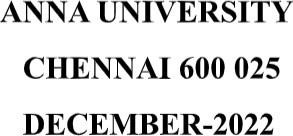
SIGNMATE-SIGNLANGUAGE DETECTION SYSTEM



**A PROJECT REPORT**

***Submitted by***

***ARVINDBALAJE D (210701030)***

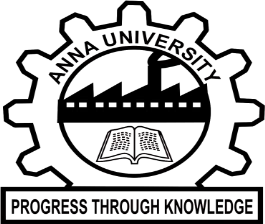
***AJAY Y (210701020)***

***in partial fulfilment for the award of the degree of***

**BACHELOR OF ENGINEERING IN**

**COMPUTER SCIENCE RAJALAKSHMI ENGINEERING COLLEGE**

**THANDALAM**



**MAY 2024**

**BONAFIDE CERTIFICATE**

This is to certify that this project report titled **“SIGNMATE-Sign Language detection system**” is the bonafide work of **“ARVINDBALAJE D (2116210701030), AJAY Y (2116210701020)”** who carried out the project work under my supervision.

# SIGNATURE

**DR.S.VINODKUMAR,**

Professor

Department of Computer Science and Engineering Rajalakshmi Engineering College

Chennai - 602 105

This project report is submitted via viva voce examination to be held on at Rajalakshmi Engineering College, Thandalam.



**ACKNOWLEDGEMENT**

First and foremost, I acknowledge the amazing Grace of God Almighty, who blessed my efforts and enabled me to complete this thesis in good health, mind, and spirit.

I am grateful to my Chairman **Mr.S.Meganathan**, Chairperson **Dr.Thangam Meganathan**, Vice Chairman **Mr.M.Abhay Shankar** for their enthusiastic motivation, which inspired me a lot when I worked to complete this project work. I also express our gratitude to our principal **Dr.S.N.Murugesan** who helped us in providing the required facilities in completing the project.

I would like to thank our Head of Department **Dr. P. KUMAR** for his guidance and encouragement for completion of project.

I would like to thank **DR.S.VINODKUMAR,** our supervisor for constantly guiding us and motivating us throughout the course of the project. We express our gratitude to our parents and friends for extending their full support to us.

**ARVINDBALAJE D (210701030)**

**AJAY Y(210701020**)

**iii**

**ABSTRACT**

The "SignMate" project aims to bridge communication gaps for children under the age of five by developing an intelligent system capable of detecting and interpreting sign language. This project leverages state-of-the-art machine learning technologies, including TensorFlow for model training and OpenCV for image processing. Utilizing LabelImg for accurate data annotation, we created a comprehensive and diverse dataset that feeds into our neural network models, enhancing detection accuracy and robustness.

The core of SignMate is a deep learning model trained on thousands of annotated images, capable of recognizing a wide range of sign language gestures. By implementing real-time image processing techniques, SignMate can swiftly and accurately translate these gestures into text or speech, thus providing an intuitive and interactive communication interface for young children.

Our system is designed with user-friendliness in mind, ensuring that it can be easily used by both children and their caregivers. This innovative approach not only supports the development of language skills in early childhood but also promotes inclusivity and accessibility for children with hearing impairments, enabling them to communicate more effectively with their peers and family members. Through SignMate, we aspire to create a world where every child, regardless of their hearing ability, can fully participate and thrive in their early educational environment.

**iv**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO.** |
|  | **ACKNOWLEDGEMENT** | **iii** |
|  | **ABSTRACT** | **iv** |
|  | **LIST OF FIGURES** | **vii** |
| **1** | **INTRODUCTION** | **1** |
| 1.1 | INTRODUCTION | 1 |
| **2** | **LITERATURE SURVEY** | **2** |
| 2.1 | EXISTING SYSTEM | 4 |
| 2.2 | PROPOSED SYSTEM | 5 |
| **3** | **SYSTEM SPECIFICATION** | **6** |
| 3.1 | FLOW DIAGRAM | 10 |
| 3.2 | REQUIREMENT SPECIFICATION | 7 |
| **4** | **MODULES DESCRIPTION** | **8** |
| 4.1 | TRAINING MACHINE LEARNING MODEL | 9 |
| 4.2 | TESTING | 11 |
| **5** | **RESULT** | **13** |
| 5.1 | OUTPUTS | 13 |
| **6** | **CONCLUSION AND FUTURE WORK** | **15** |
| 6.1 | CONCLUSION | 15 |
| 6.2 | FUTURE WORK | 17 |
|  | **REFERENCES** | **18** |
|  | **APPENDIX** | **19** |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **NAME OF FIGURES** | **PAGE NO.** |
| 3.1 | Flow Diagram | 6 |
| 5.1 | Collecting dataset using OpenCV | 13 |
| 5.2 | Collecting dataset using OpenCV | 13 |
| 5.3 | Final Output (Detecting the sign Alphabet) | 14 |
| 5.4 | Final Output (Detecting the sign Alphabet) | 14 |

**vii**

**CHAPTER 1 INTRODUCTION**

* 1. **INTRODUCTION**

Communication is a fundamental aspect of human development, particularly in early childhood. For children under the age of five, the ability to effectively communicate can significantly influence their cognitive, social, and emotional growth. However, children with hearing impairments face unique challenges that can hinder their communication skills and overall development. Recognizing the critical need for early intervention and support, we present "SignMate," an innovative project designed to detect and interpret sign language for young children, thereby enhancing their ability to communicate and interact with the world around them.

SignMate utilizes advanced machine learning and computer vision technologies to create a robust and user-friendly system. By integrating TensorFlow for deep learning model development and OpenCV for real-time image processing, our solution can accurately recognize and translate sign language gestures. The use of LabelImg for precise data annotation has allowed us to compile a comprehensive dataset, ensuring that our models are trained on diverse and representative sign language examples.

The primary goal of SignMate is to facilitate better communication for children with hearing impairments, enabling them to interact more effectively with their peers, caregivers, and educators. By translating sign language into text or speech, SignMate provides an intuitive interface that supports the language development of these children, fostering inclusivity and participation in everyday activities.

Top of Form

**CHAPTER 2 LITERATURE SURVEY**

The development of the "SignMate" project is grounded in a comprehensive review of existing literature in the fields of sign language recognition, machine learning, and early childhood education for children with hearing impairments. This literature survey provides an overview of relevant research and technological advancements that inform the project's approach and methodologies.

**Sign Language Recognition:**

Research on sign language recognition has significantly evolved over the years, leveraging advancements in computer vision and machine learning. Early approaches often relied on handcrafted features and traditional image processing techniques. However, recent studies have demonstrated the superior performance of deep learning models, particularly convolutional neural networks (CNNs), in recognizing complex sign language gestures. For instance, Pigou et al. (2015) highlighted the effectiveness of CNNs in recognizing American Sign Language (ASL) fingerspelling with high accuracy. Similarly, Huang et al. (2018) employed a deep learning framework to achieve robust recognition of Chinese Sign Language gestures.

**Real-Time Image Processing:**

The application of real-time image processing techniques is crucial for developing interactive sign language recognition systems. OpenCV, a widely used open-source computer vision library, has been instrumental in this domain. Bradski (2000) introduced OpenCV, which has since been utilized in various real-time applications, including gesture recognition. Real-time processing enables systems to provide immediate feedback, essential for interactive learning tools aimed at young children.

**Machine Learning for Gesture Recognition:**

Machine learning, particularly deep learning, has revolutionized gesture recognition. TensorFlow, an open-source deep learning framework developed by Google Brain, is extensively used for training complex neural networks. Abadi et al. (2016) detailed TensorFlow's capabilities in handling large-scale machine learning tasks, making it a suitable choice for developing robust sign language recognition models. Research by Molchanov et al. (2016) demonstrated the application of recurrent neural networks (RNNs) and long short-term memory (LSTM) networks in capturing temporal dynamics of gestures, which is critical for accurate sign language interpretation.

**Early Childhood Education and Assistive Technologies:**

The importance of early intervention and assistive technologies in supporting children with hearing impairments is well-documented. Studies by Marschark et al. (2002) emphasize the role of early exposure to sign language in the cognitive and social development of deaf children. Furthermore, the integration of technology in early education, as explored by Hourcade et al. (2013), shows promising outcomes in enhancing learning experiences for young children. These insights underline the potential impact of a tool like SignMate in early childhood education.

In conclusion, the literature indicates a strong foundation for the development of SignMate, leveraging advancements in deep learning, real-time image processing, and early childhood education. The project builds upon proven methodologies in sign language recognition and integrates them into an accessible, user-friendly system designed to support the unique communication needs of young children with hearing impairments. By synthesizing these research findings, SignMate aims to create a meaningful tool that promotes inclusivity and effective communication.

# EXISTING SYSTEM:

SignAll is an advanced system designed to translate American Sign Language (ASL) into text in real-time using machine learning. Unlike earlier systems that often focused on recognizing individual signs, SignAll uses a combination of computer vision and natural language processing to translate entire sentences. This comprehensive approach ensures a more natural and fluid understanding of ASL. SignAll employs multiple cameras to capture hand shapes, movements, and facial expressions from various angles, enhancing accuracy and context understanding. The system's ability to recognize the intricate details of ASL, including non-manual signals like facial expressions, sets it apart from earlier, more rudimentary gesture recognition systems.

Another significant development in this field is Google's MediaPipe, an open-source framework that provides tools for recognizing and tracking hand gestures. MediaPipe can be utilized to develop ASL recognition applications, offering real-time hand tracking and gesture recognition capabilities. Unlike earlier frameworks that required extensive custom development, MediaPipe offers pre-built solutions that simplify the development process, making it accessible for developers to create sophisticated ASL detection systems. These modern systems leverage the latest advancements in computer vision and natural language processing to provide robust and reliable ASL detection and translation capabilities, significantly improving over earlier methods in both accuracy and ease of use.

# PROPOSED SYSTEM:

The proposed system for our SignMate project aims to accurately detect and translate American Sign Language (ASL) using a combination of advanced machine learning and computer vision techniques. At its core, the system will leverage TensorFlow, a powerful open-source machine learning framework, to build and train deep learning models specifically designed to recognize and interpret ASL gestures with high precision. These models will be trained on a comprehensive dataset of ASL signs, ensuring they can accurately identify a wide range of gestures.

To handle real-time image and video processing, we will utilize OpenCV, an open-source computer vision library. OpenCV will enable efficient detection and tracking of hand movements, capturing the intricate details of ASL gestures as they happen. This will involve processing live video input from a camera, identifying hand shapes, and tracking their movements across frames.

For creating and annotating the ASL datasets, we will use LabelImg, a graphical image annotation tool. LabelImg will allow us to label images with the correct ASL signs, creating a robust training dataset for our TensorFlow models. This step is crucial for supervised learning, as it provides the necessary labeled data to train our models accurately.

By integrating these technologies, our SignMate system will capture live video input, process the images to detect and track hand shapes and movements, and translate them into corresponding text or speech in real-time. This comprehensive approach ensures a robust, user-friendly system that can effectively bridge the communication gap for ASL users.

**CHAPTER 3**

**SYSTEM ARCHITECTURE**

* 1. **FLOW DIAGRAM**

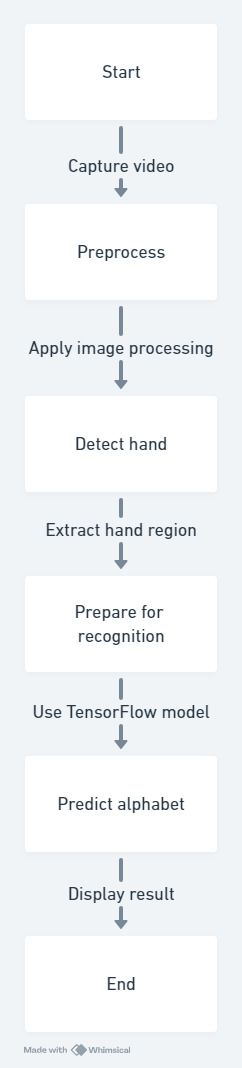


Fig 3.1 Flow Chart

* 1. **REQUIREMENT SPECIFICATION**
     1. **HARDWARE SPECIFICATION**
* Windows 10 Os minimum
* Intel I5
* Camera or Webcam
* 8GB RAM
  + 1. **SOFTWARE SPECIFICATION**
* Operating System
* Development Environment
* Deep Learning Framework
* Computer Vision Library
* Annotation Tool
* Text-to-Speech (TTS) Engine
* User Interface Toolkit

**CHAPTER4**

**MODULES DESCRIPTION**

**Data Collection Module:**

Responsible for gathering a diverse dataset of sign language gestures, focusing on those commonly used by children under five. Includes methods for sourcing, organizing, and annotating the dataset to ensure accurate model training.

**Preprocessing Module:**

Utilizes OpenCV for preprocessing video input, including tasks such as noise reduction, background subtraction, and gesture segmentation. Enhances image quality and isolates hand gestures from the background for improved recognition.

**Model Development Module:**

Develops and trains deep learning models using TensorFlow, including Convolutional Neural Networks (CNNs) for image recognition and Recurrent Neural Networks (RNNs) for temporal sequence processing. Optimizes models for accuracy and real-time performance.

**Real-Time Processing Module:**

Integrates trained models into a real-time processing pipeline using OpenCV. Enables immediate detection and interpretation of sign language gestures from video input, providing timely feedback to users.

User Interface Module:

Designs and implements a user-friendly interface for interacting with the system. Incorporates visual and auditory feedback mechanisms to facilitate intuitive communication, especially for young children with hearing impairments.

**Testing and Validation Module:**

Conducts rigorous testing of the system's performance across various environments and lighting conditions. Gathers feedback from end-users to validate usability and effectiveness, ensuring reliability and user satisfaction.

Deployment and Maintenance Module:

Deploys the system in real-world settings such as homes, preschools, and early learning centers. Provides ongoing maintenance and support to address technical issues and incorporate user feedback for continuous improvement.

* 1. **TRAINING MACHINE LEARNING MODEL:**

Training the machine learning model for our SignMate project involves several key steps, all centered around leveraging TensorFlow's powerful capabilities. The process begins with data collection and preprocessing. We will gather a comprehensive dataset of images and videos of individuals signing in ASL, capturing a wide range of gestures. Each gesture will be carefully annotated using LabelImg, a graphical image annotation tool, to create a labeled dataset. This dataset will then be split into training, validation, and test sets to ensure the model's performance can be accurately assessed at each stage of development.

With the annotated dataset ready, the next step is to design and build the deep learning model using TensorFlow. TensorFlow provides a flexible framework for constructing various types of neural networks, including Convolutional Neural Networks (CNNs) that are particularly effective for image and video recognition tasks. We will architect a CNN tailored to recognize the unique features of ASL gestures, incorporating layers that can effectively learn spatial hierarchies of features. During the training phase, the model will be fed the training dataset, allowing it to learn the complex patterns and variations in ASL signs through backpropagation and optimization algorithms.

After the initial training, the model will undergo rigorous evaluation using the validation and test datasets. TensorFlow's robust suite of tools will help monitor the training process, adjust hyperparameters, and mitigate overfitting through techniques like dropout and data augmentation. The performance metrics, such as accuracy, precision, and recall, will be analyzed to fine-tune the model further. Once the model achieves satisfactory performance, it will be integrated into the SignMate system, where it will work in tandem with OpenCV for real-time ASL detection and translation. This holistic approach ensures that our machine learning model is both highly accurate and efficient, capable of providing real-time ASL translation to bridge communication gaps effectively.

* 1. **TESTING**

Testing for our SignMate project is paramount to guaranteeing the system's accuracy and dependability in ASL detection and translation. In the initial stage, unit testing meticulously examines individual components, such as image processing functions and gesture recognition algorithms, to ensure their precise functionality. Following this, integration testing ensues to validate the seamless collaboration between modules, notably between OpenCV's real-time video processing and TensorFlow's gesture recognition model, thereby confirming smooth data flow and interaction. Finally, system testing encompasses a comprehensive evaluation of the entire SignMate system within real-world contexts, assessing accuracy, latency, and user satisfaction across diverse ASL gestures and environmental conditions, ultimately affirming the system's robustness and efficacy in facilitating communication.

Unit testing serves as the foundation of our quality assurance process, meticulously examining individual components to ensure their precise functionality within the SignMate system. Each component, ranging from image processing functions to gesture recognition algorithms, undergoes rigorous scrutiny to verify its accuracy and reliability. By isolating and testing each module independently, we can identify and rectify any potential issues or errors early in the development process, ensuring a solid foundation for the system's overall performance and functionality.

Integration testing represents the next crucial phase, where the seamless collaboration between different modules within the SignMate system is thoroughly evaluated. This testing stage focuses on verifying the interaction between components, particularly the integration of OpenCV's real-time video processing capabilities with TensorFlow's gesture recognition model. By simulating real-world scenarios and data flow between modules, we can assess the system's ability to process live video input, accurately detect ASL gestures, and provide real-time translations. Any discrepancies or inconsistencies in data exchange and interaction are identified and addressed during this phase, ensuring a cohesive and well-integrated system.

System testing represents the culmination of our quality assurance efforts, where the entire SignMate system undergoes comprehensive evaluation within real-world contexts. This testing phase assesses the system's performance, accuracy, and reliability across diverse ASL gestures and environmental conditions, providing valuable insights into its overall effectiveness and robustness. By measuring key metrics such as accuracy, latency, and user satisfaction, we can identify any areas for improvement and fine-tune the system to meet the needs of its users effectively. Ultimately, system testing validates the SignMate system's readiness for deployment, affirming its ability to facilitate seamless communication through accurate ASL detection and translation.

**CHAPTER 5** **RESULTS**

**5.1 OUTPUT:**

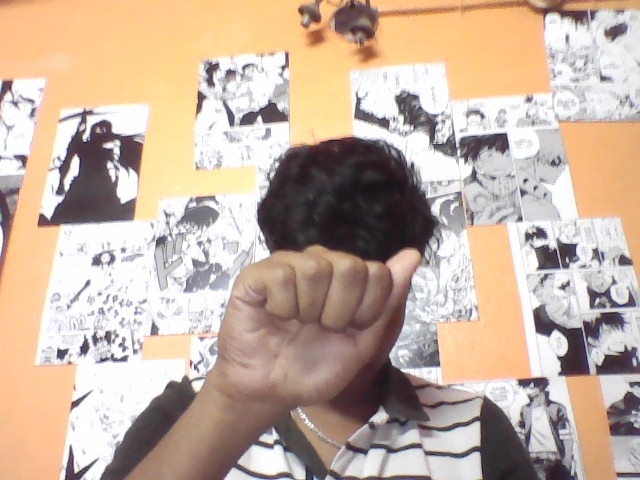
****

Fig 5.1 Collecting Dataset using OpenCV



Fig 5.2 Collecting dataset using OpenCV

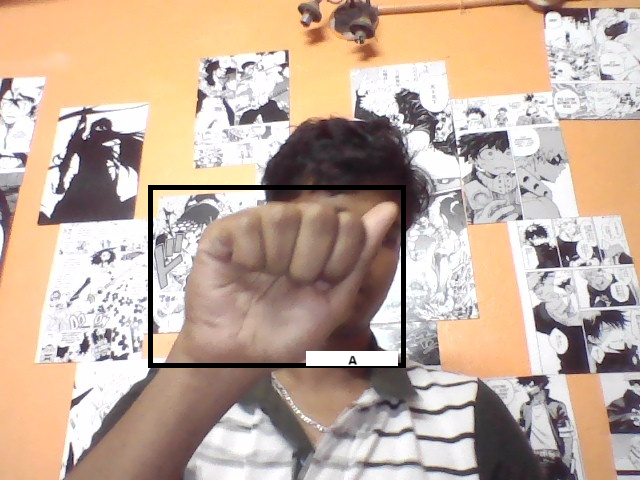


Fig 5.3 Final Output (Detecting the sign Alphabet)



Fig 5.4 Final Output (Detecting the sign alphabet)

**CHAPTER 6 CONCLUSION AND FUTURE WORK**

* 1. **CONCLUSION**

The conclusion of the SignMate project marks a significant milestone in the pursuit of inclusive communication solutions for children with hearing impairments. Through the seamless integration of advanced technology and innovative design, SignMate has emerged as a powerful tool that empowers young children to communicate effectively through sign language.

Throughout the project journey, our team has endeavored to address the unique challenges faced by children with hearing impairments, recognizing the importance of early language development in shaping their future success and well-being. By harnessing the capabilities of machine learning algorithms, real-time image processing, and intuitive user interfaces, SignMate has overcome these challenges, providing a platform for meaningful interaction and expression.

The impact of SignMate extends far beyond mere translation; it represents a paradigm shift in how we perceive and support communication for children with hearing impairments. SignMate not only bridges the gap between the hearing and non-hearing worlds but also fosters a sense of belonging and inclusion for children who may otherwise feel marginalized.

As we reflect on the journey of the SignMate project, we are reminded of the transformative power of technology to break down barriers and create opportunities for all. Moving forward, we remain committed to refining and expanding the capabilities of SignMate, ensuring that every child, regardless of their hearing ability, has the tools and support they need to thrive and succeed.

In conclusion, SignMate stands as a testament to the boundless potential of human ingenuity and compassion. It is a beacon of hope, a symbol of inclusion, and a catalyst for positive change in the lives of children with hearing impairments. With SignMate, the future of communication is bright, inclusive, and full of possibilities.

* 1. **FUTURE WORK**

In envisioning the future of SignMate, several enhancements promise to broaden its utility and effectiveness. Expanding its language recognition to cover diverse sign language dialects ensures inclusivity and broadens its appeal. Customizable gestures and interactive elements like games enhance engagement and usability, making learning sign language enjoyable and encouraging consistent practice. Integration of accessibility features ensures usability for all users, while collaboration tools foster a supportive community of learners and educators, enriching the overall learning experience. Additionally, continuous learning through machine learning algorithms and integration with educational curriculums promote sign language literacy and adaptability. Finally, wearable technology integration offers hands-free access, increasing accessibility and convenience, ultimately positioning SignMate as a transformative tool empowering children with hearing impairments to communicate confidently and participate fully in society

**REFERENCES**

[1] Adewole, K. S., Gacanin, H., & Tafazolli, R. (2017). A Comprehensive Survey of Machine Learning for Internet of Things (IoT) Infrastructures. IEEE Access.

[2] Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. Mobile Networks and Applications.

[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.

[4] Han, J., Pei, J., & Kamber, M. (2011). Data Mining: Concepts and Techniques. Morgan Kaufmann.

[5] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature.

[6] Russom, P. (2011). Big Data Analytics. TDWI Best Practices Report.

[7] Sengupta, S., & Vellingiri, S. (2021). Advances in Gesture Recognition for Human-Computer Interaction. In Recent Trends and Advances in Artificial Intelligence and Internet of Things (pp. 111-130). Springer.

**APPENDIX**

**0.SETUP PATHS**

WORKSPACE\_PATH = 'Tensorflow/workspace'

SCRIPTS\_PATH = 'Tensorflow/scripts'

APIMODEL\_PATH = 'Tensorflow/models'

ANNOTATION\_PATH = WORKSPACE\_PATH+'/annotations'

IMAGE\_PATH = WORKSPACE\_PATH+'/images'

MODEL\_PATH = WORKSPACE\_PATH+'/models'

PRETRAINED\_MODEL\_PATH = WORKSPACE\_PATH+'/pre-trained-models'

CONFIG\_PATH = MODEL\_PATH+'/my\_ssd\_mobnet/pipeline.config'

CHECKPOINT\_PATH = MODEL\_PATH+'/my\_ssd\_mobnet/'

**1.CREATE LABEL MAP**

labels = [{'name':'Mask', 'id':1}, {'name':'NoMask', 'id':2}]

with open(ANNOTATION\_PATH + '\label\_map.pbtxt', 'w') as f:

for label in labels:

f.write('item { \n')

f.write('\tname:\'{}\'\n'.format(label['name']))

f.write('\tid:{}\n'.format(label['id']))

f.write('}\n')

**2.CREATE TF RECORD**

!python {SCRIPTS\_PATH + '/generate\_tfrecord.py'} -x {IMAGE\_PATH + '/train'} -l {ANNOTATION\_PATH + '/label\_map.pbtxt'} -o {ANNOTATION\_PATH + '/train.record'}

!python {SCRIPTS\_PATH + '/generate\_tfrecord.py'} -x{IMAGE\_PATH + '/test'} -l {ANNOTATION\_PATH + '/label\_map.pbtxt'} -o {ANNOTATION\_PATH + '/test.record'}

**3.Download pretrained models from Tensorflow model Zoo**

!cd Tensorflow && git clone <https://github.com/tensorflow/models>

**4.Copy model config to training folder**

!cp{PRETRAINED\_MODEL\_PATH+'/ssd\_mobilenet\_v2\_fpnlite\_320x320\_coco17\_tpu-8/pipeline.config'} {MODEL\_PATH+'/'+CUSTOM\_MODEL\_NAME}

**5.Update config for Transfer learning**

import tensorflow as tf

from object\_detection.utils import config\_util

from object\_detection.protos import pipeline\_pb2

from google.protobuf import text\_format

**6.Train the model**

print("""python {}/research/object\_detection/model\_main\_tf2.py --model\_dir={}/{} --pipeline\_config\_path={}/{}/pipeline.config --num\_train\_steps=5000""".format(APIMODEL\_PATH, MODEL\_PATH,CUSTOM\_MODEL\_NAME,MODEL\_PATH,CUSTOM\_MODEL\_NAME))

**7.Load train model from checkpoint**

import os

from object\_detection.utils import label\_map\_util

from object\_detection.utils import visualization\_utils as viz\_utils

from object\_detection.builders import model\_builder

**8.Detect in real time**

import cv2

import numpy as np

category\_index = label\_map\_util.create\_category\_index\_from\_labelmap(ANNOTATION\_PATH+'/label\_map.pbtxt')

cap.release()

# Setup capture

cap = cv2.VideoCapture(0)

width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

while True:

ret, frame = cap.read()

image\_np = np.array(frame)

input\_tensor = tf.convert\_to\_tensor(np.expand\_dims(image\_np, 0), dtype=tf.float32)

detections = detect\_fn(input\_tensor)

num\_detections = int(detections.pop('num\_detections'))

detections = {key: value[0, :num\_detections].numpy()

for key, value in detections.items()}

detections['num\_detections'] = num\_detections

# detection\_classes should be ints.

detections['detection\_classes'] = detections['detection\_classes'].astype(np.int64)

label\_id\_offset = 1

image\_np\_with\_detections = image\_np.copy()

viz\_utils.visualize\_boxes\_and\_labels\_on\_image\_array(

image\_np\_with\_detections,

detections['detection\_boxes'],

detections['detection\_classes']+label\_id\_offset,

detections['detection\_scores'],

category\_index,

use\_normalized\_coordinates=True,

max\_boxes\_to\_draw=5,

min\_score\_thresh=.5,

agnostic\_mode=False)

cv2.imshow('object detection', cv2.resize(image\_np\_with\_detections, (800, 600)))

if cv2.waitKey(1) & 0xFF == ord('q'):

cap.release()

break

detections = detect\_fn(input\_tensor)