Face Mask Detection Based on Machine Learning and Edge Computing

Ivan Jovović, Dejan Babić, Stevan Čakić,
Tomo Popović, *Senior IEEE Member*
Faculty for Information Systems and Technologies
University of Donja Gorica
Podgorica, Montenegro
ivan.jovovic@udg.edu.me

Srdjan Krčo, Petar Knežević
DunavNET
Novi Sad, Serbia

***Abstract*— This paper describes research effort aimed at the use of machine learning, Internet of Things, and edge computing for a use case in health, mainly the prevention of the spread of infectious diseases. The main motivation for the research was the Covid-19 pandemic and the need to improve control of the prevention measures implementation. In the study, the experimentation was focused on the use of machine learning to create and utilize prediction models for face mask detection. The prediction model is then evaluated on the various platforms with a focus on the use on various edge devices equipped with a video camera sensor. Different platforms have been tested and evaluated such as standard laptop PC, Raspberry Pi3, and Jetson Nano AI edge platform. Finally, the paper discusses a possible approach to implement a solution that would utilize the face mask detection function and lays out the path for the future research steps.**

Keywords*— *artificial intelligence; Covid19 prevention, edge computing, face mask detection, image processing; Internet of Things; machine learning

I. INTRODUCTION

Edge computing assumes the processing of the data near the source instead of relaying to the servers at remote data centers. Internet of Things (IoT) was introduced to the community in late 90s for supply chain management [1], and then the concept of collecting data without human intervention was widely adopted in application domains such as health care, environment monitoring, transportation, etc. [2,3]. As of April 2021, the number of people on the Internet has grown a lot, i.e. approximately 60% of the world population is now online [4]. This is reflected on the amount data being produced every day. At the end of 2021, there was around 74 zettabytes of data, as estimated by Cisco Global Cloud Index. With this being said, it can be realized that some IoT applications require short response time, while a huge quantity of data could be created and may introduce a heavy load on communication and processing. Traditional Cloud computing, with processing servers located in remote data centers, may not be efficient enough to handle this kind of challenges. For example, one such application is autonomous cars. One gigabyte of data can be generated by a self-driving car every second, and it requires real-time processing for the car to make correct decision [5]. If all the data needs to be send to the Cloud for the processing, it could also introduce waiting for response for a long time. So, there is a variety of use cases where the data needs to be processed at the edge to get response in shorter time, and for the processing to be more efficient [6]. In recent years, especially due to the advancements in computing equipment and machine learning (ML), artificial intelligence (AI) is finding its practical uses in variety of applications that employ IoT, edge computing, and big data analytics [7].

Edge computing provides numerous of benefits to the information technology. Some of those benefits are: reduced latency, improved bandwidth, longer lifespan, etc. [8]. In this research, we focused on the use of edge computing device called Jetson Nano, which is a small, powerful computer designed for entry-level edge IoT and AI applications. Jetson Nano is an Nvidia product aimed at implementation of IoT solutions with the power of GPU computation [9]. This device has GPIO (General Purpose Input/Output) pins and GPU core to help users build programs and solutions quite easily. The Nvidia Jetson Nano was introduced to public in mid-March of 2019 and this product was intended for IoT makers. Since it is said that Jetson Nano is a small general-purpose computer, it can be treated as an everyday computer. It can be used to browse internet, write documents, print, etc. In order to act like real computer user needs to provide keyboard, mouse and monitor. However, the designated use of this device is to be used in building various IoT solutions, where edge processing is needed. Sensors and additional modules can be added directly to a Nano board. The board provides GPIO interface that is used for connecting external device modules. Programming platforms based on modern programming languages such as Python and C++ are supported by Nvidia Jetson Nano software image containing the system software and development environment. Since Jetson's system software image is based on Ubuntu operating system, it supports installation of wide range of standard Linux programs, compilers, databases, etc. Nvidia Jetson Nano is equipped with a GPU containing 128 cores that enables implementation of edge AI applications [10]. Speaking of AI and machine learning software tools, Jetson can run Python with Pandas, NumPy, TensorFlow, and other similar tools. As such, Jetson Nano can be used for various computer vision applications in the AI and IoT edge setup. It just needs to be connected with external

This research was funded in part by the EUROCC project, European High-Performance Computing Joint Undertaking (JU) under grant agreement No. 951732. The JU receives support from the EU's Horizon 2020 research and innovation programme and EUROCC project participating institutions.

camera sensor and equipped with valid program scripts and ML trained prediction models.

In the study, the research was focused on the use of machine learning to create and utilize prediction models for face mask detection that can then be ported to an edge AI/IoT device equipped with camera in order to create a face mask detection system. Such a system could find its use in prevention of the spread of infectious diseases.

II. Materials and Methods

A. IoT/AI Edge Platform

As for the computing IoT/AI platform, we selected Nvidia Jetson Nano system [10]. This is a powerful computing board, but it does not have an internal storage, and because of that an external storage is needed. The most common storage that is finding its use with this system is MicroSD card with the minimum of 16GB, but 32GB is used for this example. This storage is used for operating system, application and/or service software, and data. In order to prepare the system, a PC computer with MicroSD card reader/writer is needed in order to prepare the software setup configuration and fully enable Nvidia Jetson Nano for edge computing. In addition, some hardware requirements such as keyboard, mouse, Ethernet cable, Power adapter, HDMI cable and monitor are needed for the first boot and the development process. Later, once everything is developed and configured, Jetson Nano can be used in headless mode.

The operating system can be obtained in ISO file format from the official Jetson Nano pages at the Nvidia website., where the latest version of JetPack software development kit (SDK) [11]. After the ISO file has been downloaded it is used to flash MicroSD card. Balena Etcher software tool can be used for this step. Once the MicroSD card is prepared, it is installed into the dedicated port, and configuring process of the Jetson Nano board can start. For the configuration and development, the system is connected to keyboard, mouse, and monitor, and it is used as a regular PC. The remaining process of the configuration is mostly straightforward, and it includes setting up the user's personal information and preferences.

The implementation architecture assumed for this study is illustrated in Fig. 1. For ML process, we selected the Faster R-CNN network as it will be explained in the next section. Resulting ML prediction model is ported to Jetson Nano that processes images obtained from the camera.

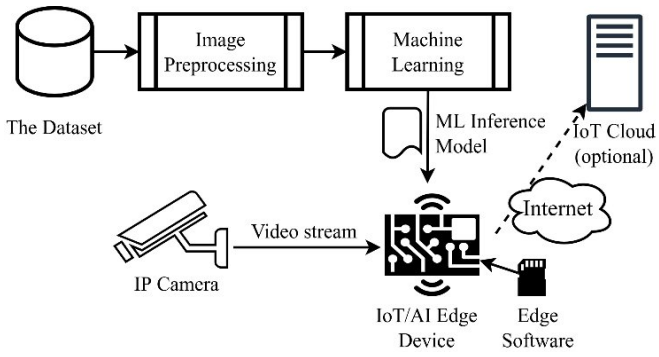

Figure 1. IoT/AI Edge Platform Setup for Face Mask Detection

The dataset of images with people wearing or not wearing face masks with corresponding annotations is used for creation and training of inference models. Prior to the use for ML process, the data in the data set is preprocessed in order to augment and enhance the dataset to get better results. After the successful ML session, the output ML inference model can be ported to the IoT/AI edge device, in this case Jetson Nano. The camera sensor can then be used to obtain video stream, i.e., real-life images aimed at locations we expect to find people wearing face masks. The inference model is used to process the input images. The result of this process can be an action or alert performed locally where the device is installed. Optionally, the results can be communicated via Internet to a remote IoT data platform in the cloud, and integrated into different applications that require face mask detection.

B. The Dataset Selection and Preparation

The dataset for machine learning in this study is obtained from Kaggle-s open-source database. This dataset contains 853 images and prior to their use, these images were further preprocessed in order to gain a richer dataset [12]. The dataset contained only partially labeled images and it was needed to label all the images according to the desired target function of the prediction model, in our case face mask detection. Computer Vision Annotation Tool (CVAT) is tool that was used for labeling images in this example.

As mention, in order to create a richer dataset, the Roboflow's augmentation tool was used to triple our previously labeled dataset. This tool is rotating existing images and labels in two other directions, so after this step, the dataset contained 2559 images out of which 2029 were used for training, 270 were used for validation and 260, were for the model testing, which was approximately 80/10/10 split.

C. Methods and Tools

The main tools used in this example were: Detectron2, PyTorch, Google Colab and OpenCV [7]. The experimentation, training and testing of the models, was performed both locally on PC workstation and remotely using Google Colab environment. The PC workstation specifications include Intel I7 CPU, Nvidia GeForce RTX 2060 GPU card, and 16GB of RAM. Training the model locally would take up to 180 minutes, while training of the model using Google Colab would take about 160 minutes. Figure 2 provides visualization of how the model training process went.

The image classification was done with the use of a Faster R-CNN machine learning algorithm. The Faster R-CNN is a class of deep learning neural networks which is and upgrade to previous R-CNN algorithm. Faster R-CNN algorithm contains multiple improvements and innovations implemented to speed up training and testing process, while also increasing detection accuracy [13]. More details about Faster R-CNN can be found in [13,14]. Detectron2 is software library used for purpose of building the model. Detectron2 is Facebook's next generation library that enables various detection and segmentation algorithms. The previous versions were Detectron and maskr-cnn benchmark. Detectron2 supports multiple computer vision projects and applications on Facebook. The latest version of the

platform is implemented in PyTorch. Detectron2 is flexible and extensible and it is able to provide fast training on single and multiple GPUs. The software library includes a variety of network models such as Faster R-CNN, Mask R-CNN, RetinaNet, Panoptic FPN, TensorMask, etc. Models built using Detecron2 can be exported in TorchScript format or in Caffe2 format for the later use in production deployment. The face detection mask machine learning experiment described in this paper was based on the adaptation of the official demonstration scripts for Detectron2 software library [15].

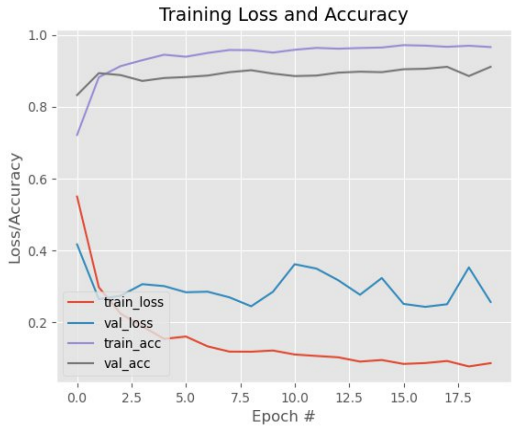

Figure 2. Visualization of the model training process

D. Installation and System Configuration

Conceptualized as an AI/IoT edge device, the Jetson Nano can compute multiple neural networks in parallel for AI applications such as image classification, object detection, object segmentation, and speech processing. It is designed to provide powerful computing resources in a small package. In comparison with other similar devices, it has shown far more better results, for example, the comparison with a well-known Raspberry Pi platform is shown in Table 1 [16]. Here, the Jetson Nano is compared with Raspberry Pi3 based on how many frames per second (FPS) can each device compute using different types of machine learning models.

In order for everything to run on the edge devices, it is needed to make sure that all of the required software components are installed properly. In addition to the operating system and Python, we needed Detectron2 software library. As Detectron2 was released recently, the process of installation on Jetson Nano was not as straightforward as on the other platforms. Jetson Nano CPU is a quad-core ARM A57 64-bit CPU (aarch64) and most of the software needs to be built from the source. The first step is to make sure that the latest version of JetPack SDK is installed with adequate CUDA version. After that, it is needed to install PyTorch, TorchVision and Cython, and these tools need to be installed in that order. TorchVision has to be compatible with Pytorch aarch64 version. Next step is to installation of PyCoco tools and pyyaml. PyCoco tools and PyYaml are needed for importing the input images and visualizing the output image. With all the previously mentioned software components installed, the installation of Detectron2 can take place. Detectron2 can be cloned and installed from its official GitHub repository. This process can take up to 3 hours depending on Internet speed. There is a suggestion to install everything inside virtual environment in case of something has gone wrong during the installation. Detailed information about versions can be found in [17]. Another way of installing Detectron2 on Jetson Nano is by pulling and configuring Docker container provided by Detectron2 official page. Since both, Jetson Nano and Detectron2 are relatively new, there are not many documented projects that involve both of these tools together, and this is the reason why the installation can be challenging.

TABLE I. COMPARISON JETSON NANO VS. RASPBERRY PI 3

Model	Application	Framework	Jetson Nano	RPi 3
ResNet50 (224x224)	Classification	TensorFlow	36 FPS	1.4 FPS
MobileNet-v2(300x300)	Classification	TensorFlow	64 FPS	2.5 FPS
SSD ResNet (960x544)	Object detection	TensorFlow	5 FPS	DNR
SSD ResNet (480x272)	Object detection	TensorFlow	16 FPS	DNR
SSD ResNet (300x300)	Object detection	TensorFlow	18 FPS	DNR
SSD MNet (960x544)	Object detection	TensorFlow	8 FPS	DNR
SSD MNet (480x272)	Object detection	TensorFlow	27 FPS	DNR
SSD MNet (300x300)	Object detection	TensorFlow	39 FPS	1 FPS
Inception 4 (299x299)	Classification	PyTorch	11 FPS	DNR
Tiny Yolo (416x416)	Object detection	Darknet	25 FPS	0.5 FPS
VGG-19 (224x224)	Classification	MXNet	10 FPS	0.5 FPS

III. RESULTS AND DISCUSSION

In this study, the focus of the experimentation was to create inference models and then port them into the AI edge platform in order to integrate everything into the proposed implementation architecture. Such a system could be further developed into a production level for the use in face mask detection alert system. The evaluation of the accuracy of the inference model was not in the focus as it can be improved by using enhanced and additional datasets. The experimental setup was evaluated using an actual video stream recording at the university entrance (University of Donja Gorica, Podgorica), and the initial results and performance of the model was satisfactory. As illustrated in the example shown in Fig. 4, it can be seen that model managed to detect 80% of people wearing masks with accuracy of 70-90%. Such accuracy was expected, having in mind the size and quality of the dataset, and the accuracy of the model could be improved.

As for the timings, we tested the end-to-end script on the system several times in a row, and during these tests we obtained the similar results. It took around 52 seconds on average to process the predict function based on the created inference model using Detectron2. It took a bit more time then expected to process the script from the start to the end, around 2 minutes in each test. The main reason for this additional time and difference between computing prediction function and execution of the complete script is that the end-to-end script

was programmed to draw the boundary boxes around detected faces/masks, and then save output picture, as illustrated in Fig. 3. This functionality can be optimized in the future.

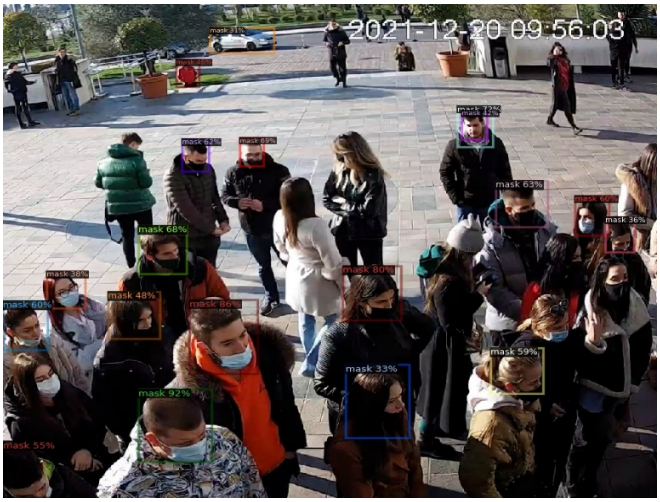

Figure 3. An example output from the proposed system (UDG, Podgorica)

The other way of evaluation of the performance is in measuring and representing behavior of different hardware parts of Jetson Nano as illustrated in Table 2. The hardware parts include CPU and GPU in this case. Since the model was based on GPU computation completely, the result of 98% GPU usage was as expected, while CPU was used only around 40% during the time our model was working. The temperature of the device was around 45°C degrees during the execution of the testing script. These results were acquired using Jetson Nano statistic software. The results from similar project on Jetson Nano were more-less like our results, but in this case, author used YOLOv4 for detection. [18].

TABLE II. PERFORMANCE RESULTS

	Test 1	Test 2	Test 3
CPU [%]	42	39	41
GPU [%]	98	98	98
Temperature [C]	44	46	45
RAM	2.1	2.1	2.1
Power [W]	6.2	6.1	6.3

IV. CONCLUSIONS

This paper describes experimentation with AI edge platform and proposes a solution for computer vision system aimed at face mask detection and prevention of infectious disease spreading. Such a system could be used in health care institutions, but also in public spaces, shopping malls, schools and universities, etc. The paper discusses the selection of the IoT/AI edge device platform based on the Jetson Nano embedded system on module board, and all of the software tools needed to create and implement such a system. The inference model was created using publicly available dataset, Detectron2 software library and Faster R-CNN network model. The initial results are promising and such an approach to implementing a face mask detection system based on the proposed architecture and selected tools is possible.

The next steps for future research will include selection of additional datasets, experimentation with various machine learning parameters, aiming for better accuracy of the prediction model. We foresee the use of Google Colab and HPC resources for these experiments. As for the system implementation and IoT/AI edge platform, the solution needs a better power management and the script for image pre- and post- processing can be optimized. The device will need better cooling using additional heat sink and adding a fan. The combination of a camera sensor and this edge device with well-trained inference model, can result in a new AI edge computer vision sensor that can be integrated into various IoT and disease prevention solutions.

REFERENCES

[1] K. Ashton, “That ‘internet of thing’ thing,” RFiD journal, vol. 22, no. 7, pp. 97-114, 2009.
[2] H. Sundmaeket, P. Guillemin, P. Friess and S. Woelffie, “Vision and challenges for realising the Internet of Things,” 2010.
[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of Things (iot): A vision, architectural elements, and future directions,” Future Generation Computer systems, vol. 29, no. 7, pp. 1645- 1660, 2013.
[4] S. Kemp, “60% of the world’s population is now online”, We are Social, April 2021.
[5] R. Cheruvu, “Big data Applications in Self-Driving cars”, Harvard University, pp. 1-4, 2016.
[6] Jie Cao, Quan Zhang, Weisong Shi, “Edge computing- a primer”, Springer Intermational Publishing, pp. 5-8, 2018.
[7] M.Merenda, C.Porcaro, D.Iero, “Edge machine learining for AI-enabled IoT devices: A review”,pp. 2-5, 2021.
[8] Shijun Lio, Bedir Tekinerdogan, Mikio Aoyama, Liang-Jie Zhang, “Edge Computing- EDGE 2018”, Springer International Conference, pp. 7-9, 2018.
[9] A. Kurniawan, “IoT projects with Nvidia Jetson Nano”, pp. 4-12, 2021
[10] Jetson Nano Developer Kit, [Online] Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit
[11] ISO file, [Online] Available: https://developer.nvidia.com/embedded/develop/software
[12] Face-Mask dataset”, [Online] Available: ,https://www.kaggle.com/andrewmvd/face-mask-detection.
[13] Ren Shaoqing, “Faster r-cnn: Towards realtime object detection with region proposal networks”,pp. 2-4, 2015
[14] Girshick Ross, “Fast r-cnn”, Proceedings of the IEEE international conference on computer vision,2015.
[15] “How to train detectron2 with custom dataset”, [Online] Available: https://colab.research.google.com/github/Tony607/detectron2_instance_segmentation_demo/blob/master/Detectron2_custom_coco_data_segmentation.ipynb
[16] “Jetson Nano Deep learning inference benchmarks”, [Online] Available: https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks
[17] “Install and run Detectron2 on Nvidia Jetson Nano”, [Online] Available: https://che-adrian.medium.com/install-and-run-detectron2-on-nvidia-jetson-nano-1c66b522598d
[18] S. Valladares, M.Toscano, R. Tufino, D. Vallejo, “Performance Evaluation of the Nvidia Jetson Nano through a real-time machine learing algorithm”, pp. 4-5, 2021.