DEEP LEARNING TECHNIQUE COMPRISING OBJECT DETECTION AND IDENTIFICATION USING MOVIDIUS NEURAL COMPUTE STICK

SABARISH KUMARAN R
VIT,VELLORE
sabarishkumaran11a12@gmail.com

RAMYA S
VIT,VELLORE
ramyashekar99@gmail.com

KISHOR KUMAR R
VIT,VELLORE
kishorkumar007.r@gmail.com

ABSTRACT: **Due to the vagueness in the research and development of image processing techniques and applications, detecting and identifying real-time objects with high accuracy at faster rate is an efficient way to acquire a revolution in technology. In this study, the device used for the process of detecting and recognizing objects is Movidius neural compute stick (NCS) which is a tiny offline, low power VPU, fanless deep learning device very effective for deploying Artificial Intelligence and deep learning applications using Neural Networks (CNN and DNN). Main objective of the process is to classify, detect and recognize the objects in real time which is trained prior as datasets, later used for identification using SSD (Single Shot Detector) algorithm trained with Caffe or TensorFlow framework and MobileNet architecture. And it incorporates some of the procedures of YOLO (Tiny YOLO). It is very interesting to understand the VPU (vision as importing processing unit) of Neural compute stick that is myriad 2 VPU. The NCS is plugged into a host machine using the USB interface on the VPU. OpenCV libraries was an essential part of it and this process can be achieved using UBUNTU (16.04) platform which is designed for strong focus in computational efficiency.**

Keywords: Neural Compute Stick (NCS), AI and Deep learning, CNN and DNN, YOLO (You Only Look Once), Single shot Detector, Vision processing Unit (VPU).

I. INTRODUCTION:

Object detection and identification is the process of identifying the presence of any objects, recognizing and labeling it with the help of pre-trained datasets in NCS SDK (Neural Compute Stick Software Development Kit) through deep learning. It considers everything as object and detects it using SSD algorithm and MobileNet technique. To achieve this we use an USB like hardware device called Neural Compute Stick (NCS) produced by Movidius Intel which works on Ubuntu 16.04 or Single board computers (Raspberry pi 3) and this work was implemented on Ubuntu 16.04. Being human, sensing and recognizing the presence of all objects in front of us is an easy task, but when it comes to computer system it cannot recognize object as it is. As we are trained enough from our childhood to recognize and classify objects, the system needs to be trained with thousands of datasets and images to become frequent with the objects and eventually detects them with good accuracy in terms of probability.

NCS is designed for low power devices to achieve high FPS (frame per second).Initially, to use NCS in a stand-alone environment, installing NCS SDK and generating graph files is necessary. Later, NCS was developed to reduce the burden of training the neural network and datasets as it is comprised of a pre-trained network. Generally object detection has totally three stages constituting Training (here pre-trained datasets), Inference and deployment. Training of a neural network is a separate process that has not been discussed here. Inference is concluding results through all formulations and comparisons of images. Deployment stage is implementing the process using python script file and establishing it in our day to day use.

NCS overcomes all the difficulties of training process, and produces effective results in terms of accuracy, speed, processing time and complexity. Moreover some of the existing techniques have good accuracy but lags in other criteria. NCS compares the images with the pre-trained parameters of the other images, where the existing techniques compare the images directly.

This is the reason why the proposed method does not face any accuracy issues.

II. ARCHITECTURE

Neural Compute Stick (NCS) is comprised of unique features thus incorporating them into a legible architecture which has Single/Dual Intel Movidius Myriad 2 VPU which is an AI-embedded chip comprising of high-quality architecture that is used to accelerate the image processing and computer vision techniques PMIC (MAX77620). The operating temperature for this NCS as recorded by the research is found to be -20°C to +70°C in the modern smart devices.

The boot image device used for loading images is a simple USB port and PCIe mini-card is developed provided with the PCIe x 1 interface. Integrated memory with LPDDR3 4GB is incorporated in the architecture with maximum power up to 2 x PMIC (MAX77620). This device is also certified by CE (Charted Engineers) & RoHS (Restriction of Hazardous Substances) Compliance. The form factor that is used is USB which has the capability to attach to the host systems, and the VPU present inside is responsible for providing machine learning on a low-power deep learning inference engine [10].

The VPU (Myriad 2 VPU-28 nm capable of 80-150 GFLOPS performance) present inside the device uses only 1 W of power and 10-15 inferences to generate nearly 100 GFLOPS. Low-Power DDR3 DRAM (4GB), image and computer-vision accelerator and an array comprised of 12 VLIW (Very Long Instruction Word) processors called SHAVE (Streaming Hybrid Architecture Vector Engine) processors, which supports parallel computations that are included in the VPU (Vision Processing Unit). This device entirely consumes 2.5 W through its USB 3.0 port. Framework used for this device like Caffe or TensorFlow will be trained through CNN and later compiled into an embedded neural network, optimized to run on the VPU (Vision Processing Unit) inside NCS (Neural Compute stick) [13].

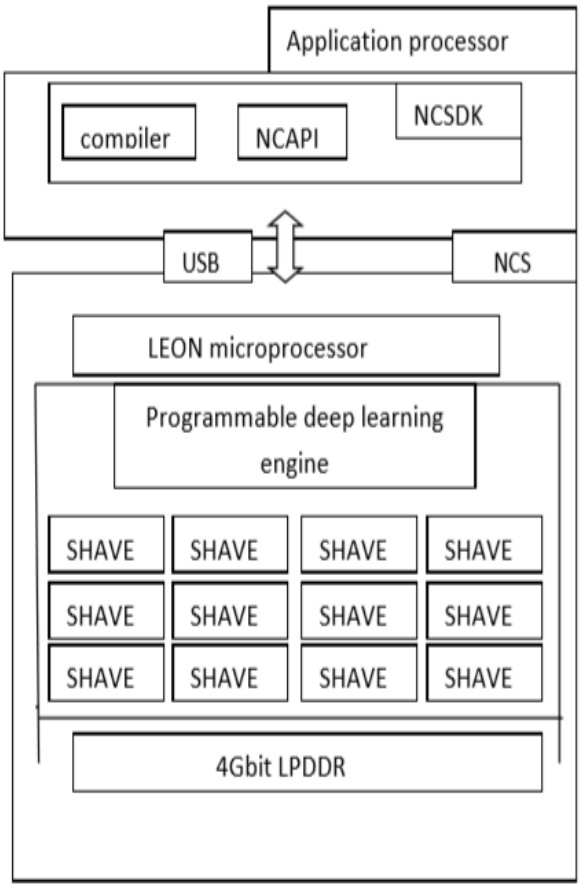


Fig.Architecture of NCS [1].

NCS (Neural Compute Stick) device lets execution to be controlled by the Leon Microprocessor and computations to be performed by the SHAVE processor. LEON Microprocessor is a 32 bit microprocessor. This design is used for embedded applications provided along with low complexity and power consumption with high performance. SHAVE (Streaming Hybrid Architecture Vector Engine) processor consists of wide and deep register files integrated with a Very Long Instruction Word (VLIW) used for code-size efficiency. VLIW packets control multiple functional units containing SIMD (Single instruction, multiple data) capability for high parallelism and throughput at a functional unit and processor level. Each of these units can be launched in parallel in a single instruction packet [1].

III. NEURAL COMPUTE STICK

NCS (Neural compute Stick) is a device used to work quickly with plug and play technique made for developing a simple infrastructure which looks like a USB stick that can be easily attached to PC's. And training cannot be done with NCS, in spite it has pre-trained datasets and networks. PC's are low-power devices which are expected to work with high performance and this device is designed for such capability provided with no internet requirement. Myriad 2 VPU is responsible for its computing capability. Integrating low-graph processors and deep-learning devices efficiently is still a challenging task and this device helps this process in an effective way.

NCS is therefore vision as GPU (Graphics Processing Unit) running on a portable USB where model that is trained would work optimally on the physical environment comprising of physical devices. This device was mainly designed with the aim to achieve inference being done in the infrastructure of the stick. This device is only accessible when connected through USB 3.0 Type-A port. Industries are migrating from cloud to edge for the major advantages like accuracy, efficient computations, model efficiency and model size [11].

INTEL developed the NCS (Neural Compute Stick) with the motive to perform inference inside its own infrastructure regardless of where the model is being trained, so that the optimization is achieved for the process of inference at the edge. This device became more specialized with the creation of Model Zoo which turns out to be a repo of models, which are fully optimized, ready to use and can be made extensible and customized as per the requirement [8].

This device supports CNN profiling, prototyping, tuning workflow and inference at the edge without cloud connectivity. The key feature of this device includes capacity to run multiple devices on the same platform to scale performance and quick deployment of trained Caffe framework-based feed-forward Convolutional Neural Network (CNN) models and uniquely trained networks as well [12].

NCS has pre-trained datasets for detection of these objects: airplanes, bicycles, birds, boats, bottles, buses, cars, cats, chairs, cows, dining tables, dogs, horses, motorbikes, people, potted plants, sheep, sofas, trains, and television monitors, *w*hich means that sample video consisting of these objects are recognized and classified by NCS.[3]

IV. METHODOLOGY OF EXISTING WORKS

Various methodologies are used to project the process of object detection through NCS i.e. before the incoming of NCS to the field of work, developers have used these techniques to propose the work and the notable markings by SSD, YOLO (You look only once), DNN(Deep Neural Network) and CNN(Conventional Neural Network) accompanying some of the Neural Network frameworks and libraries such as MobileNet, Caffe, and OpenCV. Earlier object detection methodologies have been following some of the neural network concepts. In this paper, we incorporated those concepts with the NCS and achieved the real-time object detection.

MobileNet:

This proposed system using MobileNet assumed that the spatial dimension maps the output feature that is same as the input feature and those feature maps are the square. With aspect ratios and arbitrary sizes, feature maps generalize their model results [14].

CNN:

This study ensures that the TensorFlow-based object detection and CNN-based object tracking algorithm is performed. These techniques help to detect the objects in complex backgrounds. And the CNN has been used to extract the local characteristics and visual features and these have been used in the recognition algorithm. CNN combines these (local respective fields, shared weights and spatial sub sampling) three architectural ideas that gives us the distortion and a certain degree of invariance of change [15].

CNN for Multi Class Recognition Object:

This proposed system is based on CNN and utilizes the most popular framework TensorFlow. They successfully applied five layers of CNN model for recognition with non-linear activation function and Rectified Linear Unit (ReLU) for recognition purposes. They used data augmentation and dropout to give the improved model accuracy and potential over fitting. They employed normalized weight initialization approach to get the weights at every layer of the model. And to improve the performance of the detection, they applied several deep learning techniques [16].

SSD:

In this system, SSD uses grid structure to make the regions whereas the other end-to-end detectors like Faster-CNN uses proposal mechanism for the additional region. With this approach, SSD is faster by 41 FPS compared to the R-CNN. And it provides the object maps with various scales, which overcome the disadvantages of YOLO that derives the object regions in only one feature map [17].

V. DEEP LEARNING ON NCS:

In the field of machine learning, deep learning technique plays a major role and that has been achieved with the help of NCS in this paper. Some of the deep learning fields are object detection, facial recognition, bioinformatics, etc. With the help of Deep learning on NCS we are going to detect the objects and show the identified object in the output with bounding boxes around it. Deep learning through NCS has been very helpful to automate tasks and execute it. In recent times, deep CNN has been used for the object detection since it has the better image processing algorithm. So we tried to implement that algorithm in the process to detect the objects. And to extract the feature maps in the object detection and recognition there are some CNN architectures has been used. They are MobileNet, VGG (Visual Geometry Group), Inception and ResNet (Residual Network). In this paper, we tried to elicit the feature maps from input frames with the MobileNet architecture as it is designed for resource constrained devices where as other architectures are unsuitable because of their sheer size. To perform the deep learning techniques there are some python framework deep learning libraries (Caffe, TensorFlow, Theano, etc) which can be used. In this proposed work, it is with Caffe [2].

VI. INSTALLATION OF NCS SDK

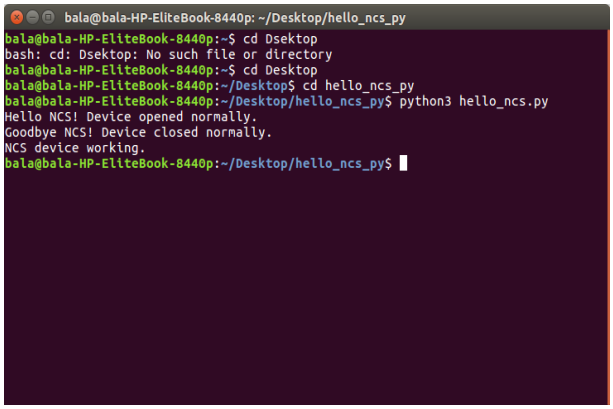
NCS SDK is the content model and software development kit comprising many other files which is supported by NCS and provided by Movidius. NCS SDK is used to generate graph files and later it is used for object detection. It carries additional projects other than object detection, coded with pre-trained datasets incorporating a particular framework library for deep learning and neural networks. Following is the procedure to use it and get NCS installed in the Ubuntu 16.04 Environment.

To download NCS SDK , two primary ways are github repository which comes as open source project with all required tools and libraries to produce a graph from framework models and executing particular commands in terminal (in Ubuntu 16.04). An exclusive Ubuntu platform is necessary for NCS SDK to work. Having other virtual python Environments or OpenCV installed, used by other applications may lead to inability of the present libraries and could cause conflict with system PYNOPATH.

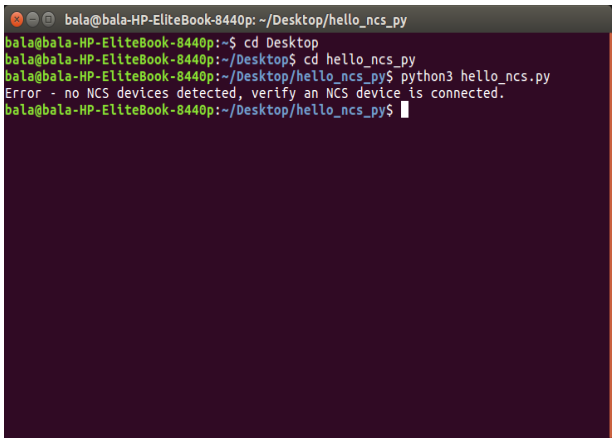
- sudo apt-get install git
- cd ~
- mkdir ncsproject
- cd ncsproject
- git clone https://github.com/movidius/ncsdk.git
- git clone https://github.com/movidius/ncappzoo.git
- cd ~/ncsproject/ncsdk
- make install

Executing this in Ubuntu terminal, constituting good network speed could take 20-30 mins of time [9].Once it gets installed, verify its connectivity by executing type “dmesg” in terminal and wait for the output then run the hello_ncs.py file inside the ncsdk directory(cd/ncsdk/examples/apps/hello_ncs_py/ python3 hello_ncs.py) to test the NCSSDK working.

(i) when the NCS Connected:



(ii) when NCS disconnected:



After this process, NCS and the python code inside NCS SDK put forth the object detection process and this makes NCS to compute for the frameworks available (caffe, tensor flow etc.).

VII. NCS SDK WORK FLOW

Fig.NCS SDK Workflow [4]

The steps involved in the workflow are:

- Having pre-trained data framework models or training a new network with the help of a powerful GPU (in case of Raspberry pi 3)with the use of tensor flow or caffe framework and standard algorithm. Network training phase does not utilize NCSDK.
- Convert the trained model into graph file which can be deployed using SDK and NCS.
- To deploy the graph file, writing a python script is primary source.
- Deploy the python script, and graph file on the platform used constituting NCS.[4]

VIII. NCS SDK

[1,5] It is the collection of files and tools aiding development and deployment of deep learning based real time applications. Primarily, this SDK has three command line tools named as mvNCCompile, mvNCProfile, mvNCCheck.

mvNCCompile:

Used for converting Tensorflow/Caffe framework and its weights to an internal Movidius compiled format for the purpose of Movidius Neural Compute API

mvNCProfile:

Used for the performance evaluation of framework models by layer by layer statistics on NCS.

mvNCCheck:

Used for the comparison of results obtained from running the network on NCS versus framework models for network verification.

4Gbit LPDDR:
Used for the reduction of supply voltage from 2.5V to 1.8V, where LPDDR stand for Low power dual data rate (LPDDR) and it is the modified form of dual data rate for SDRAM

IX. OPENCV

Open Source Computer Vision Library, an open source machine learning Software library which supports computer vision, image processing. It is

very efficient, as it supports wide range of algorithms related to machine learning and image processing. There is growing demand for OpenCV as it enhances computational efficiency and very useful for real time applications. It supports C++, python, java in its current version and it works on different platforms such as Windows, Linux and Android. Here we accommodated Python on Ubuntu 16.04 Environment. Python became very popular nowadays as it is very easy and simple to learn, and to notice that NCS supports OpenCV where it can run on more than 2000 algorithms. In this work, SSD algorithm in OpenCV is used. Where SSD combines with MobileNet makes it efficient deep learning method for object detection. [1]

X. PROPOSED SYSTEM:

Earlier, object detection has been performed using many other techniques such as YOLO, SSDs. Faster R-CNN etc. But considering the key capabilities of Neural Compute Stick (NCS) and its total involvement in deep learning makes it superior and effective technique over others which had been deployed. We have used NCS incorporating SSD Algorithm with MobileNet Architecture for object detection [1]. The reason behind usage of MobileNets is that they are designed for resource constrained devices where as other existing architectures such as VGG or ResNet are very large which makes them unsuitable for resource constrained devices [3]. As NCS is tiny and has a low power VPU, it can be used for various applications and fields for accurate outcomes such as Real time Vehicle detection, security cameras, modern drone etc. Eventually, it is composed of detection, recognition, classification as working fields with increase in accuracy, speed and reducing processing time.

XI. METHODOLOGY

The proposed system considers everything as objects i.e. Person, vehicles, birds etc and it can detect multiple objects within same image. The following methodologies are followed

- Understanding other possible methods on detection and classification of objects
- Studying the already presented works on object detection
- Comparing the above methods with our proposed system of work
- Proposed work seems superior over existing methods on the basis of accuracy, speed, efficiency. So implementing the proposed work of object detection and classification
- Later, moving an video inside NCS SDK (video_objetcs inside ncapzoo) proceeds the work.[1]

XII. PROCEDURE:

Having a platform, either Ubuntu 16.04 or Raspberry pi 3 with Raspbian OS where proposed work can be performed and our work is performed on Ubuntu 16.04.There is an option of installing virtual box in case of having multiple OS. Confirming that having exclusive stand-alone environment for NCS SDK and don't install the SDK on a "daily development and productivity use" machine [4]. Aforesaid, Installing NCS SDK has two ways, follow any one of it and after that ensure NCS SDK connectivity to the system. Later, move the sample video to the NCS SDK (video_objects in ncapzoo) directory. It uses SSD algorithm and MobileNet architecture together to detect objects [2]. And it uses FPS to get objects from the video. Open terminal, check the connectivity status and working of NCS and direct into ncapzoo, the folder which has sample video and python code file (video_objects directory), run the python extension file , where the file contains python libraries and code for deploying the graph. The python extension file is present default in ncapzoo provided by Intel Movidius in this Pre-trained network. If it is to be trained with a new network and datasets, then the scenario is completely different which we have not addressed. Once you open the python extension file in terminal, then wait for another window for the output we have expected with group of default bounding boxes covering every objects in the video representing the object type from the trained datasets with accuracy rate**.**

XIII. PROPOSED SYSTEM VS EXISTING SYSTEM

In the existing systems mentioned above, they trained the network with datasets consisting hundreds of sample images to recognize the given input image with the sample image. But the criterion with NCS is totally different which has pre-trained datasets within its NCS SDK's python code file, which makes the works easier and faster. Moreover, in the existing systems object detection proceeds by object recognition constituting comparison of sample image and the input image whereas through NCS, object detection takes place by considering parameters of images in spite of comparing images directly which is a tedious process compared to object detection by NCS, which is accurate, faster. .

Precise note on the difference among the proposed system and existing system [1]:

Existing system:

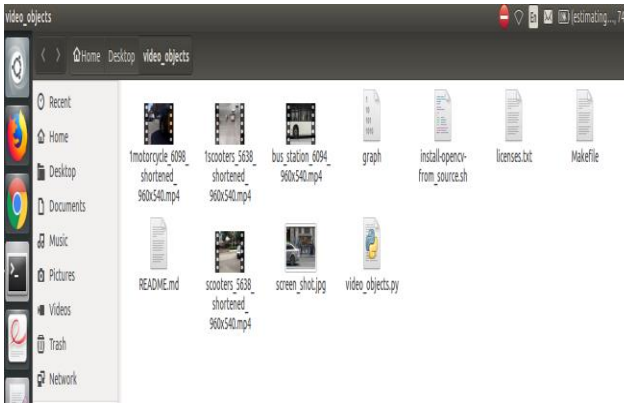
- No NCS is used, so manual training.
- Network is required
- Time consuming, tedious, complex

Proposed system:

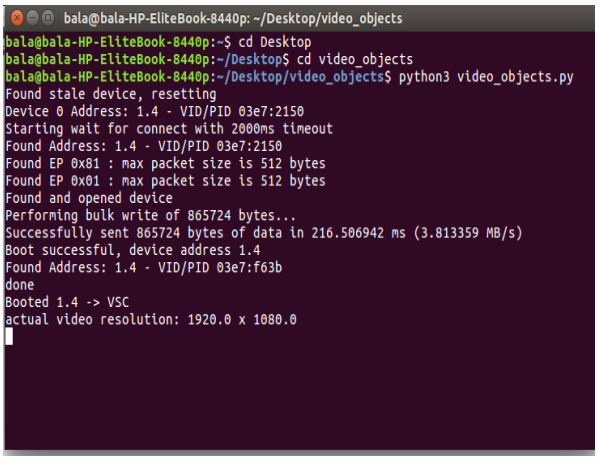
- Caffe framework is used
- Pre-trained datasets in NCS
- No network is required and seems faster, accurate and less complex.

XIV. IMPLEMENTATION:

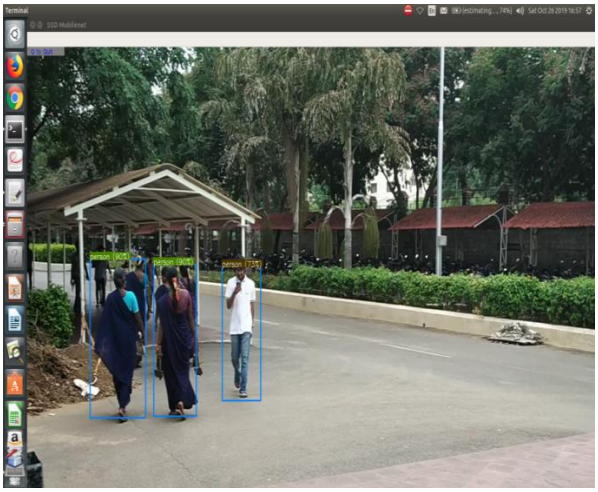
(i) place the sample video in the respective folder or SDK(video_objects):



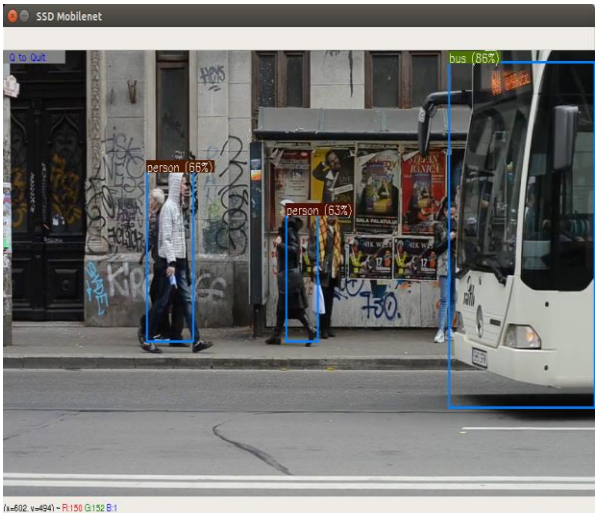
(ii) Access the python extension file from the video_objects folder:



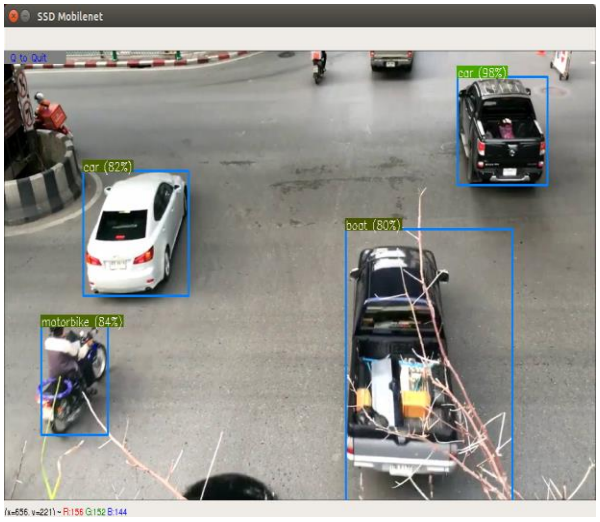
(iii) SSD MobileNet output window with bounding boxes and object detection (Scenario-1):



(iv) Scenario-2:



(v) Scenario-3:



XV. APPICATIONS OF NCS:

NCS SDK contains other working models and processes too, rather than object detection. In particular it has “Bird detection” which produces the output from the input image specifying the kind of bird with accuracy rate. And this “Bird detection” process needs the power of two NCS to perform the output.

Another project is that “Face detection”, where primarily sample image should be given in the NCS SDK(respective process folder) directory, later through web cam input image is taken continuously. Output is produced with default box around the output window with two colors denoting, green for inputs matching and red for mismatch.

And more processes are available in NCS SDK. In recently updated NCS SDK by Movidius Intel some more crucial processes are also added.

NCS is used for wide range of applications because of its vital properties [12]

- Smart home and consumer robotics
- Surveillance and security
- Retail
- Healthcare
- Robot assistant

XVI. ADVANCEMENT AND FUTUREWORK

INTEL has come up with next generation of NCS which is NCS 2 (Neural Compute Stick 2) that is assured to provide 8 times performance when compared to its previous generation NCS. This version release was intended to justify the improvement with the use of latest VPU (Vision Processing Unit) Movidius Myriad X which contains 16 SHAVE cores (Streaming Hybrid Architecture Vector Engine). It has 4 pieces on top of the 12 cores present in its predecessor Movidius NCS (Neural Compute Stick). [6]

The third Generation of VPU is introduced as Intel Movidius Myriad X VPU which was developed by INTEL. A dedicated hardware accelerator is featured as specialization of its class to perform inference for deep neural network models. This VPU is considered as the industry expert for on-device ML models and neural network applications. Intel’s Movidius Myriad X VPU is advancing with features like performance gain in imaging and vision accelerators, increased SHAVE cores and a new 4k pipeline that supports up to 8 HD sensors directly attached to the VPU. This VPU is operated via MDK (Myriad Development Kit) which supports all required development tools, frameworks and APIs to process computer vision, imaging and neural network workloads on the chip. [7]

NCS need external tools and SDK to work but the new release by INTEL which is NCS 2 doesn’t need such exclusive tools to work on. OpenVINO toolkit is used as a supporting software platform to achieve optimization in machine learning models. This toolkit was developed for the purpose of attaining hardware platform-independency which includes INTEL Arria and FPGA runtime environments. This OpenVINO toolkit also supports different software platforms like Ubuntu, CentOS and Yocto Linux along with Microsoft Windows and Raspbian 32-bit OS. This toolkit also includes ONNX (Open Neural Network Exchange) for deploying ML models across multiple frameworks. This toolkit consists of tools for generating an Intermediate

Representation (IR) from different frameworks like TensorFlow, Caffe and Apache MXNet models. [8]

In future, many other ML models are queued to be implemented using this INTEL NCS 2 and Movidius Myriad X VPU along with OpenVINO toolkit. INTEL would gradually migrate from OpenVINO toolkit to a unified platform for model deployment and inferencing at the edge. Future advancements may include deployment of certain applications like fully automated driverless cars and many challenging NLP (Natural Language Processing) applications.

XVII. RESULTS

The proposed system is used for object detection and recognition using Neural Compute Stick (NCS)in real time scenarios which comes under the field of deep learning. NCS is a USB like device which works with USB 3.0 type A plug. Installing a stand-alone Ubuntu 16.04 environment for NCS and checking its connectivity and working with the system put forth the work of object detection. Execution of commands in terminal with the framework models and deploying the graph with python script projects the work, with the code discussed above. Collaborating SSD with Raspberry pi 3 CPU produces very low FPS(nearing 0.5), it is very low for object detection and to overcome this we use NCS(produces almost 3.5FPS) [2]. Existing techniques such as SSD, YOLO has good speed but lags in precision whereas Faster R-CNN has good precision and low speed. We get good precision with speed by incorporating SSD algorithm with MobileNet technique through NCS [1].

[15] The Quantitative measures are computed using parameters like accuracy, sensitivity, specificity, precision and recall. The attributes used to compute these parameters are true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

Statements for attributes:

TP: Outcome of a model which is correctly predicted as positive class.

FP: Outcome of a model which is incorrectly predicted as positive class.

TN: Outcome of a model which is correctly predicted as negative class.

FN: Outcome of a model which is incorrectly predicted as negative class.

The mathematical representation of the above given metrics are as follows:

- Accuracy is the overall performance of the system including sensitivity and specificity.

  a. $Accuracy = \frac{TP+TN}{TP+FP+TN+FP}$

- Precision is the positive predictive value or the fraction of the positive predictions that are actually positive.

  b. $Precision = \frac{TP}{TP+FP}$

- Sensitivity is the ratio of truly object present in the scene who are correctly identify as an object. This term present the number of positive samples correctly identified. Higher the true positive element higher is the sensitivity.

  c. $Sensitivity/Recall = \frac{TP}{TP+FN}$

- Specificity is the ratio of truly stationary object present in the scene that are correctly identify as a stationary object. This term present the number of negative samples correctly identified. Higher the true positive element higher is the Specificity.

  d. $Specificity = \frac{TN}{TN+FP}$

XVIII. CONCLUSION:

In this proposed System we have used an USB like hardware device called Neural Compute Stick (NCS) for object detection overcoming the lags and constraints from the existing systems. The framework models used here are addressed

for the Ubuntu platform 16.04. Even this NCS can be used for real time applications. It considers everything as objects, deploys the graph using the frameworks models and pre-trained datasets inside the python code for detecting the objects [1]. It can even be used with single board computers like Raspberry pi 3, depending upon developers interest. Taking FPS into considerations, Ubuntu or single board computer with NCS, stuns with result stating higher FPS(3.5) than SSD MobileNet with Raspberry pi 3 CPU without NCS(0.5), which is almost 7 times efficient [2]. Eventually it produces results with less processing time, high accuracy in terms of probabilities and with default bounding boxes around the detected objects with object type and accuracy.

XIX. ACKNOWLEDGEMENT

We sincerely thank our supervisor Prof. P.Prabavathy, VIT Vellore for giving us the opportunity to work on this project and also her constant supervision and valuable guidance during the course of the thesis.

XX. REFERENCES

[1] Banumathi, K., Bharathi, B., Daisy Deve Priya, ,. S., & Gogulalakshmi, P. (2018). Real Time Vehicle Detection using Movidius Neural Compute Stick. International Journal of Engineering Research & Technology (IJERT), (p. 5).

[2] Othman, N. A., & AYDIN, I. (2018, October). A New Deep Learning Application Based on Movidius NCS for Embedded Object Detection and Recognition. In 2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT) (pp. 1-5). IEEE.

[3] Combining MobileNets and Single Shot Detectors for fast, efficient deep-learning based object detection,(n.d.). Retrieved from pyimagesearch website: https://www.pyimagesearch.com/2017/09/11/object-detection-with-deep-learning-and-opencv/

[4] Installing the Intel Movidius SDK, ,(n.d.). Retrieved from pyimagesearch website: https://www.pyimagesearch.com/2018/02/19/real-time-object-detection-on-the-raspberry-pi-with-the-movidius-ncs

[5] Introduction , (n.d.). Retrieved from movidius.github.io website: https://movidius.github.io/ncsdk/tools/tools_overview.html

[6] Object detection using the Intel Neural Compute Stick 2 and OpenVINO, (n.d.). Retrieved from Good Audience website: https://blog.goodaudience.com/a-test-drive-of-the-intel-neural-compute-stick-2-b694983e5f9b

Intel® Movidius™ Myriad™ X VPUs
Intel® Movidius™ Myriad™ X VPUs
Intel® Movidius™ Myriad™ X VPUs
Intel® Movidius™ Myriad™ X VPUs

[7] Intel Movidius Myriad X VPUS,(n.d.). Retrieved from intel.ai website: https://www.intel.ai/intel-movidius-myriad-vpus/

[8] Intel NCS 2, (n.d.). Retrieved from THENEWSTACK Website: https://thenewstack.io/a-closer-look-at-intel-movidius-neural-compute-stick/

[9] (n.d.). Retrieved from academia website: https://www.academia.edu/40065515/Raspberry_PI3_and_a_Movidius_Neural_Compute_Stick_for_Real-Time

[10] (n.d.). Retrieved from www.adlinktech.com: https://www.adlinktech.com/Products/Deep_Learning_Accelerator_Platform_and_Server/Deep_Learning_Accelerator/EDL-mPCIe-MA2485?Lang=en

[11] Darren Crews. (n.d.). Retrieved from https://qconsf.com/: https://qconsf.com/sf2017/system/files/presentation-slides/qconf_presentation.pdf

[12] Rungta, M. (n.d.). Retrieved from http://www.cse.iitd.ac.in/: http://www.cse.iitd.ac.in/mavi/docs/thesis_sarvesh_mukund.pdf

[13] Hochstetler, J., Padidela, R., Chen, Q., Yang, Q., & Fu, S. (2018, October). Embedded deep learning for vehicular edge computing. In 2018

IEEE/ACM Symposium on Edge Computing (SEC) (pp. 341-343). IEEE.

[14] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

[15] Mane, S., & Mangale, S. (2018, June). Moving Object Detection and Tracking Using Convolutional Neural Networks. In 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS) (pp. 1809-1813). IEEE.

[16] Hayat, S., Kun, S., Tengtao, Z., Yu, Y., Tu, T., & Du, Y. (2018, June). A Deep Learning Framework Using Convolutional Neural Network for Multi-Class Object Recognition. In 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC) (pp. 194-198). IEEE.

[17] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.