ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

OpenVINO ToolKit: A Tool for Video Analytics

Prof. Abhilasha Kulkarni[1], Nikita Modi[2], Saloni Patil[3], Sourabh Pawar[4], Nikhil Thatte[5]

Assistant Professor, Department of Computer Engineering, MMCOE, Karve Nagar, Maharashtra, India[1]

U.G. Student, Department of Computer Engineering, MMCOE, Karve Nagar, Maharashtra, India[2,3,4,5]

ABSTRACT: Video analytics is used to analyse video streams in near real-time and for identifying characteristics, actions or patterns of specific behaviour via analysis of monitored environments. It may include technical functions such as classification and detection of objects, face recognition, vehicle recognition and motion detection using trained models. The technology used to achieve video analytics may include sensors, cameras and connectivity techniques. Video analytics comes with many advantages to businesses, especially in security surveillance systems. This paper presents an overview of a Video Analytics tool OpenVINO developed by Intel®. OpenVINO combines concepts such as Computer Vision and Deep Learning to analyse the video content and to extract meaningful information.

KEYWORDS*:* Video Analytics, OpenVINO, Face Detection, Object Detection, Motion Detection.

I. INTRODUCTION

Challenging and changing times need smart security solutions. Video Analytics can help with risk mitigation and management and can be an effective way to reduce risk, enable efficiencies, ensure compliance, and keep employees and assets safe.Video Analytics is used to analyse video streams in near real-time and for identifying characteristics, actions or patterns of specific behaviour via analysis of monitored environments. This technical ability has a wide range of application in domains such as entertainment, health-care, retail, automotive, transport, safety, and security.
Video Analytics plays a very important part in face detection and recognition, object detection wherein objects can be identified based on the dataset provided to the learning models. Video Analytics can be made simpler and more effective with tools such as KWIVER, VIAME and OpenVINO. We have explored and analyzed the working of Intel Distribution OpenVINO Toolkit.OpenVINO stands for Open Visual Interface and Neural Network Optimization. OpenVINO allows developers the power to create cutting-edge AI-powered computer vision applications. Computer vision and Deep learning techniques can be integrated and used with Intel's®OpenVINO. This paper provides an overview of the Intel's® video analytics toolkit which isOpenVINO.

II. RELATED WORK

In paper [1] the authors have described the various phases of object detection and object tracking. They have described the various methods through which object tracking can be done like Kernel tracking, colour edges and texture. The research done in this paper will help in the design and development of robust object detection algorithms.The authors of this paper [2] have described various methods to carry out face detection.One method makes use of background subtraction in the haar face detector and the second is the Eigenface method. They concluded that the subtraction requires a lot of time which in turn affects the system performance as compared to the Eigenface method. Inpaper [3] the authors have elaborated a visual surveillance system to find a motion made by an object and generate an alert. Inthis paper the captured images are getting stored in a HDD so that the user has knowledge about the object responsible for the movement.The object detection algorithm proposed paper [4] showed remarkable results in the real world. The algorithm was able to track objects of various sizes, shapes and colour with great accuracy.In paper [5] the authors have

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

given an overview of an open source framework for the development of computer vision and machine learning systems. The paper describes the various feature of the tool, the releases, and their repositories.

III. SYSTEM OVERVIEW

A. System Requirements:

Compatible Operating System:

- Ubuntu 16.04.3 LTS (64 bit)
- Windows 10 (64 bit)
- CentOS 7.4 (64 bit)

Compatible Processors:

- 6th to 8th generation Intel® Core™ and Intel® Xeon® processors

A Linux Environment Requires:

- OpenCV 3.4 or higher
- GNU Compiler Collection (GCC) 3.4 or higher
- CMake 2.8 or higher
- Python 3.5 or higher

A Windowsbuild environment needs these components:

- Intel HD Graphics Driver (latest version)
- OpenCV 3.4 or higher
- IntelC++ Compiler 2017 Update 4
- CMake 2.8 or higher
- Python 3.5 or higher
- Visual Studio 2015

B. System Components

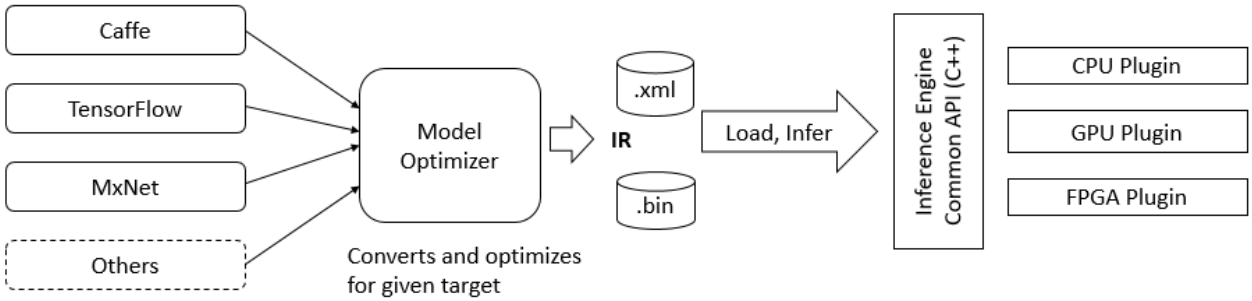

Figure 1: Procedure for Deploying a Trained Model

Figure 1 Description: The figure gives an overview of the OpenVINO toolkit components.It shows the flow of operations that take place when deploying a trained model.

a) Model Optimizer

Model Optimizer is a cross-platform command-line tool that allows the evolution between the training and deployment environment and carries out static model analysis and optimizes deep learning models for efficient execution on the target devices. Model Optimizer process presumes that you have a network model which is trained using one of the supported frameworks. The frameworks include TensorFlow, Caffe, etc. Model Optimizer generates an Intermediate Representation (IR) of the network as output.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

b) Inference Engine

After using the Model Optimizer to generate an Intermediate Representation, utilize the Inference Engine to deduce input data.The Inference Engine is a C++ library which contains a set of C++ classes that are used to infer input data (images) and obtain a result. The C++ library offers functionalities to read the Intermediate Representation, set the input and output formats, and execute the model on devices.The Inference Engine reads, loads, and deduces the Intermediate Representation. The Inference Engine API provides a unified API which is supported across all Intel® platforms.

Intermediate Representation is a collection of files that describe the entire model. It includes:

- .xml: Describes the network topology
- .bin: Contains the weights and biases binary data

C. Installation for Linux Operating System

1. Download the latest version of OpenVINO Toolkit from https://software.intel.com
2. Go to the downloads directory and unpack the .tgz file.
3. Your files will be unpacked to the following directory: l_openvino_toolkit_p_<version>
4. Go to the above directory. The required dependencies will be installed here.
5. To set up the environment variables refer to the documentation provided by Intel® onhttps://software.intel.com

D. Directory Structure

a) Intel Models
/opt/intel/computer_vision_sdk/deployment_tools/intel_models

b) Actual Code
/opt/intel/computer_vision_sdk/deployment_tools/inference_engine/samples

c) Executables
/home/user/inference_engine_samples/intel64/Release

E. Pre-Trained Models in OpenVINO

- age-gender-recognition-retail: Age and gender classification.[2]
- face-detection-retail: Face Detection.
- person-detection-retail: Person detection.
- license-plate-recognition-barrier: Chinese license plate recognition.
- face-detection-adas: Face Detection.
- person-detection-retail: Person Detection.
- head-pose-estimation-adas: Head and yaw + pitch + roll.
- vehicle-attributes-recognition-barrier: Vehicle attributes (type/color) recognition.
- person-vehicle-bike-detection-crossroad: Multiclass (person, vehicle, non-vehicle) detector.
- vehicle-license-plate-detection-barrier: Multiclass (vehicle, license plates) detector.

F. Converting TensorFlow Models into OpenVINO compatible format

1. Download the TensorFlow model to be made compatible to OpenVINO
2. Extract the zip file in the home/user/tmp directory and execute the following command: /opt/intel/computer_vision_sdk/deployment_tools/model_optimizer/mo_tf.py --input_model=/tmp/ssd_mobilenet_v1_coco_2018_01_28/frozen_inference_graph.pb --tensorflow_use_custom_operations_config/opt/intel/computer_vision_sdk/deployment_tools/model_optimizer /extensions/front/tf/ssd_support.json --tensorflow_object_detection_api_pipeline_config /tmp/ssd_mobilenet_v1_coco_2018_01_28/pipeline.config --reverse_input_channels
3. An xml and bin file will be generated which is compatible to OpenVINO.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

IV. EXPERIMENTAL RESULTS

We have executed some of the pretrained models in OPENVINO. The input, output, and commands to run them are given below.

A. Face Detection

Figure 2: Input Image for Face Recognition

Figure 2 Description: The figure is the input given to the face recognition algorithm. The algorithm works the face detection module on the figure above.

Figure 3: Output image for Face Recognition

Figure 3 Description: The figure describes the output of detection as a result of execution of the face recognition algorithm. Bounding boxes describe the age, gender and emotions are seen.

```
nikita@nikita-HP-Pavilion-15-Notebook-PC: ~/inference_engine_samples/intel64/Release
nikita@nikita-HP-Pavilion-15-Notebook-PC:~/inference_engine_samples/intel64/Rele
ase$ ./interactive_face_detection_sample -i facesample.jpg -m /opt/intel/compute
r_vision_sdk/deployment_tools/intel_models/face-detection-adas-0001/FP32/face-de
tection-adas-0001.xml -m_ag /opt/intel/computer_vision_sdk/deployment_tools/inte
l_models/age-gender-recognition-retail-0013/FP32/age-gender-recognition-retail-0
013.xml -m_hp /opt/intel/computer_vision_sdk/deployment_tools/intel_models/head-
pose-estimation-adas-0001/FP32/head-pose-estimation-adas-0001.xml -m_em /opt/int
el/computer_vision_sdk/deployment_tools/intel_models/emotions-recognition-retail
-0003/FP32/emotions-recognition-retail-0003.xml
InferenceEngine:
        API version ............ 1.2
        Build .................. 13911
[ INFO ] Parsing input parameters
[ INFO ] Reading input
[ INFO ] Loading plugin CPU

        API version ............ 1.2
        Build .................. lnx_20180510
        Description ....... MKLDNNPlugin
[ INFO ] Loading network files for Face Detection
[ INFO ] Batch size is set to 1
[ INFO ] Checking Face Detection inputs
[ INFO ] Checking Face Detection outputs
[ INFO ] Loading Face Detection model to the CPU plugin
```

Figure 4: Terminal Command for Face Recognition

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

Figure 4 description: The figure describes the command that is required to execute the face recognition module. The first parameter denotes the executable file created. '-i' is used to take image as an input and '-m' specifies the path at which the training model for the algorithm is located.

B. Object Detection
(Refer section II-F)

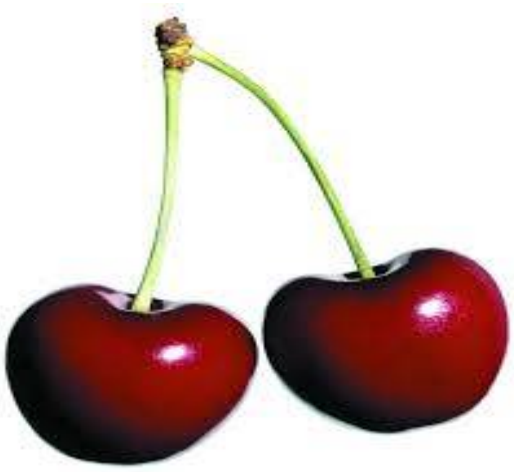

Figure 5: Input Image for Object Detection

Figure 5 Description: The figure is the input given to the object detection algorithm.The algorithm works the object detection module on the figure above.

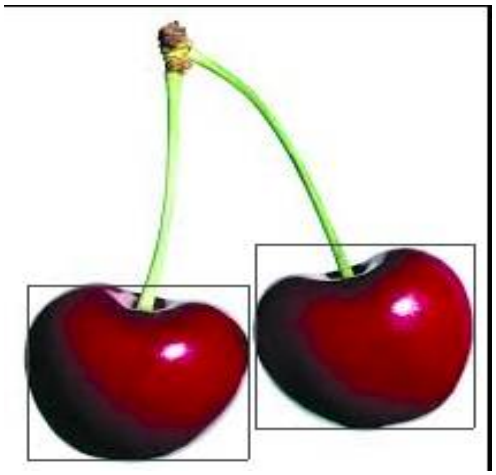

Figure 6: Output Image for Object Detection

Figure 6 Description: The figure describes the output of detection as a result of execution of the object detection algorithm. Bounding boxes describethe detected objects are seen.

```
nikita@nikita-HP-Pavilion-15-Notebook-PC:~/inference_engine_samples/intel64/Release$ ./object_detection_sample_ssd -i download.jpg -m /home/nik
ita/frozen_infere
nce_graph.xml
[ INFO ] InferenceEngine:
        API version ............ 1.2
        Build .................. 13911
Parsing input parameters
[ INFO ] Loading plugin

        API version ............ 1.2
        Build .................. lnx_20180510
        Description ....... MKLDNNPlugin
[ INFO ] Loading network files:
        /home/nikita/frozen_inference_graph.xml
        /home/nikita/frozen_inference_graph.bin
[ INFO ] Preparing input blobs
[ INFO ] Batch size is 1
[ INFO ] Preparing output blobs
[ INFO ] Loading model to the plugin
[ WARNING ] Image is resized from (212, 237) to (300, 300)
[ INFO ] Batch size is 1
[ INFO ] Start inference (1 iterations)
[ INFO ] Processing output blobs
[0,86] element, prob = 0.651132    (12.4661,142.101)-(109.014,232.054) batch id : 0 WILL BE PRINTED!
[1,86] element, prob = 0.590475    (111.198,121.665)-(206.3,215.784) batch id : 0 WILL BE PRINTED!
[ INFO ] Image out_0.bmp created!

total inference time: 61.642
Average running time of one iteration: 61.642 ms

Throughput: 16.2227 FPS

[ INFO ] Execution successful
nikita@nikita-HP-Pavilion-15-Notebook-PC:~/inference_engine_samples/intel64/Release$
```

Figure 7: Terminal Command for Object Detection

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

Figure 7 description: The figure describes the command that is required to execute the object detection module. The first parameter denotes the executable file created. '-i' is used to take image as an input and '-m' specifies the path at which the training model for the algorithm is located.

C. Motion Detection

Figure 8: Output for Motion Detection

Figure 8 description: The figure describes the output of execution of motion detection module. It can be seen that faces are detected as unknown and an action viz standing is displayed as a label for the bounding box.

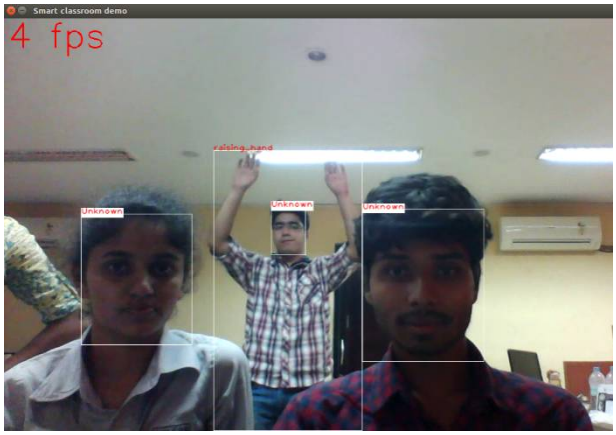

Figure 9: Output for Motion Detection

Figure 9 description: The figure describes the output of execution of motion detection module. It can be seen that faces are detected as unknown and an action viz raising hands is displayed as a label for the bounding box.

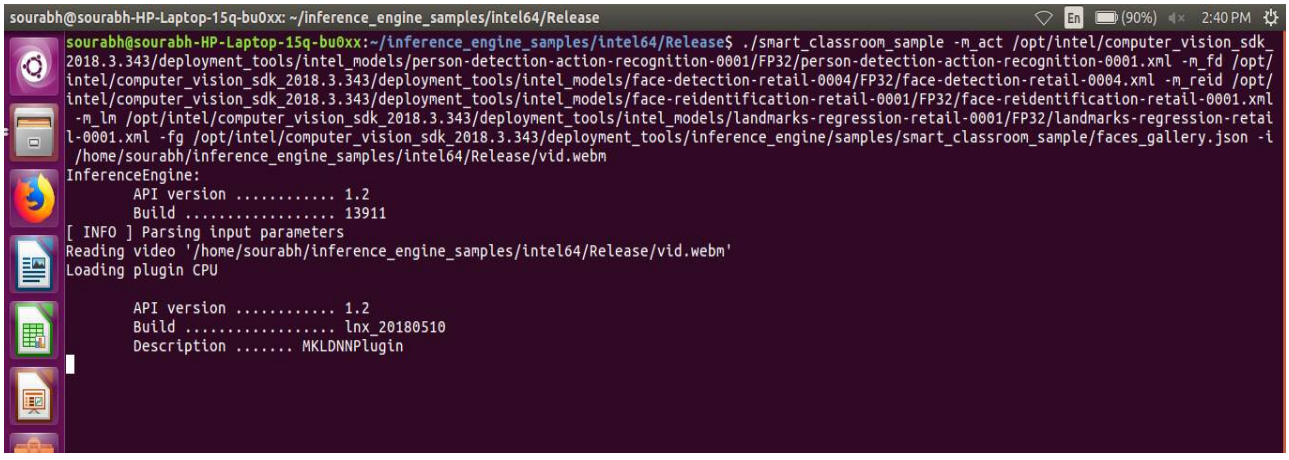

Figure 10: Terminal Command for Motion Detection

Figure 10 description: The figure describes the command that is required to execute the motion detection module. The first parameter denotes the executable file created. '-i' is used to take image as an input and '-m' specifies the path at which the training model for the algorithm is located. It should be noted that the face detection module has also been executed in this module. Parameters like m_fd, m_reid are used for detection of actions.

ISSN(Online): 2319-8753
ISSN (Print): 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Visit: www.ijirset.com

Vol. 7, Issue 11, November 2018

V. APPLICATIONS AND FUTURE SCOPE

OpenVINO can be used to provide surveillance in various sensitive areas such as Banks, Hospitals, Museums, Jewellery Stores, Malls, Parking Lots, etc.
Future Scope can include integrating the various modules present so that they can run simultaneously according to the customer's requirement.

VI. CONCLUSION

Video Analytics can be a complex task, however, the use of tools like OpenVINO can be used to simplify analysis ofthe video content. OpenVINO Toolkit can be used in a number of applications and can be customized according to our requirements.

REFRENCES

[1] G.P. Saroha and Pawan Kumar Mishra, "A study of Surveillance System for Object Detection and Tracking", International Conference on Computing for Sustainable Global Development (INDIACom), 2016
[2] S. V. Tathe, A. S. Narote and S. P. Narote, "Human Face Detection and Recognition in Videos", Conference on Advances in Computing, Communications and Informatics (ICACCI), 2016
[3] Rupali Nikhare, Satishkumar Varma and Shraddha G. Mhatre, "Visual Surveillance Using Absolute DifferenceMotion Detection", International Conference on Technologies for Sustainable Development, 2015
[4] Shashank Prasad and Shubhra Sinha, "Real-time Object Detection and Tracking in an Unknown Environment", World Congress on Information and Communication Technologies, 2011
[5] Keith Fieldhouse, Matthew J. Leotta, Arslan Basharat, Russell Blue, David Stoup, Charles Atkins, Linus Sherrill, Benjamin Boeckel, Paul Tunison, Jacob Becker, Matthew Dawkins, Matthew Woehlke, Roderic Collins, Matt Turek and Anthony Hoogs, " KWIVER: An Open Source Cross-Platform Video Exploitation Framework", IEEE, 2014