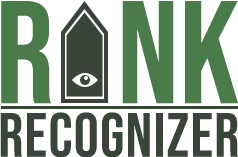
# TDK DISSERTATION 2023.

Development of a rank recognizer software



# RANK RECOGNIZER

AI/ML/DL technology on edge devices

Table of Contents

[Abstract 4](#_Toc151190268)

[Introduction 5](#_Toc151190269)

[Goals 5](#_Toc151190270)

[Tasks 5](#_Toc151190271)

[Targeted groups 6](#_Toc151190272)

[Utilities/Information added-value (estimation) 6](#_Toc151190273)

[Motivations 7](#_Toc151190274)

[Literature/backgrounds 7](#_Toc151190275)

[Diving into the AI/ML/DL 7](#_Toc151190276)

[ML is what I choose - and the off-the-shelf product 12](#_Toc151190277)

[About recognition – in general 12](#_Toc151190278)

[About testing - in general 14](#_Toc151190279)

[About benchmarking 14](#_Toc151190280)

[Own developement 15](#_Toc151190281)

[What we need: 15](#_Toc151190282)

[Scenario: 15](#_Toc151190283)

[Preparation 16](#_Toc151190284)

[Think before code 16](#_Toc151190285)

[An other problem I can define 18](#_Toc151190286)

[Testing TensorFlow 19](#_Toc151190287)

[A little success after struggling 21](#_Toc151190288)

[Creating and apply own datasets for custom objects 23](#_Toc151190289)

[Crossing fingers 27](#_Toc151190290)

[Conclusion 28](#_Toc151190291)

[Discussion 29](#_Toc151190292)

[Conclusions, future planning 31](#_Toc151190293)

[List of Abbreviations 32](#_Toc151190294)

[List of figures 33](#_Toc151190295)

[References (downloaded at: 2013.11.15): 34](#_Toc151190296)

## Abstract

The goal is to use some ML and AI with predefined python libraries to recognize military ranks (with at least 70% accuracy) on edge devices. I want to show how simple yet powerful is to use „big datas”, unique techniques, predefined data-sets on Raspberry Pi. Cheap, fast, accurate.

The method is to use predefined off-the-shelf products and modify it on the code level to customize for or my own purpose. The idea behind this project is to make a software to recognize predefined parts of a real-time video input.

But that is not enough. The main concept is to use it everywhere where it needs to use, so for the resource saving I want this project to run a little, optimized, portable device, which is fast with pre-trained models and enough to do its job.

According to the resource saving concept it is not just about energy, computational capacity, or phisical ram/cpu/gpu performance. Is about how to work absolutely alone, in "the dark". Without no internet connection. Under the radar.

If I am succeed, this project will do this, and will fill the gap between the bigdata-analysis and the edge devices.

How I mean that:

Cheap:

Most of the edge devices are cheap (excluding the smartphones which count as edge devices by they limitations, but absolutely not cheap). Usually they created for teaching and/or learning purposes and school projects (for example: arduino, raspberry pi) . If I can do something with the AI/ML/DL concept, that will be a huge thing for me.

Fast:

Fast enough. The time-perspective (or management) is depends on the problem structure what I want to solve. Edge devices attachments (objectives, cameras, passive-infrared sensors so on...) can replaced by expensive and professional tools. It means the computing is for the cheap device, but if anybody wants to feed the device with more accurate data, the limit is only the money. These devices can process inputs from almost any kind of devices.

Accurate

I mean accurate enough, as I mentioned before. What, when, why I want to do it depends on the actual problem I want to solve. In this project I just want to prepare the Raspberry Pi to recognize 3 different kind of rank symbol of a specific military. My little goal is to train the model and recognize the symbol in normal circumstances. If the problem went hardcore or difficult, I do not need to enrich the program, only the tools (as I also mentioned before)

In this case I have enough time to train the model, do the rest of the job on high level computer capacity, but when I want to use in real, the thing needs to be cheap, fast, accurate.

## Introduction

### Goals

In this case the goals is to recognize the predefined, pretrained custom model set, without any other object (without any noise).

### Tasks

* Gather some HW tool to make it happen. I need a Raspberry Pi4, usb webcamera, some python programming skills.
* I need to familiarize with online tools (Google Colab[[1]](#footnote-1), Github Codespaces[[2]](#footnote-2)) to borrow some capacity (mostly GPU) for the model training.
* Make different scripting and programming languages and dependencies working together.
* Learn a lot about AI/ML/DL
* Define more and more gaps, problems.

### Targeted groups

In this case there is no specific target group, but if I need to say something, the target is to reveal: AI/ML/DL is not a kind of extremely difficult to use (not to learn, learning and understand it is extremly high level). Using premade tools is not extremely hard.

### Utilities/Information added-value (estimation)

As the perspective of utilities there is a couple of them on the net, so everybody can find and use anything they want. Or if somebody is skilled enough, they can write they own code/tool to make things happens.

The information added-value is a different kind of perspective. It depends on the problem we want to solve. I have two example, one is a kind of 'peaceful' the other one maybe more realistic:

#### added value #1: peaceful version:

Let's say, our job is to optimize supply chains during a catastrophic situation (I know that is not too peaceful until that point, but wait..). At this kind of disaster there are multiple nations, multiple organizations appeared at once and try to organize they work to do they best. If the program can count and make some statistic about who, wher, from when comes, maybe later on we can learn about that kind of information. For example 20 WHO member shows up and 50 local disaster-prevetion team member and than we got a final result how many people was saved during the operation, later we can manage the proportion of the on-side teams.

#### added value #2: less peaceful version:

Let's say, our job is to observe a hostile military base. Our friendly forces planning a raid, when multiple high level MVP (most valuable person) are at the same place. We train our program to the enemy forces ranks, and when -for example- the program counts more than 3 generals at the base, we can alert our forces to start the attack manoveur.

### Motivations

My motivation to make this program is to learn how AI/ML/DL really works with tools. After this I wanted to create it for a specific segment and I choose edge devices. There are multiple opportunities in this case, because the trained objects can be endless, and also the edge devices (cellphones, smart gadgets, raspberry so on) can be thousands.

Among other things, recognizing a specific rank or company logo can be a kind of helpful/military planning to solve problems automatically (as mentioned above).

## Literature/backgrounds

### Diving into the AI/ML/DL

At first when I want to understand what I want to do with this recognizer, and what "kind of machine help" do I need, the main step is to look after the differences and define which aspect of the learning is fits for my project.

#### The AI

The main perspective of the AI is in simple words is about two parts. The artifical part is stands for the basic learning/data process, the intelligence part is to mimic the operation of the human brain. As far as I thought to recognize pre-trained objects does not needs so complex algorithms. These programs can learn, adapt, rethink and find new solutions to complex problems. The common thing is -until this point- is a kind of dataset to feed the program. And also there are three basic skill what AI stands on and in this case I do not need: reasoning, learning, self-correction. My goal is to train simple objects, so in this case I do not need to the three things mentioned above.

#### The ML

Basically the ML is what can "learn" only in it's own experiences. In this case the human cooperation is more important during the learning process. During this, the program can recognize patterns between the dataset just to observing it. The success is mostly based on the data-set I provide. More data=more accurate results (more data=more patterns to search, and connect with each other to the future decision).

#### The DL

This is a part of the ML family. It learns with neural networks (just like our brains). The DL works from much larger datasets then the ML, and the pattern-recognizing and pattern-connecting algorithms is much sophisticated and more complex. The last part which is a huge difference between the DL and the ML is the DL contains a prediction mechanism which is controlled by itself (it sounds like almost a "decision making" process, but it cannot correct itself by other inputs, just get sophisticated "answers" by the recognition algorithms.

#### Main differences [[3]](#footnote-3)

|  |  |  |
| --- | --- | --- |
| Artificial Intelligence | Machine Learning | Deep Learning |
| AI stands for Artificial Intelligence, and is basically the study/process which enables machines to mimic human behaviour through particular algorithm. | ML stands for Machine Learning, and is the study that uses statistical methods enabling machines to improve with experience. | DL stands for Deep Learning, and is the study that makes use of Neural Networks(similar to neurons present in human brain) to imitate functionality just like a human brain. |
| AI is the broader family consisting of ML and DL as it’s components. | ML is the subset of AI. | DL is the subset of ML. |
| AI is a computer algorithm which exhibits intelligence through decision making. | ML is an AI algorithm which allows system to learn from data. | DL is a ML algorithm that uses deep(more than one layer) neural networks to analyze data and provide output accordingly. |
| Search Trees and much complex math is involved in AI. | If you have a clear idea about the logic(math) involved in behind and you can visualize the complex functionalities like K-Mean, Support Vector Machines, etc., then it defines the ML aspect. | If you are clear about the math involved in it but don’t have idea about the features, so you break the complex functionalities into linear/lower dimension features by adding more layers, then it defines the DL aspect. |
| The aim is to basically increase chances of success and not accuracy. | The aim is to increase accuracy not caring much about the success ratio. | It attains the highest rank in terms of accuracy when it is trained with large amount of data. |
| Three broad categories/types Of AI are: Artificial Narrow Intelligence (ANI), Artificial General Intelligence (AGI) and Artificial Super Intelligence (ASI) | Three broad categories/types Of ML are: Supervised Learning, Unsupervised Learning and Reinforcement Learning | DL can be considered as neural networks with a large number of parameters layers lying in one of the four fundamental network architectures: Unsupervised Pre-trained Networks, Convolutional Neural Networks, Recurrent Neural Networks and Recursive Neural Networks |
| The efficiency Of AI is basically the efficiency provided by ML and DL respectively. | Less efficient than DL as it can’t work for longer dimensions or higher amount of data. | More powerful than ML as it can easily work for larger sets of data. |
| Examples of AI applications include: Google’s AI-Powered Predictions, Ridesharing Apps Like Uber and Lyft, Commercial Flights Use an AI Autopilot, etc. | Examples of ML applications include: Virtual Personal Assistants: Siri, Alexa, Google, etc., Email Spam and Malware Filtering. | Examples of DL applications include: Sentiment based news aggregation, Image analysis and caption generation, etc. |
| AI refers to the broad field of computer science that focuses on creating intelligent machines that can perform tasks that would normally require human intelligence, such as reasoning, perception, and decision-making. | ML is a subset of AI that focuses on developing algorithms that can learn from data and improve their performance over time without being explicitly programmed. | DL is a subset of ML that focuses on developing deep neural networks that can automatically learn and extract features from data. |
| AI can be further broken down into various subfields such as robotics, natural language processing, computer vision, expert systems, and more. | ML algorithms can be categorized as supervised, unsupervised, or reinforcement learning. In supervised learning, the algorithm is trained on labeled data, where the desired output is known. In unsupervised learning, the algorithm is trained on unlabeled data, where the desired output is unknown. | DL algorithms are inspired by the structure and function of the human brain, and they are particularly well-suited to tasks such as image and speech recognition. |
| AI systems can be rule-based, knowledge-based, or data-driven. | In reinforcement learning, the algorithm learns by trial and error, receiving feedback in the form of rewards or punishments. | DL networks consist of multiple layers of interconnected neurons that process data in a hierarchical manner, allowing them to learn increasingly complex representations of the data. |

### ML is what I choose - and the off-the-shelf product

After a little research I choosed the TensorFlow [[4]](#footnote-4). That is because it is free, open-source, and developed by the Google Brain Team, since 2015.

I was also happy because it supports edge-computing, which is necessary for this project. The edge-computing is a network of micro-datacenters. The main concept is to store and process critical data on these networks, and they forwards only the results of the calculation. Edge-computing enables the connected-devices to work with the data closer where the data is created and provides an alternative to send the data to big datacenters (Amazon, Google, Microsoft).

In May of 2017 Google announced the TensorFlow Lite, which is created specifically for mobile/edge/embedded devices. The created and trained models compressed then optimized to reach higher performance on smaller devices.

source: CB Insights

Figure 1. source: CB Insights

### About recognition – in general

Object recognition is an absolutely AI related solution, it can identify specific objects by its propertities. There are multiple techniques to use to detect an object propertly:

* Feature extraction & machine learning[[5]](#footnote-5)  
  The process converts the raw data (image, video, etc.) to numeric values to help process the data and keep the original information in the original data set.
* DL models (CNN)[[6]](#footnote-6)  
  This technique is based on the differences between the pixel and the environment.
* Bag-of-Visual-Words model[[7]](#footnote-7)  
  Same as BoW models, the extracted propertities informations are classified into groups, then it counts the visual words. The final step is to compare the extracted data to the original picture.
* Gradient and derivative based approach[[8]](#footnote-8)  
  The gradient part looks for the edge differences between the data and the original object. In shortly: it looks for the "similarity of the changes". The derivative based approach is about the velocity of the changes (derive).
* The Viola-Jones algorithm[[9]](#footnote-9)  
  Recognize a variety of objects, including faces and upper bodies.
* Template matching[[10]](#footnote-10)  
  This image recognition technique is tries to find small parts of an image which are matching one pattern of the original dataset. The more pattern matches the more accuracy the algorythm has.
* Image segmentation and blob analysis[[11]](#footnote-11)  
  The purpose to isolate objects (regions) in the input image.

### About testing - in general

At the first part of the testing process we need to create a test-dataset which is never "seen" by the software. After that, the created detection model needs to be loaded and testing with the test-dataset to verify how accurate the recognition is.

The output is used to evaulate the operation and the performance. The next step is to analyse the output errors, and do some fine tuning.

All of the testing process steps repeated until we are statisfied with the results.

### About benchmarking

For the object detection models tested, the performance was evaluated using a custom dataset on a Raspberry Pi 4.

The testing models was:

* SSD-MobileNet-v2
* EfficientDet-D0
* SSD-MobileNet-v1
* EfficientDet-Lite-D0

These models were assessed based on inference speed, FPS achieved in different scenarios, and accuracy per the COCO metric (mAP @ 0.5:0.95)[[12]](#footnote-12), along with the total number of objects correctly labeled in 75 test images.

Summarized results:

* SSD-MobileNet-v2 (quantized):  
  Inference time of 68.96 ms, COCO mAP score of 60.99%. This model balances speed and accuracy well.
* SSD-MobileNet-v2-FPNLite:  
  High accuracy, detecting 306 out of 335 total objects in the test images. It's slightly slower than SSD-MobileNet-v2 but offers excellent accuracy.
* EfficientDet-Lite-D0:  
  Placed fifth in accuracy and fourth in speed among the tested models, showing middle-of-the-road performance.

Additionally, in a specific use case of TensorFlow Lite for on-device fetal ultrasound assessment, leveraging a TensorFlow Lite GPU delegate resulted in a 2x speed improvement with no loss in model accuracy, achieving real-time inference of more than 30 frames/second with both the gestational age and fetal presentation models running in parallel on Pixel devices.

These results offer insight into the trade-offs between speed and accuracy in TensorFlow Lite models but do not provide a comprehensive overview of all TensorFlow Lite model accuracies.

As we can see, the results shows us a couple of opportunity to get higher accuracy like using GPU instead of CPU, or making more pictures about the objects. There are a couple of way to fine-tune our object model recognition accuracy and benchmark, depends on what is enough for us.

## Own developement

### What we need:

* some programming skills in Python
* Raspberry Pi 4
* any kind of usb webcam
* 3 different rank symbols

### Scenario:

3 of our soldiers has birthdays at the same week, and we organized a suprise party for they. The general orders them to come at the canteen between 18:00-19:00. Everything is locked and loaded for the party, and when they are in a same place at the same time, the party-balloon will shows up, and the confetti-cannon comes into operation. (Collegues, friend hidden in multiple places just for the suprise, when we start the observe process.)

In this case we got an order to observe a friendly military base (FMB) canteen, from distance. Our intel tells us to watch only the 3 different ranks, and when they are on the base together the party will automatically start.

### Preparation

First I try to learn and deploy an off-the-shelf object detection to the Pi. The main concept is to use TensorFlow which has a built-in object detection library for common objects (car, keyboard, mouse, apple etc.). After that I will train my custom object detection, to recognize the three different ranks. At the last step the training data will be only my custom, so just the ranks will be recognized by the program. Little differences between the good quality svg images and the reality (the reality what we need to work on..)

### Think before code

As the object-detection is a complex part of the AI/ML, I defined the possible problems before start coding and making images to the dataset. Accodring to a couple of learning material its reveals some „quality” checks which I need to define. The following table shows us what can be the possible problems between recognizing a virtual, good quality picture and the real life „captured” picture.

Tabelle 1. Good quality pictures vs real life pictures

|  |  |  |  |
| --- | --- | --- | --- |
|  | OR-08\_HUN-tzls | OF-01\_HUN-fhdgy | OF-02\_HUN-szds |
| HQ images | Picture 11 | Picture 12 | Picture 13 |
| reality | Picture 14 | Picture 16 | Picture 17 |

As we see, the program needs to prepared to deal with a couple of quality-problems. The golden rule of the datasets is: more data=more accuracy. In this case I will make 15 pictures / objects, with different angles and different distances (and then crossing fingers, I hope it will work).

The following table shows us the method of the „picturing”. I choose 3 different distances, and 5 different POVs.

The „orig” is the original object, bigger circle is refers to the object in 3d, the green circle is the position and distance of the camera (the more the green-circle is smaller, the distance is increasing).

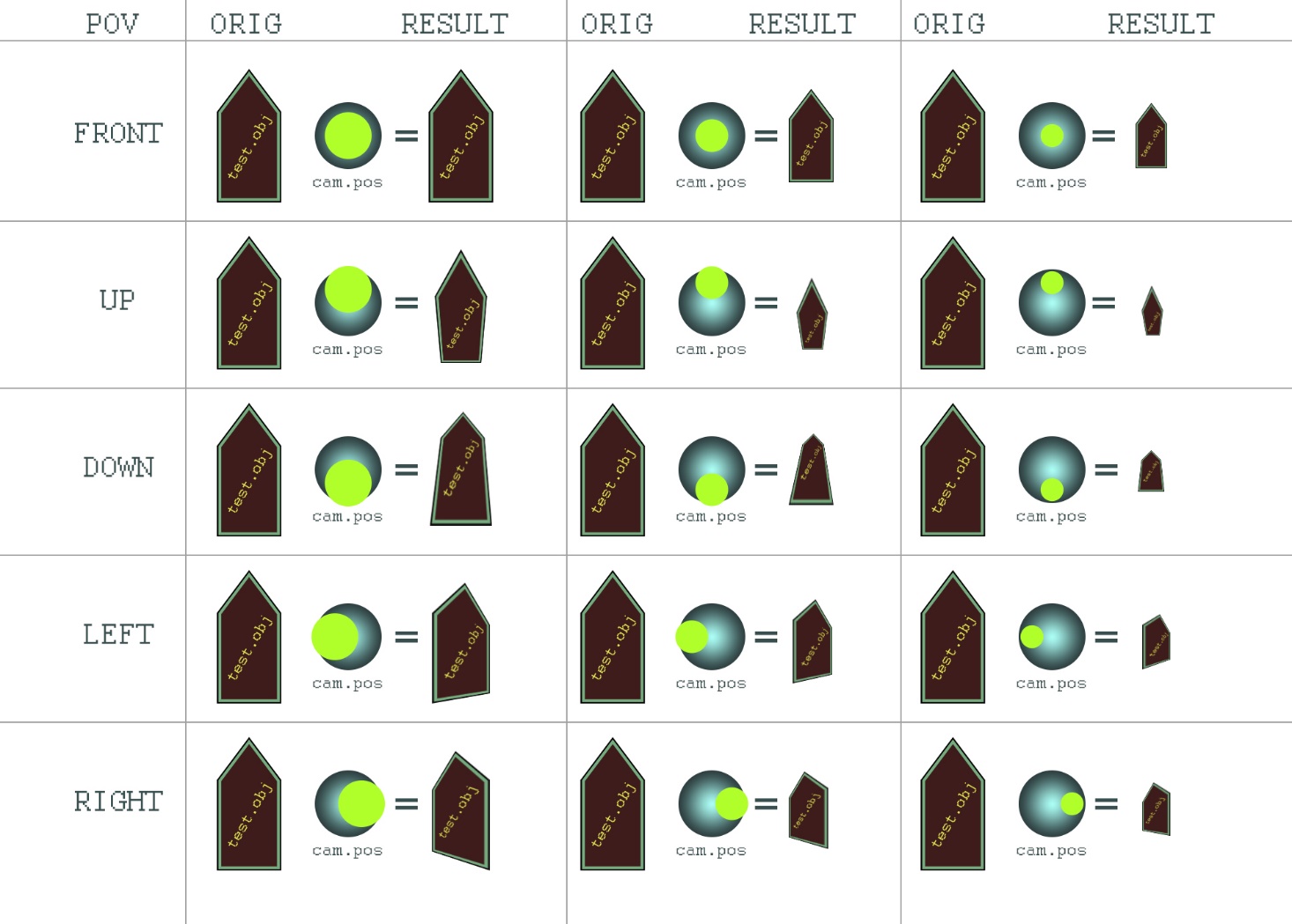


Figure 2: "picturing"method. The size and position of the object and the camera.

This is the pseudo-table how the custom-object dataset was formed. At this point I have 15 picture/object so I have 45 picture.

### An other problem I can define

I have not enough knowledge about how ML really works, but as far as I know, everything which can be a problem to recognize an object for the human eye, can be a problem for the machines too. I choose these 3 ranks because its has obvious differences, but also has obvious common signs. Here is an animated version about the ranks, if you focus what is the difference, it is hard to tell what is the common, if you focus the common, it is hard to tell what is the difference.

Figure 3: differences and similarities

|  |  |  |
| --- | --- | --- |
| Picture 11 | Picture 12 | Picture 13 |

### Testing TensorFlow

Just to make sure, I start my favorit part: coding/make things practical. I downloaded and installed the latest Tensorflow module with pretrained models (casulal objects, more than 500). After installing there was a detect.py, to start the program.... but...

At this point after a couple of error messages, I realized the fresh, newely boughtRaspberry Pi Camera Module 3 NoIR camer module will not work.

cmd: v4l2-ctl --list-devices

msg: bcm2835-codec-decode (platform:bcm2835-codec):

/dev/video10

/dev/video11

/dev/video12

8M Web Camera: 8M Web Camera (usb-0000:01:00.0-1.3):

/dev/video0

/dev/video1

As you see, the output of the v412-ctl --list-devices shows bcm2835-codec-decode devices (at this time the video10 is the attached NoIR module, the usb webcam is the video0.

Without the usb device there was a couple of error message:

msg: ImportError: /lib/aarch64-linux-gnu/libstdc++.so.6: version `GLIBCXX\_3.4.29' not found  
This is related to C++ and also has some Python dependencies.

After sudo apt-get install libgtk-4-dev and edit the path like GTK\_PATH=/usr/lib/x86\_64-linux-gnu/gtk-4 this error was gone, but...

msg: global /tmp/pip-req-build-r02f5qx8/opencv/modules/videoio/src/cap\_v4l.cpp (890) open VIDEOIO(V4L2:/dev/video0): can't open camera by index

The situation went hardcore, because I satisfied all of the programs dependencies, that seems like some kind of permission related thing. I do not know what was the problem, but I got a diferrent error msg when is started the detect.py with sudo. Error message gone, but...

msg: ModuleNotFoundError: No module named 'cv2'

Yeah, that is a kind of Python2 vs Python3 related error (the cv2 was installed before). After a little workaround (more than 4 hours), this also gone, but...

msg: Created TensorFlow Lite XNNPACK delegate for CPU.

That is related an old TensorFlow bug, which shows up specific devices with specifik python and os versions. After running into some dead github issue forums this error msg **never went away**, at this point I decided to stop the hunting and plug an usb device in.

### Picture 1A little success after struggling

Figure 4: recognition works on default training data

After successfully starting the program, the detect.py started to work with the pretrained samples. As I mentioned before, according to the description, there are more than 500 common object trained.

In these pictures you can see, it can recognize tv, bottle, and also persion and dog in the „tv”, which is -lets say- very nice.

It is called object-in-object detection, so it is related to my project.

The other interesting thing is, I think the easiest „object” it can detect is a person/human (in this case my favorite actress, Emily Blunt on the screen). Maybe it has the most sample from humans (let’s mention FaceApp, social media etc).

A simple ring finger is more than enough to recognize, which is also nice.

Taking a closer look to the code I try to figure out how an what can I modify to add some fun to the project. My other (optional) goal is to speak if the specific custom object was found.

And also there is a sign of an other problem. This default detect.py can run approximately 4-6 fps, which is maybe not enough to recognize objects in movement.

I tested the human detection in move, and it works very well, later on we will see if it works or not.

Another lesson learned: never ever try to initialize your camera at the first stage via rdp. Without knowing too much about the protocol I believe there is no acutal physical display, all of the picture is generated via rdp.

Therefore at the first time your built-in camera will not found as a display or output device by the program. This is not about the usb device, is about the additional hw connected / embedded cameras.

### Creating and apply own datasets for custom objects

#### Step 1.: Collecting/labeling training images

For the labeling I use LabelIMG[[13]](#footnote-13) tool which is a lightweight linux program. I need select and label all of the images one-by-one.

|  |  |  |
| --- | --- | --- |
| Picture 1  Figure 5: First Lieutenant rank in close | A screenshot of a computer  Description automatically generated  Figure 6: Captain rank from distance | A close up of a camouflage jacket  Description automatically generated  Figure 7: skewed Chief Warrant Officer 2 rank |

Not the most interesting part of the project...

After thats was saved, a couple of xml file created with the following contents:

<annotation>

<folder>tzls</folder>

<filename>IMG\_20231028\_153723.jpg</filename>

<path>/home/kali/Desktop/orig/tzls/IMG\_20231028\_153723.jpg</path>

<source>

<database>Unknown</database>

</source>

<size>

<width>2250</width>

<height>4000</height>

<depth>3</depth>

</size>

<segmented>0</segmented>

<object>

<name>tzls</name>

<pose>Unspecified</pose>

<truncated>0</truncated>

<difficult>0</difficult>

<bndbox>

<xmin>381</xmin>

<ymin>430</ymin>

<xmax>1943</xmax>

<ymax>3389</ymax>

</bndbox>

</object>

</annotation>

Obviously the xmin, ymin is the lower left corner of the selection, xmax and ymax is the upper right corner. This xml file called Pascal VOC (PASCAL Visual Object Classes Challenge[[14]](#footnote-14))

Additional interesting fields and descriptions:

*segmented*

This field signifies if the images contain annotations that are non-linear (irregular) in shape - commonly referred to as polygons.

*object: pose*

Specifies the skewness or orientation of the image. By default, it is specified as Unspecified, which means that the image is not skewed.

*object: truncated*

Tells if an object is fully or partially visible (can be either 0 or 1 respectively).

*object: difficult*

Tells if an object is difficult to recognize from an image (can be either 0 - easy or 1 - difficult).

#### Step 2.: Training the custom model

Instead of write a lot of code, and by a lot of GPU rig I use TensorFlow Lite Model Maker[[15]](#footnote-15) to create the object model from the dataset. Also Google Colab will be my friend, because they offer computing capacity for free.

At the first step I install the necessary packages. Of course everything is free, but many times free things not working.. Somewhy the colab disk space becomes full at the middle of the tflite-model-maker installation process.

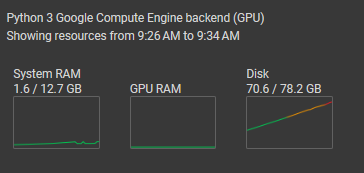


Figure 8: indicator of fail, the disk become full

A screen shot of a computer

Description automatically generated

Figure 9: mounting the own google drive storage as an external disk wont help

|  |
| --- |
| Picture 6  Figure 10: os error said no space left on device |

#### Step 3.b.: Keep doing, never give up

I kept struggling at least two days to get some workaround, because there was a couple of problems/errors/solutions:

The default Google Colab disk space is not enough to build the tflite-model-maker (it has an endless loop because some kind of bug.

* solution: subscribe to Colab for ~11 EUR / month

TensorFlow updated version does not meets the newer Python, what causes dependency problem

* solution:
  + create a custom train.py
  + install a custom (virtual) environment to run another Python versions
  + uploading the trainining data
  + run the training script

Make sure to make some changes to the code. Because I modified and there is 3 object I want to recognize, not one, the code was modified something like:

|  |  |
| --- | --- |
| original code | modified code |
| # Load Dataset  train\_data = object\_detector.DataLoader.from\_pascal\_voc( 'default/train',  ['objectTest']  ) | # Load Dataset  train\_data = object\_detector.DataLoader.from\_pascal\_voc( 'RankRecognizer/train',  ['tzls', 'fhdgy', 'szds']  ) |

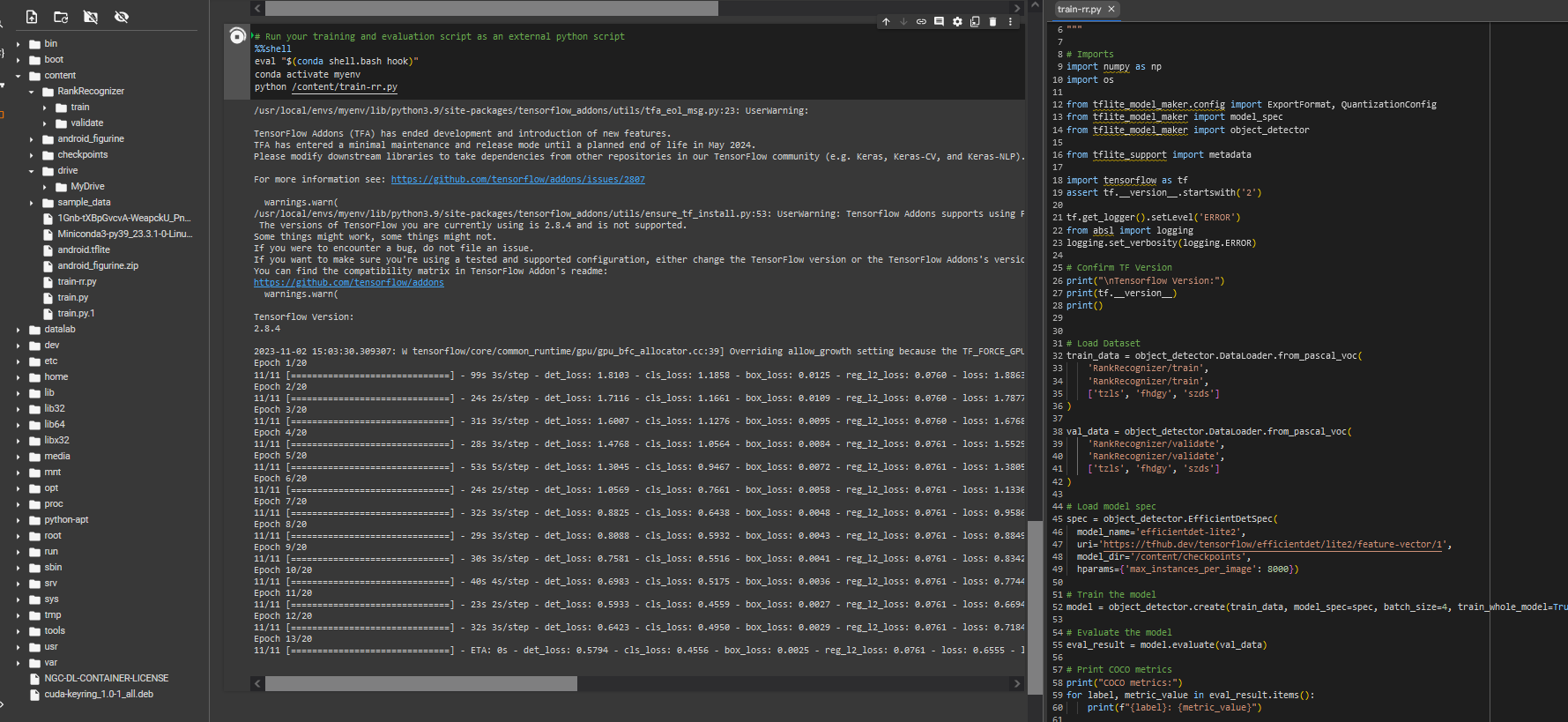


Figure 11: This is how my Google Colab flow looks like.

### Crossing fingers

Copy the .tflite file to the raspberry, then run the detect.py againt with an additional parameter:

cmd: python detect.py --model rankrecognizer

Then, magic.

|  |  |
| --- | --- |
| Picture 2  Figure 12: Chief Warrant Officer 2 rank recognized | Picture 3  Figure 13: First Lieutenant rank recognized |
| Picture 4  Figure 14: Captain rank recognized | Picture 5  Figure 15: All of the three ranks recognized at once |

It can recognize the objects even if they are showing up at once, which is.. let say very nice.

### Conclusion

AI, ML, DL is a very wide topic in the IT industry, and also an extremly fast growing. Anybody with a regular IT skills and little agility can train and use it for they own purposes (even on edge devices!).

So as a tool, it can be very useful.

Somebody with a widespread math skills can do more, deeper in the machine learning there are big opportunities.

The point is, the main goal (use some ML and AI with predefined python libraries to recognize military ranks) was successfully made.

## Discussion

* What can the current solution do?  
  It can recognize three differrent but also similar HDF ranks in real time. The accuracy is

|  |  |  |
| --- | --- | --- |
| captain | first lieutenant | chief warrant officer 2 |
| AP\_/szds: 0.4522112309932709  result: 45% | AP\_/fhdgy: 0.37683168053627014 result: 37% | AP\_/tzls: 0.800000011920929 result: 80% |

It can run on a Raspberry 4 edge device, with a cheap usb webcamera.

* Why it is not able to detect more objects?  
  This project is for educational and/or testing and also for fun purposes. I wanted to know how it works, and what are the barriers. The data training process is including a lot of things what is also non technical (and that is not the favorite part of my style), for example shooting images about object is a kind of boring. Among this I think I capable to sophisticate the dataset/software to reach more accuracy, more sensitive working flow.
* Is it working with an infra camera (in visibility conditions, poor darkness)?  
  Not tested yet. My theoretical approach is a 100% percent there is a solution. I am sure somehow I can train the model also well visible and less visible dataset. Acutally observing things in the dark is like observing non-colored, but monochrome colored objects. The infracam is able to provide grayscale pictures, and I think during the recognizing process the focus is not on the objects colors, but more on the object shape and other reference marks.
* Can it recongize the difference between other countries military ranks?  
  Not tested yet. Before this, it needs to a wide-range research about other military ranks/signs. If there is any which is similar to the observed one, we need to focust at the differences, and training the object model to do this.
* What about that if the observed object is truncated, or partially visible?  
  Right now the Rank Recognizer does not able to detect partially visible objects. This is because I am working only with the "clean" images. The previously mentioned LabelImg (which is for to label the images with bounding boxes) has an optional parameter: truncated (yes or no). It means is the object partially visible or fully. So everything is given to train models to recognize truncated objects.
* How about that if the object is moving, or rotated?  
  The still and rotated objects can be recognized, but the speed of the observing (like the FPS of the camera, or the speed of the object in move) can be a hard part. In this case I think if the camera is not a cheap usb one, but a modern, expensive with good parameters, the observing process can be more likely accurate.

## Conclusions, future planning

First of all, this project costs 4 days from 0 to the working state. My thoughts about AI, DL, ML and edge devices relations to sooner or later this premade solutions will work, it was true.

Using off-the-self products was a good idea, because they have a lot of detailed descriptions and also free, video and text based learning materials (like TensorFlow). That is also amazing to find a lot of free solutions, and I am happy to find this "performance-borrowing" opportunity like Google Colab or Github Codespace.

This part of the IT is new and fast growing, therefore a lot of discussions, forums, pages, groups, video tutorials etc. on the Internet.

My final conclusion is everybody with an endurance, massive interest in IT and some programming skills can develop and learn they own AI, ML, DL basics.

These are basics because for the deep understanding these topics needs an extremely high level deep mathematical knowledge in chaos theory, probability theory, calculus, algebra so on.. This is because I think this project (and the off-the-shelf products) as a usage of a "tool". Putting things together and make it work is like a samurai who use his sword. The AI, DL, ML developer is like the smith who create the samurai's sword from scratch.

This project shows the direction how to put tools together, and also touches some math topics.

There are a couple of ways where this project can be progress. What is the purpose, what is the goal(s), what we want to do with it and how. These circumstances are the "borders" or the "limitations" of the future project. As the circumstances is from also our creative fantasy, that can be as complex as we want.

## List of Abbreviations

* AI - Artificial Intelligence
* DL - Deep Learning
* ML - Machine Learning
* RAM - Random Access Memory
* CMD - command
* MSG - message
* CPU - Central Processor Unit
* GPU - Graphical Processor Unit
* WHO - World Health Organization
* MVP - Most Valuable Person
* ANI - Artificial Narrow Intelligence (ANI) AGI - Artificial General Intelligence (AGI)
* ASI - Artificial Super Intelligence (ASI)
* CNN - Convolutional Neural Network
* FPS - Frames Per Second
* COCO - Common Objects in Context
* SVG - Scalable Vector Graphics
* HQ - Headquaters
* PoV - Point of View
* VOC - Visual Object Classes Challenge
* XML - extended markup language
* HDF - Hungarian Defence Forces
* SZDS - "százados" (colonel in hungarian)
* FHDGY - "főhadnagy" (first lieutenant in hungarian)
* TZLS - "törzszászlós" (chief warrant officer 2 in hungarian)

## List of figures

* Figure 1. cloud computing, source: CB Insights
* Figure 2: "picturing"method. The size and position of the object and the camera.
* Figure 3: differences and similarities
* Figure 4: recognition works on default training data
* Figure 5: First Lieutenant rank in close
* Figure 6: Captain rank from distance
* Figure 7: skewed Chief Warrant Officer 2 rank
* Figure 8: indicator of fail, the disk become full
* Figure 9: mounting the own google drive storage as an external disk wont help
* Figure 10: os error said no space left on device
* Figure 11: This is how my Google Colab flow looks like.
* Figure 12: Chief Warrant Officer 2 rank recognized
* Figure 13: First Lieutenant rank recognized
* Figure 14: Captain rank recognized
* Figure 15: All of the three ranks recognized at once

## References (downloaded at: 2013.11.15):

* AI, ML, DL comparison  
  https://www.geeksforgeeks.org/difference-between-artificial-intelligence-vs-machine-learning-vs-deep-learning/
* Benchmarking  
  https://www.ejtech.io/learn/tflite-object-detection-model-comparison
* A Gentle Introduction to Object Recognition With Deep Learning  
  https://machinelearningmastery.com/object-recognition-with-deep-learning/#:~:text=,one%20object%20in%20an%20image
* A Review of Object Detection Techniques  
  https://ieeexplore.ieee.org/document/8901325
* Tools, Techniques, Datasets and Application Areas for Object Detection  
  https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9033309/#:~:text=,in%20the%20past%2020%20years
* A Review of Object Detection Based on Deep Learning  
  https://link.springer.com/article/10.1007/s11042-020-08976-6#:~:text=,applications%20and%20future%20development%20directions
* Deep Learning: A Comprehensive Overview on Techniques​.  
  https://link.springer.com/article/10.1007/s42979-021-00815-1#:~:text=,0
* Optical Character Recognition with TensorFlow Lite: A New Example App  
  https://blog.tensorflow.org/2020/09/how-to-use-tensorflow-lite-for-optical-character-recognition.html
* How to Build a Simple Image Recognition System with TensorFlow (Part 1)  
  https://www.freecodecamp.org/news/how-to-build-a-simple-image-recognition-system-with-tensorflow/
* Image Classification | TensorFlow Lite  
  https://www.tensorflow.org/lite/examples/image\_classification/overview
* TensorFlow Lite Examples | Machine Learning Mobile Apps  
  https://www.tensorflow.org/lite/examples
* Achieving Lightning-Fast, On-Device Image Recognition - TensorFlow  
  https://blog.tensorflow.org/2018
* What is COCO?  
  https://cocodataset.org

1. https://colab.research.google.com/ [↑](#footnote-ref-1)
2. https://github.com/features/codespaces [↑](#footnote-ref-2)
3. https://www.geeksforgeeks.org/difference-between-artificial-intelligence-vs-machine-learning-vs-deep-learning/ [↑](#footnote-ref-3)
4. https://www.tensorflow.org/lite/ [↑](#footnote-ref-4)
5. https://www.mathworks.com/discovery/feature-extraction.html [↑](#footnote-ref-5)
6. https://www.mathworks.com/discovery/deep-learning.html [↑](#footnote-ref-6)
7. https://www.mathworks.com/discovery/bag-of-words.html [↑](#footnote-ref-7)
8. https://www.researchgate.net/publication/276576665\_Gradient\_Based\_Image\_Edge\_Detection [↑](#footnote-ref-8)
9. https://medium.com/@aaronward6210/facial-detection-understanding-viola-jones-algorithm-116d1a9db218 [↑](#footnote-ref-9)
10. https://www.mathworks.com/help/vision/ref/templatematching.html?s\_tid=srchtitle\_site\_search\_2\_Template%20matching [↑](#footnote-ref-10)
11. https://www.mathworks.com/matlabcentral/fileexchange/25157-image-segmentation-tutorial [↑](#footnote-ref-11)
12. https://cocodataset.org/ [↑](#footnote-ref-12)
13. https://github.com/HumanSignal/labelImg [↑](#footnote-ref-13)
14. http://host.robots.ox.ac.uk/pascal/VOC/ [↑](#footnote-ref-14)
15. https://www.tensorflow.org/lite/models/modify/model\_maker [↑](#footnote-ref-15)