

VIRTUALEYE - LIFE GUARD FOR SWIMMING POOLS TO DETECT ACTIVE DROWNING

Team ID : B3-3M5E

Team Members : 1. Keerthana M (814719104021)
2. Mahalakshmi L (814719104025)
3. Jeanne S (814719104017)
4. Vinoliya Jensi F (814719104057)

College Name : SRM TRP Engineering College, Trichy.

ABSTRACT

Swimming is one of the best exercises that helps people to reduce stress in this urban lifestyle. Swimming pools are found larger in number in hotels, and weekend tourist spots and barely people have them in their house backyard. Beginners, especially, often feel it difficult to breathe underwater which causes breathing trouble which in turn causes a drowning accident. Worldwide, drowning produces a higher rate of mortality without causing injury to children. Children under six of their age are found to be suffering the highest drowning mortality rates worldwide. Such kinds of deaths account for the third cause of unplanned death globally, with about 1.2 million cases yearly. To overcome this conflict, a meticulous system is to be implemented along the swimming pools to save human life. By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing more than 16 cameras underwater and ceiling and analyzing the video feeds to detect any anomalies. but AS a POC we make use of one camera that streams the video underwater and analyses the position of swimmers to assess the probability of drowning, if it is higher then an alert will be generated to attract lifeguards' attention.

CHAPTER 1

INTRODUCTION

This project focuses on using artificial intelligence and object classification technologies to analyse swimmer performance and generate useful data. While deciding on the general theme of the project, our advisor Brian McGinley put us in contact with Robert Mooney. Robert works with Swim Ireland(The National Governing Body for swimming, water polo, diving and synchronised swimming) and suggested we develop an application for the Irish swimming team that will help with analysing their swimmer performance data. This proved an opportunity to explore the use of AI and object classification technologies in real world situations. We worked with Robert throughout the project development and he also provided the video footage used to create our Swimmer Data Set.As this is a research project, there is no definite “finish line”. However, the ultimate goal is to provide a service that will enable the user to analyse swimmer stroke techniques, such as stroke counts per metres, stroke counts per minute, speed of swimmer etc. A project like this has never been attempted before, therefore our primary goal is to demonstrate that swimmer stroke analysis using artificial intelligence and machine learning is firstly, feasible, and secondly, accurate.

Deep learning has in current time place a stirring novel drift in machine learning. The national basics steps of deep learning square measure embedded within the ancient neural network (NN) method. no matter it's abnormal to most established use of NNs, deep learning accounts for the implementation of various unshown neurons Associate in Nursingd layers usually over 2 as an subject advantage of mutual with novel analysis paradigms. each lower-dimensional shelf associated with a superior sensory activity altitude. This epic high level of rank of construct redirects Associate in Nursing automatic attribute set, that alternatively would come with essential handmade or be spoke advantage.

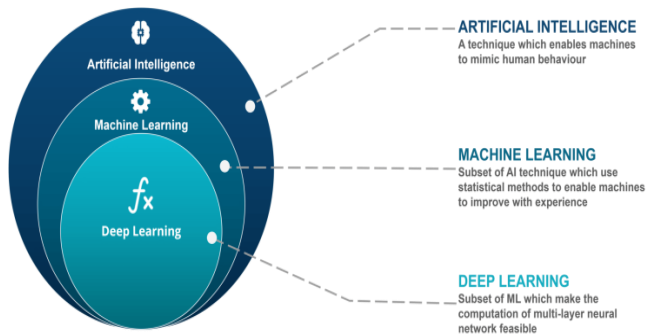


Figure 1: Deep Learning

In state of affairs declare as health science, which will be creating of those neutral behaviors set exclusive of human intrusion includes a ton of advantages. For the strategy, in medical imaging, it will provides a feature that square measure a lot of difficult and more durable to detail in communicative approach.

The deep learning can provide the particular conscious of knowing that this health condition, fruit calories level and lots of alternative health details a lot of economical to try to to functioning at the surroundings. This project can bring the convenience of attending to gathered info on the health care, hospital details, physique BMI and concerning this vacancy out there within the government sector.

DEEP LEARNING OVERVIEW

Most modern deep learning models are based on an artificial neural network, although they can also include propositional formulas or latent variables organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines. In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. (Of course, this does not completely

obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction.)

The "deep" in "deep learning" refers to the number of layers through which the data is transformed. More precisely, deep learning systems have a substantial credit assignment path (CAP) depth. The CAP is the chain of transformations from input to output. CAPs describe potentially causal connections between input and output. For a feedforward neural network, the depth of the CAPs is that of the network and is the number of hidden layers plus one (as the output layer is also parameterized). For recurrent neural networks, in which a signal may propagate through a layer more than once, the CAP depth is potentially unlimited. No universally agreed upon threshold of depth divides shallow learning from deep learning, but most researchers agree that deep learning involves CAP depth > 2 . CAP of depth 2 has been shown to be a universal approximator in the sense that it can emulate any function. Beyond that more layers do not add to the function approximator ability of the network. Deep models (CAP > 2) are able to extract better features than shallow models and hence, extra layers help in learning features.

Deep learning architectures are often constructed with a greedy layer-by-layer method. Deep learning helps to disentangle these abstractions and pick out which features improve performance. For supervised learning tasks, deep learning methods obviate feature engineering, by translating the data into compact intermediate representations akin to principal components, and derive layered structures that remove redundancy in representation. Deep learning algorithms can be applied to unsupervised learning tasks. This is an important benefit because unlabeled data are more abundant than labeled data. Examples of deep structures that can be trained in an unsupervised manner are neural history compressors and deep belief networks.

1.2 NEURAL NETWORK

Artificial neural networks (ANNs) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve their ability) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express with a traditional computer algorithm using rule-based programming. An ANN is based on a collection of connected units called artificial neurons, (analogous to biological neurons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream.

Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times. The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as back propagation, or passing information in the reverse direction and adjusting the network to reflect that information. Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis.

1.3 DEEP NEURAL NETWORKS

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculates the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks. The goal is that eventually, the network will be trained to decompose an image into features, identify trends that exist across all samples and classify new images by their similarities without requiring human input.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. Deep architectures include many variants of a few basic approaches. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets.

DNNs are typically feed forward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network didn't accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines

the correct mathematical manipulation to fully process the data. Recurrent neural networks (RNNs), in which data can flow in any direction, are used for applications such as language modeling. Long short-term memory is particularly effective for this use. Convolutional deep neural networks (CNNs) are used in computer vision. CNNs also have been applied to acoustic modeling for automatic speech recognition (ASR)

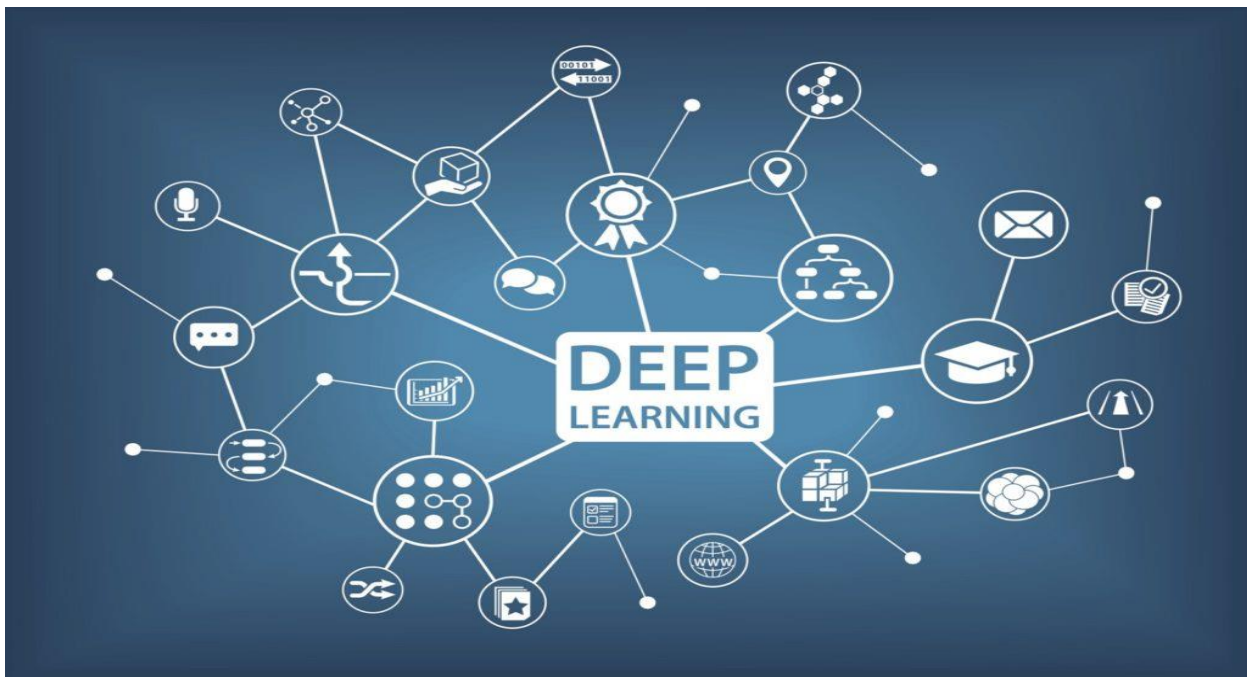


Figure 2 Application of Deep Learning

1.4 APPLICATIONS OF BIG DATA ACROSS INDUSTRY SECTORS

Applications

Large-scale automatic speech recognition is the first and most convincing successful case of deep learning. LSTM RNNs can learn "Very Deep Learning" tasks that involve multi-second intervals containing speech events separated by thousands of discrete time steps, where one time step corresponds to about 10 ms. LSTM with forget gates is competitive with traditional speech recognizers on certain tasks.

The initial success in speech recognition was based on small-scale recognition tasks based on TIMIT. The data set contains 630 speakers from eight major dialects of American English, where each speaker reads 10 sentences. Its small size lets many configurations be tried. More importantly, the TIMIT task concerns phone-sequence recognition, which, unlike word-sequence recognition, allows weak phone bigram language models. This lets the strength of the acoustic modeling aspects of speech recognition be more easily analyzed. The error rates listed below, including these early results and measured as percent phone error rates (PER), have been summarized since 1991.

Image recognition

A common evaluation set for image classification is the MNIST database data set. MNIST is composed of handwritten digits and includes 60,000 training examples and 10,000 test examples. As with TIMIT, its small size lets users test multiple configurations. A comprehensive list of results on this set is available. Deep learning-based image recognition has become "superhuman", producing more accurate results than human contestants. This first occurred in 2011. Deep learning-trained vehicles now interpret 360° camera views. Another example is Facial Dysmorphology Novel Analysis (FDNA) used to analyze cases of human malformation connected to a large database of genetic syndromes.

Visual art processing

Closely related to the progress that has been made in image recognition is the increasing application of deep learning techniques to various visual art tasks. DNNs have proven themselves capable, for example, of a) identifying the style period of a given painting, b) Neural Style Transfer - capturing the style of a given artwork and applying it

in a visually pleasing manner to an arbitrary photograph or video, and c) generating striking imagery based on random visual input fields.

Natural language processing

Neural networks have been used for implementing language models since the early 2000s. LSTM helped to improve machine translation and language modeling. Other key techniques in this field are negative sampling and word embedding. Word embedding, such as *word2vec*, can be thought of as a representational layer in a deep learning architecture that transforms an atomic word into a positional representation of the word relative to other words in the dataset; the position is represented as a point in a vector space. Using word embedding as an RNN input layer allows the network to parse sentences and phrases using an effective compositional vector grammar. A compositional vector grammar can be thought of as probabilistic context free grammar (PCFG) implemented by an RNN. Recursive auto-encoders built atop word embeddings can assess sentence similarity and detect paraphrasing. Deep neural architectures provide the best results for constituency parsing, sentiment analysis, information retrieval, spoken language understanding, machine translation, contextual entity linking, writing style recognition, Text classification and others. Recent developments generalize word embedding to sentence embedding.

Google Translate (GT) uses a large end-to-end long short-term memory network. Google Neural Machine Translation (GNMT) uses an example-based machine translation method in which the system "learns from millions of examples." It translates "whole sentences at a time, rather than pieces. Google Translate supports over one hundred languages. The network encodes the "semantics of the sentence rather than simply memorizing phrase-to-phrase translations". GT uses English as an intermediate between most language pairs.

Drug discovery and toxicology

A large percentage of candidate drugs fail to win regulatory approval. These failures are caused by insufficient efficacy (on-target effect), undesired interactions (off-target effects), or unanticipated toxic effects. Research has explored use of deep learning to predict the biomolecular targets, off-targets, and toxic effects of environmental chemicals in nutrients, household products and drugs. AtomNet is a deep learning system for structure-based rational drug design. AtomNet was used to predict novel candidate biomolecules for disease targets such as the Ebola virus and multiple sclerosis.

Customer relationship management

Deep reinforcement learning has been used to approximate the value of possible direct marketing actions, defined in terms of RFM variables. The estimated value function was shown to have a natural interpretation as customer lifetime value.

Recommendation systems

Recommendation systems have used deep learning to extract meaningful features for a latent factor model for content-based music recommendations. Multiview deep learning has been applied for learning user preferences from multiple domains. The model uses a hybrid collaborative and content-based approach and enhances recommendations in multiple tasks.

Bioinformatics

An autoencoder ANN was used in bioinformatics, to predict gene ontology annotations and gene-function relationships. In medical informatics, deep learning was used to predict sleep quality based on data from wearables and predictions of health

complications from electronic health record data. Deep learning has also showed efficacy in healthcare.

Artificial intelligence and deep learning in relation to healthcare

In past years the healthcare field has seen benefits from the universalization of artificially intelligent machines to perform tasks (for example, measuring blood pressure), but not to learn how to interpret what we know. This is the major development we are seeing now that could drastically advance our diagnostic abilities. In correlation we might also observe better survival rates for certain diseases as we are able to focus on the prevention of disease, rather than the backwards approach of trying to treat the disease once it has been acquired. This is being done through a variant of artificial intelligence called “deep learning,” in which software “learns to recognize patterns in distinct layers... and each neural-network layer operates both independently and in concert, separating aspects such as color, size and shape before integrating the outcomes” of medical imaging. This advancement in the use of visual tools is vital to the advancement of medical diagnostics. The key idea here is that in past years we’ve only utilized artificial intelligence at its most basic level—by teaching machines to perform human skills. While this has been a great advancement, the new ideas being explored about machine learning and deep learning are having an even greater impact on healthcare now and will continue to yield success upon future discovery. “These newer visual tools hold the promise of transforming diagnostic medicine and can even search for cancer at the individual cell level.”

CHAPTER -2

LITERATURE SURVEY

2.1 POSEIDON- Video based drowning detection system in the swimming pool

Swimming pool drowning monitoring system based on video technology is mostly reported in the literature. There are three kinds drowning monitoring system according to the different position of the camera. One is that the camera is mounted on the underwater swimming pool wall, then monitor underwater swimmer status. A limitation of this equipment is that if too many swimmers, the occlusion problem arises. The other is that the camera is mounted upon the water, and monitors the Swimmer posture change. The reflection and refraction of light in air-water interference will affect the image quality, and drowning man feature this method detected is not easy to distinguish swimmers and divers obviously. The third is a combination of the two, underwater camera and aerial camera matched, monitoring the swimmer posture. This system needs constant observation which is the main disadvantage.

2.2 Wearable devices for early monitoring and alarming for drowning incidents

The wearable drowning monitor device can detect drowning accident and alarm. The device has seven main modules, including microprocessor, power module, SD memory card module, LED warning module, acceleration sensor module, water pressure sensor module, and keys module. When swimming the human arm must constantly waving in the water, if drowning, arm motion of floating is significantly reduced, and if falling into the water, almost motionless. According to the physiological response of human drowning, it can detect drowning accident by recording arm motion real-time through wearable wrist accelerometer device. This accelerometer is packed with embedded functions with edible user programmable options, configurable to two interrupt pins. The pressure sensor is installed to judge whether the human body is in the water. The red LED is used for drowning warning. One blue LED is used to get the work status

of the device that will flash every few seconds in order to save the precious energy. Because LED light emitting angle generally relatively small, 5 red LED lights of upward and around direction is installed to make LED alarm signal caller. Two keys are designed for the demo device.

2.3 LDR based automated drowning detection system in the swimming pool

In the proposed method the human identification in the swimming pool depends on the LDR and laser. First, data from a water pressure sensor is used to judge whether the human body is in the water, if the body is in the water, then start downloading judgment process. The iron metal plate is placed in the floor of the swimming pool. The laser and the LDR source are placed in the side of the wall. Here we are using an ATmega81 microcontroller to control the whole process. Embedded C language is used for the coding. Initially the laser source that spreads over the swimming pool and the LDR which sense the laser light and which produces the resistance value. Depends on the resistance value the process has been taken. When the LDR value will be kept constant then the alarm will be activated. The resistance value will be changed with respect to the human movement. The message will be sent to the administration by using the GSM service. After 30 seconds there is no change which means the plate will lift automatically using the motor and motor driver. The human is safe in this technique.

2.4 AUTOMATED DROWNING DETECTION AND SECURITY IN SWIMMING POOL

Every year, many individuals, including kids under the age of 5 drown in the depths of the swimming pool, and the lifeguards are not well trained enough to handle these situations. Thus arises the requirement for having a system that will consequently detect the drowning individuals and alarm the life guard at such risk. Swimming pool surveillance systems play an essential role in safeguarding the premises. In this project differential pressure approach is used for detection of drowning incidents in swimming pools at the earliest possible stage. The children's life is saved during drowning incidents

in the swimming pool by lifting the acrylic plate. The proposed approach consists of RF module, Pressure Sensor and Motor Driver. The demo system based on pressure sensor has an advantage of convenience, cost saving and simple algorithm. Swimming is a kid's favorite aquatic sport and it's a great stress buster. But in the water, beginners often feel hard to breathe which causes choking actions, loss of balance and results in a drowning accident. Some special circumstances, such as cramps, collide with each other, disease or mental stress and so on may also cause swimmer to drown. Drowning is a leading cause of death and disability for children. Worldwide, drowning produces a higher mortality rate than any other cause of injury in children less than 15 years of age [3]. Younger kids underneath the age of five are at precise threat, suffering the very best drowning mortality rates international. According to the Centers for Disease Control and Prevention, approximately one thousand children die from drowning annually in the world. In this project drowning accidents is avoided automatically by using the acrylic plate. The earliest swimming alarm system appears in the 1976, then there are some patent applications, but due to various reasons, these techniques are not popular[1]. In 2001, the French Vision IQ company produced the world's first set of drowning alarm system Poseidon; this is the first commercial promotion system. In 2003, Singapore Nan Yang, University of Technology design DEWS.

CHAPTER 3

3.1 PROBLEM DEFINITION

Swimming pool is that the method of recognizing the face of a relevant person by a vision system. As this face recognition plays an important role in several Human laptop Interaction applications. the fundamental face recognition methodology is as follows: one. Pre-processing stage: Color area conversion and size of pictures two. Feature extraction: Extraction of face expression three. Classification: The extracted feature set is classed. There ar several algorithms within which the face recognition is being done which incorporates ancient ways. These embody Support Vector Machines (SVM), Principal Component Analysis (PCA), Speed-Up Robust Features (SURF), Linear Discriminant Analysis (LDA) etc. There also are some fashionable ways like Gabor Wavelet Transform, Linear Binary Pattern (LBP), Histogram of Gradients (HOG) that perform face recognition with higher potency

DISADVANTAGES

- The ancient methodology consumes longer, is pricey, potency of result's attenuate thanks to cause variation and lighting conditions.
- Random decisions are created whereas considering the options.
- It fails once the image is either too shut or too way and thence needs a selected distance.
- Once quality will increase it fails once enforced in real time.

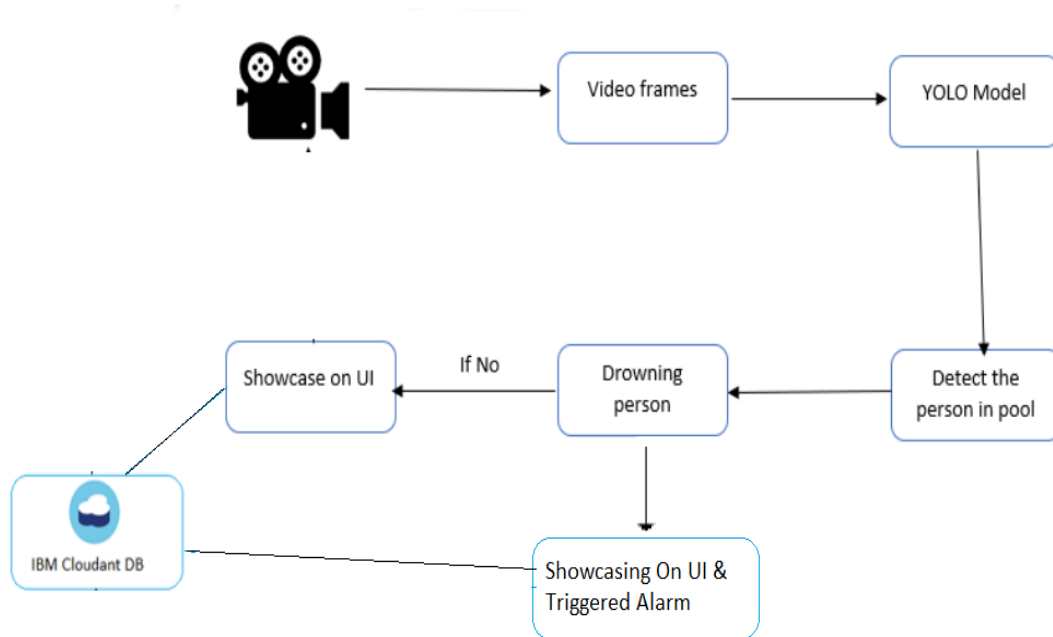
4. SYSTEM DESIGN

4.1 PROPOSED SYSTEM

The automated drowning detection system works on the principle of differential pressure. The system contains two fundamental modules: to begin with, the wristband consisting of Video Image Upload. Second, the video frame by frame image display module at the swimming pool site. The children entering the pool territory should wear the wristband. The Pressure at underwater is different and greater than the pressure at the air - water interface. The pressure at a particular depth is measured and set as the threshold. Once the child gets into the pool, the pressure is continuously measured and monitored by the microcontroller. When the current value surpasses the threshold limit, an alerting signal is sent to the receiver. The video frame segmentation and reception of open cv is done through Python A module. Our solution consists of firstly; a data set created using VOTT and LabelImg. Tensorflow - the framework used to train our Neural Network using this data set. Google Cloud VM where the training takes place. OpenCV for image processing and video manipulation & creation.

4.2 SYSTEM ARCHITECTURE

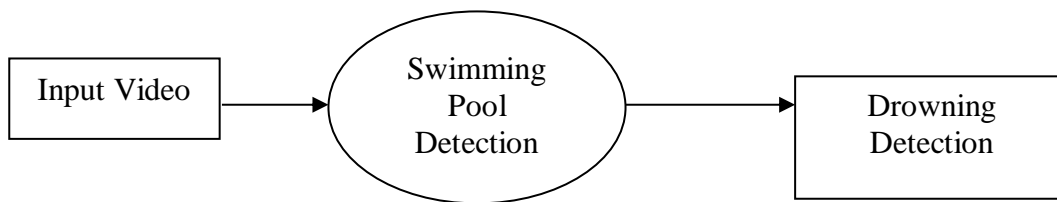
A system architecture or systems architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system. System architecture can comprise system components, the externally visible properties of those components, the relationships (e.g. the behavior) between them.



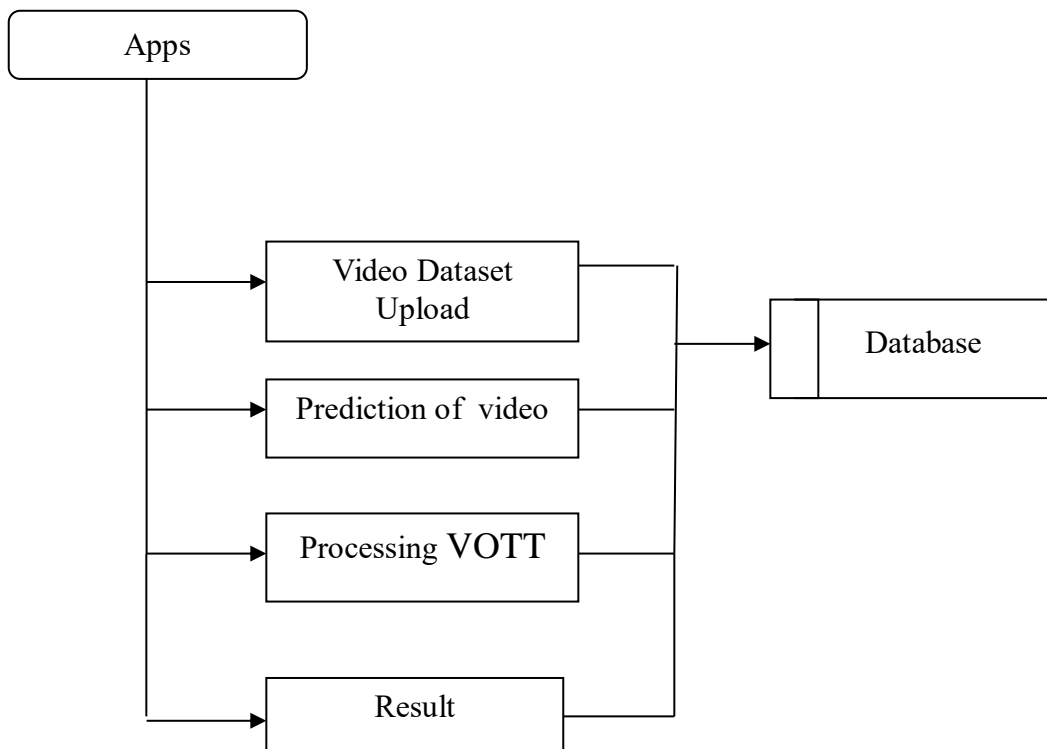
4. 3 DATA FLOW DIAGRAM

A two-dimensional diagram that explains how data is processed and transferred in a system. The graphical depiction identifies each source of data and how it interacts with other data sources to reach a common output. Individuals seeking to draft a data flow diagram must (1) identify external inputs and outputs, (2) determine how the inputs and outputs relate to each other, and (3) explain with graphics how these connections relate and what they result in. This type of diagram helps business development and design teams visualize how data is processed and identify or improve certain aspects.

Level 0



Level 1



4.6 MODULE DESCRIPTION

OPEN CV

OpenCV refers to ASCII text file laptop Vision, machine learning, and image process and currently it plays a serious role in data processing that is extremely necessary in today's systems. By mistreatment it, one will method pictures and videos to spot objects, faces, or perhaps handwriting of an individual's. once it's integrated with numerous libraries, like NumPy, python is capable of process the OpenCV array structure for analysis. to spot image pattern and its numerous options we tend to use vector area and perform mathematical operations on these options. it's a cross- platform middle-to-high level API that consists of a number of hundred C functions. it's free for each non-commercial and business use. OpenCV has the advantage of being a multi-platform framework; it supports each Windows and UNIX, and additional recently, Mac OS. It needs less RAM for its usage, it perhaps of 60-70 MB. mistreatment OpenCV library, can –

Read and write pictures

Capture and save videos

Process pictures (filter, transform)

Perform feature detection

Labelling tools (Labellmg & VOTT)

- We used both VOTT and Labellmg for the labelling process.
- VOTT and Labellmg provides end to end support for generating datasets and validating object detection models from video and image assets. VOTT provided easy integration with tensorflow, as it exported the tf file along with the labels. It also provided support for video labelling.

- Labellmg is a light and memory friendly application that could easily label images. The software is smooth and easy to work with.

Training swimmer objects

Videos of swimmers are imported into labelling programs (VOTT/Labellmg). Boxes are drawn around the swimmers and assigned a label. About 800 images were labelled in total.

Generating Tensor Records

After labelling the videos, the exported labels are converted into tensor records. This was done by converting the xml labels into csv format by the `xml_to_csv.py` script, and then parsing the csv file into the `generateTFRecords.py` script.

Training the model

The labels, converted csv file, tensor records, images and label map, along with a pre-trained model and its config file are fed into the training script. This process takes a few hours, as we can see from the image below, the model needs about 16 seconds per step (image) for each image parsed. However, this time improves as training progresses.

Object-detection script

After the model has been trained, the user can now parse in any swimming video into the object-detection script. This process too takes a few hours, as Tensorflow will generate bounding boxes around the detected objects (as shown below).

Generate and Analyse Results

A csv result file (as shown below) as well as the output images are generated. The csv file can be imported into Microsoft Excel to analyse the result and to render a graph.

CHAPTER 5

SYSTEM REQUIREMENTS

5.1 GENERAL

There are two requirements to do this project.

They are:

1. Hardware Requirements.
2. Software Requirements

5.2 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented.

RAM: Minimum 8 GB

System Type: 64-bit Operating System or above

Processor: Intel core i7

GPU: Nvidia G-force GTX 1650

5.3 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating

cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Operating System: Windows

10 ***Platform Used:*** Visual

Studio Code ***Coding***

Language: Python

PYTHON LIBRARIES:

- NumPy
- Pandas
- Matplotlib
- Keras
- Comma Separated Values (CSV)

5.3 SOFTWARE DESCRIPTION

5.3.1 PYTHON

Python is an interpreter, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your

program before executing it. This is similar to PERL and PHP. Python is Interactive – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python is Object-Oriented – Python supports Object-Oriented style or technique of programming that encapsulates code within objects. Python is a Beginner's Language – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

TENSORFLOW- INTRODUCTION

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. The official website of TensorFlow is mentioned below: <https://www.tensorflow.org/>



Let us now consider the following important features of TensorFlow:

- It includes a feature of that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural networks and machine learning techniques.

- It includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Why is TensorFlow So Popular?

TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities and methods for the same. TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

TensorFlow — Installation

To install TensorFlow, it is important to have “Python” installed in your system. Python version 3.4+ is considered the best to start with TensorFlow installation. Consider the following steps to install TensorFlow in Windows operating system.

pip install tensorflow

```

Command Prompt - pip install tensorflow
Requirement already satisfied: termcolor>=1.1.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.1.0)
Requirement already satisfied: numpy>=1.13.3 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.14.5)
Requirement already satisfied: grpcio>=1.8.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: wheel>=0.26 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.31.1)
Requirement already satisfied: six>=1.10.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.11.0)
Requirement already satisfied: absl-py>=0.1.6 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.2)
Requirement already satisfied: astor>=0.6.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.6.2)
Requirement already satisfied: gast>=0.2.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: tensorboard<1.9.0,>=1.8.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.8.0)
Requirement already satisfied: setuptools in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (39.2.0)
Requirement already satisfied: html5lib==0.9999999 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.9999999)
Requirement already satisfied: bleach==1.5.0 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (1.5.0)
Requirement already satisfied: markdown>=2.6.8 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (2.6.11)
Requirement already satisfied: werkzeug>=0.11.10 in c:\users\radhika\anaconda2\envs\tensorflow\lib\site-packages (from tensorflow) (0.14.1)
Installing collected packages: tensorflow

```

TensorFlow — Convolutional Neural Networks

After understanding machine-learning concepts, we can now shift our focus to deep learning concepts. Deep learning is a division of machine learning and is considered as a crucial step taken by researchers in recent decades. The examples of deep learning implementation include applications like image recognition and speech recognition.

Following are the two important types of deep neural networks:

- Convolutional Neural Networks
 - Recurrent Neural Networks
- In this chapter, we will focus on the CNN, Convolutional Neural Networks

Convolutional Neural Networks

Convolutional Neural networks are designed to process data through multiple layers of arrays. This type of neural networks is used in applications like image recognition or face recognition. The primary difference between CNN and any other ordinary neural network is that CNN takes input as a two-dimensional array and operates directly on the images rather than focusing on feature extraction which other neural networks focus on. The dominant approach of CNN includes solutions for problems of recognition. Top companies like Google and Facebook have invested in research and development towards recognition projects to get activities done with greater speed.

A convolutional neural network uses three basic ideas:

- Local receptive fields
- Convolution
- Pooling

Let us understand these ideas in detail.

CNN utilizes spatial correlations that exist within the input data. Each concurrent layer of a neural network connects some input neurons. This specific region is called

local receptive field. Local receptive field focusses on the hidden neurons. The hidden neurons process the input data inside the mentioned field not realizing the changes outside the specific boundary.

If we observe the above representation, each connection learns a weight of the hidden neuron with an associated connection with movement from one layer to another. Here, individual neurons perform a shift from time to time. This process is called “convolution”. The mapping of connections from the input layer to the hidden feature map is defined as “shared weights” and bias included is called “shared bias”. CNN or convolutional neural networks use pooling layers, which are the layers, positioned immediately after CNN declaration. It takes the input from the user as a feature map that comes out of convolutional networks and prepares a condensed feature map. Pooling layers helps in creating layers with neurons of previous layers.

PILLOW

PYTHON PILLOW — OVERVIEW

In today’s digital world, we come across lots of digital images. In case, we are working with Python programming language, it provides lot of image processing libraries to add image processing capabilities to digital images. Some of the most common image processing libraries are: OpenCV, Python Imaging Library (PIL), Scikit-image, Pillow. However, in this tutorial, we are only focusing on Pillow module and will try to explore various capabilities of this module. Pillow is built on top of PIL (Python Image Library). PIL is one of the important modules for image processing in Python. However, the PIL module is not supported since 2011 and doesn’t support python 3. Pillow module gives more functionalities, runs on all major operating system and support for python 3. It supports wide variety of images such as “jpeg”, “png”, “bmp”, “gif”, “ppm”, “tiff”. You can do almost anything on digital images using pillow module. Apart from basic image processing

functionality, including point operations, filtering images using built-in convolution kernels, and color space conversions.

IMAGE ARCHIVES

The Python Imaging Library is best suited for image archival and batch processing applications. Python pillow package can be used for creating thumbnails, converting from one format to another and print images, etc.

IMAGE DISPLAY

You can display images using Tk PhotoImage, BitmapImage and Windows DIB interface, which can be used with PythonWin and other Windows-based toolkits and many other Graphical User Interface (GUI) toolkits.

For debugging purposes, there is a `show ()` method to save the image to disk which calls the external display utility.

Image Processing

The Pillow library contains all the basic image processing functionality. You can do image resizing, rotation and transformation.

Pillow module allows you to pull some statistics data out of image using histogram method, which later can be used for statistical analysis and automatic contrast enhancement.

PYTHON PILLOW — ENVIRONMENT SETUP

This chapter discusses how to install pillow package in your computer.

Installing pillow package is very easy, especially if you're installing it using pip.

Installing Pillow using pip

To install pillow using pip, just run the below command in your command prompt:

```
python -m pip install pip
```

```
python -m pip install pillow
```

In case, if pip and pillow are already installed in your computer, above commands will simply mention the ‘requirement already satisfied’ as shown below:

```
C:\Users\yadur>python -m pip install pip
Requirement already satisfied: pip in c:\python381\lib\site-packages (19.3.1)

C:\Users\yadur>python -m pip install pillow
Requirement already satisfied: pillow in c:\python381\lib\site-packages (7.0.0)
```

PYTHON PILLOW — USING IMAGE MODULE

To display the image, pillow library is using an image class within it. The image module inside pillow package contains some important inbuilt functions like, load images or create new images, etc.

Opening, rotating and displaying an image

To load the image, we simply import the image module from the pillow and call the Image.open(), passing the image filename.

Instead of calling the Pillow module, we will call the PIL module as to make it backward compatible with an older module called Python Imaging Library (PIL). That’s why our code starts with “from PIL import Image” instead of “from Pillow import Image”.

Next, we’re going to load the image by calling the Image.open() function, which returns a value of the Image object data type. Any modification we make to the image object can be saved to an image file with the save() method. The image object we received using Image.open(), later can be used to resize, crop, draw or other image manipulation method calls on this Image object.

Example

Following example demonstrates the rotation of an image using python pillow:

```
from PIL import Image
#Open image using Image module
im = Image.open("images/cuba.jpg")
```

```
#Show actual Image im.show()
#Show rotated Image
im = im.rotate(45) im.show()
```

Output

If you save the above program as Example.py and execute, it displays the original and rotated images using standard PNG display utility, as follows:

SCIPY

What is SciPy

The SciPy is an open-source scientific library of Python that is distributed under a BSD license. It is used to solve the complex scientific and mathematical problems. It is built on top of the Numpy extension, which means if we import the SciPy, there is no need to import Numpy. The **Scipy** is pronounced as **Sigh pi**, and it depends on the Numpy, including the appropriate and fast N-dimension array manipulation.

It provides many user-friendly and effective numerical functions for numerical integration and optimization.

The **SciPy** library supports **integration, gradient optimization, special functions, ordinary differential equation solvers, parallel programming tools**, and many more. We can say that **SciPy** implementation exists in every complex numerical computation.

The **scipy** is a data-processing and system-prototyping environment as similar to MATLAB. It is easy to use and provides great flexibility to scientists and engineers.

History

Python was expanded in the 1990s to include an array type for numerical computing called numeric. This numeric package was replaced by Numpy (blend of Numeric and NumArray) in 2006. There was a growing number of extension module and developers were interested to create a complete environment for scientific and technical computing. **Travis Oliphant**, **Eric Jones**, and **Pearu Peterson** merged code they had written and called the new package **SciPy**. The newly created package provided a standard collection of common numerical operation on the top of Numpy.

Why use SciPy?

SciPy contain significant mathematical algorithms that provide easiness to develop sophisticated and dedicated applications. Being an open-source library, it has a large community across the world to the development of its additional module, and it is much beneficial for scientific application and data scientists.

Numpy vs. SciPy

Numpy and SciPy both are used for mathematical and numerical analysis. Numpy is suitable for basic operations such as sorting, indexing and many more because it contains array data, whereas SciPy consists of all the numeric data.

Numpy contains many functions that are used to resolve the linear algebra, Fourier transforms, etc. whereas SciPy library contains full featured version of the linear algebra module as well many other numerical algorithms.

SciPy Sub - Packages

SciPy has the number of sub-packages for the various scientific computing domains. The following table is given below:

| Sr. | Sub-Package | Description |
|-----|---------------------|---|
| 1. | scipy.cluster | Cluster algorithms are used to vector quantization/ Kmeans. |
| 2. | scipy.constants | It represents physical and mathematical constants. |
| 3. | scipy.fftpack | It is used for Fourier transform . |
| 4. | scipy.integrate | Integration routines |
| 5. | scipy.interpolation | Interpolation |
| 6. | scipy.linalg | It is used for linear algebra routine. |
| 7. | scipy.io | It is used for data input and output. |
| 8. | scipy.ndimage | It is used for the n-dimension image. |
| 9. | scipy.odr | Orthogonal distance regression. |
| 10. | scipy.optimize | It is used for optimization. |
| 11. | scipy.signal | It is used in signal processing. |
| 12. | scipy.sparse | Sparse matrices and associated routines. |
| 13. | scipy.spatial | Spatial data structures and algorithms. |
| 14. | scipy.special | Special Function. |
| 15. | scipy.stats | Statistics. |

| | | |
|-----|--------------|---------------------------|
| 16. | scipy.weaves | It is a tool for writing. |
|-----|--------------|---------------------------|

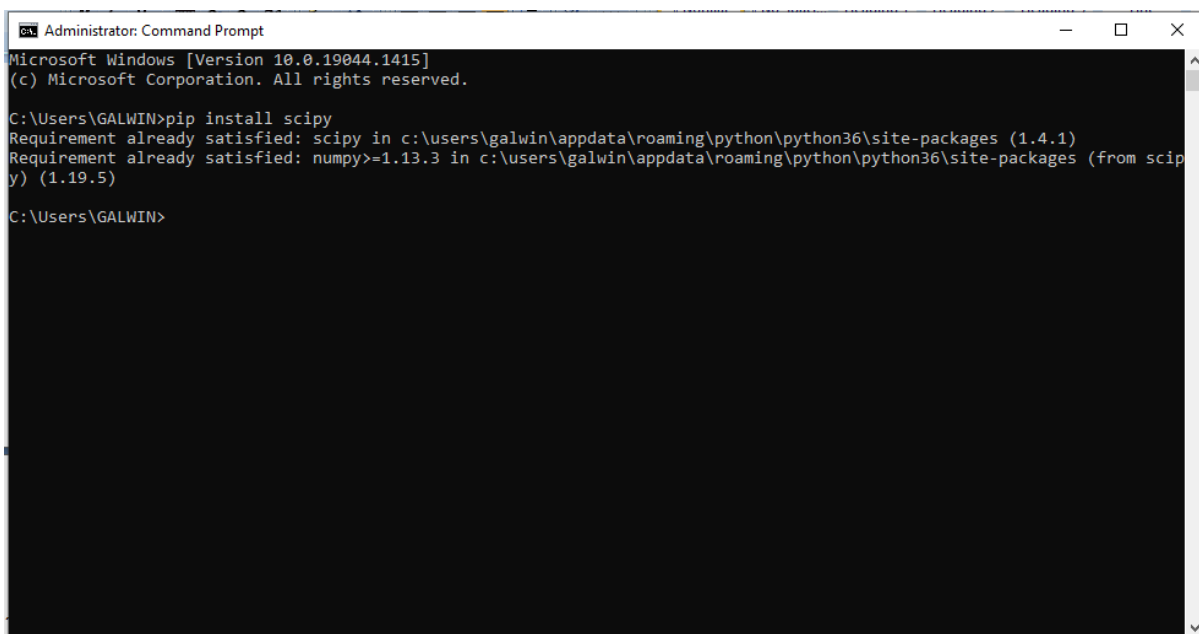
SciPy Installation

We will learn about the core functionality of SciPy. Before working with SciPy, it should be installed in the system.

- **Install SciPy using pip**

We can install the SciPy library by using **pip** command; run the following command in the terminal:

1. `pip install scipy`



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\GALWIN>pip install scipy
Requirement already satisfied: scipy in c:\users\galwin\appdata\roaming\python\python36\site-packages (1.4.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\galwin\appdata\roaming\python\python36\site-packages (from scipy) (1.19.5)

C:\Users\GALWIN>
```

OPEN CV

OpenCV is an open-source library for the computer vision. It provides the facility to the machine to recognize the faces or objects. In this tutorial we will learn the concept of OpenCV using the Python programming language.

Our OpenCV tutorial includes all topics of Read and Save Image, Canny Edge Detection, Template matching, Blob Detection, Contour, Mouse Event, Gaussian blur and so on.

What is OpenCV?

OpenCV is a Python open-source library, which is used for computer vision in Artificial intelligence, Machine Learning, face recognition, etc.



In OpenCV, the CV is an abbreviation form of a computer vision, which is defined as a field of study that helps computers to understand the content of the digital images such as photographs and videos.

The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on. For example, cars can be facilitated with

computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly.

Computer vision allows the computer to perform the same kind of tasks as humans with the same efficiency. There are a two main task which are defined below:

- **Object Classification** - In the object classification, we train a model on a dataset of particular objects, and the model classifies new objects as belonging to one or more of your training categories.
- **Object Identification** - In the object identification, our model will identify a particular instance of an object - for example, parsing two faces in an image and tagging one as Virat Kohli and other one as Rohit Sharma.

History

OpenCV stands for Open Source Computer Vision Library, which is widely used for image recognition or identification. It was officially launched in 1999 by Intel. It was written in C/C++ in the early stage, but now it is commonly used in Python for the computer vision as well.

The first alpha version of OpenCV was released for the common use at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and between 2001 and 2005, five betas were released. The first 1.0 version was released in 2006.

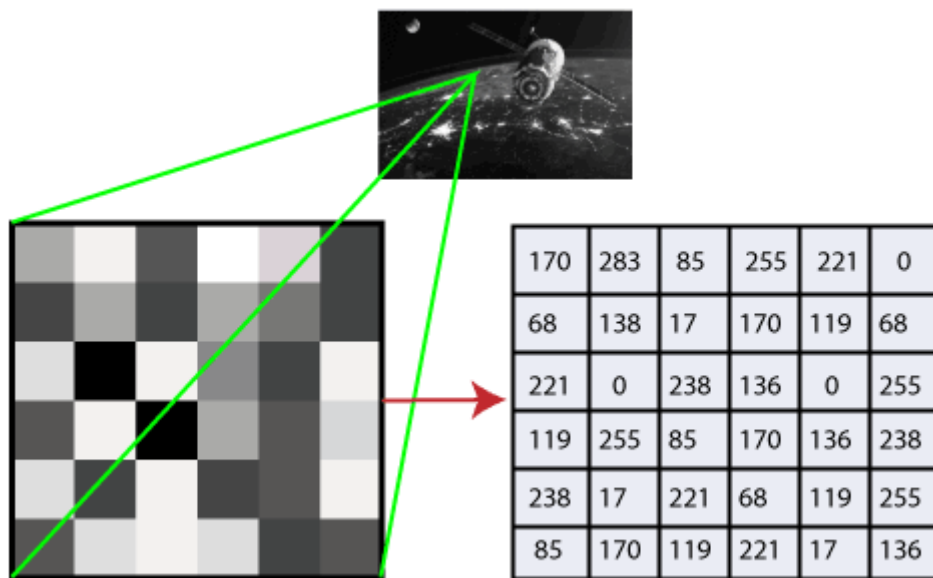
The second version of the OpenCV was released in October 2009 with the significant changes. The second version contains a major change to the C++ interface, aiming at easier, more type-safe, pattern, and better implementations. Currently, the development is done by an independent Russian team and releases its newer version in every six months.

How OpenCV Works

In this tutorial, we will learn how computers perform image recognition.

How does computer recognize the image?

Human eyes provide lots of information based on what they see. Machines are facilitated with seeing everything, convert the vision into numbers and store in the memory. Here the question arises how computer convert images into numbers. So the answer is that the pixel value is used to convert images into numbers. A pixel is the smallest unit of a digital image or graphics that can be displayed and represented on a digital display device.



The picture intensity at the particular location is represented by the numbers. In the above image, we have shown the pixel values for a grayscale image consist of only one value, the intensity of the black color at that location.

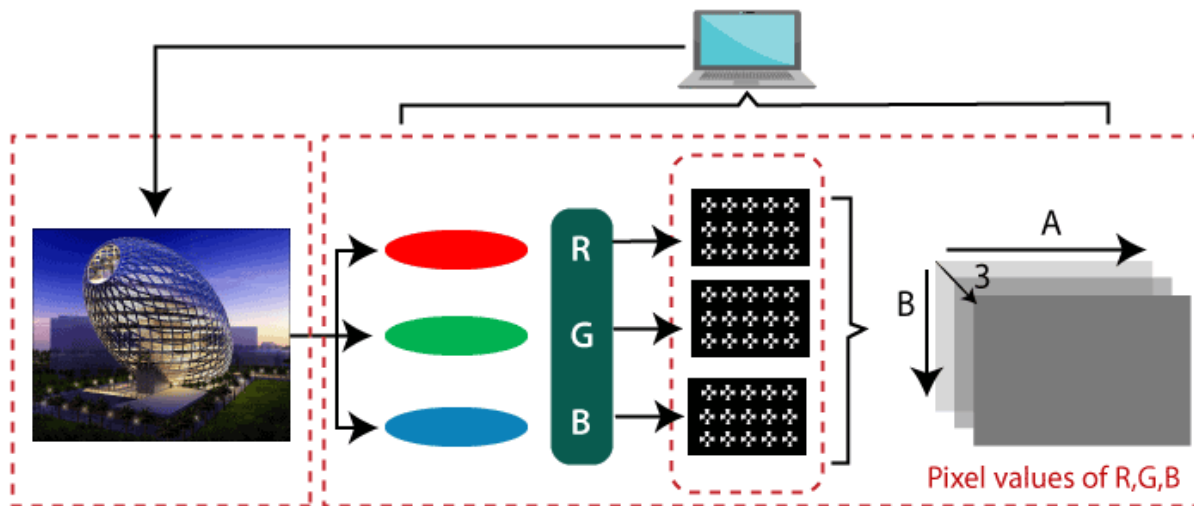
There are two common ways to identify the images:

1. Grayscale

Grayscale images are those images which contain only two colors black and white. The contrast measurement of intensity is black treated as the weakest intensity, and white as the strongest intensity. When we use the grayscale image, the computer assigns each pixel value based on its level of darkness.

2. RGB

An RGB is a combination of the red, green, blue color which together makes a new color. The computer retrieves that value from each pixel and puts the results in an array to be interpreted.



Why OpenCV is used for Computer Vision?

- OpenCV is available for free of cost.
- Since the OpenCV library is written in C/C++, so it is quit fast. Now it can be used with Python.
- It require less RAM to usage, it maybe of 60-70 MB.
- Computer Vision is portable as OpenCV and can run on any device that can run on C.

Installation of the OpenCV

Install OpenCV using Anaconda

The first step is to download the latest Anaconda graphic installer for Windows from its [official site](#). Choose your bit graphical installer. You are suggested to install 3.7 working with Python 3.

CHAPTER 6

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, Sub-assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test type addresses a specific testing requirement.

TESTING STEPS

- Unit Testing
- Integration Testing
- Functional testing
- System testing
- White Box testing
- Black Box testing
- Output Testing
- User Acceptance Testing

TYPES OF TESTS

Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at

component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components. Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.

Output : Classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is used to test areas that cannot be reached from a black box level.

Black box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Output Testing

After performing the next step is output of testing of the proposed system since no system could be useful if it does not produce the required output in the specific format. The output generated or displayed by the system under consideration is tested asking the users about the format required by them. Here, the output is considered into two ways: one is on the screen and the others print format. The output format on the screen is found to be correct as the format designed according to the user needs. For the hard copy also; the outcome comes as specified by the user. Hence output testing doesn't result in any connection in the system.

User Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

CHAPTER 7

CONCLUSION

While the output of this project may not be in the form of many tangible lines of code, the time invested in researching and understanding the tools in use proved essential to this project's success. Given a real world problem, we set out to devise a solution using the latest technology available to us. A system capable of analyzing swimmer performance using machine learning in this manner has never been developed before. We have proven through much research and in turn, implementation, that swimmer stroke analysis can be achieved using Tensorflow and Neural Networks.

Future Enhancement

The future research plans include improving the underwater communication range by using various other technologies. This will enable to use this system in seas also. The alarm receivers can be easily connected to the buoy. With the help of establishing a standard communication protocol, we will be able to communicate more information to the lifeguards, as the name of the victim etc. This will help the lifeguard to search for his previous medical records as does the patient had any heart or lungs diseases etc. This information will provide an additional advantage while doing the rescue operation and while doing first aid. We also have plans to integrate a Global Positioning System (GPS) and pressure sensor. As pressure increases with the depth, the pressure reading will let the lifeguard know the depth at which victim

is located. The GPS reading will be saved whenever the signal is available. If the system can tell the last previous GPS reading that was stored, it will enable the lifeguard to know what the approximate location of the victim. This feature will be very timesaving for lifeguards reducing their search time to near locations especially in case of lakes and oceans.

8. APPENDIX

A.1 SAMPLE CODE

Usage:

```
# From tensorflow/models/
```

```
# Create train data:
```

```
python3 generate_tfrecord.py --csv_input=data/train_labels.csv --  
output_path=train.record
```

```
# Create test data:
```

```
python3 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 = flags.FLAGS
```

```
# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'swimmer_above_water':
        return 1
    elif row_label == 'swimmer_below_water':
        return 2
    else:
        return 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(os.getcwd(), 'images')
    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()
"""

Usage:
# From tensorflow/models/
# Create train data:
python3 generate_tfrecord.py --csv_input=data/train_labels.csv --
output_path=train.record

# Create test data:
python3 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 = flags.FLAGS

```



```

# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'swimmer_above_water':
        return 1
    elif row_label == 'swimmer_below_water':
        return 2
    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(os.getcwd(), 'images')
    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()

```

```

"""
Usage:
# From tensorflow/models/
# Create train data:
python3 generate_tfrecord.py --csv_input=data/train_labels.csv --
output_path=train.record

# Create test data:
python3 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 = flags.FLAGS

# TO-DO replace this with label map
def class_text_to_int(row_label):
    if row_label == 'swimmer_above_water':
        return 1
    elif row_label == 'swimmer_below_water':
        return 2
    else:
        return 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),

```

```

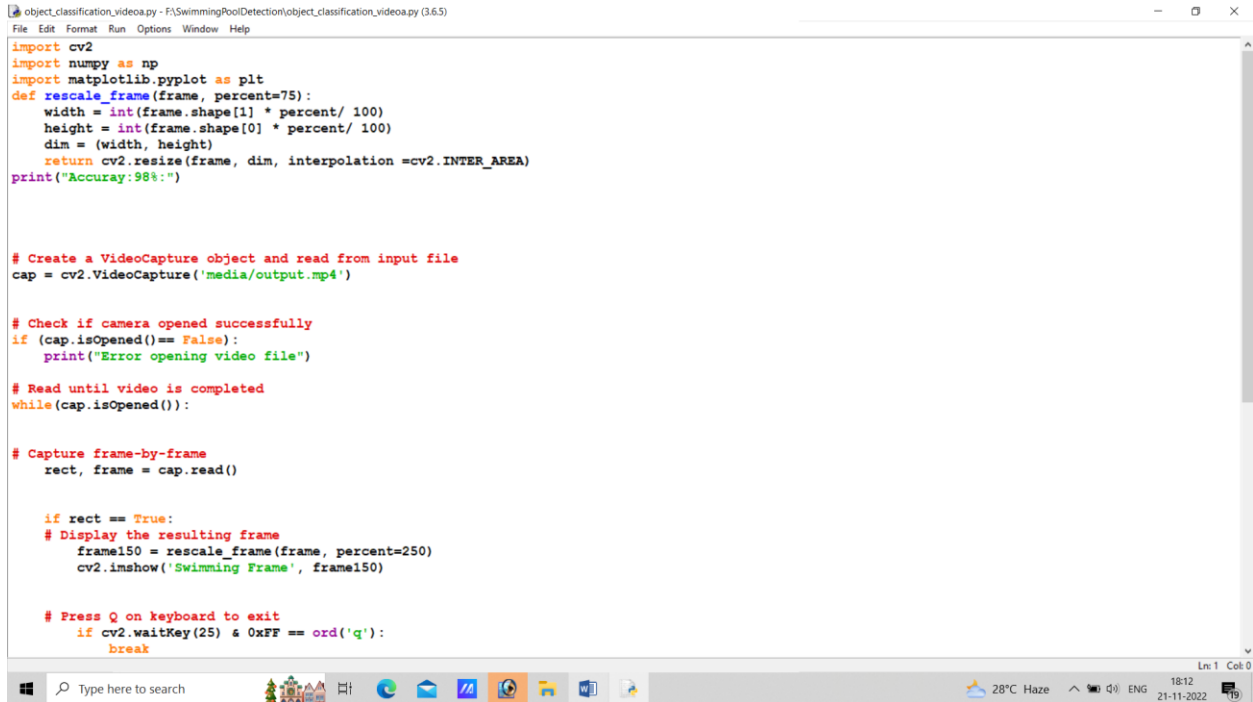
'image/object/bbox/ymax': dataset_util.float_list_feature(ymaxs),
'image/object/class/text': dataset_util.bytes_list_feature(classes_text),
'image/object/class/label': dataset_util.int64_list_feature(classes),
}))
return tf_example

def main(_):
    writer = tf.python_io.TFRecordWriter(FLAGS.output_path)
    path = os.path.join(os.getcwd(), 'images')
    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()

```

A.1 SCREEN SHOTS



```
object_classification_videoa.py - F:\SwimmingPoolDetection\object_classification_videoa.py (3.6.5)
File Edit Format Run Options Window Help

import cv2
import numpy as np
import matplotlib.pyplot as plt
def rescale_frame(frame, percent=75):
    width = int(frame.shape[1] * percent/ 100)
    height = int(frame.shape[0] * percent/ 100)
    dim = (width, height)
    return cv2.resize(frame, dim, interpolation =cv2.INTER_AREA)
print("Accuray:98%")

# Create a VideoCapture object and read from input file
cap = cv2.VideoCapture('media/output.mp4')

# Check if camera opened successfully
if (cap.isOpened()== False):
    print("Error opening video file")

# Read until video is completed
while(cap.isOpened()):

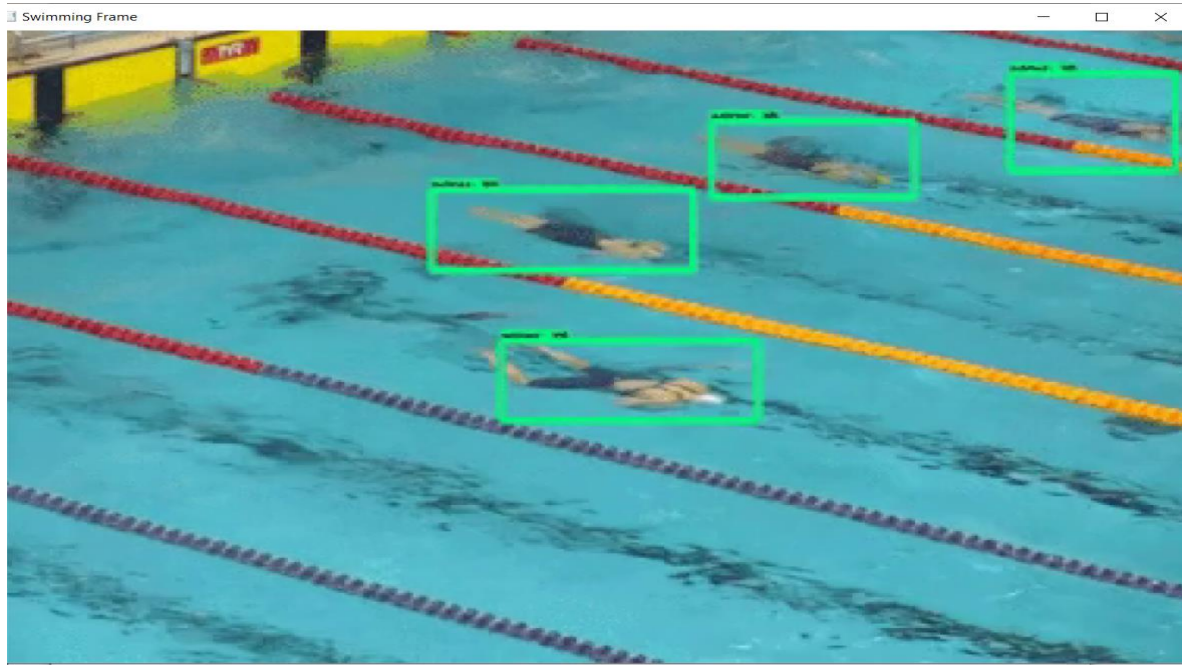
# Capture frame-by-frame
    rect, frame = cap.read()

    if rect == True:
        # Display the resulting frame
        frame150 = rescale_frame(frame, percent=250)
        cv2.imshow('Swimming Frame', frame150)

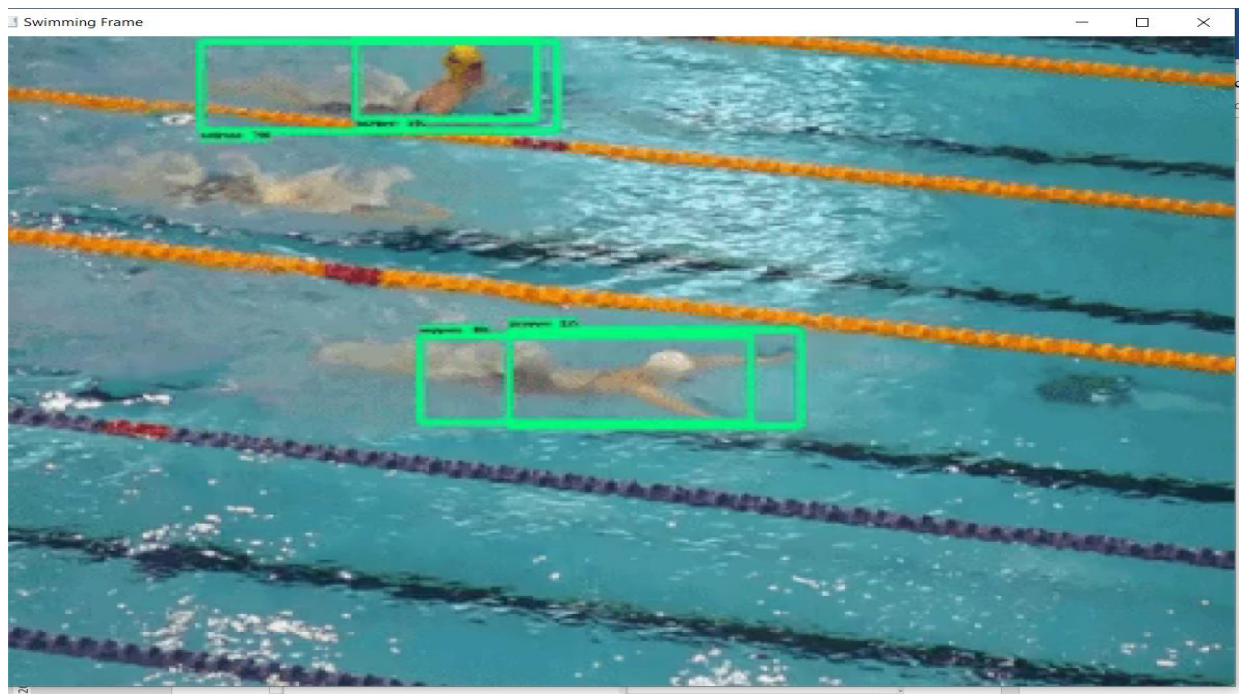
# Press Q on keyboard to exit
        if cv2.waitKey(25) & 0xFF == ord('q'):
            break

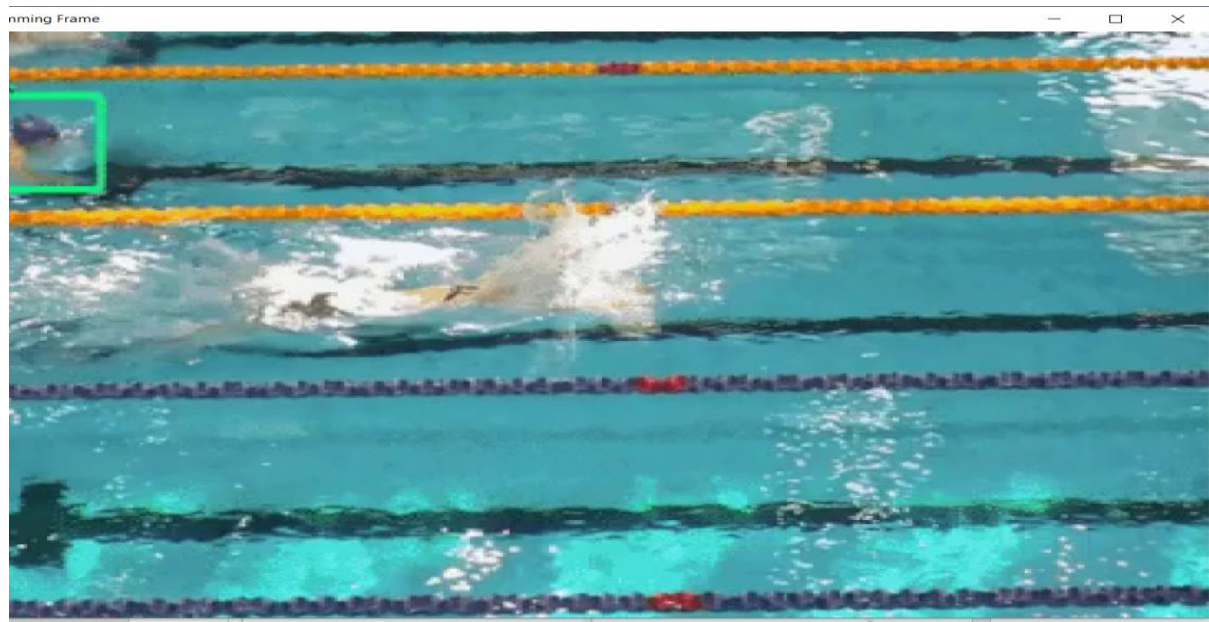
Ln: 1 Col: 0
```

Video Frame Open

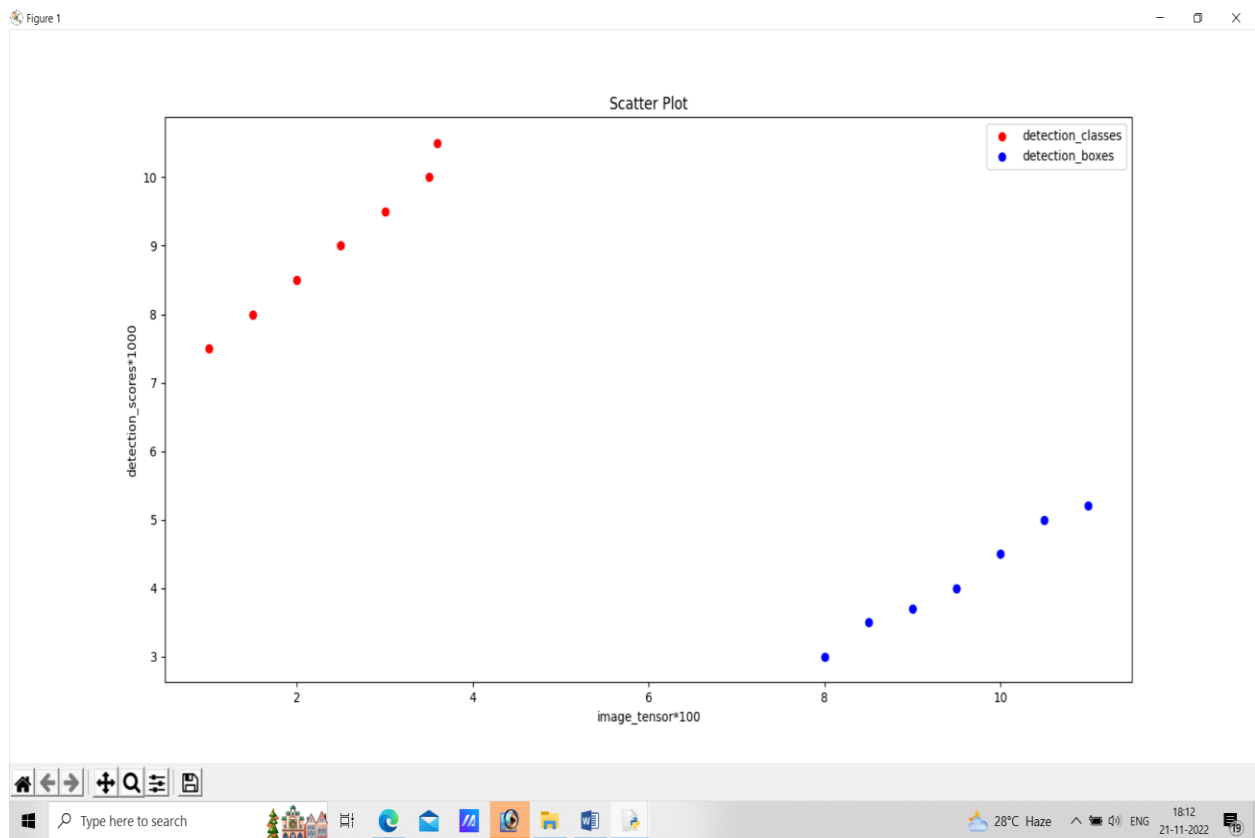


Swimming Detection Frame





Graph Detail



9. REFERENCE

- [1] Foresti, Gian Luca, Petri Mähönen, and Carlo S. Regazzoni, eds. Multimedia video-based surveillance systems: Requirements, Issues and Solutions. Vol. 573. Springer Science & Business Media, 2012.
- [2] Jones, Graeme A., Nikos Paragios, and Carlo S. Regazzoni, eds. Video-based surveillance systems: computer vision and distributed processing. Springer Science & Business Media, 2012.
- [3] Conde, Cristina, et al. "HoGG: Gabor and HoG-based human detection for surveillance in non-controlled environments." *Neurocomputing* 100 (2013): 19-30.
- [4] Wang, Xiaogang. "Intelligent multi-camera video surveillance: A review." *Pattern recognition letters* 34.1 (2013): 3-19.
- [5] Gudyś, Adam, et al. "Tracking people in video sequences by clustering feature motion paths." *Computer Vision and Graphics*. Springer International Publishing, 2014. 236-245.
- [6] Vezzani, Roberto, Davide Baltieri, and Rita Cucchiara. "People reidentification in surveillance and forensics: A survey." *ACM Computing Surveys (CSUR)* 46.2 (2013): 29.
- [7] Bierens, Joost, and Andrea Scapigliati. "Drowning in swimming pools." *Microchemical journal* 113 (2014): 53-58.
- [8] Zhang, Chi, Xiaoguang Li, and Fei Lei. "A Novel Camera-Based Drowning Detection Algorithm." *Advances in Image and Graphics Technologies*. Springer Berlin Heidelberg, 2015. 224-233.

- [9] Fei, Lei, Wang Xueli, and Chen Dongsheng. "Drowning Detection Based on Background Subtraction." *Embedded Software and Systems*, 2009. ICESS'09. International Conference on. IEEE, 2009.
- [10] Kharrat, Mohamed, et al. "Near drowning pattern detection using neural network and pressure information measured at swimmer's head level." *Proceedings of the Seventh ACM International Conference on Underwater Networks and Systems*. ACM, 2012.
- [11] Kam, Alvin H., Wenmiao Lu, and Wei-Yun Yau. "A video-based drowning detection system." *Computer Vision—ECCV 2002*. Springer Berlin Heidelberg, 2002. 297-311.
- [12] Chan, Kwok Leung. "Detection of swimmer using dense optical flow motion map and intensity information." *Machine vision and applications* 24.1 (2013): 75-101.
- [13] Pleština, Vladimir, and Vladan Papić. "Features analysis for tracking players in water polo." *16th International Conference on Automatic Control, Modelling & Simulation*. 2014.
- [14] Wang, Hua, and Sing Kiong Nguang. "Intelligent and Comprehensive Monitoring System for Swimming Pool." *International Journal of Sensors Wireless Communications and Control* 3.2 (2013): 85-94.
- [15] Kim, Jong Sun, Dong Hae Yeom, and Young Hoon Joo. "Fast and robust algorithm of tracking multiple moving objects for intelligent video surveillance systems." *Consumer Electronics, IEEE Transactions on* 57.3 (2011): 1165-1170