ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

Τμήμα Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

«ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ»

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«

Smart Fridge, an object detection application for android»

ΜΟΥΖΑΚΑ ΘΕΟΔΩΡΑ

AM: icsdm417007

Επιβλέπων: Μαραγκουδάκης Μανώλης

Τριμελής Επιτροπή:

Μαραγκουδάκης Μανώλης

Βουγιούκας Δημοσθένης

Γκουμόπουλος Χρήστος

ΣΑΜΟΣ 2020

Ευχαριστίες

Η παρούσα εργασία αποτελεί διπλωματική εργασία στα πλαίσια του μεταπτυχιακού προγράμματος «ΠΛΗΡΟΦΟΡΙΑΚΑ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΑ ΣΥΣΤΗΜΑΤΑ» του τμήματος Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων.

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή Μανώλη Μαραγκουδάκη για τη καθοδήγηση και την συνεργασία μας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου για τη στηριξή της.

Table of Contents

0.0 Introduction

0.1 Summary

The aim of this thesis entitled <<Smart Fridge, an object detection application for android>> is to create a machine learning algorithm, train it with a suitable dataset in order to be able to detect and classify objects that can be found inside a fridge and create an android application that can detect and classify the objects from a mobile phone's camera or have the option to import an image for classification.

0.2 Structure

This thesis has four sections.

The first section refers to:

- What is *Machine Learning* and where it is being used
- What is *Deep Learning* and where it is being used
- What is *Tensorflow* and where it is being used
- The programming language *python*

The second section refers to:

- Collecting the dataset
- Training the model
- Evaluation of the results

The third section refers to:

- What is IDE Android Studio
- The programming language *Java*
- The implementation of the application

Section One

1.0 Machine Learning

1.0.1 What is Machine Learning

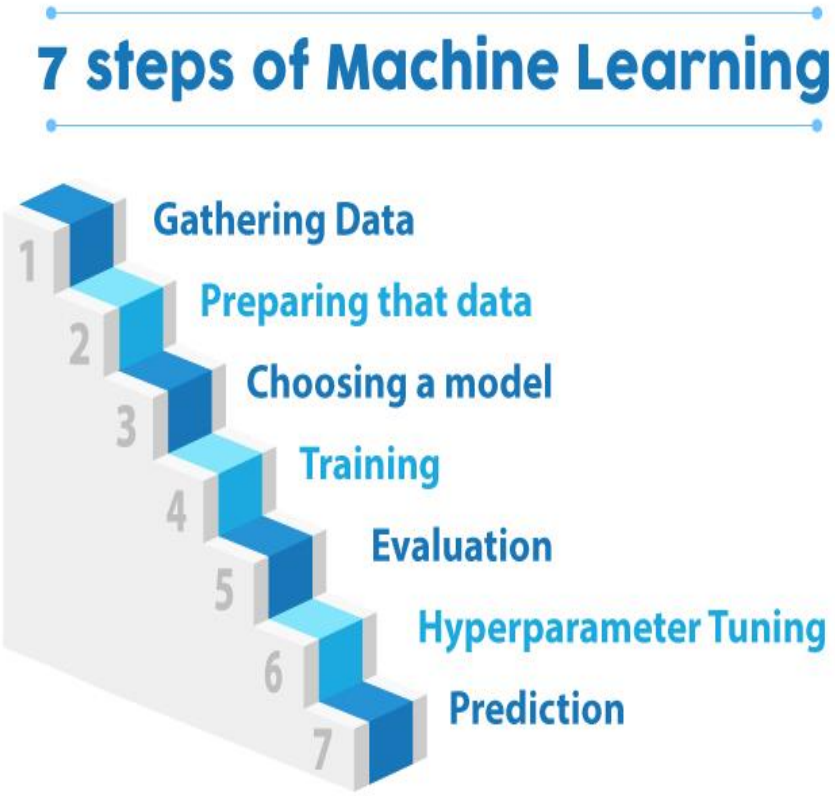
Machine Learning is a data analysis method, which is based on the idea that the machine will learn from the data, will identify patterns and it will be able to take decisions with minimal human supervision.
The machine learning algorithms use statistics in order to find patterns in huge amount of data, and by the word data here, we mean text, images, numbers, clicks and everything that is digitally stored.
Examples:

- **Image Recognition**
  This is one of the most common use of machine learning.
  It is used to classify objects inside a digital image, for example in the black and white images; the intensity of each pixel is one of the measurements and in colored images, each pixel provides 3 measurements of intensities in three different colors – red, green and blue (RGB).
  In addition, it is possible to detect a face into an image. There is a separate category for each person in a database of several people.
- **Speech recognition.**
  This is the translation of spoken words into text. The measurement in this application can be a set of numbers that represent the speech signal. We can also segment the speech signal by intensities in different time-frequency

bands. Speech recognition can be used in voice search, appliance control, data entry or the preparation of structured documents.

- **Medical diagnosis**
  In the healthcare sector, machine learning is used to analyze disease progression, patient monitoring and therapy planning. The goal is to achieve successful integration of computer-based systems in this sector.
- **Classification**
  Classification is the way to analyze the measurements of an object and place it to the category, which it belongs. For example, in the banking sector it is being used to identify if a customer that requests a loan will be able to pay it by using as factors customer's financial history, savings and earnings.
- **Prediction**
  Making predictions is one of the top machine learning applications. We can used it after classification and calculate the probability of fault.
- **Information extraction**
  It is the process of extracting structured information from the unstructured data like emails, reports, blogs etc.
- **Financial Services**

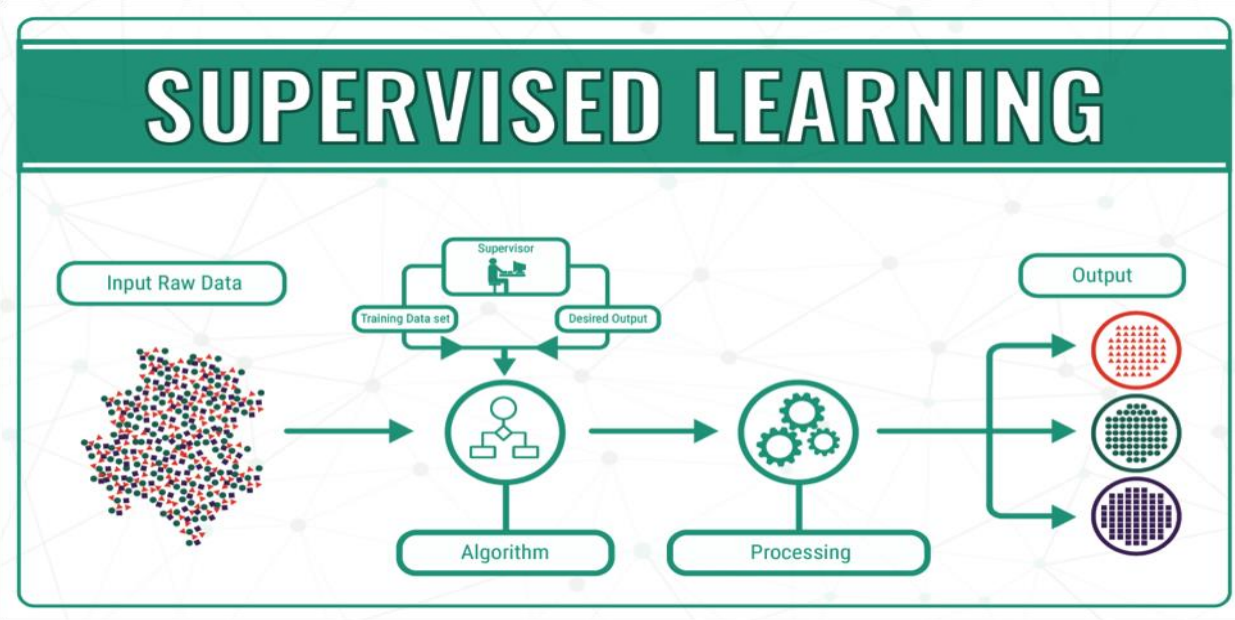
Financial institutions and banks have great advantages using machine-learning techniques. They can make smarter decisions through machine learning, predict customer's behavior, track spending patterns and predict market analysis.
The algorithms can identify the trends easily and can react in real time.



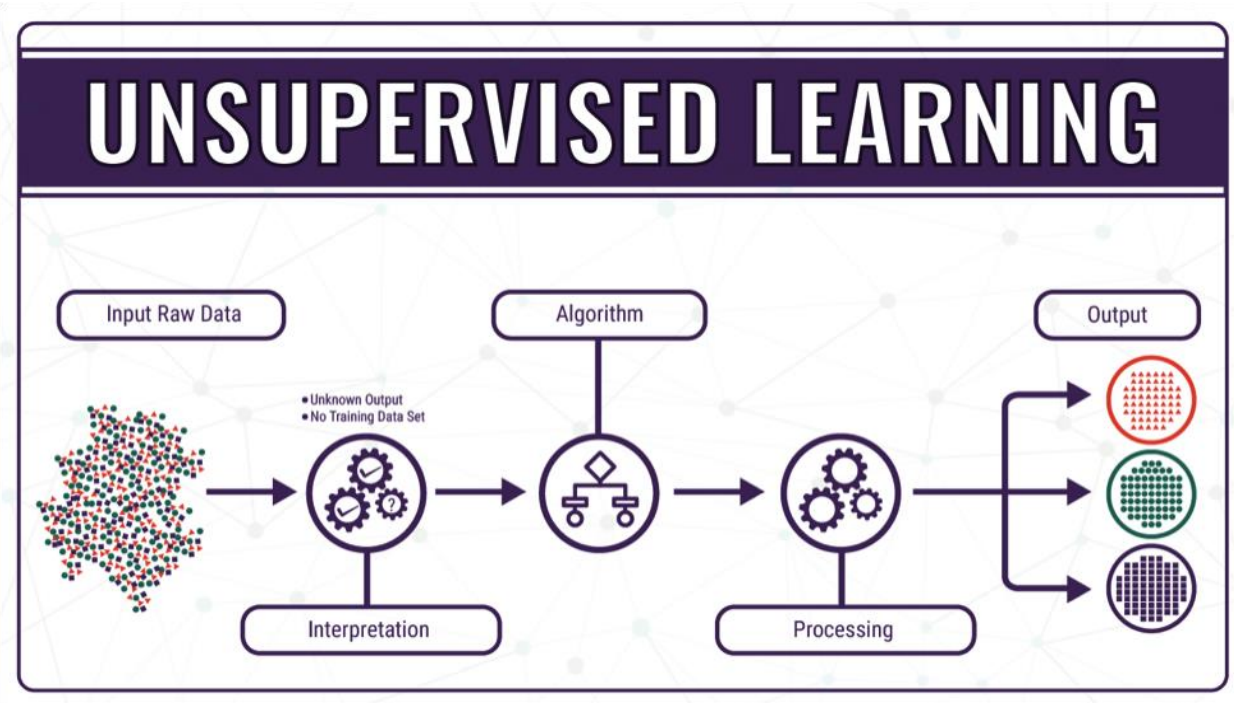
1.0.2 Types of machine learning algorithms

Algorithms can be divided into four main categories:

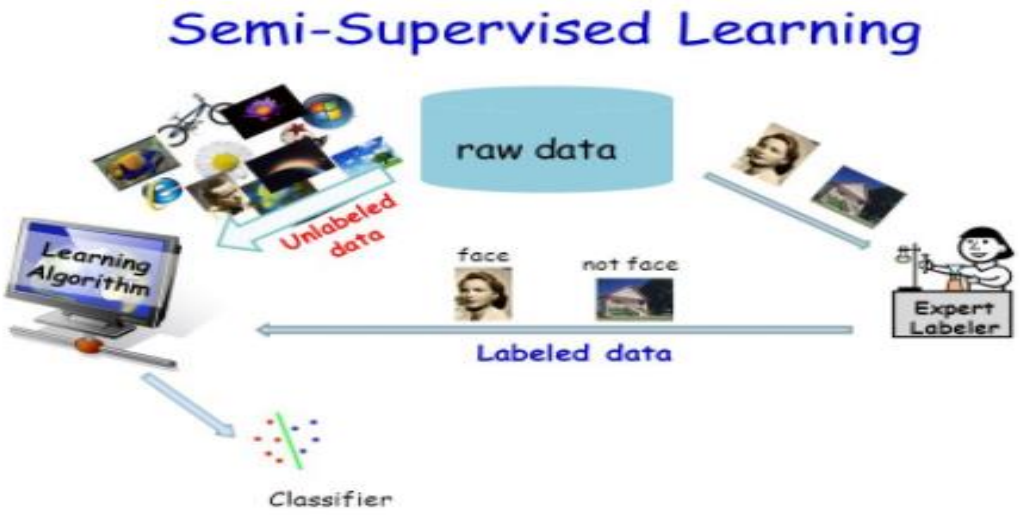
1. **Supervised machine learning algorithms**
   In the supervised machine learning algorithms, we have a monitored learning and the developer acts like a «teacher» providing the appropriate training data and the desired output. Each training example has one or more inputs and the desired output, also known as a supervisory signal.
   Supervised learning algorithms include classification and regression. Classification algorithms are used when the outputs are restricted to a limited set of values, and regression algorithms are used when the outputs may have any numerical value within a range.



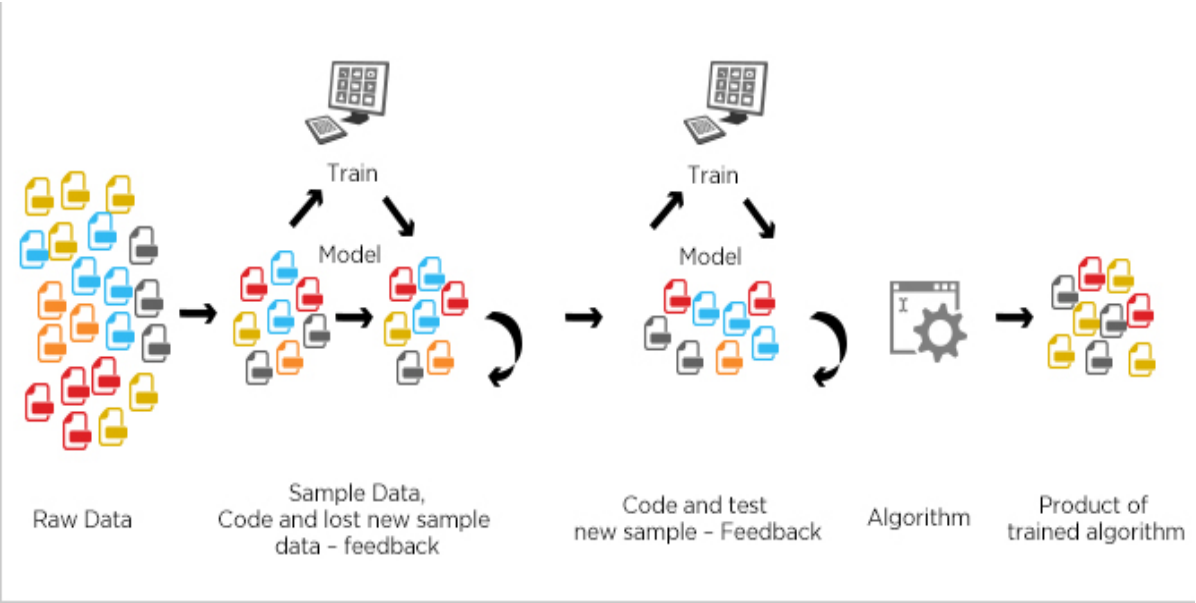
2. **Unsupervised machine learning algorithms**
   Unsupervised machine learning algorithms infer patterns from a dataset without reference to known or labeled outcomes. In contrast with the supervised machine learning, this method cannot be applied directly for regression or classification since we have no idea what the output will be and this makes difficult to determine how accurate the outcome is.



3. **Semi-supervised machine learning algorithms**
   This method is a combination of supervised machine learning and unsupervised machine learning. A small amount of labeled data is used together with a large amount of unlabeled data during training. This method produces better learning accuracy.



4. **Reinforcement machine learning algorithms**
   Reinforcement machine learning is called also *approximate dynamic programming*, or *neuro-dynamic programming*. It differs from supervised learning in not needing labelled input/output pairs be presented, and in not needing sub-optimal actions to be explicitly corrected. The main goal of this method is finding a balance between unknown territory and current knowledge (exploration and exploitation).
   Software agents and machines are working together to determine the ideal behavior within a specific context and maximize its performance.



1.0.3 Machine learning in real life

2. Social Media

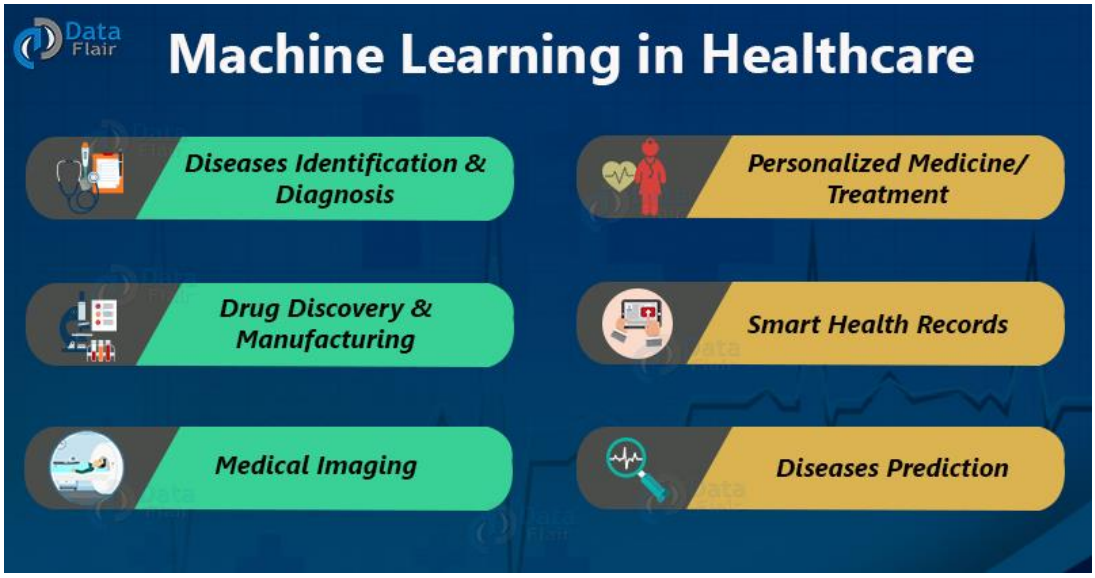
- **Brand monitoring tools**
  The question here : With so much traffic in social media feeds at any given time, how can a brand keep track of what people think?
  Tools such as Brands Eye and BuzzSumo can send an alert when your brand name is mentioned on any social network, and provide you with information about the context and the sentiment of the post.
- **Recommendations**
  With machine learning in social media playing a major role, the recommendations based on user's profile, page likes, places they visited, etc. are more than common.
- **Face Recognition**
- **Chatbots**
  Instead of browsing a brand's website to look for new products, customers can enable a Facebook messenger chatbot and ask it questions about what's in stock, what's discounted, what's new, and much more questions.

3. Automotive Industry

In the automotive industry, Artificial Intelligence and machine learning techniques together are trying to achieve the recognition of road signs, identification of patterns of human behavior to make data-driven decisions on the road. AI algorithms can predict travel time, traffic congestion, and even vehicle breakdowns. The adoption of machine learning and AI in automotive industry can also offer route recommendations based on fuel consumption and even parking availability.
Hand free driving will make drivers more focused to the road; also the safety will be highly improved with the help of machine learning algorithms, since multiple vehicle behavior models will help cars recognize the world around them.

4. Healthcare Industry

Today, machine learning is helping to streamline administrative processes in hospitals, map and treat infectious diseases and personalize medical treatments. The rate of progress keeps increasing in this sector, for example, KenSci uses machine learning to predict illness and treatment to help physicians and payers intervene earlier, predict population health risk by identifying patterns and surfacing high risk markers and model disease progression and more.



2.0 Deep Learning

2.0.1 What is Deep Learning

Deep learning, also known as deep structured learning or differential programming, is a machine learning method based on neutral networks. The goal of deep learning is to imitate the human's brain process of analyzing information and decision-making. Andrew Ng from Coursera says «Using brain simulations, hope to:

– Make learning algorithms much better and easier to use.

– Make revolutionary advances in machine learning and AI.

I believe this is our best shot at progress towards real AI»

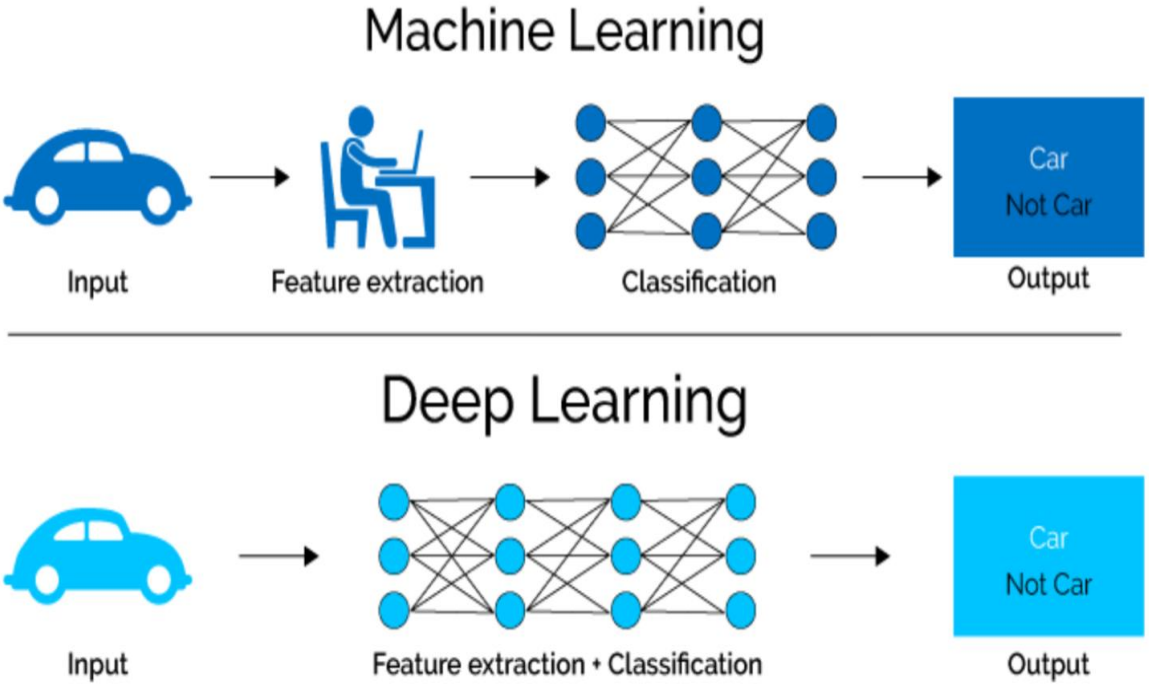
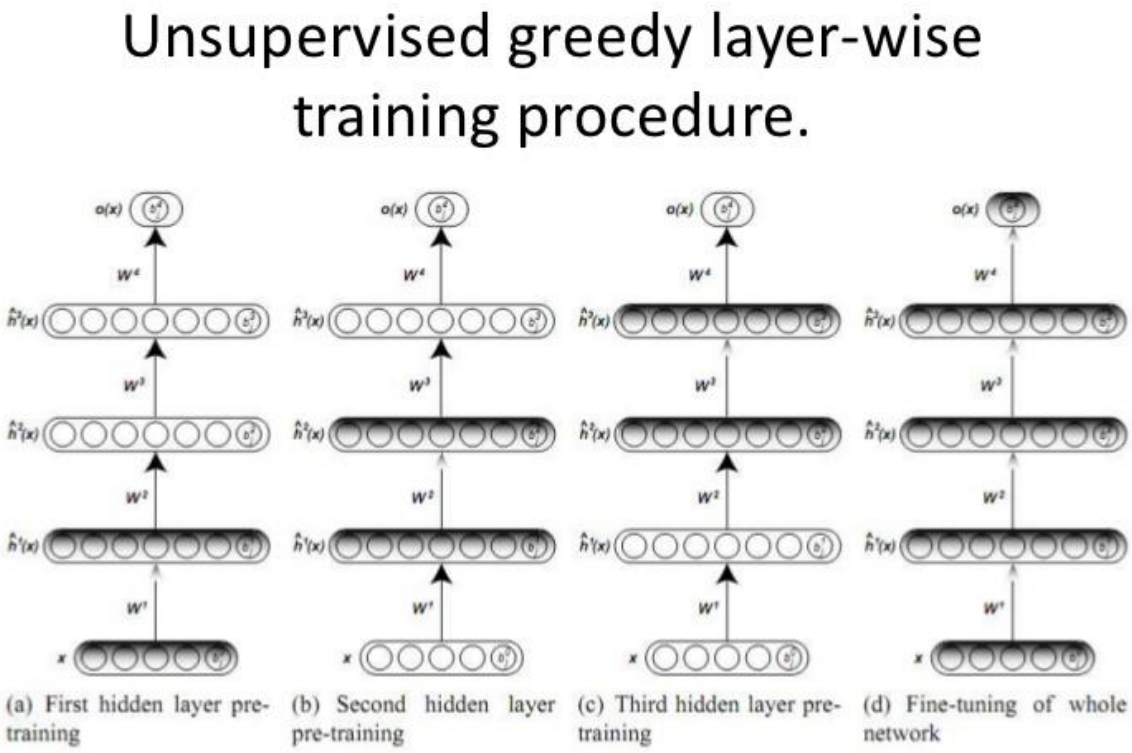


Figure 1: Machine Learning VS Deep Learning

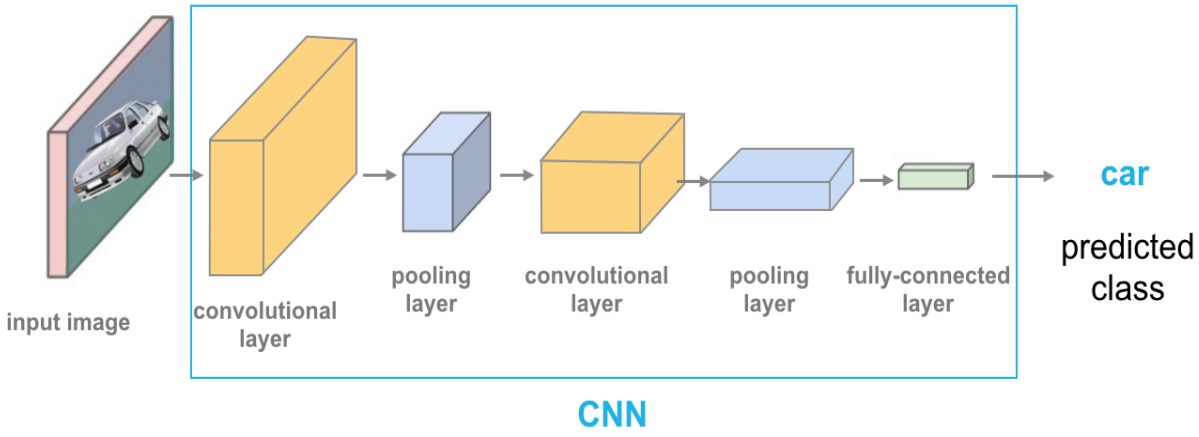
2.0.2 Types of deep learning techniques

Deep learning techniques can be divided into four main categories:

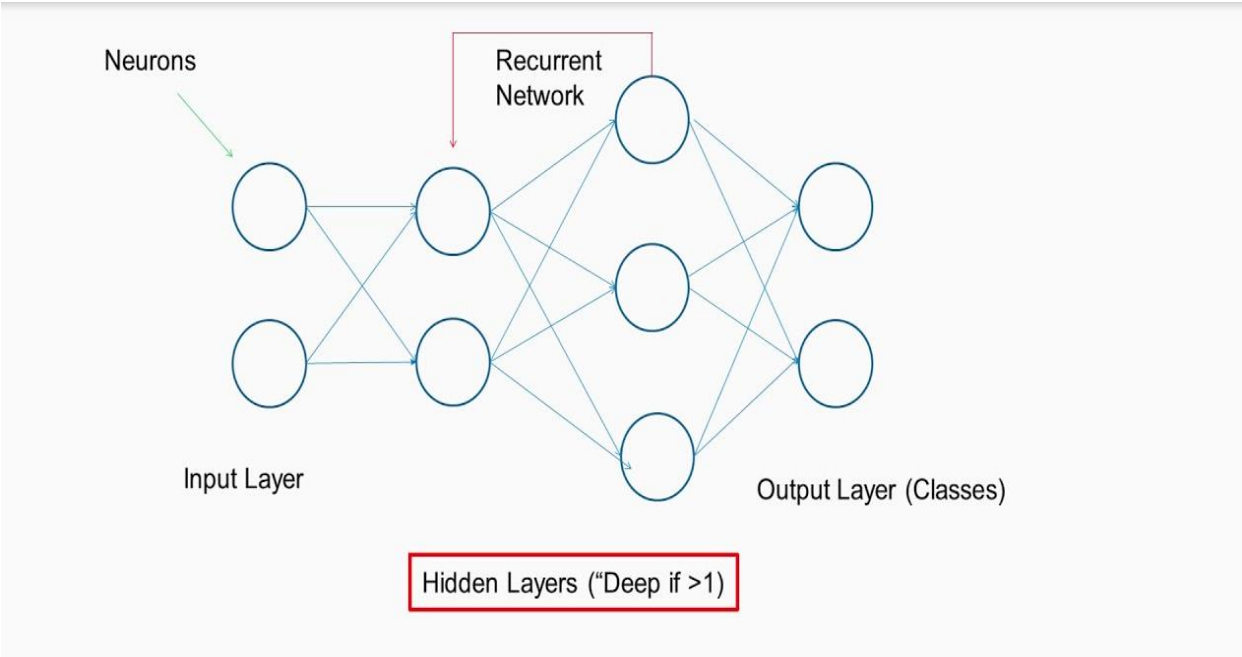
1. **Unsupervised Pre-trained Networks**
   Unsupervised pre-training sets a discriminative neural net from another which was trained using an unsupervised condition, such as a deep belief network or a deep auto encoder. This method can sometimes help with both the optimization and the overfitting issues.



2. **Convolutional Neural Networks**
   A convolutional neural network (CNN) is a technique of artificial neural network used in image recognition and processing that is designed to process mainly pixel data.
   It works by taking an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and differentiate one from the other



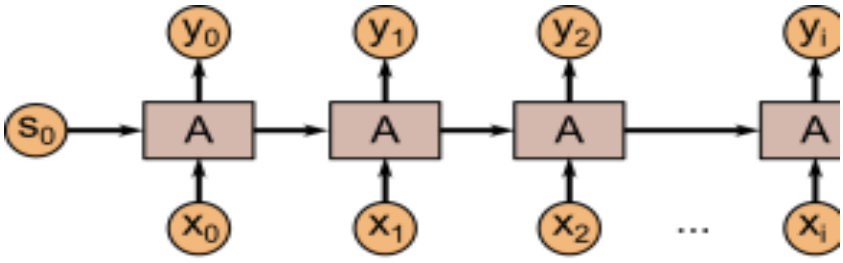
3. **Recurrent Neural Networks**
   Recurrent Neural Networks or RNNs can use their internal memory to process variable length. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.



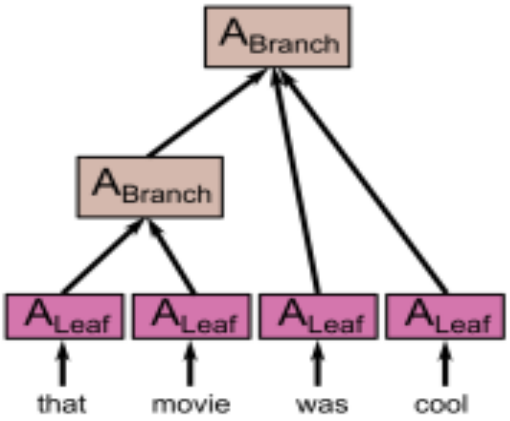
4. **Recursive Neural Networks**

As Wikipedia describes it:

«A recursive neural network is a kind of deep neural network created by applying the same set of weights recursively over a structured input, to produce a structured prediction over variable-size input structures, or a scalar prediction on it, by traversing a given structure in topological order. RNNs have been successful, for instance, in learning sequence and tree structures in natural language processing, mainly phrase and sentence continuous representations based on word embedding»



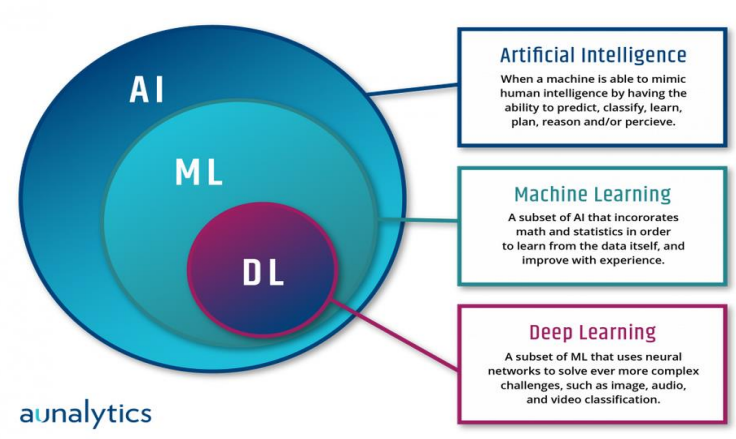
Recurrent Neural Net



2.0.3 Deep learning in real life

Here are some deep learning examples in practice:

- **Translations**
  Although automatic translation is not something new, deep learning is helping improve automatic translation of text by using stacked networks of neural networks and making possible the translations from images.
- **Colorization**
  The process of adding color to an image that was a time consuming task for humans can now be done automatically with deep-learning models.
- **Language recognition**
  Deep learning machines are able to differentiate dialects of a language. The machine can decide that the language that a human is speaking is for example Greek and can tell as the differences between dialects.
- **Computer vision**
  Deep learning has achieved great accuracy for image classification, object detection, image restoration and image segmentation—even handwritten digits can be recognized. Deep learning using enormous neural networks is teaching machines to automate the tasks performed by human visual systems.
- **Deep-learning robots**
  In the robotics field, deep learning applications are great and powerful. Through deep learning models, we have a housekeeping robot that it was trained by observing human actions. Similar to how a human brain processes input from past experiences, current input from senses and any additional data that is provided, deep-learning models will help robots execute tasks based on the input of many different AI opinions.
- **News aggregator based on sentiment**
  Advanced natural language processing and deep learning are being used to filter the news. News aggregators using sentiment analysis, can help you create news streams that only cover the good news happening.
- **Text generation**
  The machines can learn the punctuation, grammar and style of a piece of text and can use the model it developed to automatically create entirely new text with the proper spelling, grammar and style of the example text.
- **Autonomous vehicles**
  Artificial intelligence, machine learning algorithms and deep learning methods are combined together in order to make models used by cars that are able to recognize pedestrians, signs and other obstacles in the road.



3.0 Tensorflow

3.0.1 What is Tensorflow

According to Wikipedia: «TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015». It allows developers to create large-scale neural networks with many layers. The main use of Tensorflow is for Classification, Perception, Understanding, Discovering and Prediction.

3.0.2 How Tensorflow works

TensorFlow allows programmers to create dataflow structures that describe how data moves through a graph, or a series of processing nodes. Each node in the graph represents a mathematical operation, and each connection or edge between nodes is a multidimensional data array, or tensor. TensorFlow provides all of this to the developer using Python language. The math operators however are not written in Python but in C++. TensorFlow applications can be ran on most targets like a local machine, a cluster in the cloud, iOS and Android devices, CPUs or GPUs.

As described in **Infoworld**: « The single biggest benefit TensorFlow provides for machine learning development is abstraction. Instead of dealing with the nitty-gritty details of implementing algorithms, or figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application. TensorFlow takes care of the details behind the scenes. It also offers additional conveniences for developers who need to debug and gain introspection into TensorFlow applications. The eager execution mode lets you evaluate and modify each graph operation separately and transparently, instead of constructing the entire graph as a single opaque object and evaluating it all at once. The TensorBoard visualization suite lets you inspect and profile the way graphs run by way of an interactive, web-based dashboard. »

Noticeable algorithms supported by Tensorflow:

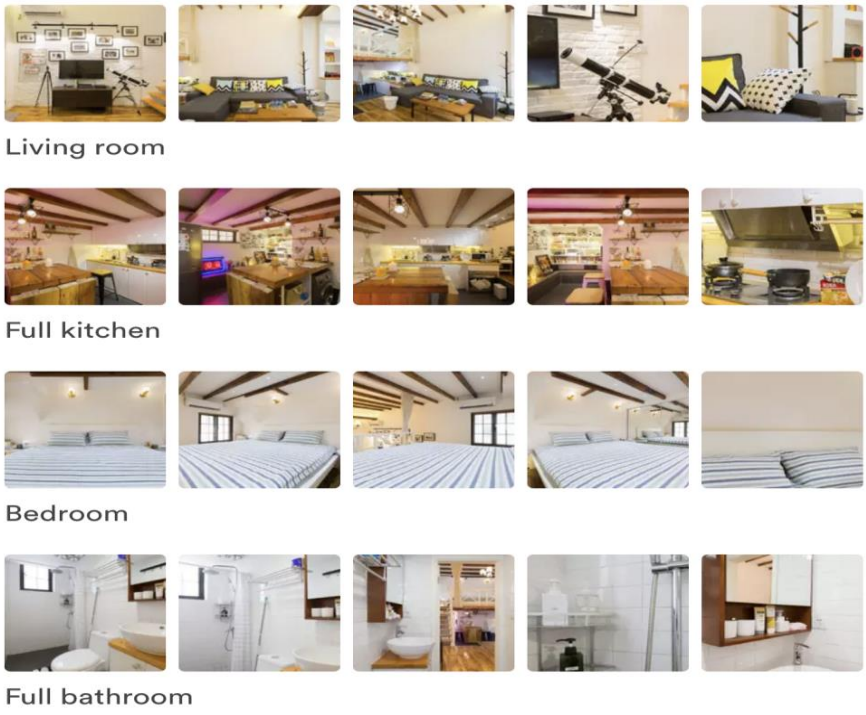
- Linear regression: tf.estimator.LinearRegressor
- Classification:tf.estimator.LinearClassifier
- Deep learning classification: tf.estimator.DNNClassifier
- Deep learning wipe and deep: tf.estimator.DNNLinearCombinedClassifier
- Booster tree regression: tf.estimator.BoostedTreesRegressor
- Boosted tree classification: tf.estimator.BoostedTreesClassifier

3.0.3 Where Tensorflow is being used

We will analyze below some examples of platforms/applications that use Tensorflow;

- **Airbnb**
  Until recently, when a customer was searching photos of houses in Airbnb there was no way he can see the most informative ones.
  With image classification by Tensorflow, we have the room-type classification. The developers trained a model based on Tenroflow in order to recognize which is the room type displayed in every photo (living room,

bedroom, bathroom etc.) so now when the user is searching for a house in Airbnb he has the photos available listed by room type and this has highly improved the his experience.



- **Twitter**
  Twitter like any other social media, uses machine learning for the following different aspects:

Ads: Personalize ads according to the user's interest.

Timelines: Providing with interesting relevant context of timelines to the users.

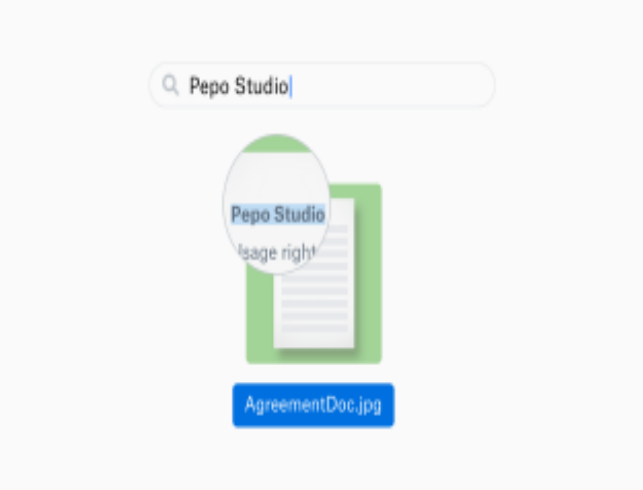
Abuse: The information reached should be safe for work and healthy for the platform.

Recommendations: Provides recommendations to tweets based on the user interest.

Twitter deals with petabytes of data, each tweet has to be addressed in very few seconds before being rendered on the timeline. Thus, the models besides being accurate in prediction should also be supersonic. This is why machine learning is always required to platforms like this.

- **Dropbox**

Dropbox now with the tensorflow deep learning framework can automatically recognize text in images (including PDFs containing images). People have stored more than 20 billion image and PDF files in Dropbox. Of those files, 10-20% are photos of documents—like receipts and whiteboard images—as opposed to documents themselves. These are now candidates for automatic image text recognition. Similarly, 25% of these PDFs are scans of documents that are also candidates for automatic text recognition. Making document images searchable is the first step towards a deeper understanding of the structure and content of documents. Automatic image text recognition is a prime example of the type of large scale projects involving computer vision and machine learning



- **Airbus**

  This is what is written in Airbus's website:

  «A further area where deep learning has made considerable progress is in image recognition – and the Group is benefitting here, too. Since 29 September 2016, 'One Atlas' has been providing employees and customers with a large database of satellite images that can be accessed online. However, before these images from space end up in the database, clouds need to be identified and removed.

  "That's a lot harder than it sounds," says Laurent Gabet, who manages optical research and development at Airbus Defence and Space. In fact it is sometimes very difficult to determine, even for the human eye, whether a particular area on a satellite image is cloud or snow.

  Airbus Defence and Space uses an algorithm to identify these objects, but it has an error rate of 11%, which has prompted Gabet and his colleagues to test the deep-learning platform TensorFlow. This open source software from Google can be used and changed by anyone; Google provides the toolbox for others to work with. TensorFlow analyses a large number of images, looks for recurring patterns, and uses this to learn how to identify objects by itself.

In the TensorFlow test phase, the error rate has already improved to just 3%. Gabet is convinced that these new applications will make a huge difference: “Now that these programmes are freely available, the field of image recognition will progress in leaps and bounds and open up new opportunities,” he says»

4.0 Python

4.0.1 What is Python

According to Python’s official website: «Python is an interpreted, high-level, general-purpose programming language. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.».

Python's design and philosophy have influenced many other programming languages:

- Ruby
- Kotlin
- Cobra
- Boo
- CoffeeScript
- Swift
- Groovy
- Nim

4.0.2 Python and Data Science

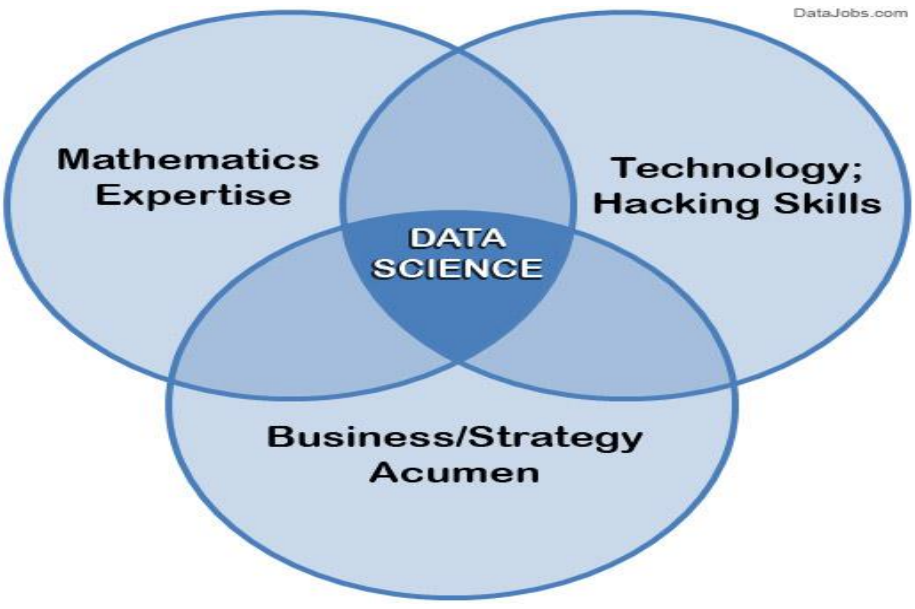
Python language is getting more and more popular for many reasons as it is described in the previous chapter but it is considered also a language that somebody needs to master if want to be involved with the data analytics/data science field since it the most flexible programming language.

- **Data Analysis**

Data analysis is the methodology of gathering data and processing it in order to get useful information. A Data Analyst utilizes the major techniques related to data visualization and manipulation. All these insights allow the companies to formulate better strategies and to make even better decisions. The steps here are:

- Data Requirements
- Data collection
- Data processing
- Data cleaning
- Exploratory data analysis (the variety of techniques to understand what data contain)
- Modeling and algorithms
- Data product

- **Data Science**
  Data Science is a scientific field that uses high level methods and techniques to process structured and unstructured data to extract insights. It combines statistics, data analysis and machine learning in order to «understand data».



Section Two

5.0 Dataset

5.0.1 Collecting the dataset

Creating the dataset is the first of the many steps required to successfully train the model. For this project, I collected photos of fridges inside from my friends, family, work and downloaded some from Google. The total photos were 50 and they include many objects multiple times inside.

The objects that are included in the photos are the following:

- Mustard
- Beer
- Juice
- Ketchup
- Lemon
- Orange
- Jam
- Eggs
- Yogurt
- Lettuce
- Tomato
- Milk
- Butter
- Tabasco
- Bread
- Cucumber
- Pineapple
- Olives
- Watermelon
- Banana

- Mango
- Meat
- Carrot
- Water
- Coca Cola
- Apple
- Cheese
- Sausage
- Pepper

We need to note here that for better accuracy the photos needed for training a model to detect an object with 90% and higher precision are about 200-250 per object but this was not feasible in this project.

5.0.2 Labeling the images

Now that the dataset is ready, the next step is to label the images. Since we are doing object detection, we need a ground truth of what exactly the object is. For this, we need to draw a bounding box around the object, in order system to understand that this object inside the box is the actual object we want to learn. The software used for this task is labelImg (https://github.com/tzutalin/labelImg). LabelImg is a graphical image annotation tool. It is written in Python and uses Qt for its graphical interface. Annotations are saved as XML files in PASCAL VOC format, the format used by ImageNet. Besides, it also supports YOLO format. The project is implemented in Linux environment so we performed the build from source suggested by the developer for Linux\Ubuntu:

Python 3 + Qt5 (Recommended)

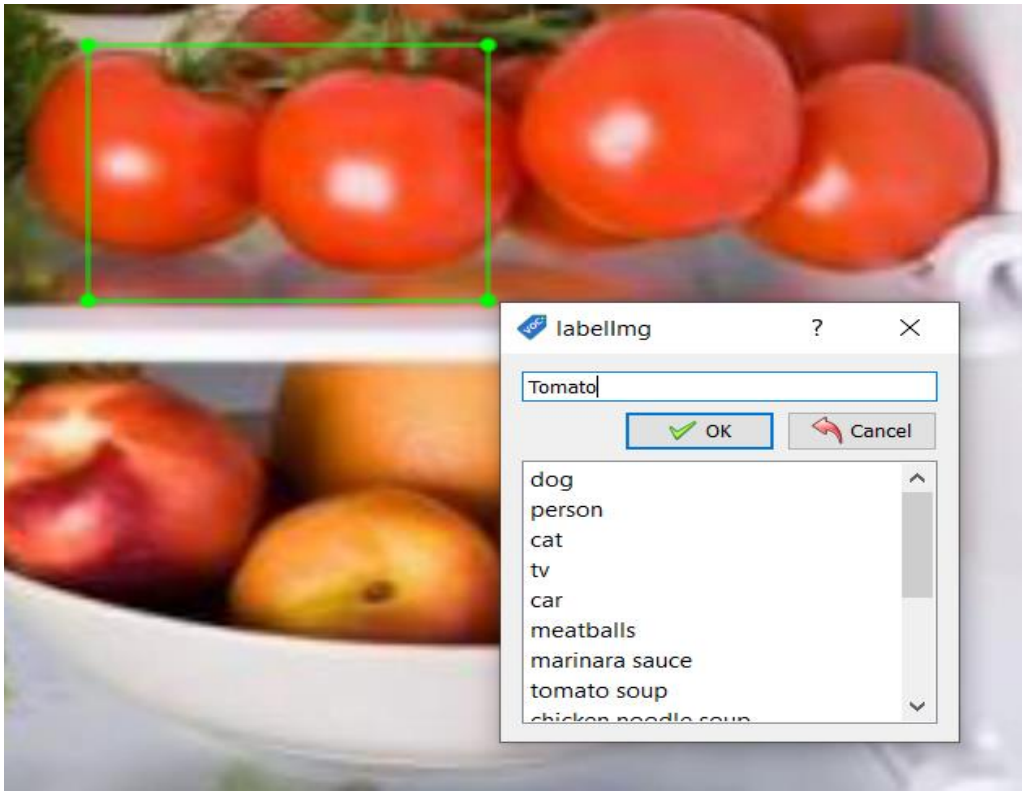
```
sudo apt-get install pyqt5-dev-tools
sudo pip3 install -r requirements/requirements-linux-python3.txt
make qt5py3
python3 labelImg.py
python3 labelImg.py [IMAGE_PATH] [PRE-DEFINED CLASS FILE]
```

After the installation, launch the application and follow the steps:

1. Click 'Change default saved annotation folder' in Menu/File
2. Click 'Open Dir'

3. Click 'Create RectBox'
4. Click and release left mouse to select a region to annotate the rect box
5. You can use right mouse to drag the rect box to copy or move it

The annotation will be saved to the folder you specify.



Once we have labeled all the images and have their corresponding xml files, the next step is to split the dataset into a train and test dataset. Inside the same directory where the images are, we need to create a directory named "*train*" and "*test*" and add around 70% of the images and their respective XML to the training directory and the remaining 30% to the test directory.

5.0.3 Generate TFRECORD

In this step, we need to covert the images and their XML files to a format that is readable by Tensorflow. This format is called *tfrecord.* In order to proceed with the creation of these files we first need to convert the xml of Train images to a CSV file and then the xml of the Test images to another CSV file. This was done by the tool xml_to_csv.py that can be found (https://github.com/qdraw/tensorflow-face-object-detector-tutorial/blob/master/003_xml-to-csv.py). Below we can see the script (The only change required is replace the paths with ours)

The code:

```
import os
import glob
import pandas as pd
import xml.etree.ElementTree as ET

def xml_to_csv(path):
xml_list = []
for xml_file in glob.glob(path + '/*.xml'):
tree = ET.parse(xml_file)
root = tree.getroot()
for member in root.findall('object'):
value = (root.find('filename').text,
int(root.find('size')[0].text),
int(root.find('size')[1].text),
member[0].text,
int(member[4][0].text),
int(member[4][1].text),
int(member[4][2].text),
int(member[4][3].text)
)
xml_list.append(value)
column_name = ['filename', 'width', 'height', 'class', 'xmin', 'ymin', 'xmax', 'ymax']
xml_df = pd.DataFrame(xml_list, columns=column_name)
return xml_df

def main():
image_path = os.path.join(os.getcwd(), 'annotations')
xml_df = xml_to_csv(image_path)
xml_df.to_csv('raccoon_labels.csv', index=None)
print('Successfully converted xml to csv.')

main()
```

After we have the two CSV files (Train and Test) we need to convert the tfrecords. Then, using the script generate_tfrecord.py (https://github.com/datitran/raccoon_dataset/blob/master/generate_tfrecord.py). Please note that before running the script you have to specify the class of your objects in the function class_text_to_int. The script:

```
Usage:
# From tensorflow/models/
# Create train data:
python generate_tfrecord.py --csv_input=data/train_labels.csv --output_path=train.record

# Create test data:
python generate_tfrecord.py --csv_input=data/test_labels.csv --output_path=test.record
"""
from __future__ import division
from __future__ import print_function
from __future__ import absolute_import

import os
import io
import pandas as pd
import tensorflow as tf

from PIL import Image
from object_detection.utils import dataset_util
from collections import namedtuple, OrderedDict

flags = tf.app.flags
flags.DEFINE_string('csv_input', '', 'Path to the CSV input')
flags.DEFINE_string('output_path', '', 'Path to output TFRecord')
flags.DEFINE_string('image_dir', '', 'Path to images')
```

```
FLAGS = flags.FLAGS

# TO-DO replace this with label map
def class_text_to_int(row_label):
if row_label == 'raccoon':
return 1
else:
None

def split(df, group):
data = namedtuple('data', ['filename', 'object'])
gb = df.groupby(group)
return [data(filename, gb.get_group(x)) for filename, x in zip(gb.groups.keys(), gb.groups)]

def create_tf_example(group, path):
with tf.gfile.GFile(os.path.join(path, '{}'.format(group.filename)), 'rb') as fid:
encoded_jpg = fid.read()
encoded_jpg_io = io.BytesIO(encoded_jpg)
image = Image.open(encoded_jpg_io)
width, height = image.size

filename = group.filename.encode('utf8')
image_format = b'jpg'
xmins = []
xmaxs = []
ymins = []
ymaxs = []
classes_text = []
classes = []

for index, row in group.object.iterrows():
xmins.append(row['xmin'] / width)
xmaxs.append(row['xmax'] / width)
ymins.append(row['ymin'] / height)
ymaxs.append(row['ymax'] / height)
```

```
classes_text.append(row['class'].encode('utf8'))
classes.append(class_text_to_int(row['class']))

tf_example = tf.train.Example(features=tf.train.Features(feature={
'image/height': dataset_util.int64_feature(height),
'image/width': dataset_util.int64_feature(width),
'image/filename': dataset_util.bytes_feature(filename),
'image/source_id': dataset_util.bytes_feature(filename),
'image/encoded': dataset_util.bytes_feature(encoded_jpg),
'image/format': dataset_util.bytes_feature(image_format),
'image/object/bbox/xmin': dataset_util.float_list_feature(xmins),
'image/object/bbox/xmax': dataset_util.float_list_feature(xmaxs),
'image/object/bbox/ymin': dataset_util.float_list_feature(ymins),
'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
'image/object/class/label': dataset_util.int64_list_feature(classes),
}))
return tf_example

def main(_):
writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
path = os.path.join(FLAGS.image_dir)
examples = pd.read_csv(FLAGS.csv_input)
grouped = split(examples, 'filename')
for group in grouped:
tf_example = create_tf_example(group, path)
writer.write(tf_example.SerializeToString())

writer.close()
output_path = os.path.join(os.getcwd(), FLAGS.output_path)
print('Successfully created the TFRecords: {}'.format(output_path))

if __name__ == '__main__':
tf.app.run()
```

5.0.4 Creation of label map

A *labels* map indicating the labels and their indexes is required. Always start with index 1 because 0 is reserved. The file was saved as object-detection.pbtxt in a new directory named "*training*". This is how our label map looks:

```
item {
  id: 1
  name: 'Mustard'
}

item {
  id: 2
  name: 'Beer'
}

item {
  id: 3
  name: 'Juice'
}

item {
  id: 4
  name: 'Ketchup'
}

item {
  id: 5
  name: 'Lemon'
```

```
}

item {
  id: 6
  name: 'Orange'
}

item {
  id: 7
  name: 'Jam'
}

item {
  id: 8
  name: 'Eggs'
}

item {
  id: 9
  name: 'Yogurt'
}

item {
  id: 10
  name: 'Lettuce'
}

item {
  id: 11
  name: 'Tomato'
}

item {
```

```
  id: 12
  name: 'Milk'
}

item {
  id: 13
  name: 'Butter'
}

item {
  id: 14
  name: 'Tabasco'
}

item {
  id: 15
  name: 'Bread'
}

item {
  id: 16
  name: 'Cucumber'
}

item {
  id: 17
  name: 'Pineapple'
}

item {
  id: 18
  name: 'Olives'
}
```

item {
  id: 19
  name: 'Watermelon'
}

item {
  id: 20
  name: 'Banana'
}

item {
  id: 21
  name: 'Mango'
}

item {
  id: 22
  name: 'Meat'
}

item {
  id: 23
  name: 'Carot'
}

item {
  id: 24
  name: 'Water'
}

item {
  id: 25

```
  name: 'Coca Cola'
}

item {
  id: 26
  name: 'Apple'
}

item {
  id: 27
  name: 'Cheese'
}

item {
  id: 28
  name: 'Sausage'
}

item {
  id: 29
  name: 'Pepper'
}
```

5.1 Training

The complete training process of the model is being handled by a configuration file known as *pipeline*. The pipeline is divided into five main structures that are responsible for defining the model, the training and evaluation process parameters, and both the training and evaluation dataset inputs. It is not recommended though to create our own model from scratch. TensorFlow developers suggest using one of their own and already trained models as a starting point. TensorFlow provides several configuration files that only need a minimal amount of changes to make it work in a new training environment. The model I used was ssd_mobilenet_v1_coco.config.

The model:

```
# SSD with Mobilenet v1 configuration for MSCOCO Dataset.
# Users should configure the fine_tune_checkpoint field in the train config as
# well as the label_map_path and input_path fields in the train_input_reader and
# eval_input_reader. Search for "PATH_TO_BE_CONFIGURED" to find the fields that
# should be configured.

model {
ssd {
num_classes: 90
box_coder {
faster_rcnn_box_coder {
y_scale: 10.0
x_scale: 10.0
height_scale: 5.0
width_scale: 5.0
}
}
matcher {
argmax_matcher {
matched_threshold: 0.5
unmatched_threshold: 0.5
ignore_thresholds: false
negatives_lower_than_unmatched: true
force_match_for_each_row: true
}
}
similarity_calculator {
iou_similarity {
}
}
anchor_generator {
ssd_anchor_generator {
num_layers: 6
```

```
min_scale: 0.2
max_scale: 0.95
aspect_ratios: 1.0
aspect_ratios: 2.0
aspect_ratios: 0.5
aspect_ratios: 3.0
aspect_ratios: 0.3333
}
}
image_resizer {
fixed_shape_resizer {
height: 300
width: 300
}
}
box_predictor {
convolutional_box_predictor {
min_depth: 0
max_depth: 0
num_layers_before_predictor: 0
use_dropout: false
dropout_keep_probability: 0.8
kernel_size: 1
box_code_size: 4
apply_sigmoid_to_scores: false
conv_hyperparams {
activation: RELU_6,
regularizer {
l2_regularizer {
weight: 0.00004
}
}
```

initializer {
truncated_normal_initializer {
stddev: 0.03
mean: 0.0
}
}
batch_norm {
train: true,
scale: true,
center: true,
decay: 0.9997,
epsilon: 0.001,
}
}
}
}
feature_extractor {
type: 'ssd_mobilenet_v1'
min_depth: 16
depth_multiplier: 1.0
conv_hyperparams {
activation: RELU_6,
regularizer {
l2_regularizer {
weight: 0.00004
}
}
initializer {
truncated_normal_initializer {
stddev: 0.03
mean: 0.0
}

```
}
batch_norm {
train: true,
scale: true,
center: true,
decay: 0.9997,
epsilon: 0.001,
}
}
}
loss {
classification_loss {
weighted_sigmoid {
}
}
localization_loss {
weighted_smooth_l1 {
}
}
hard_example_miner {
num_hard_examples: 3000
iou_threshold: 0.99
loss_type: CLASSIFICATION
max_negatives_per_positive: 3
min_negatives_per_image: 0
}
classification_weight: 1.0
localization_weight: 1.0
}
normalize_loss_by_num_matches: true
post_processing {
batch_non_max_suppression {
```

score_threshold: 1e-8
iou_threshold: 0.6
max_detections_per_class: 100
max_total_detections: 100
}
score_converter: SIGMOID
}
}
}

train_config: {
batch_size: 24
optimizer {
rms_prop_optimizer: {
learning_rate: {
exponential_decay_learning_rate {
initial_learning_rate: 0.004
decay_steps: 800720
decay_factor: 0.95
}
}
momentum_optimizer_value: 0.9
decay: 0.9
epsilon: 1.0
}
}
fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED/model.ckpt"
from_detection_checkpoint: true
Note: The below line limits the training process to 200K steps, which we
empirically found to be sufficient enough to train the pets dataset. This
effectively bypasses the learning rate schedule (the learning rate will
never decay). Remove the below line to train indefinitely.
num_steps: 200000

```
data_augmentation_options {
random_horizontal_flip {
}
}
data_augmentation_options {
ssd_random_crop {
}
}
}

train_input_reader: {
tf_record_input_reader {
input_path: "PATH_TO_BE_CONFIGURED/mscoco_train.record-?????-of-00100"
}
label_map_path: "PATH_TO_BE_CONFIGURED/mscoco_label_map.pbtxt"
}

eval_config: {
num_examples: 8000
# Note: The below line limits the evaluation process to 10 evaluations.
# Remove the below line to evaluate indefinitely.
max_evals: 10
}

eval_input_reader: {
tf_record_input_reader {
input_path: "PATH_TO_BE_CONFIGURED/mscoco_val.record-?????-of-00010"
}
label_map_path: "PATH_TO_BE_CONFIGURED/mscoco_label_map.pbtxt"
shuffle: false
num_readers: 1
}
```

The changes made here were to the num_classes where I put 29,The num_steps to halt the training earlier, the fine_tune_checkpoint to point to the location of the model downloaded,

and the input_path and label_map_path variables of the train_input_reader and eval_input_reader to point to the training and test dataset and the labels map.

SSD, which stands for Single Shot Detector, is a neural network architecture, based on a single feed-forward convolutional neural network. It is called “single shot” because it predicts the class of the image and the position of the box that represents the detection (known as the anchor) during the same step. The opposite of this is an architecture that requires a second component known as the “proposal generator” to predict the exact position of the box.

MobileNet is a convolutional feature extractor, designed to work on mobile devices, that is used to obtain the high-level features of the images.

Once the pipeline is ready, we add it to the “*training*” directory. Then, we start the training using the following command:

python object_detection/train.py --logtostderr
--train_dir=path/to/training/
--pipeline_config_path=path/to/training/ ssd_mobilenet_v1_coco.config.

5.2 Evaluation

Firstly we start by exporting our model.

Before exporting it, we need to make sure we have the following files in the training directory:

model.ckpt-${CHECKPOINT_NUMBER}.data-00000-of-00001,

model.ckpt-${CHECKPOINT_NUMBER}.index

model.ckpt-${CHECKPOINT_NUMBER}.meta

then we execute the following command:

```
python object_detection/export_inference_graph.py
input_type image_tensor
pipeline_config_path=path/to/training/ssd_mobilenet_v1_pets.config
trained_checkpoint_prefix=path/to/training/model.ckpt-xxxxx
output_directory path/to/output/directory
```

The output will be a file that holds a “frozen” version of the model named `frozen_inference_graph.pb`.

5.2.1 Tensorboard

The evaluation of the training we performed can be done using TensorFlow's visualization platform, TensorBoard. In this platform, we can monitor several metrics such as the training time, total loss, number of steps and many more. Different Views of TensorBoard

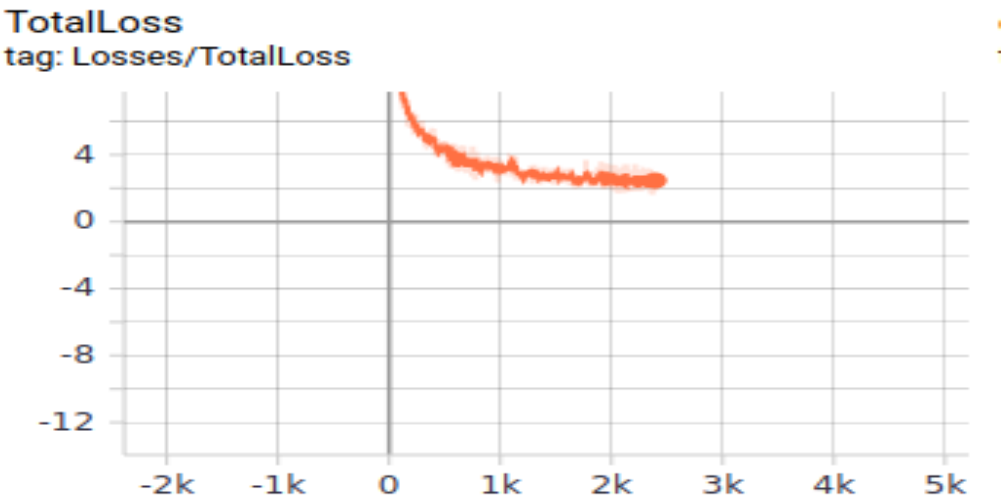
Different views take inputs of different formats and display them differently. You can change them on the orange top bar.

- Scalars - Visualize scalar values, such as classification accuracy.
- Graph - Visualize the computational graph of your model, such as the neural network model.
- Distributions - Visualize how data changes over time, such as the weights of a neural network.
- Histograms - A fancier view of the distribution that shows distributions in a 3-dimensional perspective
- Projector - Can be used to visualize word embeddings (that is, word embeddings are numerical representations of words that capture their semantic relationships)
- Image - Visualizing image data
- Audio - Visualizing audio data
- Text - Visualizing text (string) data

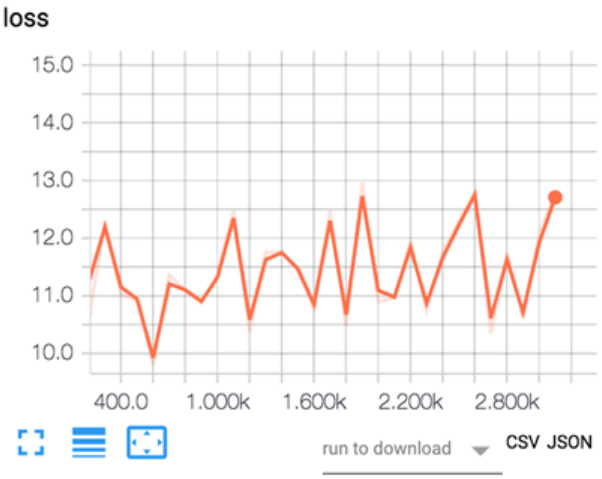
To execute TensorBoard, we run the following command:

tensorboard --logdir=path/to/training/

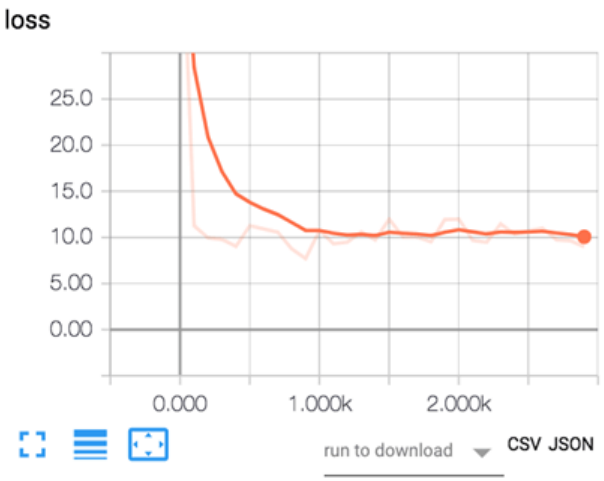
Once the tensorboard opens we go to the first tab named «Scalars» and we check the Losses. Below is presented a screenshot from the TotalLoss of our model VS the one that should be according to Tensorflow in order to have a good trained model :



Model does not learn anything

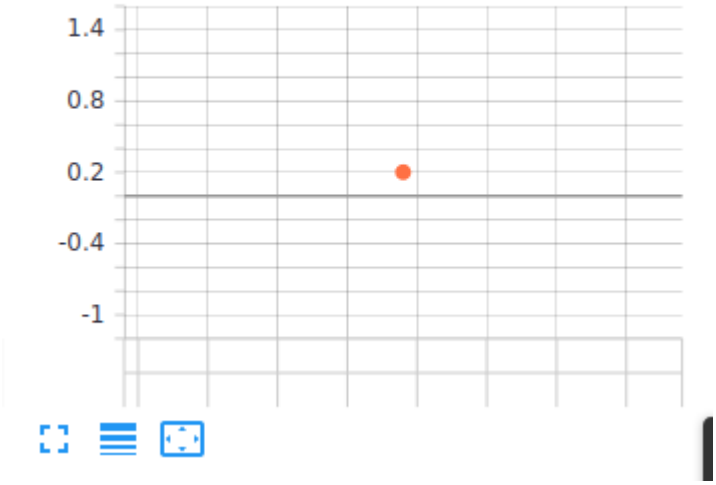


Model learns all the informations

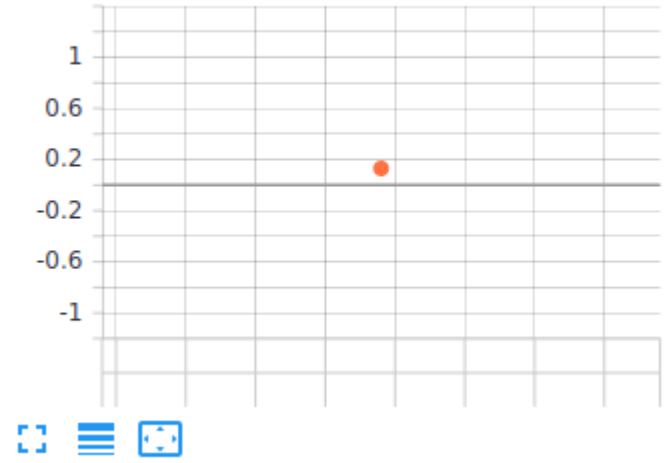


After the total loss, moving to the performance per category section, some examples of the categories are shown in the below screenshots:

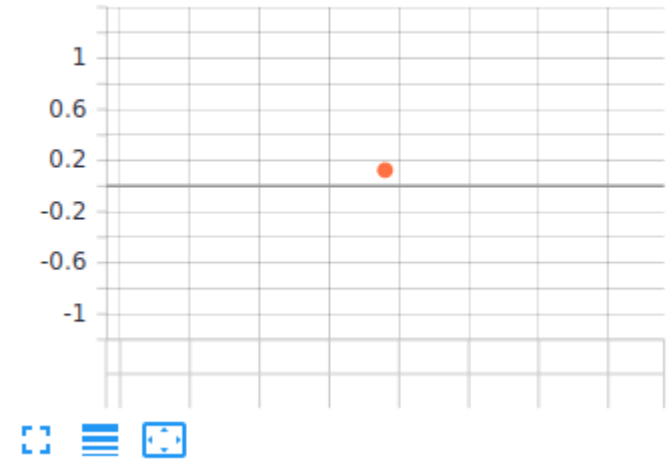
AP@0.5IOU/Banana
tag: PascalBoxes_PerformanceByCategory/AP@0.5IOU/Banana



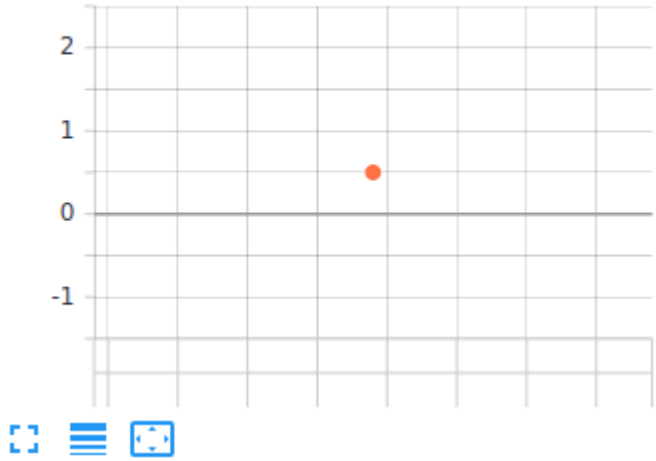
AP@0.5IOU/Juice
tag: PascalBoxes_PerformanceByCategory/AP@0.5IOU/Juice



AP@0.5IOU/Eggs
tag: PascalBoxes_PerformanceByCategory/AP@0.5IOU/Eggs

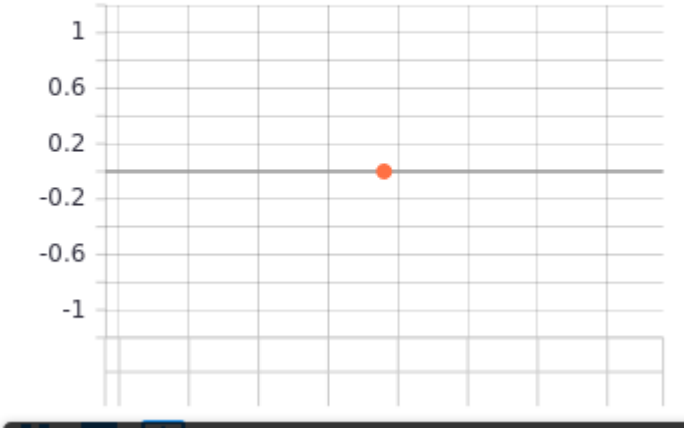


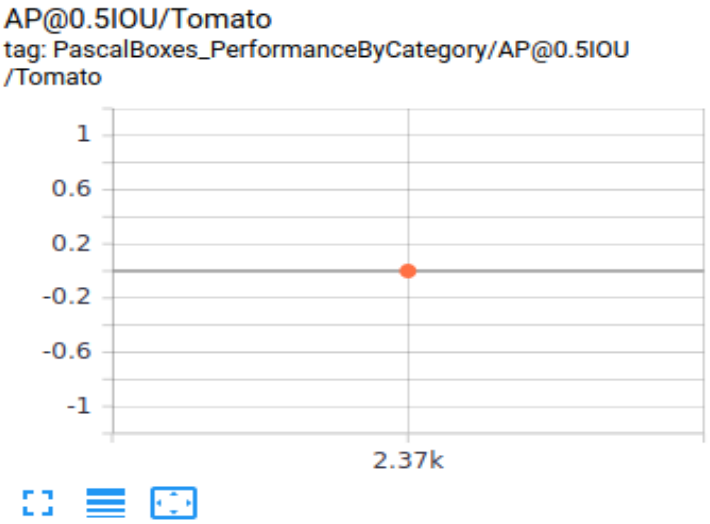
AP@0.5IOU/Pineapple
tag: PascalBoxes_PerformanceByCategory/AP@0.5IOU
/Pineapple



AP@0.5IOU/Yogurt
tag: PascalBoxes_PerformanceByCategory/AP@0.5IOU
/Yogurt

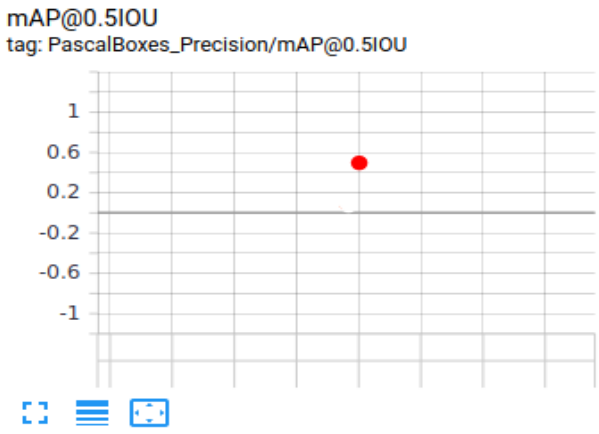
AP@0.5IOU/Apple
tag: PascalBoxes_PerformanceByCategory/AP@0.5IOU
/Apple



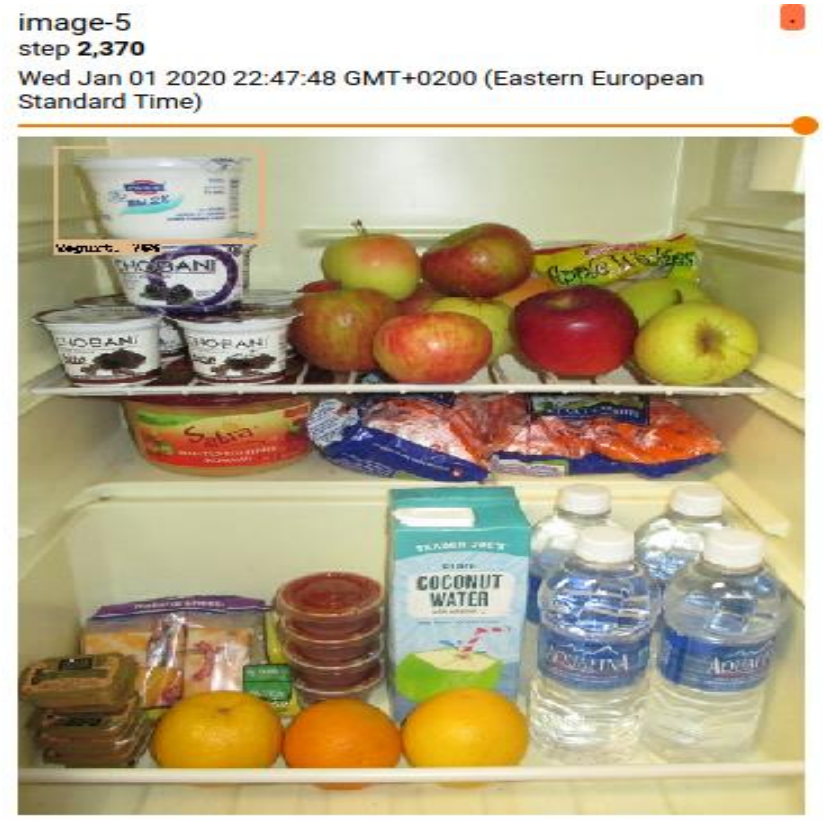


The conclusions of the above statistics about performance per category are that as we can observe for items that are similar like apple and tomato the performance is nearly zero, this issue can be resolved by having more training dataset. The ideal pictures per item are about 200-250 in order to have high accuracy.

In the section of Precision, we see below that the total precision of the model we trained is about 58%.



In the Images tab we can see how the model worked and what detected in the images we have in the test directory.



We see in the above photos that the trained model detected the Yogurt in both photos, in the first picture with 78% accuracy and in the second with 58%. We also see though that in the first photo, the model did not recognize the apples or the orange in this specific picture.

5.2.2 Jupyter Notebook

According to Wikipedia and the official site of Jupyter the project Jupyter is a nonprofit organization created to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages". Spun-off from IPython in 2014 by Fernando Pérez, Project Jupyter supports execution environments in several dozen languages. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter. Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab, the next-generation version of Jupyter Notebook.

Jupyter Notebook (formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebook documents. The "notebook" term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter Python web server, or Jupyter document format depending on context. A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the ".ipynb" extension.

A Jupyter Notebook can be converted to a number of open standard output formats (HTML, presentation slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through "Download As" in the web interface, via the nbconvert library or "jupyter nbconvert" command line interface in a shell.

To simplify visualisation of Jupyter notebook documents on the web, the nbconvert library is provided as a service through NbViewer which can take a URL to any publicly available notebook document, convert it to HTML on the fly and display it to the user.

Jupyter Notebook interface

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

- IPython
- ØMQ
- Tornado (web server)
- jQuery
- Bootstrap (front-end framework)
- MathJax

Jupyter Notebook can connect to many kernels to allow programming in many languages. By default Jupyter Notebook ships with the IPython kernel.

The jupyter notebook do not give us statistics but we can evaluate the model in terms that we can import photos and check how the model works.

We can open Jupyter Notebook by writing Jupyter Notebook to the command prompt.

In the page that opens, we go to the path that we have the object_detection_tutorial.ipynb file and open it.

The code :

```
import numpy as np
import os
import six.moves.urllib as urllib
import sys
import tarfile
import tensorflow as tf
import zipfile

from distutils.version import StrictVersion
from collections import defaultdict
from io import StringIO
from matplotlib import pyplot as plt
from PIL import Image

# This is needed since the notebook is stored in the object_detection folder.
sys.path.append("..")
from object_detection.utils import ops as utils_ops

if StrictVersion(tf.__version__) < StrictVersion('1.12.0'):
  raise ImportError('Please upgrade your TensorFlow installation to v1.12.*.')
# This is needed to display the images.
%matplotlib inline
from utils import label_map_util
```

```
from utils import visualization_utils as vis_util
# What model to download.
MODEL_NAME = 'smart_fridge_graph'

# Path to frozen detection graph. This is the actual model that is used for the object detection.
PATH_TO_FROZEN_GRAPH = MODEL_NAME + '/frozen_inference_graph.pb'

# List of the strings that is used to add correct label for each box.
PATH_TO_LABELS = os.path.join('training', 'object-detection.pbtxt')
detection_graph = tf.Graph()
with detection_graph.as_default():
  od_graph_def = tf.GraphDef()
  with tf.gfile.GFile(PATH_TO_FROZEN_GRAPH, 'rb') as fid:
    serialized_graph = fid.read()
    od_graph_def.ParseFromString(serialized_graph)
    tf.import_graph_def(od_graph_def, name='')
PATH_TO_TEST_IMAGES_DIR = 'test_images'
TEST_IMAGE_PATHS = [ os.path.join(PATH_TO_TEST_IMAGES_DIR,
'image{}.jpg'.format(i)) for i in range(3, 9) ]

# Size, in inches, of the output images.
IMAGE_SIZE = (12, 8)
def run_inference_for_single_image(image, graph):
  with graph.as_default():
    with tf.Session() as sess:
      # Get handles to input and output tensors
      ops = tf.get_default_graph().get_operations()
      all_tensor_names = {output.name for op in ops for output in op.outputs}
      tensor_dict = {}
      for key in [
          'num_detections', 'detection_boxes', 'detection_scores',
          'detection_classes', 'detection_masks'
      ]:
```

```
      tensor_name = key + ':0'
      if tensor_name in all_tensor_names:
        tensor_dict[key] = tf.get_default_graph().get_tensor_by_name(
            tensor_name)
    if 'detection_masks' in tensor_dict:
      # The following processing is only for single image
      detection_boxes = tf.squeeze(tensor_dict['detection_boxes'], [0])
      detection_masks = tf.squeeze(tensor_dict['detection_masks'], [0])
      # Reframe is required to translate mask from box coordinates to image coordinates and fit the image size.
      real_num_detection = tf.cast(tensor_dict['num_detections'][0], tf.int32)
      detection_boxes = tf.slice(detection_boxes, [0, 0], [real_num_detection, -1])
      detection_masks = tf.slice(detection_masks, [0, 0, 0], [real_num_detection, -1, -1])
      detection_masks_reframed = utils_ops.reframe_box_masks_to_image_masks(
          detection_masks, detection_boxes, image.shape[1], image.shape[2])
      detection_masks_reframed = tf.cast(
          tf.greater(detection_masks_reframed, 0.5), tf.uint8)
      # Follow the convention by adding back the batch dimension
      tensor_dict['detection_masks'] = tf.expand_dims(
          detection_masks_reframed, 0)
    image_tensor = tf.get_default_graph().get_tensor_by_name('image_tensor:0')

    # Run inference
    output_dict = sess.run(tensor_dict,
                           feed_dict={image_tensor: image})

    # all outputs are float32 numpy arrays, so convert types as appropriate
    output_dict['num_detections'] = int(output_dict['num_detections'][0])
    output_dict['detection_classes'] = output_dict[
        'detection_classes'][0].astype(np.int64)
    output_dict['detection_boxes'] = output_dict['detection_boxes'][0]
    output_dict['detection_scores'] = output_dict['detection_scores'][0]
    if 'detection_masks' in output_dict:
```

```
      output_dict['detection_masks'] = output_dict['detection_masks'][0]
  return output_dict
for image_path in TEST_IMAGE_PATHS:
  image = Image.open(image_path)
  # the array based representation of the image will be used later in order to prepare the
  # result image with boxes and labels on it.
  image_np = load_image_into_numpy_array(image)
  # Expand dimensions since the model expects images to have shape: [1, None, None, 3]
  image_np_expanded = np.expand_dims(image_np, axis=0)
  # Actual detection.
  output_dict = run_inference_for_single_image(image_np_expanded, detection_graph)
  # Visualization of the results of a detection.
  vis_util.visualize_boxes_and_labels_on_image_array(
     image_np,
     output_dict['detection_boxes'],
     output_dict['detection_classes'],
     output_dict['detection_scores'],
     category_index,
     instance_masks=output_dict.get('detection_masks'),
     use_normalized_coordinates=True,
     line_thickness=8)
  plt.figure(figsize=IMAGE_SIZE)
  plt.imshow(image_np)
```

When we run the above code in Jupyter, it takes as inputs the images we want to see if our model recognizes and runs the model. We see an example below, how it detected the orange and the yogurt in the below image:

Section Three

6.0 Android Studio

6.0.1 What is Android Studio

According to Wikipedia, Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.

The following features are provided in the current stable version:

Gradle-based build support

Android-specific refactoring and quick fixes

Lint tools to catch performance, usability, version compatibility and other problems

ProGuard integration and app-signing capabilities

Template-based wizards to create common Android designs and components

A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations

Support for building Android Wear apps

Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine

Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin[20] and "all Java 7 language features and a subset of Java 8 language features that vary by platform version." External projects backport some Java 9 features. While IntelliJ that Android Studio is built on supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.

6.0.2 Building the application

Firsty we download the Android Studio, clone the Tensorflow repository and import it to Android Studio as a new project using the directory from the repository we cloned.

In the MainActivity we have the following code:

```
import android.app.Activity;
import android.app.ProgressDialog;
import android.content.Intent;
import android.content.res.Configuration;
import android.database.Cursor;
import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.provider.MediaStore;
import android.support.v4.content.FileProvider;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import com.nostra13.universalimageloader.core.DisplayImageOptions;
import com.nostra13.universalimageloader.core.ImageLoader;
import com.nostra13.universalimageloader.core.ImageLoaderConfiguration;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.text.SimpleDateFormat;
import java.util.Date;

public class MainActivity extends Activity {

    private ProgressDialog dialog;
    static final int REQUEST_IMAGE_CAPTURE = 2;
    private String mCurrentPhotoPath;
    private ImageView mImageView;
    private TextView image_name_tv;
    private int column_index;
    private String imagePath;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        dialog = new ProgressDialog(this);
        dialog.setIndeterminate(true);
        dialog.setCancelable(false);
        dialog.setMessage("Loading. Please wait..."); // showing a dialog for loading the data
        // Create default options which will be used for every
        //  displayImage(...) call if no options will be passed to this method
        DisplayImageOptions defaultOptions = new DisplayImageOptions.Builder()
                .cacheInMemory(true)
```

```
            .cacheOnDisk(true)
            .build();
        ImageLoaderConfiguration config = new
ImageLoaderConfiguration.Builder(getApplicationContext())
            .defaultDisplayImageOptions(defaultOptions)
            .build();
        ImageLoader.getInstance().init(config); // Do it on Application start



        Button startDetector;
        startDetector = (Button) findViewById(R.id.startDetector);
        startDetector.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startDetector();
            }
        });

       /* Button startClassifier;
        startClassifier = (Button) findViewById(R.id.startClassifier);
        startClassifier.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startClassifier();
            }
        });*/

        Button getPictureIntent = (Button) findViewById(R.id.capturePhoto);
        getPictureIntent.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                pickPhoto();
            }
```

```
    });
    setTitle("Smart Fridge Application");
}

private void startDetector() {
    Intent detectorActivity = new Intent(this, DetectorActivity.class);
    startActivity(detectorActivity);
}

private void startClassifier() {
    Intent classifierActivity = new Intent(this, ClassifierActivity.class);
    startActivity(classifierActivity);
}

private void pickPhoto() {
    Intent photoPickerIntent = new Intent(Intent.ACTION_PICK);
    photoPickerIntent.setType("image/*");
    startActivityForResult(photoPickerIntent, 1);
}



@Override
public void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig);
    // your code here, you can use newConfig.locale if you need to check the language
    // or just re-set all the labels to desired string resource
}

/* @Override
 public void onActivityResult(int requestCode, int resultCode, Intent data) {
     super.onActivityResult(requestCode, resultCode, data);
     if (requestCode == 1)
         if (resultCode == Activity.RESULT_OK) {
```

```
            Uri selectedImage = data.getData();

            String filePath = getPath(selectedImage);
            String file_extn = filePath.substring(filePath.lastIndexOf(".") + 1);
            image_name_tv.setText(filePath);

            if (file_extn.equals("img") || file_extn.equals("jpg") || file_extn.equals("jpeg") ||
file_extn.equals("gif") || file_extn.equals("png")) {
                //FINE
            } else {
                //NOT IN REQUIRED FORMAT
            }
        }
    }

    public String getPath(Uri uri) {
        String[] projection = {MediaStore.MediaColumns.DATA};
        Cursor cursor = managedQuery(uri, projection, null, null, null);
        column_index = cursor
                .getColumnIndexOrThrow(MediaStore.MediaColumns.DATA);
        cursor.moveToFirst();
        imagePath = cursor.getString(column_index);

        return cursor.getString(column_index);
    }*/

    private void resultActivity() {
        Intent goToResults = new  Intent (this, ResultActivity.class);
        startActivity(goToResults);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == REQUEST_IMAGE_CAPTURE) {
            if (resultCode == Activity.RESULT_OK) {
                File file = new File(Environment.getExternalStorageDirectory().getPath(),
"photo.jpg");
                Uri uri = Uri.fromFile(file);
                Bitmap bitmap;
                try {
                    bitmap = MediaStore.Images.Media.getBitmap(getContentResolver(), uri);
                    //  bitmap = crupAndScale(bitmap, 300); // if you mind scaling
                    mImageView.setImageBitmap(bitmap);
                } catch (FileNotFoundException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IOException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }

            }
        }
    }
    @Override
    public void onBackPressed(){
        Intent intent = new Intent(MainActivity.this, ResultActivity.class);
        startActivity(intent);
    }
}
```

We also have the DetectorActivity which is called from the MainActivity,

This activity does all the detection job and in here we have our trained frozen model.

Below the code:

```
package org.tensorflow.demo;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.Bitmap.Config;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.Paint.Style;
import android.graphics.RectF;
import android.graphics.Typeface;
import android.media.ImageReader.OnImageAvailableListener;
import android.os.SystemClock;
import android.util.Size;
import android.util.TypedValue;
```

```
import android.view.Display;
import android.view.Surface;
import android.widget.Toast;
import java.io.IOException;
import java.util.LinkedList;
import java.util.List;
import java.util.Vector;
import org.tensorflow.demo.OverlayView.DrawCallback;
import org.tensorflow.demo.env.BorderedText;
import org.tensorflow.demo.env.ImageUtils;
import org.tensorflow.demo.env.Logger;
import org.tensorflow.demo.tracking.MultiBoxTracker;
import org.tensorflow.demo.R; // Explicit import needed for internal Google builds.

/**
 * An activity that uses a TensorFlowMultiBoxDetector and ObjectTracker to detect and then track
 * objects.
 */
public class DetectorActivity extends CameraActivity implements OnImageAvailableListener {
  private static final Logger LOGGER = new Logger();

  // Configuration values for the prepackaged multibox model.
  private static final int MB_INPUT_SIZE = 224;
  private static final int MB_IMAGE_MEAN = 128;
  private static final float MB_IMAGE_STD = 128;
  private static final String MB_INPUT_NAME = "ResizeBilinear";
  private static final String MB_OUTPUT_LOCATIONS_NAME = "output_locations/Reshape";
  private static final String MB_OUTPUT_SCORES_NAME = "output_scores/Reshape";
  private static final String MB_MODEL_FILE = "file:///android_asset/multibox_model.pb";
  private static final String MB_LOCATION_FILE =
      "file:///android_asset/multibox_location_priors.txt";
```

```
  private static final int TF_OD_API_INPUT_SIZE = 300;
  private static final String TF_OD_API_MODEL_FILE =
      "file:///android_asset/ssd_mobilenet_v1_android_export.pb";
  private static final String TF_OD_API_LABELS_FILE =
"file:///android_asset/coco_labels_list.txt";
  // Configuration values for tiny-yolo-voc. Note that the graph is not included with
TensorFlow and
  // must be manually placed in the assets/ directory by the user.
  // Graphs and models downloaded from http://pjreddie.com/darknet/yolo/ may be converted
e.g. via
  // DarkFlow (https://github.com/thtrieu/darkflow). Sample command:
  // ./flow --model cfg/tiny-yolo-voc.cfg --load bin/tiny-yolo-voc.weights --savepb --verbalise
  private static final String YOLO_MODEL_FILE = "file:///android_asset/graph-tiny-yolo-
voc.pb";
  private static final int YOLO_INPUT_SIZE = 416;
  private static final String YOLO_INPUT_NAME = "input";
  private static final String YOLO_OUTPUT_NAMES = "output";
  private static final int YOLO_BLOCK_SIZE = 32;

  // Which detection model to use: by default uses Tensorflow Object Detection API frozen
  // checkpoints.  Optionally use legacy Multibox (trained using an older version of the API)
  // or YOLO.
  private enum DetectorMode {
    TF_OD_API, MULTIBOX, YOLO;
  }
  private static final DetectorMode MODE = DetectorMode.TF_OD_API;

  // Minimum detection confidence to track a detection.
  private static final float MINIMUM_CONFIDENCE_TF_OD_API = 0.6f;
  private static final float MINIMUM_CONFIDENCE_MULTIBOX = 0.1f;
  private static final float MINIMUM_CONFIDENCE_YOLO = 0.25f;

  private static final boolean MAINTAIN_ASPECT = MODE == DetectorMode.YOLO;
```

```
  private static final Size DESIRED_PREVIEW_SIZE = new Size(640, 480);

  private static final boolean SAVE_PREVIEW_BITMAP = false;
  private static final float TEXT_SIZE_DIP = 10;

  private Integer sensorOrientation;

  private Classifier detector;

  private long lastProcessingTimeMs;
  private Bitmap rgbFrameBitmap = null;
  private Bitmap croppedBitmap = null;
  private Bitmap cropCopyBitmap = null;

  private boolean computingDetection = false;

  private long timestamp = 0;

  private Matrix frameToCropTransform;
  private Matrix cropToFrameTransform;

  private MultiBoxTracker tracker;

  private byte[] luminanceCopy;

  private BorderedText borderedText;
  @Override
  public void onPreviewSizeChosen(final Size size, final int rotation) {
    final float textSizePx =
        TypedValue.applyDimension(
            TypedValue.COMPLEX_UNIT_DIP, TEXT_SIZE_DIP,
getResources().getDisplayMetrics());
```

```
    borderedText = new BorderedText(textSizePx);
    borderedText.setTypeface(Typeface.MONOSPACE);

    tracker = new MultiBoxTracker(this);

    int cropSize = TF_OD_API_INPUT_SIZE;
    if (MODE == DetectorMode.YOLO) {
      detector =
          TensorFlowYoloDetector.create(
              getAssets(),
              YOLO_MODEL_FILE,
              YOLO_INPUT_SIZE,
              YOLO_INPUT_NAME,
              YOLO_OUTPUT_NAMES,
              YOLO_BLOCK_SIZE);
      cropSize = YOLO_INPUT_SIZE;
    } else if (MODE == DetectorMode.MULTIBOX) {
      detector =
          TensorFlowMultiBoxDetector.create(
              getAssets(),
              MB_MODEL_FILE,
              MB_LOCATION_FILE,
              MB_IMAGE_MEAN,
              MB_IMAGE_STD,
              MB_INPUT_NAME,
              MB_OUTPUT_LOCATIONS_NAME,
              MB_OUTPUT_SCORES_NAME);
      cropSize = MB_INPUT_SIZE;
    } else {
      try {
        detector = TensorFlowObjectDetectionAPIModel.create(
            getAssets(), TF_OD_API_MODEL_FILE, TF_OD_API_LABELS_FILE,
TF_OD_API_INPUT_SIZE);
```

```
      cropSize = TF_OD_API_INPUT_SIZE;
    } catch (final IOException e) {
      LOGGER.e(e, "Exception initializing classifier!");
      Toast toast =
          Toast.makeText(
              getApplicationContext(), "Classifier could not be initialized",
Toast.LENGTH_SHORT);
      toast.show();
      finish();
    }
  }

  previewWidth = size.getWidth();
  previewHeight = size.getHeight();

  sensorOrientation = rotation - getScreenOrientation();
  LOGGER.i("Camera orientation relative to screen canvas: %d", sensorOrientation);

  LOGGER.i("Initializing at size %dx%d", previewWidth, previewHeight);
  rgbFrameBitmap = Bitmap.createBitmap(previewWidth, previewHeight,
Config.ARGB_8888);
  croppedBitmap = Bitmap.createBitmap(cropSize, cropSize, Config.ARGB_8888);

  frameToCropTransform =
      ImageUtils.getTransformationMatrix(
          previewWidth, previewHeight,
          cropSize, cropSize,
          sensorOrientation, MAINTAIN_ASPECT);

  cropToFrameTransform = new Matrix();
  frameToCropTransform.invert(cropToFrameTransform);

  trackingOverlay = (OverlayView) findViewById(R.id.tracking_overlay);
```

```
trackingOverlay.addCallback(
    new DrawCallback() {
      @Override
      public void drawCallback(final Canvas canvas) {
        tracker.draw(canvas);
        if (isDebug()) {
          tracker.drawDebug(canvas);
        }
      }
    });

addCallback(
    new DrawCallback() {
      @Override
      public void drawCallback(final Canvas canvas) {
        if (!isDebug()) {
          return;
        }
        final Bitmap copy = cropCopyBitmap;
        if (copy == null) {
          return;
        }

        final int backgroundColor = Color.argb(100, 0, 0, 0);
        canvas.drawColor(backgroundColor);

        final Matrix matrix = new Matrix();
        final float scaleFactor = 2;
        matrix.postScale(scaleFactor, scaleFactor);
        matrix.postTranslate(
            canvas.getWidth() - copy.getWidth() * scaleFactor,
            canvas.getHeight() - copy.getHeight() * scaleFactor);
        canvas.drawBitmap(copy, matrix, new Paint());
```

```
        final Vector<String> lines = new Vector<String>();
        if (detector != null) {
          final String statString = detector.getStatString();
          final String[] statLines = statString.split("\n");
          for (final String line : statLines) {
            lines.add(line);
          }
        }
        lines.add("");

        lines.add("Frame: " + previewWidth + "x" + previewHeight);
        lines.add("Crop: " + copy.getWidth() + "x" + copy.getHeight());
        lines.add("View: " + canvas.getWidth() + "x" + canvas.getHeight());
        lines.add("Rotation: " + sensorOrientation);
        lines.add("Inference time: " + lastProcessingTimeMs + "ms");

        borderedText.drawLines(canvas, 10, canvas.getHeight() - 10, lines);
      }
    });
}

OverlayView trackingOverlay;

@Override
protected void processImage() {
  ++timestamp;
  final long currTimestamp = timestamp;
  byte[] originalLuminance = getLuminance();
  tracker.onFrame(
      previewWidth,
      previewHeight,
      getLuminanceStride(),
```

```
        sensorOrientation,
        originalLuminance,
        timestamp);
    trackingOverlay.postInvalidate();

    // No mutex needed as this method is not reentrant.
    if (computingDetection) {
      readyForNextImage();
      return;
    }
    computingDetection = true;
    LOGGER.i("Preparing image " + currTimestamp + " for detection in bg thread.");

    rgbFrameBitmap.setPixels(getRgbBytes(), 0, previewWidth, 0, 0, previewWidth,
previewHeight);

    if (luminanceCopy == null) {
      luminanceCopy = new byte[originalLuminance.length];
    }
    System.arraycopy(originalLuminance, 0, luminanceCopy, 0, originalLuminance.length);
    readyForNextImage();

    final Canvas canvas = new Canvas(croppedBitmap);
    canvas.drawBitmap(rgbFrameBitmap, frameToCropTransform, null);
    // For examining the actual TF input.
    if (SAVE_PREVIEW_BITMAP) {
      ImageUtils.saveBitmap(croppedBitmap);
    }

    runInBackground(
        new Runnable() {
          @Override
          public void run() {
```

```
        LOGGER.i("Running detection on image " + currTimestamp);
        final long startTime = SystemClock.uptimeMillis();
        final List<Classifier.Recognition> results =
detector.recognizeImage(croppedBitmap);
        lastProcessingTimeMs = SystemClock.uptimeMillis() - startTime;

        cropCopyBitmap = Bitmap.createBitmap(croppedBitmap);
        final Canvas canvas = new Canvas(cropCopyBitmap);
        final Paint paint = new Paint();
        paint.setColor(Color.RED);
        paint.setStyle(Style.STROKE);
        paint.setStrokeWidth(2.0f);

        float minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
        switch (MODE) {
          case TF_OD_API:
            minimumConfidence = MINIMUM_CONFIDENCE_TF_OD_API;
            break;
          case MULTIBOX:
            minimumConfidence = MINIMUM_CONFIDENCE_MULTIBOX;
            break;
          case YOLO:
            minimumConfidence = MINIMUM_CONFIDENCE_YOLO;
            break;
        }

        final List<Classifier.Recognition> mappedRecognitions =
            new LinkedList<Classifier.Recognition>();

        for (final Classifier.Recognition result : results) {
          final RectF location = result.getLocation();
          if (location != null && result.getConfidence() >= minimumConfidence) {
            canvas.drawRect(location, paint);
```

```
            cropToFrameTransform.mapRect(location);
            result.setLocation(location);
            mappedRecognitions.add(result);
          }
        }

        tracker.trackResults(mappedRecognitions, luminanceCopy, currTimestamp);
        trackingOverlay.postInvalidate();

        requestRender();
        computingDetection = false;
       }
      });
}

@Override
protected int getLayoutId() {
  return R.layout.camera_connection_fragment_tracking;
}

@Override
protected Size getDesiredPreviewFrameSize() {
  return DESIRED_PREVIEW_SIZE;
}

@Override
public void onSetDebug(final boolean debug) {
  detector.enableStatLogging(debug);
}

@Override
public void onBackPressed(){
```

```
    Intent intent = new Intent(DetectorActivity.this, ResultActivity.class);
    startActivity(intent);
  }
}
```

Here is the layout of the first screen activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="org.tensorflow.demo.MainActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="166dp"
        android:layout_height="200dp"
        android:layout_marginBottom="483dp"
        android:layout_marginLeft="40dp"
        android:src="@drawable/aegean_logo" />

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="325dp"
        android:layout_height="356dp"
        android:layout_above="@+id/author"
        android:layout_alignParentStart="true"
        android:layout_marginStart="29dp"
        android:layout_marginBottom="115dp"
        android:src="@drawable/logo" />
```

```
<TextView
    android:id="@+id/editText4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignBottom="@+id/imageView"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="-55dp"
    android:clickable="false"
    android:ems="17"
    android:focusable="false"
    android:focusableInTouchMode="false"
    android:gravity="center"
    android:text="@string/smart_fridge_detector"
    android:textColor="@color/black"
    android:textSize="30sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/capturePhoto"
    android:layout_width="86dp"
    android:layout_height="52dp"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="46dp"
    android:layout_marginBottom="70dp"
    android:background="@drawable/camera" />

<!-- <Button
    android:id="@+id/startClassifier"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
```

```
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="261dp"
        android:layout_marginBottom="22dp"
        android:text="@string/startClassifier"
        android:textAllCaps="false" />-->

    <Button
        android:id="@+id/startDetector"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="180dp"
        android:layout_marginBottom="71dp"
        android:text="@string/startDetector"
        android:textAllCaps="false" />

    <TextView
        android:id="@+id/author"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:textColor="@color/control_background"
        android:text="@string/author" />


</RelativeLayout>
```

Here is the final screen layout, activity_result.xml, appears after the model has run and detect the objects:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="org.tensorflow.demo.ResultActivity">


    <ImageView
        android:id="@+id/imageView"
        android:layout_width="166dp"
        android:layout_height="200dp"
        android:layout_marginBottom="483dp"
        android:layout_marginLeft="40dp"
        android:src="@drawable/aegean_logo" />

    <ImageView
        android:id="@+id/imageView3"
        android:layout_width="266dp"
        android:layout_height="277dp"
        android:layout_above="@+id/author"
        android:layout_alignParentStart="true"
        android:layout_marginStart="81dp"
        android:layout_marginBottom="8dp"
        android:src="@drawable/logo" />

    <TextView
        android:id="@+id/choose_what_todo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/imageView"
        android:layout_centerHorizontal="true"
```

```
    android:layout_marginBottom="-101dp"
    android:clickable="false"
    android:gravity="center"
    android:ems="17"
    android:focusable="false"
    android:focusableInTouchMode="false"
    android:text="@string/choose_what_todo"
    android:textColor="@color/black"
    android:textSize="22sp"
    android:textStyle="bold" />

<Button
    android:id="@+id/go_for_shopping"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="42dp"
    android:layout_marginBottom="355dp"
    android:text="@string/go_for_shopping"
    android:textAllCaps="false" />

<Button
    android:id="@+id/goBack"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentEnd="true"
    android:layout_alignParentBottom="true"
    android:layout_marginEnd="304dp"
    android:layout_marginBottom="358dp"
    android:text="@string/goBack"
    android:textAllCaps="false" />
```

```
    <TextView
        android:id="@+id/author"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        android:textColor="@color/control_background"
        android:text="@string/author" />


</RelativeLayout>
```

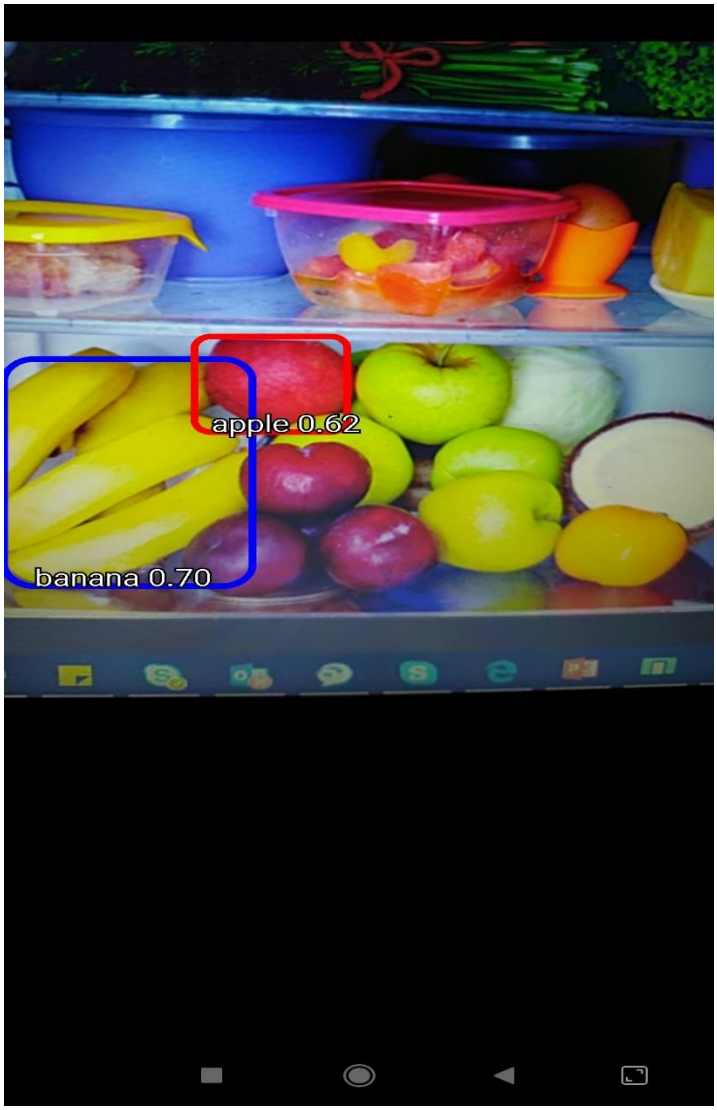
Below we will see the available screens of the application:

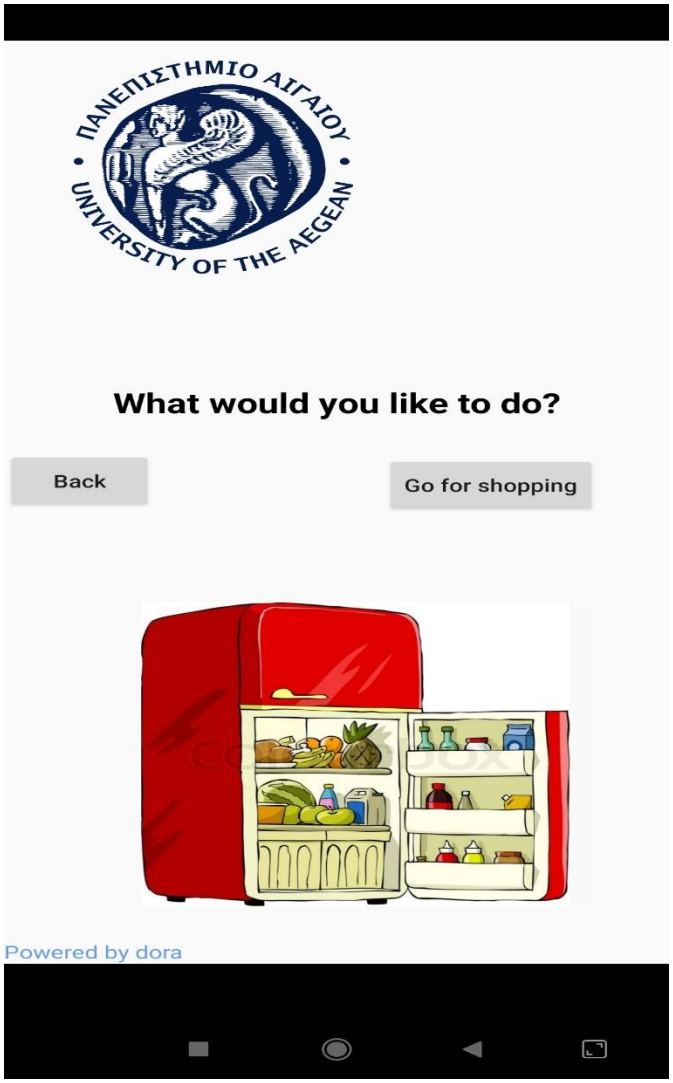
ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ
UNIVERSITY OF THE AEGEAN

Smart Fridge Detector

Detect

Powered by dora


apple 0.62
banana 0.70



In the first picture, the user have the option to detect via mobile phone's camera pressing the button Detect or he can import a photo by pressing the button with the camera icon.

In the second picture, we see how the model recognizes the objects inside the fridge by phone's camer.

In the third picture, we can see the final screen that the user can either go back to the first screen or press the button Go for shopping and a website of an online supermarket will open in order to shop.

7.0 Conclusion

The conclusion of this thesis is that we can train an object detection model with tensroflow, evaluate it and can make it run from an android application. This makes machine learning easy for everyone and the especially the fact that the model can be successfully deployed in an android application gives million opportunities for building applications that will make our lives much better.

8.0 Future Implementation

In the future the goal is for the the user to be able to create a profile in the application and add the items that wants to have always inside his fridge. The user will open his camera, the application will detect and list the object he has inside and the object he has not (based on the profile he have created) and finally it will connect him to an online supermarket so he can shop the missing items.

9.0 Bibliography

https://www.technologyreview.com

https://www.spotlightmetal.com

https://en.wikipedia.org/wiki/Machine_learning

https://bigdata-madesimple.com

https://en.wikipedia.org/wiki/Unsupervised_learning

https://en.wikipedia.org/wiki/Semi-supervised_learning

https://en.wikipedia.org/wiki/Reinforcement_learning

https://www.blog.consultants500.com/web-mobile-design-and-development/machine-learning-social-media-machines-impacting-social-networks/

https://medium.com/datadriveninvestor/the-place-of-machine-learning-and-artificial-intelligence-in-the-automotive-industry-618368db80f9

https://builtin.com/artificial-intelligence/machine-learning-healthcare

https://en.wikipedia.org/wiki/Deep_learning

https://machinelearningmastery.com/what-is-deep-learning/

https://en.wikipedia.org/wiki/Recursive_neural_network

https://www.forbes.com/sites/bernardmarr/2018/08/20/10-amazing-examples-of-how-deep-learning-ai-is-used-in-practice/#24c1f508f98a

https://www.tensorflow.org/

https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

https://medium.com/airbnb-engineering/categorizing-listing-photos-at-airbnb-f9483f3ab7e3

https://analyticsindiamag.com/twitters-adoption-of-tensorflow-might-have-improved-its-user-experience-dramatically/

https://blogs.dropbox.com/tech/2018/10/using-machine-learning-to-index-text-from-billions-of-images/

https://en.wikipedia.org/wiki/Python_(programming_language)

https://www.python.org/doc/essays/blurb/

https://thriveglobal.com/stories/why-python-is-good-for-data-analytics/

https://en.wikipedia.org/wiki/Data_analysis

https://github.com/tzutalin/labelImg

https://towardsdatascience.com/detecting-pikachu-on-android-using-tensorflow-object-detection-15464c7a60cd

https://jupyter.org/

https://en.wikipedia.org/wiki/Android_Studio