Pynq-YOLO-Net: An Embedded Quantized Convolutional Neural Network for Face Mask Detection in COVID-19 Pandemic Era

Yahia Said
Electrical Engineering Department, College of Engineering
Northern Border University, Arar, Saudi Arabia[1]
Laboratory of Electronics and Microelectronics (LR99ES30)
Faculty of Sciences of Monastir, University of Monastir, TUNISIA[2]

***Abstract*—The recent Coronavirus COVID-19 is a very infectious disease that is transmitted through droplets generated when an infected person coughs, sneezes, or exhales. So, people must wear a face mask to reduce the power of the transition of this virus. Governments around the world have imposed the use of face masks in public spaces and supermarkets. In this paper, we propose to build a face mask detection system based on a lightweight Convolutional Neural Network (CNN) and the YOLO object detection framework to implement it on an embedded low power device. The object detection framework was designed using a single Convolutional Neural Network for object detection in real-time. To make the YOLO framework suitable for embedded implementation, we propose to build a lightweight Convolutional Neural Network and quantize it by using a single bit for weight and 2 bits for activations. The proposed network called Pynq-YOLO-Net was implemented on the Pynq Z1 platform. The computation was divided between the software and the hardware. The features extraction part was executed on the hardware device and the output part was executed on the software. This configuration has allowed reaching real-time processing with a very good detection accuracy of 97% when tested on the combination of collected datasets.**

***Keywords*—Face mask detection; Coronavirus COVID-19; YOLO; Convolutional Neural Network (CNN); embedded device; Pynq Z1 board**

I. Introduction

According to the World Health Organization (WHO) [1], the COVID-19 is causing a world crisis because of its fast infection and the absence of a cure. This new virus is considered as the fastest infecting virus all over time. Based on the latest statistics [2], more than 25 million in 190 countries are infected, and more than 844000 deaths, until the writing of this paper. As a protection step, the authorities oblige people to wear face masks in public spaces to reduce the transmission impact of the virus. Many peoples are ignoring the rules and do not wear face masks. So, it is important to detect those peoples that do not use facemasks and warn them about the importance of this step to stay uninfected by the coronavirus. Thus, an automatic face mask detector must be installed in public streets, supermarkets, and all public service agencies. Based on surveillance systems of all public spaces, it is possible to process visual data and detect peoples that do not use face masks.

Recently, the performance of computer vision applications has been boosted to high-level thanks to the use of deep learning [3]. The deep learning is based on a deep neural network with tens of hidden layers. Deep learning models can learn directly from input data without any handcrafted features. For image processing, the Convolutional Neural Network (CNN) is the most used. It was inspired by the biological nervous system, and based on mathematics and informatics representation, it mimics the vision cortex of an animal. The CNN was successfully deployed to solve many computer vision applications such as traffic light detection and recognition [4], [5], medical image segmentation [6], indoor object detection and identification [7], [8], scene identification [9], face detection and recognition [10].

CNN models are characterized by their high performance and intensive computation. The feature extraction part (convolution layers, activation layers, pooling layers) uses the most computation effort and the output part (fully connected layers) uses the most of memory storage. Until today, Graphical Processing Units (GPU) are considered as the best target platform for the deployment of CNN. But GPUs need a lot of power and expansion. At this end, CNN must be optimized for low power platforms such as Field-Programmable Gate Arrays (FPGA). Many techniques were proposed to optimize CNN for low power implementation. The quantization technique is a very useful technique which aims to reduce the number of bits used for the representation of the weights and the activations on CNN. Many works have been proposed in this context with different methodologies. Doyun et al. [11] proposed a quantization algorithm based on the generalized gamma distribution. The proposed algorithm was tested with different representation and the achieved result were courageous. As reported in [11], the performance of the algorithm can be improved by tuning the parameters of the quantizer. A Kernel Density Estimation based Non-uniform Quantizer was proposed in [12]. In this work, a 4-bits representation of the weights and activations were used. The proposed quantization algorithm was tested on the ImageNet dataset and it was very effective in compressing the model without a big loss in performance. In [13], a quantization

algorithm based on trainable scaling factors and a nested-means clustering strategy was proposed. To quantize the weights, the nested-means clustering strategy was deployed to achieve high parameter compression. To quantize the activations, a linear quantization technique was used which take into account the statistical priorities of the batch normalization technique. There are many variants of the quantization technique and each of them has its impact on the model compression.

In this work, we propose a lightweight Convolutional Neural Network and quantize it for implementation on an edge device. The proposed CNN is composed of a convolution layer, 3 lightweight blocks, and a regression layer for output. The detection technique is based on You Look Only Once (YOLO) framework [14] which is designed to achieve real-time processing with good detection accuracy. The YOLO framework treats the detection task as a regression problem. For our case, it is perfect since we look for detecting if the person is wearing a face mask or not. That is a binary classification problem with a focus on the prediction of the bounding box used to locate the face mask. So, to speed up the processing, we ignored the classification and we focus on the localization task. The proposed network was called Pynq-YOLO-Net.

The proposed CNN was tested on the Pynq board which a hybrid board (software/hardware) equipped with an ARM processor and an FPGA in a single ship. This configuration allows taking the advantage of the FPGA blocks alongside the CPU. The Pynq board can be programmed using a high-level programming language (Python) or hardware description language (VHDL/Verilog).

The motivation behind optimizing the CNN for embedded implementation is to make it available for all surveillance systems without the need for high-performance computers and to reduce the power consumption of those systems. Also, it can be implemented in mobile devices such as smartphones and smart cameras.

The main contributions of this work are the following: (1) design a lightweight Convolutional Neural Network targeting embedded device; (2) the proposed CNN was quantized to fit in the Pynq board; (3) implementation of the proposed CNN inference for face mask detection on the Pynq board.

The rest of the paper is organized as follows. Section 2 was reserved to discuss related works about face mask detection methods. The proposed approach was described and discussed in Section 3. In Section 4, the experiment and results were presented and discussed. The paper was concluded in Section 5.

II. Related Works

Recently, the detection of the face mask was an important application for reducing the transmission of the COVID-19. Building an automatic face mask detector is a challenging task and many works were proposed to achieve high results.

Loey et al. [15] proposed a hybrid system for face mask detection. The proposed system is composed of a CNN for feature extraction and decision trees, Support Vector Machine (SVM), and an ensemble algorithm for the detection. The transfer learning technique was applied to the ResNet 50 model [16] to finetune it for face mask detection. The proposed system was trained and evaluated on 3 datasets, the Real-World Masked Face Dataset (RMFD) [25], the Simulated Masked Face Dataset (SMFD) [25], and the Labeled Faces in the Wild (LFW) [26]. The proposed system has achieved a high accuracy of more than 99% but it was very complex and hard to train. In addition, the proposed system is computationally intensive and cannot be used for real-time processing.

The Single Shot Multi-box Detector (SSD) [17] was proposed for face mask detection in public spaces [18]. The SSD model was pre-trained on the MSCOCO dataset for object detection and finetuned on a custom-made dataset for face mask detection. The MobileNet V2 model [19] was used as a backbone for the SSD to limit the computation complexity. The proposed model was implemented on a Raspberry PI 4 equipped with a quad-core ARM processor and 4GB of RAM. An accuracy of 85% was achieved when testing the model on the custom-made dataset. This work was a good step for implementing facemasks on embedded devices. But the Raspberry PI 4 is considered as a software device and its power consumption is too high compared to low power devices.

Jiang et al. [19] proposed the use of the RetinaNet model [20] for face mask detection. The RetinaNet was finetuned for face mask detection trough the transfer learning technique. Two backbones were tested, the Resnet and the MobileNet. In addition, a new technique was added to the RetinaNet to reject predictions with low confidences and the high intersection of a union. The RetinaNet was pre-trained on the ImageNet [21] dataset and then fine-tuned on the face mask dataset. The proposed RetinaFaceMask has achieved good results with both backbones while the best results were achieved using the ResNet model. The achieved result was good but the RetinaFaceMask was not suitable for implementation on low power devices because of its computation intensively and the need for large storage memory.

IN [22], a CNN model was proposed to detect if a person wears a face mask or not. Also, the proposed network was used to detect if the mask is correctly worn or not. The proposed CNN has a simple architecture with a convolution layer, an activation layer, a pooling layer, a fully connected layer, and a softmax layer. The proposed approach was designed to detect faces and face masks separately. The proposed CNN model was trained using publicly available datasets, Masked Face Detection Dataset (MFDD) [25], Real-world Masked Face Recognition Dataset (RMFRD) [25], and Simulated Masked Face Recognition Dataset (SMFRD) [25]. The reported results are good in terms of accuracy and speed. The main disadvantage of the proposed model is the need for high-performance computers and large memory usage.

All the mentioned methods are designed to be implemented on a high-performance computer with a very high-power consumption. In this work, we propose to quantize a lightweight CNN for implementation on low power devices with a focus on high performance. In the next section, we will

present the proposed approach and detailing the different optimization applied to achieve an embedded implementation.

III. Proposed Approach

In this section, we will describe the proposed lightweight CNN model and the compression techniques applied to make this model fit in the resource constraint of a low power device while maintaining high performance and real-time processing.

Recently, many techniques are proposed to build lightweight CNN models. The most important technique in the use of Bottlenecks instead of normal convolution layers. In this work, we adopt the Bottlenecks concept proposed by the MobileNet v2 model [19]. The main contribution of the Inverted Residuals and Linear Bottlenecks is the use of depthwise convolution and point convolutions instead of simple convolution layers with adding a residual connection. The depthwise convolution is similar to the normal convolution layer but the main difference that depthwise convolution reserves the number of channels and does not compress them. For normal convolution it the number of input channels is n then the number of output channels is 1 but for depthwise convolution, if the number of the input channels is n then the number of output channels is n. The pointwise convolution is a normal convolution layer with a kernel size of 1x1. The combination of a depthwise convolution with a pointwise convolution makes the same functionality of a normal convolution layer but 9 times faster as they claim [19]. Also, the separation of the filtering and combining functionalities allow the implementation of more than one activation layer and batch normalization layer which results in enhancing the performance of the model and reducing the computation complexity. The proposed Inverted Residuals and Linear Bottlenecks are presented in Fig. 1. Another technique was deployed by the MobileNet model was the use of strided convolution layers and eliminate the use of pooling layers. As proves in [23], the use of strided convolution layers instead of max-pooling layers is more efficient to build CNN models for embedded implementation and helps to enhance the accuracy of those models.

In this work, we propose to use a convolution layer with a kernel size of 3x3 and a stride of 2, three inverted residual bottlenecks, and three linear bottlenecks as a backbone for the YOLO framework. The YOLO framework takes an input image, applies a feature extraction through a backbone based on a CNN model to generate an output grid of NxN dimension. For each cell of the obtained grid, it predicts only one object with the parameters of the bounding box (the x, y coordinates, the height, the width, and the confidence score) and the class probability. For the Pascal VOC dataset [24] the YOLO framework generated 7x7 grid and used 2 bounding boxes (B) for 20 classes (C). The architecture of the YOLO framework is presented in Fig. 2.

In this work, we propose to eliminate the calculation of the class probability because we are solving a binary classification problem. The YOLO framework computes the score confidence for each predicted bounding box. Since there is one object to detect, we consider the confidence score the class probability. This step allows reducing the calculation at the output layer and speeds up the processing speed.

Besides, more optimizations were applied to the YOLO framework to reduce the computation complexity and to enhance the performance. First, the input image was resized to a power of 2 sizes. Thus, we propose to use 128x128 size. After applying the features extraction module, the YOLO framework generates 8x8 grid. Using an input image with a size power of 2 facilitates the implementation of the convolutional layers on the hardware device because it is easier to perform multiplications by using only shifting registers. Second, the use of an all convolution backbone allows enabling the data reuse technique to reduce the communication between the external memory and use only the on-chip memory to compute the convolutions. Third, the pruning technique was applied to eliminate weak connections and to avoid the overfitting problem in the finetuning step. Finally, the model was quantized by replacing 32 bits floating-point representation by a 2-bits fixed-point representation for the activations (A) and 1-bit fixed-point representation for the weights (W). For the input image and the output layer (grid), we used an 8-bits fixed-point representation. The backbone architecture and configuration were presented in Fig. 3.

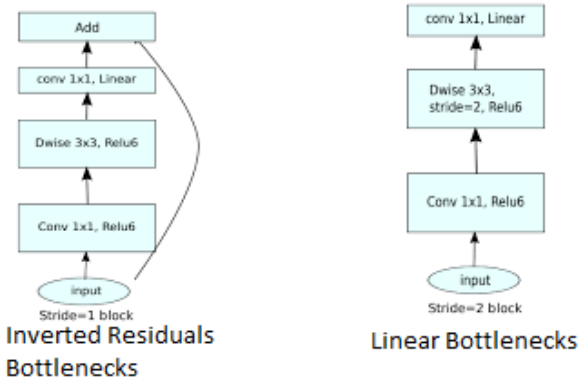

Fig. 1. Inverted Residuals and Linear Bottlenecks.

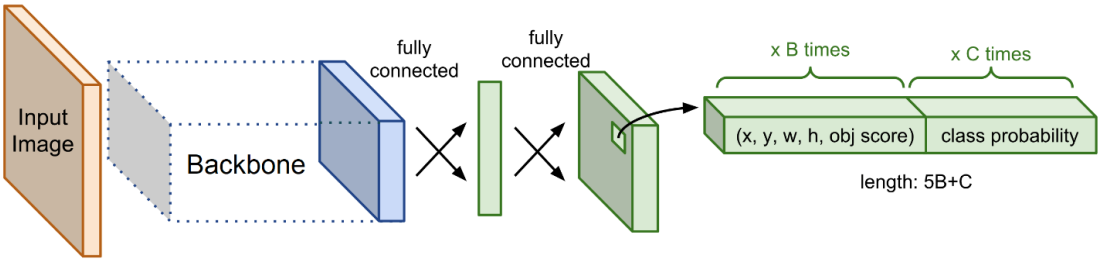

Fig. 2. YOLO Architecture.

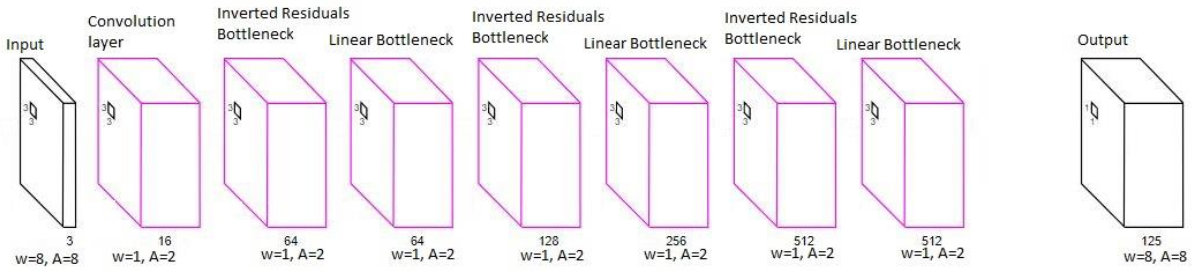

Fig. 3. Backbone Architecture for Embedded Implementation.

In this work, we propose to use the post-training quantization technique which is based on training and fine-tuning after reducing the presentation of the weights and activations. The YOLO was trained using 32 bits floating-point representation then it was reduced to the proposed representations and retrained again to recover the accuracy. In the experiments, we will report the accuracies obtained with different representations. The retraining process is very important to recover the degradation of the accuracy caused by the quantization technique.

The workflow of the proposed approach is divided into 5 steps. The first step is to develop the proposed model based on the YOLO framework. The second step is to train the proposed model using a specific dataset. The third step is to optimize the model for embedded implementation by applying the pruning technique and the quantization technique. The fourth step is the retraining of the model on the same dataset used in step 2 to recover the accuracy degraded by the optimization techniques. The final step is to implement the model on the Pynq Z1 board. The workflow of the proposed approach is illustrated in Fig. 4.

IV. Experiments and Results

A. Training Data

To train the proposed model, we proposed to combine publicly available datasets to increase the amount of training data. The performance of CNN models increases with the amount of training data. The used datasets are presented in the following:

- Real-World Masked Face Dataset (RMFD) [25]: The images of the dataset were collected automatically from the internet for public famous figures with and without a mask. The dataset contains 5,000 images of 525 persons wearing masks, and 90,000 images of the same 525 persons without masks. The dataset was manually filtered and annotated using semi-automatic labeling tools.
- Masked Face Detection Dataset (MFDD) [25]: The dataset was designed for the detection of masked faces during the era of the coronavirus. This dataset combines existing face detection datasets with images collected from the internet. The collected images were manually annotated where the coordinate of the face with the mask was defined in addition to the condition of wearing a mask or not. It contains 24771 images for masked faces.
- Simulated Masked Face Recognition Dataset (SMFRD) [25]: to increase the amount of training data for face mask detection, an automatic wearing tool was designed to add masks to faces of existing face detection and recognition datasets such as LFW [26] and Webface [27] then the collected data was added to the MFDD. This dataset allows adding 500000 annotated faces of 10000 persons to the MFDD.
- MAsked FAces (MAFA) dataset [28]: it is a dataset that contains 30,811 images and 34,806 labeled masked faces. The dataset contains faces masked by the medical mask and others masked hand or other objects. This allows enhancing the generalization power by distinguishing between the mask that it must be detected and other masks.

The mentioned datasets were combined to build a very large dataset to increase the training data. Thus, it will enhance the performance of the trained model. Fig. 5 present examples of images from the collected datasets used for the training of the model.

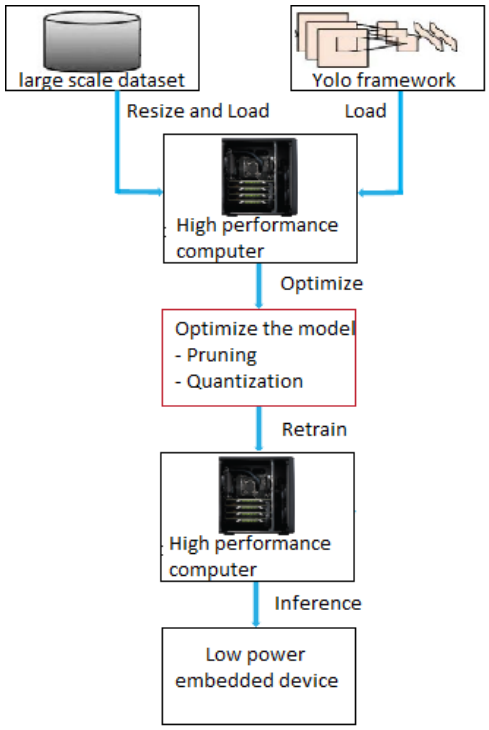

Fig. 4. Workflow of the Proposed Approach.

Fig. 5. Examples of Images from the Collected Datasets.

B. Training and Evaluation

The proposed model was trained on the combination of the collated datasets using the gradient descent algorithm. The Adam optimizer was used as a learning algorithm which is a variant of the gradient descent algorithm with many advantages. The Adam Optimizer optimizes the weights and the learning rate accordingly to achieve a better minimum of the loss function.

To evaluate the proposed Pynq-YOLO-Net, we propose to use the precision and the recall as evaluation metrics. The precision presents the percentage of relevant results and the recall presents the percentage of relevant results correctly identified. The precision and the recall are computed as (1).

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$Racall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad (1)$$

The performance of the Pynq-YOLO-Net was evaluated on the testing set which is 30% of the collected data. The Pynq-YOLO-Net achieved a Precision of 94.6% and a Recall of 95.8% for the first training process using the 32-bits floating-point representation. After compressing the model and retrained it on the same data, it achieves an accuracy of 90.7% of Accuracy and 92.3 of Recall. To find our model in the state-of-the-art, we compared against existing works. Table I present a comparison against state-of-the-art works. As shown in Table I, the proposed Pynq-YOLO-Net achieved better results than state-of-the-art works in its normal version. The compressed model achieves lower results but with the advantage of implementation on low power devices. The achieved results still good enough to generate trusted predictions.

C. Inference

The inference of the Pynq-YOLO-Net was implemented on the Pynq Z1 board. The board was connected to the internet and it was connected via an ssh node to visualize the results on the computer screen. The Pynq Z1 board is presented in Fig. 6. An operating system based on Linux kernel was loaded to the board via a pre-booted SSD card. The implementation of the Pynq-YOLO-Net on the Pynq Z1 board achieves a processing speed of 16 FPS. The achieved result can be considered as real-time processing speed. The energy consumption of the board does not exceed 5 watts.

The implementation of the proposed model on the Pynq Z1 board was divided into parts. The first part composed of the feature extraction, which is composed of the convolution layer and the bottlenecks, was implemented on the hardware to take advantage of the parallel processing of the programmable units.

The second part which is composed of the fully connected layers and the output layer, was implemented on the software because it needs more memory and less computation. The performance of the board is presented in Fig. 7.

The Pynq-YOLO-Net was tested using images that does not belong to the collected datasets to evaluate the generalization power of the model. Fig. 8 present an illustration of the obtained results. The model was very effective when tested on new images which prove that it have a good generalization power.

TABLE I. COMPARISON AGAINST STATE-OF-THE-ART WORKS

Model	Precision (%)	Recall (%)
RetinaFaceMask [19]	93.4	94.5
SSD [18]	91	91
Pynq-YOLO-Net (ours)	94.6	95.8
Pynq-YOLO-Net compressed (ours)	90.7	92.3

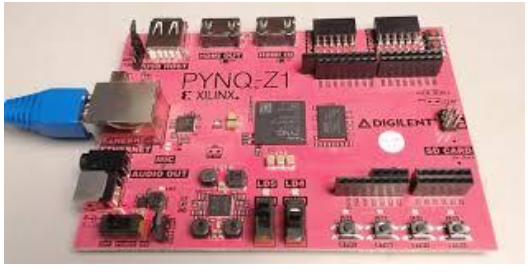

Fig. 6. Pynq Z1 Board.

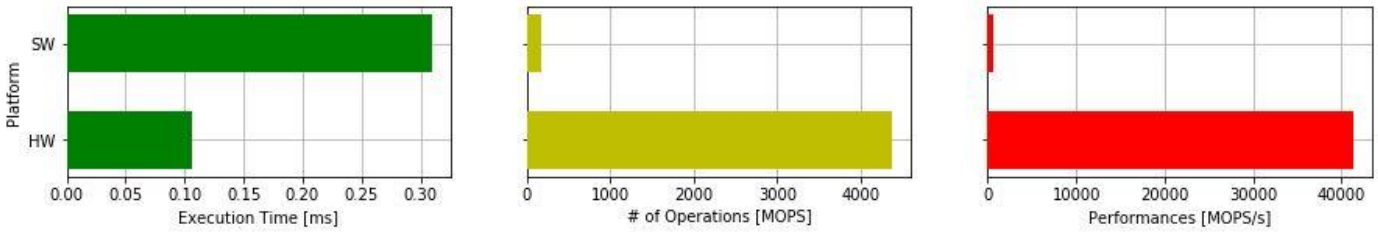

Fig. 7. Performance of the Pynq Z1 Board.

Fig. 8. Visualization of the Obtained Results of the Pynq-YOLO-Net.

D. Discussion

The reported results prove the efficiency of the proposed Pynq-YOLO-Net for implementation in low power devices. Starting by building a lightweight CNN is a very important step to reach embedded implementations. Also, the YOLO framework was a good choice since it was designed with a focus on speed. The model compression techniques used in this work allow to reduce the size of the model and speed up the processing speed without damaging the accuracy. The choice of the size of the input images was very effective for building the convolution layers on the hardware part of the board. All those factors were correlated together to achieve an embedded implementation of the proposed model.

V. Conclusions

The coronavirus COVID-19 is a very fast-spreading disease. It is important to protect ourselves from being infected by wearing masks and respecting social distances in public environments. In this paper, we propose to build a face mask detector in public spaces to detect if people are wearing masks or not. The proposed detector was based on the YOLO framework with a lightweight backbone. The proposed model, called Pynq-YOLO-Net, was designed to be implemented on the Pynq Z1 board. To achieve this implementation, some model compression techniques was applied such as pruning and quantization. Those techniques were very effective to reduce the model size and the computation complexity. The model was implemented on both hardware and software to accelerate the inference. The achieved performance has proved the efficiency of the proposed approach for mask detection in public spaces. As future work, the model will be implemented on video surveillance systems to be tested on real conditions.

Acknowledgment

The author wishes to acknowledge the help of Mr. Ayachi Riadh from Laboratory of Electronics and Microelectronics at University of Monastir for assistance with implementing the proposed model.

References

[1] Coronavirus disease (COVID-19) pandemic, Available at: https://www.who.int/emergencies/diseases/novel-coronavirus-2019 , accessed on 29/08/2020.

[2] COVID-19 CORONAVIRUS PANDEMIC, Available at: https://www.worldometers.info/coronavirus/? , accessed on 29/08/2020.

[3] Goodfellow, Ian, Aaron Courville, and Yoshua Bengio. Deep learning. Vol. 1. Cambridge: MIT press, 2016.

[4] Ayachi, Riadh, Mouna Afif, Yahia Said, and Mohamed Atri. "Traffic signs detection for real-world application of an advanced driving assisting system using deep learning." Neural Processing Letters 51, no. 1 (2020): 837-851.

[5] Ayachi, R., Y. E. Said, and M. Atri. "To perform road signs recognition for autonomous vehicles using cascaded deep learning pipeline." Artificial Intelligence Advances 1, no. 1 (2019): 1-58.

[6] Wang, Guotai, Wenqi Li, Maria A. Zuluaga, Rosalind Pratt, Premal A. Patel, Michael Aertsen, Tom Doel et al. "Interactive medical image segmentation using deep learning with image-specific fine tuning." IEEE transactions on medical imaging 37, no. 7 (2018): 1562-1573.

[7] Afif, Mouna, Riadh Ayachi, Yahia Said, Edwige Pissaloux, and Mohamed Atri. "An evaluation of retinanet on indoor object detection for blind and visually impaired persons assistance navigation." Neural Processing Letters (2020): 1-15.

[8] Afif, Mouna, Riadh Ayachi, Edwige Pissaloux, Yahia Said, and Mohamed Atri. "Indoor objects detection and recognition for an ICT mobility assistance of visually impaired people." Multimedia Tools and Applications (2020): 1-18.

[9] Afif, Mouna, Riadh Ayachi, Yahia Said, and Mohamed Atri. "Deep Learning Based Application for Indoor Scene Recognition." Neural Processing Letters (2020): 1-11.

[10] Sun, Xudong, Pengcheng Wu, and Steven CH Hoi. "Face detection using deep learning: An improved faster RCNN approach." Neurocomputing 299 (2018): 42-50.

[11] Kim, Doyun, Han Young Yim, Sanghyuck Ha, Changgwun Lee, and Inyup Kang. "Convolutional Neural Network Quantization using Generalized Gamma Distribution." arXiv preprint arXiv:1810.13329 (2018).

[12] Seo, Sanghyun, and Juntae Kim. "Efficient weights quantization of convolutional neural networks using kernel density estimation based non-uniform quantizer." Applied Sciences 9, no. 12 (2019): 2559.

[13] Schindler, Günther, Wolfgang Roth, Franz Pernkopf, and Holger Fröning. "N-Ary Quantization for CNN Model Compression and Inference Acceleration." (2018).

[14] Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

[15] Loey, Mohamed, Gunasekaran Manogaran, Mohamed Hamed N. Taha, and Nour Eldeen M. Khalifa. "A Hybrid Deep Transfer Learning Model

with Machine Learning Methods for Face Mask Detection in the Era of the COVID-19 Pandemic." Measurement (2020): 108288.

[16] He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.

[17] Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "Ssd: Single shot multibox detector." In European conference on computer vision, pp. 21-37. Springer, Cham, 2016.

[18] Yadav, Shashi. "Deep Learning based Safe Social Distancing and Face Mask Detection in Public Areas for COVID-19 Safety Guidelines Adherence." International Journal for Research in Applied Science & Engineering Technology (IJRASET). 2020.

[19] Jiang, Mingjie, and Xinqi Fan. "RetinaFaceMask: A Face Mask detector." arXiv preprint arXiv:2005.03950 (2020).

[20] Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In Proceedings of the IEEE international conference on computer vision, pp. 2980-2988. 2017.

[21] Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. "Imagenet: A large-scale hierarchical image database." In 2009 IEEE conference on computer vision and pattern recognition, pp. 248-255. Ieee, 2009.

[22] Inamdar, Madhura, and Ninad Mehendale. "Real-Time Face Mask Identification Using Facemasknet Deep Learning Network." Available at SSRN 3663305 (2020).

[23] Ayachi, Riadh, Mouna Afif, Yahia Said, and Mohamed Atri. "Strided convolution instead of max pooling for memory efficiency of convolutional neural networks." In International conference on the Sciences of Electronics, Technologies of Information and Telecommunications, pp. 234-243. Springer, Cham, 2018.

[24] Everingham, Mark, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. "The pascal visual object classes (voc) challenge." International journal of computer vision 88, no. 2 (2010): 303-338.

[25] Wang, Zhongyuan, Guangcheng Wang, Baojin Huang, Zhangyang Xiong, Qi Hong, Hao Wu, Peng Yi et al. "Masked face recognition dataset and application." arXiv preprint arXiv:2003.09093 (2020).

[26] Huang, Gary B., Marwan Mattar, Tamara Berg, and Eric Learned-Miller. "Labeled faces in the wild: A database for studying face recognition in unconstrained environments." 2008.

[27] Yi, Dong, Zhen Lei, Shengcai Liao, and Stan Z. Li. "Learning face representation from scratch." arXiv preprint arXiv: 1411.7923 (2014).

[28] Ge, Shiming, Jia Li, Qiting Ye, and Zhao Luo. "Detecting masked faces in the wild with lle-cnns." In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2682-2690. 2017.