()}
60                  detections['num_detections'] = num_detections
61
62                  # detection_classes should be ints.
63                  detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
64
65                  label_id_offset = 1
66                  image_np_with_detections = image_np.copy()
67
68                  viz_utils.visualize_boxes_and_labels_on_image_array(
69                      image_np_with_detections,
70                      detections['detection_boxes'],
71                      detections['detection_classes'] + label_id_offset,
72                      detections['detection_scores'],
73                      self.category_index,
74                      use_normalized_coordinates=True,
75                      max_boxes_to_draw=100,
76                      min_score_thresh=.5,
77                      agnostic_mode=False)
78                  j = 0
79                  if detections['detection_scores'][0] >= .5:
80                      list.insert(len(list), self.category_index[detections['detection_classes'][0] + 1]['name'])
81                  print("Start")
82                  list.sort()
83                  listt.sort()
```

objectDetection.py

```
                    max_boxes_to_draw=100,
                    min_score_thresh=.5,
                    agnostic_mode=False)
                j = 0
                if detections['detection_scores'][0] >= .5:
                    list.insert(len(list), self.category_index[detections['detection_classes'][0] + 1]['name'])
                print("Start")
                list.sort()
                listt.sort()
                if listt != list:
                    print(list)
                    print("---------")
                    speak.Speak(list)
                cv2.imshow('object detection', cv2.resize(image_np_with_detections, (800, 600)))
                cv2.waitKey(125)
            else:
                ret, frame = cap.read()

                # Display the clicked frame for 2
                # sec.You can increase time in
                # waitKey also
                # cv2.imshow('a', frame)

                # time for which image displayed
                cap.release()
                cv2.destroyAllWindows()
                break


```

6.4.3 Currency Detection:

CurrencyDetection.py

```
import os
import cv2
import numpy as np
import tensorflow as tf
import win32com.client as wincl
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as viz_utils
from object_detection.builders import model_builder
from object_detection.utils import config_util
import time

from Detection import Detect
from imageCam import ImageCam


class CurrencyDetection:
    LABEL_MAP_NAME = 'label_map.pbtxt'
    CUSTOM_MODEL_NAME = 'my_ssd_mobnet'
    numckpt='ckpt-101'


    paths = {
        'ANNOTATION_PATH': os.path.join('Egyptian Currency/Tensorflow', 'workspace', 'annotations'),
        'IMAGE_PATH': os.path.join('Egyptian Currency/Tensorflow', 'workspace', 'images'),
        'CHECKPOINT_PATH': os.path.join('Egyptian Currency/Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME)
    }
```

```
27
28      files = {
29          'PIPELINE_CONFIG': os.path.join('Egyptian Currency/Tensorflow', 'workspace', 'models', CUSTOM_MODEL_NAME, 'pi
30          'LABELMAP': os.path.join(paths['ANNOTATION_PATH'], LABEL_MAP_NAME)
31      }
32      imageCam= ImageCam()
33
34      category_index = label_map_util.create_category_index_from_labelmap(files['LABELMAP'])
35
36      configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
37      detection_model = model_builder.build(model_config=configs['model'], is_training=False)
38
39      # Restore checkpoint
40      ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
41      ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], numckpt)).expect_partial()
42
```

CurrencyDetection.py

CurrencyDetection.py > CurrencyDetection

```
43      detect = Detect()
44
45      def sum (sum , num) :
46          return sum + num
47
48      def currncydetection(self):
49          IMAGE_PATH = self.imageCam.imagePath
50          img = cv2.imread(IMAGE_PATH,cv2.IMREAD_UNCHANGED)
51          image_np = np.array(img)
52
53          input_tensor = tf.convert_to_tensor(np.expand_dims(image_np, 0), dtype=tf.float32)
54          detections = self.detect.detect_fn(input_tensor, self.detection_model)
55          # detections = self.detect_fn(input_tensor)
56
57          num_detections = int(detections.pop('num_detections'))
58          detections = {key: value[0, :num_detections].numpy()
59                        for key, value in detections.items()}
60
61          detections['num_detections'] = num_detections
62          # detection_classes should be ints.
63          detections['detection_classes'] = detections['detection_classes'].astype(np.int64)
64
65          label_id_offset = 1
66          image_np_with_detections = image_np.copy()
67          viz_utils.visualize_boxes_and_labels_on_image_array(
68              image_np_with_detections,
69              detections['detection_boxes'],
70              detections['detection_classes'] + label_id_offset,
71              detections['detection_scores'],
```

CurrencyDetection.py

```
            detections['detection_boxes'],
            detections['detection_classes'] + label_id_offset,
            detections['detection_scores'],
            self.category_index,
            use_normalized_coordinates=True,
            max_boxes_to_draw=100,
            min_score_thresh=.7,
            agnostic_mode=False)

        speak = wincl.Dispatch("SAPI.SpVoice")
        j = 0
        s = 0
        count=0
        for i in detections['detection_scores']:
            if i >= 0.8:
                count+=1
                txt = str(self.category_index[detections['detection_classes'][j] + 1]['name'])[:-3]
                speak.Speak(txt + " bound")
                print(txt + " bound", "       ", detections['detection_scores'][j])
                s = sum(s, int(txt))
            else:
                break
            j += 1
        print(s)
        if count >1 :
            speak.Speak("the sum is "+ str(s) + " bound")
        cv2.imshow('Curruncy', cv2.resize(image_np_with_detections, (800, 600)))
        cv2.waitKey(10000)
        cv2.destroyAllWindows()
```

6.4.4 Text Detection:

objectDetection.py | TextDetection.py

```
import os
import cv2
from ArabicOcr import arabicocr
from googletrans import Translator
from gtts import gTTS
import playsound
import time

from imageCam import ImageCam

class TextDetection:
    imageCam = ImageCam()

    def TextToSpeach(self):
        image_path = self.imageCam.imagePath
        out_image = 'out.jpg'
        results = arabicocr.arabic_ocr(image_path, out_image)

        f = ""
        for i in range(len(results)):
            f+= " " + results[i][1]

        translator = Translator()
        t = translator.translate(f, dest="ar").text

        myobj = gTTS(text=t, lang='ar', slow=False)
        myobj.save("welcome2.mp3")
        playsound.playsound("welcome2.mp3")
        os.remove("welcome2.mp3")
```

6.4.5 Thread:

Threading.py

```python
from time import ctime

from objectDetection import *
from CurrencyDetection import *
from TextDetection import *

class Threading:
    stop_thread_objectDetection = True

    speak = wincl.Dispatch("SAPI.SpVoice")
    imageCam = ImageCam()

    def thread_objectDetection(stop):
        object = ObjectDetection()
        object.objectDetection(stop)

    def thread_currncydetection(self):
            currency=CurrencyDetection()
            self.imageCam.SaveImage('currency')
            # currency.SaveImage()
            currency.currncydetection()

    def thread_textdetection(self):
            text = TextDetection()
            self.imageCam.SaveImage('text')
            # text.SaveTextImage()
            text.TextToSpeach()

    def thread_time(self):
            self.speak.Speak(ctime())
```

6.4.6 ImageCam:

imageCam.py

```python
import cv2
import time

class ImageCam:
    imagePath = "test.jpg"
    def SaveImage(self,lable):
        cap = cv2.VideoCapture(0)
        TIMER = int(10)
        while True:
            # ret, img = cap.read()
            # cv2.imshow('a', img)
            if True:
                prev = time.time()
                while TIMER >= 0:
                    ret, img = cap.read()

                    # Display countdown on each frame
                    # specify the font and draw the
                    # countdown using puttext
                    font = cv2.FONT_HERSHEY_SIMPLEX
                    cv2.putText(img, str(TIMER),
                                (200, 250), font,
                                7, (0, 255, 255),
                                4, cv2.LINE_AA)
                    cv2.imshow(lable, cv2.resize(img, (800, 600)))
                    cv2.waitKey(125)

                    # current time
                    cur = time.time()
```

imageCam.py

```
                    # current time
                    cur = time.time()

                    # Update and keep track of Countdown
                    # if time elapsed is one second
                    # than decrease the counter
                    if cur - prev >= 1:
                        prev = cur
                        TIMER = TIMER - 1

                else:
                    ret, img = cap.read()

                    # Display the clicked frame for 2
                    # sec.You can increase time in
                    # waitKey also
                    cv2.imshow('a', cv2.resize(img, (800, 600)))

                    # time for which image displayed
                    cv2.waitKey(10)

                    # Save the frame
                    cv2.imwrite('test.jpg', img)
                    cap.release()
                    cv2.destroyAllWindows()
                    break
```

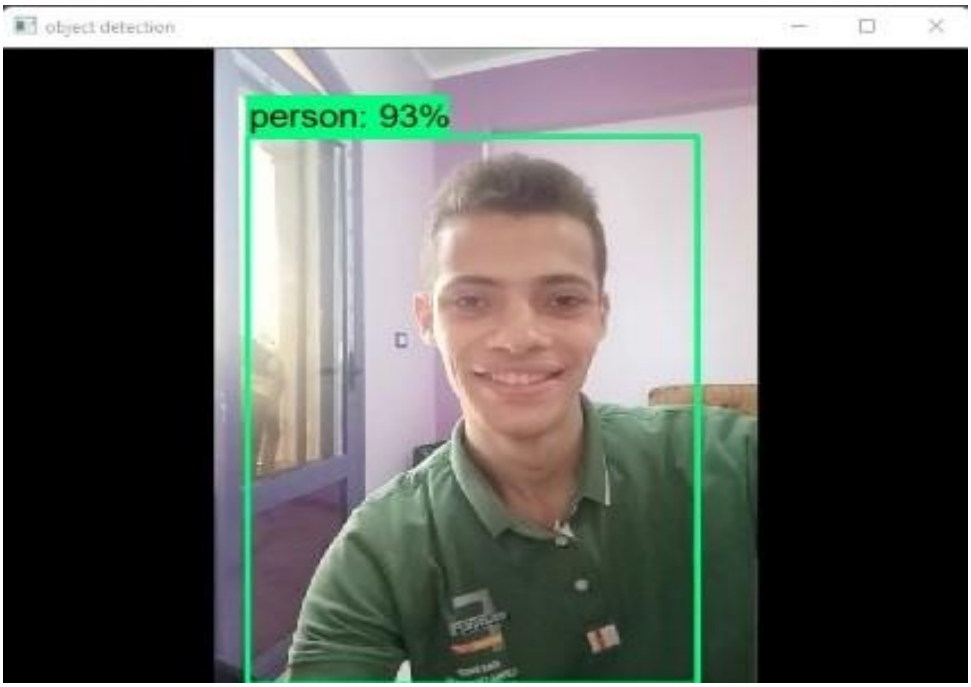
6.4.7 Detection:

Detection.py

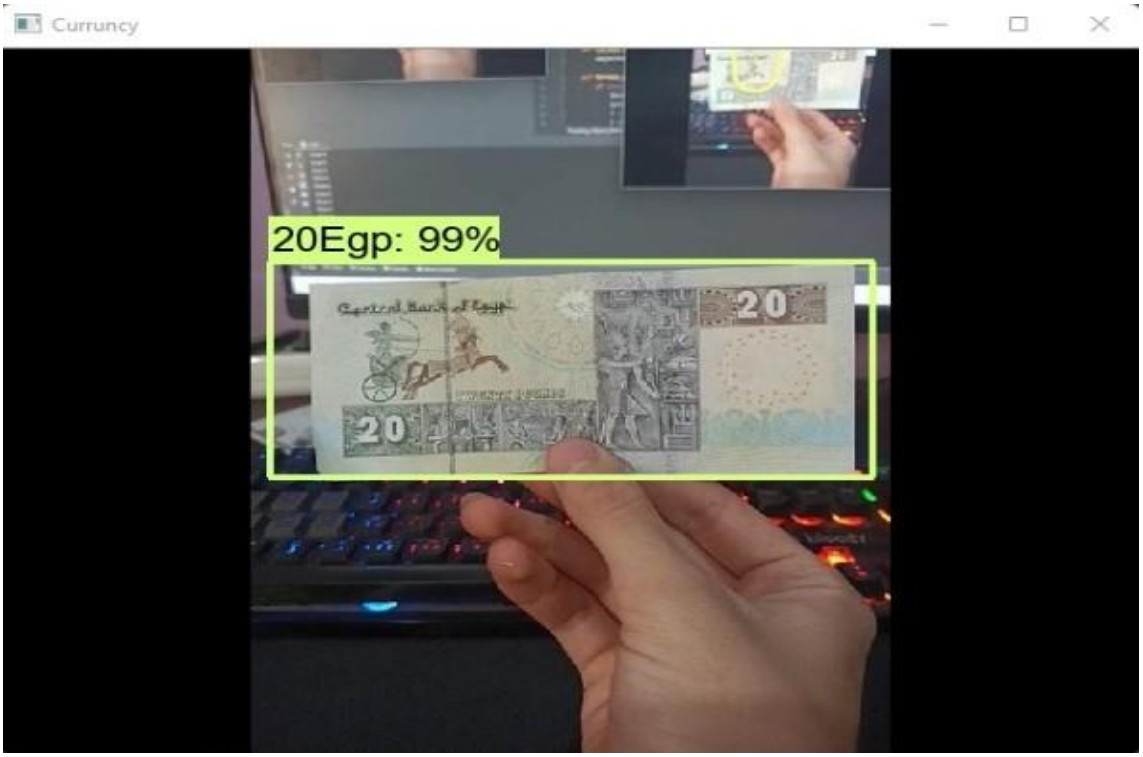
```
class Detect:
    def detect_fn(self,image,detection_model):
        image, shapes = detection_model.preprocess(image)
        prediction_dict = detection_model.predict(image, shapes)
        detections = detection_model.postprocess(prediction_dict, shapes)
        return detections
```

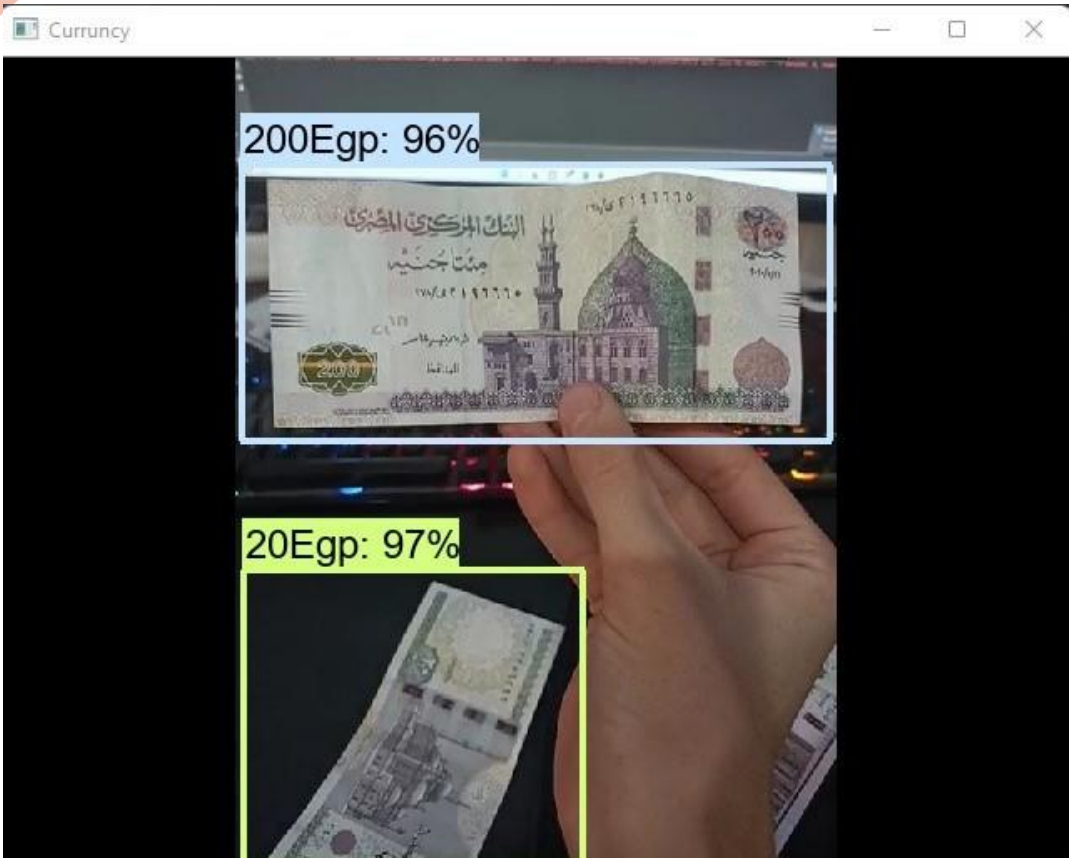
6.5 Results Snapshots:

6.5.1 Object detection:

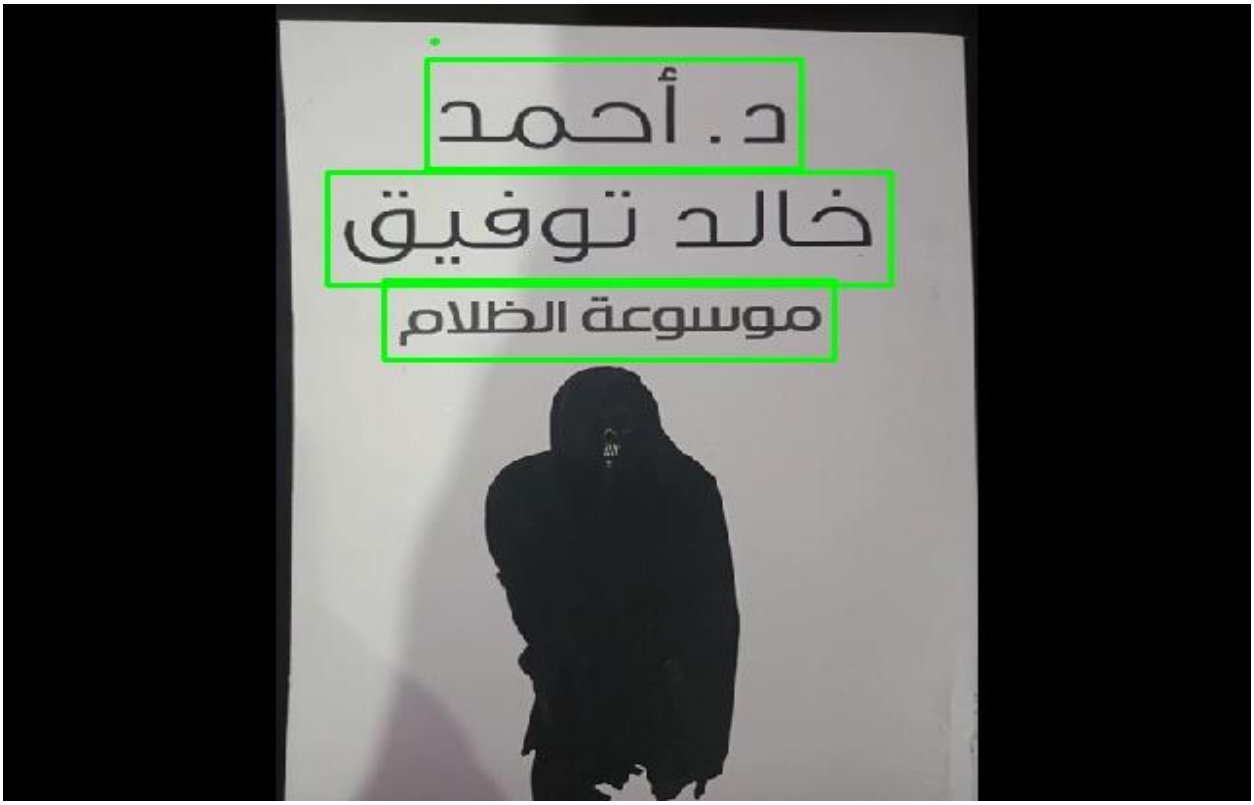


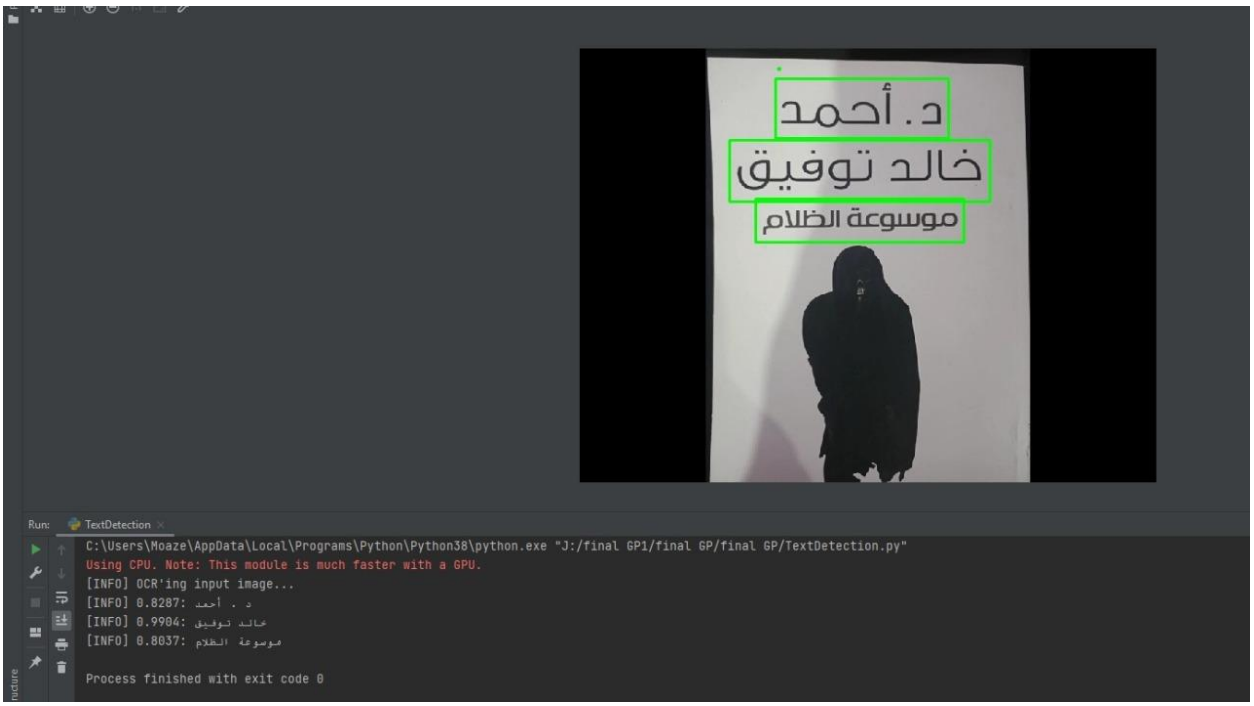
6.5.2 Currency Detection:





6.5.3 Text Detection:


د. أحمد
خالد توفيق
موسوعة الظلام


د. أحمد
خالد توفيق
موسوعة الظلام
Run:
TextDetection
C:\Users\Moaze\AppData\Local\Programs\Python\Python38\python.exe "J:/final GP1/final GP/final GP/TextDetection.py"
Using CPU. Note: This module is much faster with a GPU.
[INFO] OCR'ing input image...
[INFO] 0.8287: د . أحمد
[INFO] 0.9904: خالد توفيق
[INFO] 0.8037: موسوعة الظلام
Process finished with exit code 0

Chapter 7: Conclusion

In this work an object, currency and text detection model are presented.

In Object detection and currency detection, at first, we collect data about (person, chairs and stairs) and make label for them and make train in this with SDD+MobileNet V2.

Found the result of train is not very good of increase the steps more than 50,000 step and the learning rate does not change But it got worse, and we found that we can get better results when using prepared and trained data Egyptian currency because this data is used before and prepared.

We introduce in SSD, a fast single-shot object detector for multiple categories.

A key feature of our model is the use of multi-scale convolutional bounding box outputs attached to multiple feature maps at the top of the network, simple SSD model provides a useful building block for larger systems that employ an object.

A promising future direction is to explore its use as part of a system using recurrent neural networks to detect and track objects in video simultaneously.

In text detection, we were able to identify the texts through the image, identify it and translate it into sound, and found the performance was good.

Our Accuracy in Currency detection vs Accuracy in paper (Currency Recognition System for Visually Impaired: Egyptian Banknote as a Study Case)

	5 pound	10 pound	20 pound	50 pound	100 pound	200 pound
Quantity	20	20	20	20	20	20
True	18	17	20	16	19	17
False	2	3	0	4	1	3
Accuracy (%)	90	85	100	80	95	85

Accuracy in paper

	5 pound	10 pound	20 pound	50 pound	100 pound	200 pound
Quantity	**20**	**20**	**20**	**20**	**20**	**20**
True	**20**	**19**	**20**	**16**	**17**	**18**
False	**0**	**1**	**0**	**4**	**3**	**2**
Accutacy(%)	**100**	**95**	**100**	**80**	**85**	**90**

Our Accuracy

Chapter 8: Reference

1. Python 3.10.4 Documentation
   https://docs.python.org/3/library/threading.html
2. TensorFlow ssd_mobilenet_v2/fpnlite_320x320
   https://tfhub.dev/tensorflow/ssd_mobilenet_v2/fpnlite_320x320/1
3. Composable Multi-Threading for Python Libraries
   https://www.researchgate.net/publication/308419768_Composable_Multi-Threading_for_Python_Libraries
1. Real time text detection and recognition on handheld objects to assist blind people
   https://ieeexplore.ieee.org/abstract/document/7877741
2. Convert Text to Speech in Python
   https://data-flair.training/blogs/python-text-to-speech/
3. Python Google translator
   https://medium.com/swlh/google-translate-in-python-b5642c985c18
4. Deep Learning Based Fossil-Fuel Power Plant Monitoring in High Resolution Remote Sensing Images: A Comparative Study
   https://www.mdpi.com/2072-4292/11/9/1117
5. Detecting Parasites on Bees with Mobile Object Detection
   https://www.diva-portal.org/smash/get/diva2:1601979/FULLTEXT01.pdf

6. MobileNetV2: Inverted Residuals and Linear Bottlenecks
https://arxiv.org/abs/1801.04381
7. A Lightweight Object Detection Network for Real-Time Detection of Driver Handheld Call on Embedded Devices
https://www.hindawi.com/journals/cin/2020/6616584/
8. Pelee: A Real-Time Object Detection System on Mobile Devices
https://arxiv.org/abs/1804.06882
9. SSD: Single Shot MultiBox Detector
https://arxiv.org/abs/1512.02325
10. Object Detection using SSD Mobilenet V2
https://vidishmehta204.medium.com/object-detection-using-ssd-mobilenet-v2-7ff3543d738d
11. map (mean Average Precision) for Object Detection
https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173
12. Precision and recall
https://en.wikipedia.org/wiki/Precision_and_recall
https://www.arabicprogrammer.com/article/62471557815/
13. Currency Recognition System for Visually Impaired: Egyptian Banknote as a Study Case
https://www.researchgate.net/publication/287998433_Currency_Recognition_System_for_Visually_Impaired_Egyptian_Banknote_as_a_Study_Case

8.1 Data Set

1. Egyptian Currency
   https://www.kaggle.com/datasets/egyptiris/egyptian-currency
2. Object Detection
    1. Person
       https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F01g317&fbclid=IwAR3cYy-gtmpNHRJITCrL_gxLGnrJnk_46pfysLCuMyFvDaS7w96R7NGnsVY
    2. Chairs
       https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F01mzpv&fbclid=IwAR3ojTJUJ4_emrWr4R1NsCdNvW2PhSk4zvTfEQ4VLnCcEblelxgAyaGkUsk
    3. Stairs
       https://storage.googleapis.com/openimages/web/visualizer/index.html?set=train&type=detection&c=%2Fm%2F01lynh&fbclid=IwAR3cYy-gtmpNHRJITCrL_gxLGnrJnk_46pfysLCuMyFvDaS7w96R7NGnsVY