**Introduction of project**

Efficient and accurate object detection has been an important topic in the advancement of computer vision systems. With the advent of deep learning techniques, the accuracy for object detection has increased drastically. The project aims to incorporate state-of-the-art technique for object detection with the goal of achieving high accuracy with a real-time performance.

Motivation

In this project, we use a completely deep learning based approach to solve the problem of object detection in an end-to-end fashion. The resulting system is fast and accurate, thus aiding those applications which require object detection.

Problem statement

Many problems in computer vision were saturating on their accuracy before a decade. However, with the rise of deep learning techniques, the accuracy of these problems drastically improved. One of the major problems was that of image classification, which is defined as predicting the class of the image. A slightly complicated problem is that of image localization, where the image contains a single object and the system should predict the class of the location of the object in the image (a bounding box around the object). The more complicated problem (this project), of object detection involves both classification and localization. In this case, the input to the system will be a image, and the output will be a bounding box corresponding to all the objects in the image, along with the class of object in each box.

**Modules description**

This project took use of several software libraries, packages and programs to utilize machine learning. Python was the choice of programming language, and Tensor Flow was used for the deep learning computations, which in turn has a list of dependencies. Tensor Flow offers a version for CPU usage and another for GPU, this project used the GPU version. Said version requires extra programs from the GPU designer NVIDIA, such as CUDA Toolkit, cuDNN and their GPU drivers.

Python

**Python** is an [interpreted](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [general-purpose](https://en.wikipedia.org/wiki/General-purpose_programming_language) [programming language](https://en.wikipedia.org/wiki/Programming_language). Created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991, Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant whitespace](https://en.wikipedia.org/wiki/Off-side_rule). Its language constructs and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is [dynamically typed](https://en.wikipedia.org/wiki/Dynamic_programming_language) and [garbage-collected](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [procedural](https://en.wikipedia.org/wiki/Procedural_programming), object-oriented, and [functional programming](https://en.wikipedia.org/wiki/Functional_programming). Python is often described as a "batteries included" language due to its comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Tensor flow

**TensorFlow** is a [free](https://en.wikipedia.org/wiki/Free_software) and [open-source](https://en.wikipedia.org/wiki/Open-source_software) [software library](https://en.wikipedia.org/wiki/Library_(computing)) for [dataflow](https://en.wikipedia.org/wiki/Dataflow_programming) and [differentiable](https://en.wikipedia.org/wiki/Differentiable_programming) programming across a range of tasks. It is a symbolic math library, and is also used for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications such as [neural networks](https://en.wikipedia.org/wiki/Neural_networks). It is used for both research and production at [Google](https://en.wikipedia.org/wiki/Google).‍

TensorFlow was developed by the [Google Brain](https://en.wikipedia.org/wiki/Google_Brain) team for internal Google use. It was released under the [Apache License 2.0](https://en.wikipedia.org/wiki/Apache_License_2.0) on November 9, 2015.

CNN

In [deep learning](https://en.wikipedia.org/wiki/Deep_learning), a **convolutional neural network** (**CNN**, or **ConvNet**) is a class of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network), most commonly applied to analyzing visual imagery.

CNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptrons](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns.

SVM

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), **support-vector machines** (**SVMs**, also **support-vector networks**[[1]](https://en.wikipedia.org/wiki/Support-vector_machine#cite_note-CorinnaCortes-1)) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier) (although methods such as [Platt scaling](https://en.wikipedia.org/wiki/Platt_scaling) exist to use SVM in a probabilistic classification setting). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on the side of the gap on which they fall.

Features and functionalities

Methods for object detection generally fall into either [machine learning](https://en.wikipedia.org/wiki/Machine_learning) based approaches or [deep learning](https://en.wikipedia.org/wiki/Deep_learning)-based approaches. deep learning techniques that are able to do end-to-end object detection without specifically defining features, and are typically based on [convolutional neural networks](https://en.wikipedia.org/wiki/Convolutional_neural_networks) (CNN).  For Machine Learning approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as [support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM) to do the classification.

* Machine Learning approaches:
  + [Viola–Jones object detection framework](https://en.wikipedia.org/wiki/Viola%E2%80%93Jones_object_detection_framework) based on [Haar features](https://en.wikipedia.org/wiki/Haar-like_feature)
  + [Scale-invariant feature transform (SIFT)](https://en.wikipedia.org/wiki/Scale-invariant_feature_transform)
  + [Histogram of oriented gradients (HOG)](https://en.wikipedia.org/wiki/Histogram_of_oriented_gradients) features
* Deep Learning approaches:
  + Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
  + Single Shot Multi-Box Detector (SSD)
  + You Only Look Once (YOLO)

**Requirement analysis**

Software requirement

Before working on the Demo, let’s have a look at the prerequisites. We will be needing:

* Python
* Tensor Flow
* Tensor board
* [Protobuf v3.4 or above](https://github.com/google/protobuf/releases)

Hardware requirement

* Camera
* Quad core processor
* system

**Methodology and techniques used**

You need to go to the “**object detection**” directory and then create a new python file. You can use **Spyder** or **Jupyter** to write your code.

* First of all, we need to import all the libraries for ex. Numpy, tensor flow, zipfile, sys etc.
* Next, we will download the model which is trained on the [**COCO dataset**](http://cocodataset.org/#home). COCO stands for **Common Objects in Context.**
* Next, we provide the required model and the frozen inference graph generated by Tensor flow to use.
* It will download that model from the internet and extract the frozen inference graph of that model.
* Next, we are going to load all the labels.
* Now we will convert the images data into a numPy array for processing.
* The path to the images for the testing purpose is defined here. Here we have a naming convention “**image[i]**” for i in (1 to n+1), n being the number of images provided.
* This code runs the inference for a single image, where it detects the objects, make boxes and provide the class and the class score of that particular object.
* Our Final loop, which will call all the functions defined above and will run the inference on all the input images one by one, which will provide us the output of images in which objects are detected with labels and the percentage/score of that object being similar to the training data.

Technology used

* Python

**Limitations of project**

**Applications of project**

Facial Recognition:

A deep learning facial recognition system called the “**DeepFace**” has been developed by a group of researchers in the **Facebook,** which identifies human faces in a digital image very effectively. **Google** uses its own facial recognition system in Google Photos, which automatically segregates all the photos based on the person in the image. There are various components involved in Facial Recognition like the eyes, nose, mouth and the eyebrows.

People counting

Object detection can be also used for people counting, it is used for analyzing store performance or **crowd statistics** during festivals. These tend to be more difficult as people move out of the frame quickly.

Industrial quality check

Object detection is also used in industrial processes to identify products. Finding a specific object through visual inspection is a basic task that is involved in multiple industrial processes like sorting, inventory management, machining, quality management, packaging etc.

Inventory management can be very tricky as items are hard to track in **real time.**Automatic object counting and localization allows improving inventory accuracy.

Self driving cars

Self-driving cars are the Future, there’s no doubt in that. But the working behind it is very tricky as it combines a variety of techniques to perceive their surroundings, including radar, laser light, GPS, odometry, and computer vision.

Security

Object Detection plays a very important role in Security. Be it face ID of Apple or the retina scan used in all the sci-fi movies.

It is also used by the government to access the security feed and match it with their existing database to find any criminals or to detect the robbers’ vehicle.

The applications are limitless.

**Future scope**

The future of image processing will involve scanning the heavens for other intelligent life out in space. Also new intelligent, digital species created entirely by research scientists in various nations of the world will include advances in image processing applications. Due to advances in image processing and related technologies there will be millions and millions of robots in the world in a few decades time, transforming the way the world is managed. Advances in image processing and artificial intelligence6 will involve spoken commands, anticipating the information requirements of governments, translating languages, recognizing and tracking people and things, diagnosing medical conditions, performing surgery, reprogramming defects in human DNA, and automatic driving all forms of transport. With increasing power and sophistication of modern computing, the concept of computation can go beyond the present limits and in future, image processing technology will advance and the visual system of man can be replicated. The future trend in remote sensing will be towards improved sensors that record the same scene in many spectral channels. Graphics data is becoming increasingly important in image processing app1ications. The future image processing applications of satellite based imaging ranges from planetary exploration to surveillance applications.

Using large scale homogeneous cellular arrays of simple circuits to perform image processing tasks and to demonstrate pattern-forming phenomena is an emerging topic. The cellular neural network is an implementable alternative to fully connected neural networks and has evolved into a paradigm for future imaging techniques. The usefulness of this technique has applications in the areas of silicon retina, pattern formation, etc.

**Conclusion**

An accurate and efficient object detection system has been developed which achieves comparable metrics with the existing state-of-the-art system. This project uses recent techniques in the field of computer vision and deep learning. Custom dataset was created using labelImg and the evaluation was consistent. This can be used in real-time applications which require object detection for pre-processing in their pipeline. An important scope would be to train the system on a video sequence for usage in tracking applications. Addition of a temporally consistent network would enable smooth detection and more optimal than per-frame detection.

**Reference**

* <https://www.edureka.co/blog/tensorflow-object-detection-tutorial/>