Hindawi
Mathematical Problems in Engineering
Volume 2022, Article ID 8220677, 10 pages
https://doi.org/10.1155/2022/8220677

Research Article

Facial Mask Detection Using Image Processing with Deep Learning

Hongyu Ding,[1] Muhammad Ahsan Latif,[2] Zain Zia,[2] Muhammad Asif Habib,[3] Muhammad Abdul Qayum,[3] and Quancai Jiang[1]

[1]*College of Electrical Engineering and New Energy, China Three Gorges University, Yichang 443000, China*
[2]*Department of Computer Science, University of Agriculture, Faisalabad 38040, Pakistan*
[3]*Department of Computer Science, National Textile University, Faisalabad 37610, Pakistan*

Correspondence should be addressed to Muhammad Asif Habib; drasif@ntu.edu.pk

Received 26 February 2022; Revised 21 April 2022; Accepted 11 May 2022; Published 12 August 2022

Academic Editor: Nouman Ali

Coronavirus disease 2019 (COVID-19) has a significant impact on human life. The novel pandemic forced humans to change their lifestyles. Scientists have broken through the vaccine in many countries, but the face mask is the only protection for public interaction. In this study, deep neural networks (DNN) have been employed to determine the persons wearing masks correctly. The faster region-based convolutional neural networks (RCNN) model has been used to train the data using graphics processing unit (GPU) device. To achieve our goals, we used a multiphase detection model: first, to label the face mask, and second to detect the edge and compute edge projection for the chosen face region within the face mask. The current findings revealed that faster RCNN was efficient and precise, giving 97% accuracy. The overall loss after 200,000 epochs is 0.0503, with a trend to decrease. While the loss is falling, we are getting more accurate results. As a result, the faster RCNN technique effectively identifies whether a person is wearing face masks or not, and the training period was decreased with better accuracy. In the future, Deep Neural Network (DNN) might be used first to train the data and then compress the dimensions of the input to run it on low-powered devices, resulting in a lower computational cost. Our proposed system can achieve high face detection accuracy and coarsely obtain face posture estimation based on the specified rule. The faster RCNN learning algorithm returns high precision, and the model's lower computational cost is achieved on GPU. We use the "label-image" application to label the photographs extracted from the dataset and apply Inception V2 of faster RCNN for face mask detection and classification.

1. Introduction

COVID-19, a novel pandemic, posed a severe human pandemic threat to the entire world. This pandemic started in China at the end of 2019 and is probably one of the world's most significant health challenges this century. By January 2022, over 300 million confirmed COVID-19 cases were recorded, with almost 5.47 million deaths worldwide. Some countries' scientists enabled breaking through the vaccine in time. The research is ongoing by thousands of scientists worldwide to better understand how new virus mutations and variants like alpha, beta, gamma, delta, and omicron affect the effectiveness of the different COVID-19 vaccines. The COVID-19 pandemic has altered people's entire lifestyles. The patient has an acute lung infection for which there is currently no high-performance vaccine or treatment. Deaths are rapidly increasing, putting strain on each nation's healthcare systems [1].

Face masks have become an essential part of human life, and the only protection is the use of face masks while interacting with public people. In this challenging pandemic condition, numerous countries have mandated that citizens wear face masks when visiting any intense public spot like shopping malls taking a meal. The customer care service provider provides services to only those customers who wear a face mask correctly. The World Health Organization (WHO) has released recommendations and guidelines for the general public and healthcare professionals who use face masks. Face masks, according to medical officers, give sufficient protection against respiratory diseases. Nurses and doctors widely used face masks as the central part of droplet safety measures and precautions. Some argue that wearing a face mask will not protect you from COVID-19 infection. However, there is a significant distinction between absence and the absence of evidence [2].

We presented an accurate and efficient face mask detector algorithm in the present study. Our main task is to check whether or not people wear masks and stay away from public places using the proposed algorithm. It is a classification or object detection problem of two different classes wearing masks and not proper masks. There are several methods to detect objects, but we chose to utilize faster RCNN because of its fast, simple, accurate, and precise algorithm. We need to develop a system that could detect faces in this real world and recognize whether or not the detected faces have masks.

The paper is organized in the following fashion: Section 2 presents the literature review, and Section 3 represents the method's description. Section 4 is spare for the results we achieve in the present study, and the conclusion is given in Section 5.

2. Literature Review

Presently, the issue is connected to the general recognition of objects using deep learning and the detection of object classes [3]. A few researchers have been found to detect facial masks based on image analysis in the literature. Detection of the face or mask is one of the classes or groups of objects [4]. Detectors depend on deep learning structures rather than handcrafted features and have, in recent years, had outstanding performance due to their exceptional extraction robustness and capability. Face and object detection applications are utilized in education surveillance, autonomous driving, and several other fields [5, 6]. A surveillance camera's image processing technology could detect a person's face when not wearing a face mask.

Schneiderman and Kanade [7] proposed face scanners with characteristics shaped by a series of feature vectors trained using a view-based approach. The structure has been disclosed to enhance profile face detection accuracy. A detailed collection of features, similar to Haar, was projected, with rectangular features rotated to 45 degrees by Lienhart and Maydt [8]. He added another wing with Haar-like elements and a flexible span spatially separated the rectangles. Two separate neural networks were used to detect faces within plane rotations, as suggested by Torralba et al. [9]. Hotta [10] showed a support vector machine (SVM) method, local kernel-based, for face recognition, which was superior to global kernel-based SVM in recognizing impeded frontal faces. Felzenszwalb and Huttenlocher [11] presented a deformable design incorporating several object components, as Fischler and Elschlager's visual structure illustration indicated. Lin and Liu [12] proposed that the multiview face identifier be learned as a single tumble classifier. They built MBH Boost, a multiclass boost-up algorithm, distributing features into several classes.

Goldmann et al. [13] used the qualified detector divided into subclassifiers connected to several predefined image regions. The inputs of subclassifiers were fused, resulting in an updated Viola-Jones detection algorithm. Yang et al. [14] used the first few cascade levels, including all face markers, to estimate the pose for expediency in multiview face recognition, where all the face identifiers modified to different visions have to be measured for each scan window; they used the first few cascade levels along with all face identifiers to approximate the pose for expediency. A quick bounding box estimation method for face recognition proposed by Subburaman predicts the bounding box using a small patch-based local search [15]. Mesphil et al. [16] proposed convolutional networks. These were neural networks with at least one layer that uses convolution instead of general matrix multiplication. Zhu and Ramanan [17] presented an idea to use the deformable parts-based template to detect a face jointly, measure an estimated pose, and then localize a face sign in the wild, which was later improved to coalesce the landmark approximation and image recognition tasks in a shared supervised way to enhance face recognition through unique landmark detections.

Yang et al. [18] explored channel features to recognize faces that perform well. Despite the ease of using these techniques for unregulated face recognition, the accuracy rate is still inadequate, particularly when the identifier must account for minor false alarms. Girshick et al. [19] proposed a work of inspiration RCNN, a convolutional neural network that performs a selective search to identify candidate regions containing items. The system aims to identify healthcare workers losing their surgical masks in the operating room. Ren et al. [20] have developed a real-time face recognition and monitoring technique.

Farfade et al. [21] have developed a deep learning-based face detection technique known as deep dense face recognition technology. The method does not require any clarification of landmarks or poses, and it can detect faces in a wide variety of orientations with only one model. Zhu and Ramanan [22] gave a method for dealing with occlusions and arbitrary pose changes in direct face detection. There is a new factor called normalized pixel difference. Machine learning approaches were used to create a deep transfer, hybrid direct instruction for face mask identification by Redmon and Farhadi [23]. Dong et al. [24] have proposed a deep cascaded region detection that investigates its bounding box decrease, a localization method, to achieve image recognition of potential countenances.

Sun et al. [25] enhanced the quicker RCNN technique via profoundly learning face detection algorithms. They utilized numerous strategies, including a pretraining model, multiband training, passive extraction, accurate calibration of primary parameters, and job grouping. Zhao et al. [3] proposed the surgical mask presence or absence monitoring device in the operating room. In the linguistic image segmentation for facial mask detection, gradient descent is used for preparation, while multiple linear regression cross-entropy is used for neural networks. Ejaz et al. [26] built a new method for detecting the existence of a face mask. They classified three different types of face mask usage: proper face mask-wearing, wrong face mask-wearing, and no mask.

The two most well-known classes, two-stage human detectors, and single-stage human detectors were recently used [27]. Fan and Jiang [28] suggested the inception network that helps to find out which kernel combination is the best. The Residual Network (ResNet) trains even deeper neural networks to learn an identity function from the preceding stage.

Most recently, the RetinaFaceMask one-stage detector approach has been studied by Fan and Jiang [28]. They fused high-level semantic fusion using multiple feature maps like Feature Pyramid Network (FPN). The proposed algorithm rejects the low confidence predictions and the high intersection of the union.

The deep learning-based approaches have an inherently high degree of accuracy as compared to the other machine learning-based techniques, especially for classification and clustering. The multilayer structure in the network helps to process different tasks at different layers exclusively.

3. Description of the Method

3.1. Deep Neural Network Algorithm. DNN algorithm moves data through a sequence of "layers" of neural network models, with each layer passing a simplified summary of the data to the next layer. Several computer vision algorithms work well on datasets with a few hundred features or columns. An unstructured dataset, such as one extracted from an image, on the other hand, contains so many features that this method becomes inefficient or impossible. Traditional machine learning algorithms cannot handle 2.4 million parts in a single 800×1000 pixel RGB color image.

As the image passes through each neural network layer, DNN algorithms learn more about it. Initial layers learn how to detect low-level features such as edges, and later layers incorporate these features into a more comprehensive representation. For instance, a middle layer would detect edges to detect parts of an object in an image, such as a leg or a branch, while a deep layer might identify the entire object, such as a dog or a tree. You gather data from observations and integrate it into a single layer. The layer produces an output, which becomes the input for the following layer, and so on. This loops until the final output signal is received [29].

3.2. Types of Algorithms. There are several different types of feature extraction algorithms, which can be classified into two categories.

3.2.1. Algorithms That Rely on Classification. The regions of interest are chosen in the first stage. After that, convolutional neural networks (CNN) are used to categorize specific areas. Since prediction must be run for each selected field, this solution may be prolonged. This group includes algorithms such as the fast RCNN and faster RCNN which are enhanced variants of the region-based Convolutional Neural Network (RCNN) [30].

3.2.2. Algorithms That Rely on Regression. In contrast to the previous approach, algorithms in this category predict the class probability and define the bounding boxes surrounding the object of interest in a single run from the entire image point of view. This group includes algorithms like You Just Look Once (YOLO) and Single Shot Multibox Detector (SSD) [30]. Deep learning and computer vision are used in various applications such as objection detection, medical image analysis, and action recognition [31, 32]. Recent research is focused on the use of mid-level features and deep learning models to build robust decision support systems and IoT applications [33–35].

3.3. Faster RCNN. In object classification and recognition, a deep learning technique known as area of interest polling is gaining much attention. Detecting objects from an image scene containing several things is one example. The goal is to extract fixed-size feature maps using maximum pooling on the entire picture as reflected in Figure 1. The object detection technique used by faster RCNN is divided into three stages.

3.3.1. Region Proposal Network. Finding the spaces in the given input image where there is a possibility of finding an object is straightforward. The position of an entity in an image can be determined. The area where there is a possibility of finding an object is surrounded by the Region of Interest (ROI).

3.3.2. Classification. The next step is to assign corresponding classes to the regions of interest defined in the previous actions. Here, the CNN approach is used (Figure 2). The proposed approach includes a detailed process for identifying all spaces of object location in an image. If no regions are placed in the first stage of the algorithm, there is no need to move on to the second step. In 2015, Girschik [36] proposed the Region Proposal Network (RPN) and ROI pooling as a DLA-based object detection solution. ROI can achieve speed and usability for both training and research performance. The ROI layers take a feature map as input, which is the output of a convolution neural network with multiple convolution layers and max-pooling layers.

An $N \times N$ matrix is generated by dividing the function map space into regions of interest. The ROI is denoted by the letter N. The first column represents the image's index. In contrast, the second column, which ranges from the upper left-most coordinate to the bottom-most coordinate, represents the ROI coordinates. Region Proposal refers to the determined area of interest space. The system divides the area proposal's entire room into equal-sized partitions. The number of sections in which the whole area proposal is divided must equal the output dimension. The maximum value of each divided subregion is estimated. The maximum values are copied to the output buffer.

In RPN, the image is first fed into the convolution neural network. The input image is passed through a series of convolution layers before being sent to the final layer, which creates feature maps. Every portion of the function maps includes a sliding window. The mask size for a sliding window mask is typical. The anchors for each sliding window are created. Let it be the exact center for these

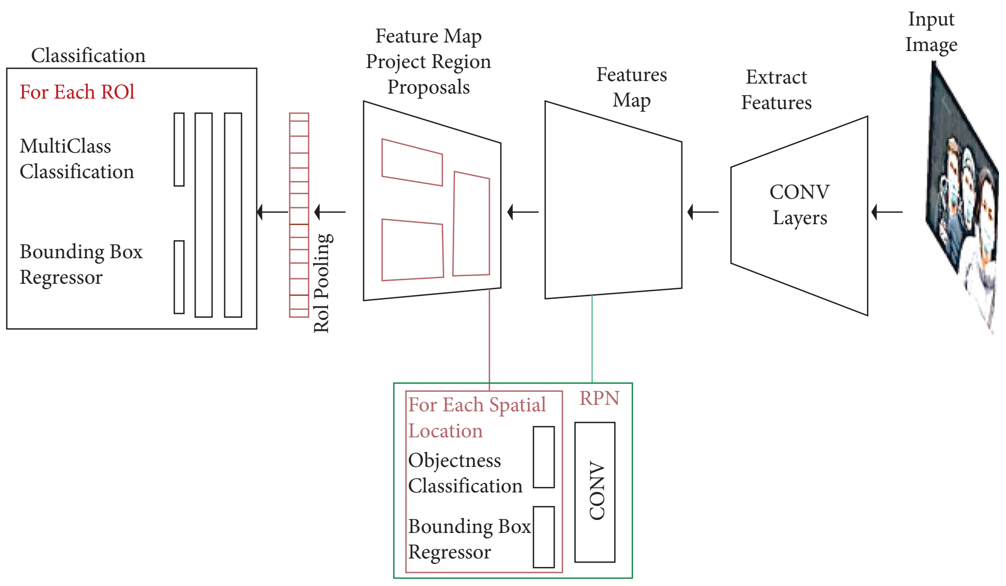

FIGURE 1: Faster RCNN.

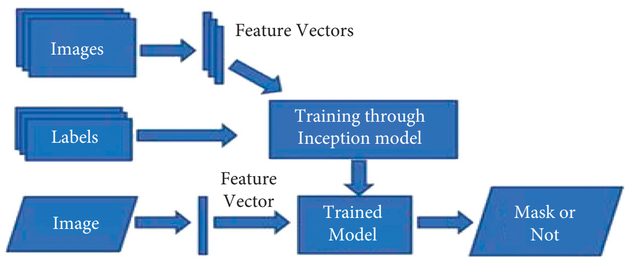

FIGURE 2: Model architecture.

anchors (x, c, y, c). However, the aspect ratios and scaling factors of the anchors produced will differ.

In addition, a value q is determined for each of these anchors, representing the likelihood of the anchors overlapping the region's boundary surrounding the objects. A region boundary with loc coordinates is the regressor's production. The classification shows whether the area contains an object or not by a probability of 0 or 1.

$$\begin{gathered} q^* = 1, \text{when } IoU > 0.7, \\ q^* = -1, \text{when } IoU < 0.7, \\ q^* = 0, \text{otherwise}, \\ t = \left[\frac{(x - x^a)}{w^a}, \frac{(y^* - y_a)}{h_a}, \frac{\log w}{w_a}, \frac{\log h^*}{h_a}\right], \\ t^* = \left[\frac{(x - x^a)}{w^a}, \frac{(y^* - y_a)}{h_a}, \frac{\log w^*}{w_a}, \frac{\log h^*}{h_a}\right]. \end{gathered} \tag{1}$$

w_a, h_a, x_a, y_a are the widths, height, and center of anchor, and h^*, w^*, x^*, y^* are the ground truth bounding box height, width, and center. Over the performance from the classification and regression networks, the loss function is established.

$$L\left(\{q^i\}, \{t^i\}\right) = \frac{1}{N_{\text{cls}} L_{\text{cls}}\left(q_i, q_i^*\right)} + \frac{\lambda 1}{N_{\text{reg}} L_{\text{reg}}\left(t_i, t_i^*\right)}. \tag{2}$$

Finally, the size 3×3 final features are extracted and fed into the networks for regression and classification.

Follow these steps to build your object detection classifier.

3.4. Inception V2 of Faster RCNN.

For object detection, the faster RCNN network is a single, centralized system. It employs the area proposal network (RPN) module. It directs the unified network's quest. On the other hand, inception comprises a 22-layer inception module with no ultimately linked layers. The main advantage of this model is that it allows better use of the computational resources available on the network. The inception module functions as a network within a network, piling modules on top of one another. It has 5 million parameters, which is a factor of 12 less than AlexNet. The combination of faster RCNN and inception V2 is computationally expensive, but the results in object detection are more reliable [37].

3.4.1. Compute Unified Device Architecture (CUDA).

CUDA is an NVidia technology that can perform a variety of challenging computations on the GPU. Every thread in CUDA uses kernels executed n times, and a unique number marks each line. CUDA's architecture comprises grids that are subdivided into smaller units called blocks. Each block is assigned to a multiprocessor by the hardware, which has a group of multiprocessors. Finally, threads make up blocks. These tiniest units can be synchronized together in a single block [38].

In general, the CUDA program begins with computer memory allocation while data on the host is being prepared. The data is then moved from the host to the computer. Since copying data from the host to the computer and the device takes time, it is essential to restrict the amount of data sent.

It is possible to launch kernels after the data on the system has been prepared. The results are copied back to the host after the calculation is completed. Finally, the results can be viewed, and the reserved memory can be freed.

The GPU implementation resembles that of a multi-threaded CPU program. The concept remains the same. We only copy to the system what is required, such as integral images, qualified classifiers, and detection windows. A CUDA kernel is run for each detection window's size. The program's first version calculates the locations of detection windows in the client framework. A set of identically sized windows is computed. Then it is sent to the computer, where the current window detection process will begin. Of course, the detection window is the same size, but the thread index determines its location. The findings are sent back to the host, and information about new detection windows is prepared after the last detection window in the kernel is checked. This procedure is repeated until the scale achieves the desired outcome. The transmission between the client and the computer is sluggish, suggesting a minor change. A count of detection windows and their size is computed for the next iteration on the client-side.

This data is transmitted to the computer. Based on the data obtained from the client and the thread index, it is now possible to calculate the location of a current detection window. This adjustment resulted in a 15-fold increase in detection speed [38].

3.4.2. CUDA Deep Neural Network (CuDNN). CuDNN is a GPU-based deep learning library created by NVIDIA. CuDNN is used by many machines learning systems, including Caffe, Tensor-Flow, and Chainer, to boost performance. We assume that the program is written in C++ and that it calls cuDNN and CUDA library functions directly in this study. For CNN computation, cuDNN includes several library functions. The cuDNN Convolution Forward function, for example, performs convolution, the cuDNN. Add Tensor function introduces biases, and the cuDNN Activation Forward function triggers sheet. CuDNN parts may only use data from GPU memory for input and output. To use cuDNN, all data, such as feature maps and weight filters, must first be loaded into GPU memory. Make sure CUDA and cuDNN versions are compatible with our Tensor-Flow edition. We should not have to worry about it because Anaconda will install the required versions of CUDA and cuDNN for the Tensor-Flow version you are using [39].

3.5. Dataset. There were a total of 3694 photos in the dataset. Another choice is to save 20% of the images (730 images) in the test folder and 80% of the images (2964 images) in the train folder, as displayed in Table 1.

3.6. Generate Training Data. Tensor-Flow needs hundreds of images of an object to train a successful detection classifier. The images used in training for a robust classifier should include random items and the target objects and several backgrounds and lighting conditions. Some photographs should have the target object partly blurred, overlapped with something else, or just halfway visible.

Table 1: Facial dataset of people with/without wearing a mask.

Source name	Class name	Amount of data	Images
Github	With mask	1847	
Github	Without mask	1847	

After we have collected all of the images, it is time to mark the items in each one (Figure 3). The tool "LabelImg" is used to mark files. Each image's label data is saved in .xml format by labeling. These .xml files will be used to generate TFRecords, a Tensor-Flow input. Once you have labeled and saved each image, there will be one .xml file in the test and train directories for each.

Now that the images have been called, it is time to make the TFRecords that will serve as input data for the Tensor-Flow training model. The image.xml data will be used to create .csv files containing all the data for the train and test images. Type the following in the Anaconda command prompt. Train record and test record are the two files used to train the current object detection classifier.

The label map is class names to class ID numbers that tell the trainer what kind of object they are dealing with. In a text editor, create a new file named labelmap.pbtxt and save it in the training folder. The numbers in the label map IDs must match those in the generate tfrecord.py format.

The object detection training pipeline, last but not least, must be set up. The training process decides the model and parameters that will be used. This is the final step before starting running training. The faster RCNN inception v2 pets. Config file has been improved with the addition of file paths to the training data and an increase in the number of classes and examples. Save the file after you have made your changes. The training role has been developed and is ready to begin!

3.7. Execute the Training. If all is set up correctly, Tensor-Flow will begin the training. The initialization process will take up to 30 seconds before the actual training begins. It will look like this when you first start training.

The loss is recorded at each stage of training. It will begin high and progressively decrease as training progresses. Our faster RCNN inception V2 model training started at around 3.0 and quickly dropped below 0.8. Enable the model to train

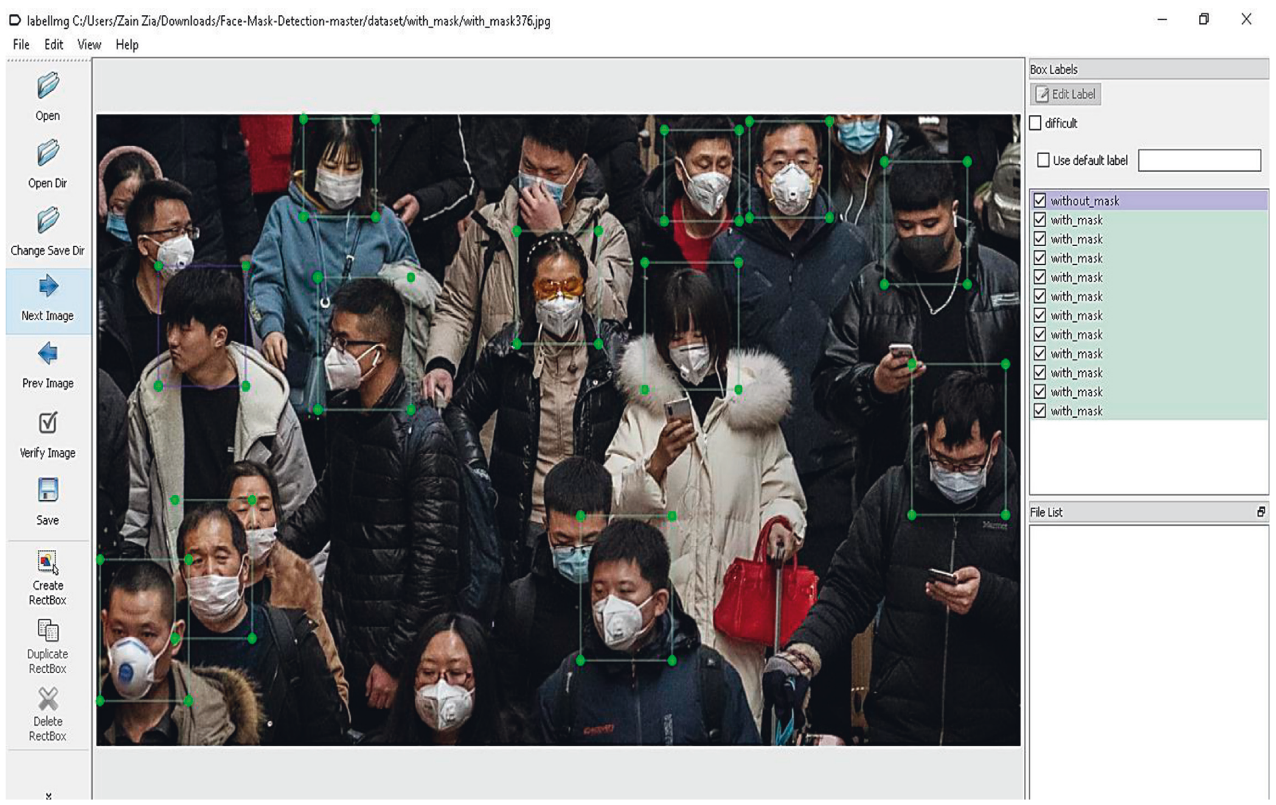

Figure 3: Labeled pictures.

for about 40,000 steps or two hours or until the loss is consistently less than 0.05, depending on CPU and GPU.

The algorithm is similar to the RCNN algorithm. Since you do not have to feed the convolutional neural network 2000 region proposals every time, "fast RCNN" is faster than RCNN. Instead, the convolution operation, performed only once per image, produces a feature map.

4. Results and Discussion

4.1. Training Results. When training is completed, Tensorboard keeps track of this operation. Using the tool made it possible to decide whether the model is ready for deployment or requires additional training or other modifications. It was possible to visualize the model's learning curve using graphs such as total loss. For example, suppose the error rate remains high and constant over time. Either the model's configuration or the data itself should be updated and corrected, and the training should be terminated.

4.2. Tensor-Board. Through Tensor-Board, we will see how the training has progressed. Tensor-Board is responsible for the visualization graphs. One of the essential graphs is the losses graph, which depicts the cumulative losses of the trained model over time during training. For a GPU-enabled OS, model training took three days. For our faster RCNN inception V2 Coco API training, the loss of the neural network (net) started at five and quickly fell below. Tensorboard is the user interface for visualizing the graph and other resources for debugging, optimizing, and understanding the model. The number of epochs is represented by the x-axis, while the y-axis represents the time. The recognition rate is calculated in real time as part of our model training. After 200 k epochs, we can see that we have achieved our desired accuracy. The model's accuracy was improved by data or image augmentation.

The panel has several tabs, each corresponding to the level of data you enter while running the model.

Scalars: During the model training, show a variety of valuable data.

Graphs: Display the model.

Histogram: A histogram can be used to show the weights.

Distribution: Show how the weights are distributed.

Projector: Show T-SNE algorithm and Principal Component Analysis. For dimensionality reduction, this technique is used.

It assists in understanding the interdependencies between operations, how weights are calculated, displaying the loss function, and much more. When you combine all of these pieces of knowledge, you have a powerful tool for debugging and improving the model. A vital graph is the loss graph, which depicts the classifier's overall loss over time.

4.3. Tensor-Board Losses Graph. While the curve continues to get closer to zero as time goes by, it will never hit that point because nothing is perfect or faultless. A total loss

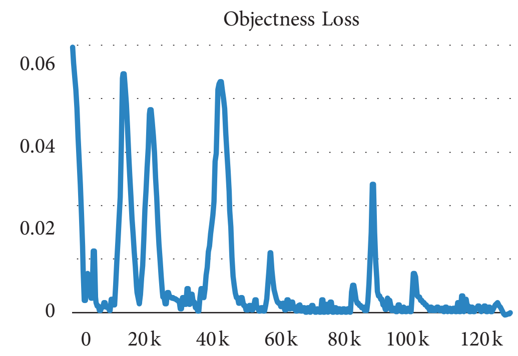

Figure 4: Objectness loss.

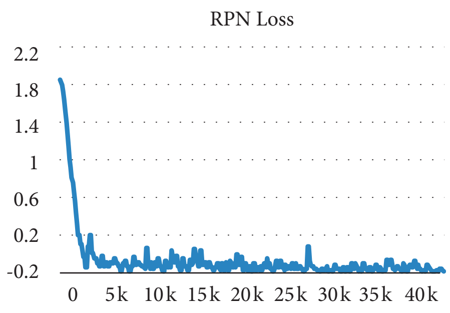

Figure 7: RPN loss.

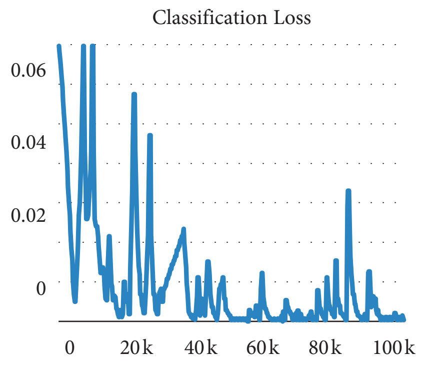

Figure 5: Classification loss.

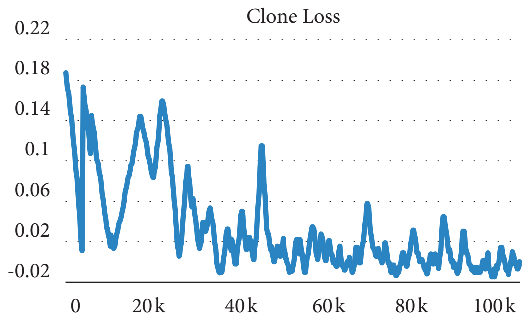

Figure 8: Clone loss.

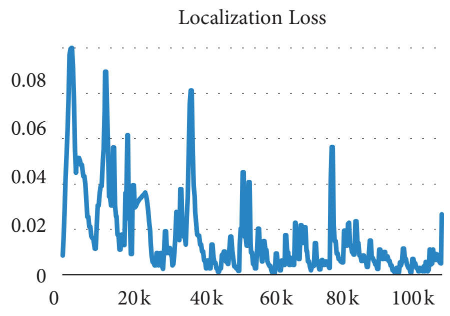

Figure 6: Localization loss.

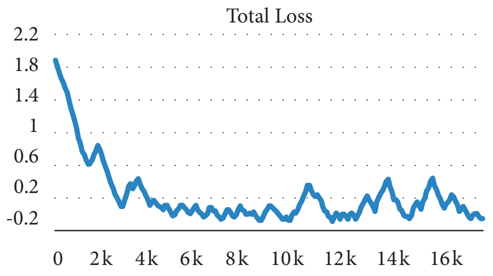

Figure 9: Total loss.

value of less than 2.5 is generally considered reliable, but it also indicates that the model may be improved by adjusting parameters or having a better dataset. After 200,000 epochs, the total loss is 0.0503. It tends to decrease. However, the map (mean average precision) does not increase as the loss decreases. When the number of epochs is 200,000, the map is 0.0502. The whole training process is reflected below through key graphs acquired after training. The objectness loss was found zero asymptotically after 120 k epochs (Figure 4). The classification loss was notably substantial during the early phase of training but ultimately comes close to zero asymptotically after 100 k epochs (Figure 5). Similarly, consistent asymptotic behavior is reflected in other

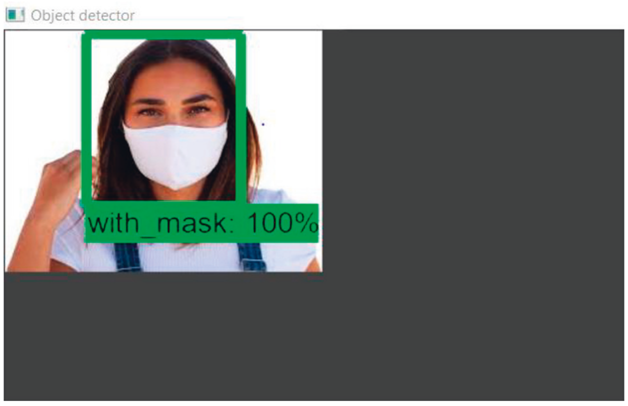

Figure 10: Output-I results.

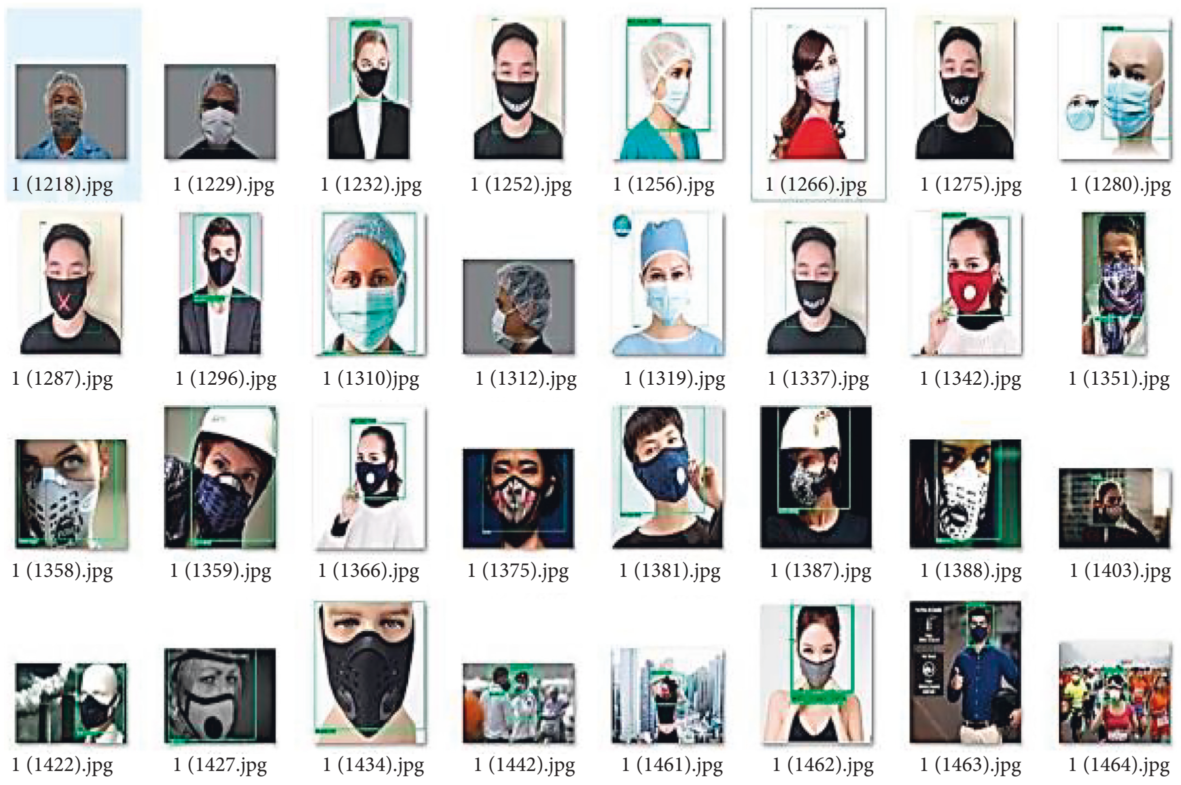

Figure 11: Output-II results.

Table 2: Comparison of accuracy achieved through different approaches.

Sr. no.	Methodology	Accuracy (%)
1	Naive Bayes	91.3
2	SVM	89.6
3	Decision tree	85.1
4	Kernel approximation method	83.5
5	Proposed approach	97

training graphs, i.e., localization loss, RPN loss, clone loss, and total loss in Figure 6–9, respectively.

4.4. Python Shell. Open the Anaconda command prompt and type “idle” (with virtual environment selected) followed by ENTER to run any scripts. This will start IDLE, allowing us to open and run some of the scripts. Image Object Detection with Tensor-flow Classifier will open when we open the Python Shell. After that, the run module was selected.

4.5. Output Results. In this research, faster RCNN proved to be more efficient and precise in providing 97% accurate results and showed that the processing time for the whole process is less than the other traditional techniques. This study proposed to decide which person is wearing the mask correctly using DNN. When we train our model after 200,000 epochs, the total loss is 0.0503, which tends to decrease. However, the mean average precision does not increase as the loss decreases. When the number of epochs is 200,000, the mean average accuracy is 0.0502. Sample results are shown in Figures 10 and 11.

5. Conclusion

In this article, we introduced a reliable DNN-based system for identifying people wearing masks. Faster RCNN was employed to train the data in this method, resulting in high accuracy. This model is trained on a GPU to obtain a low computational cost. To achieve our goals, we used a multiphase detection model: First, to label the face mask, and second to detect the edge and compute edge projection for the chosen face region within the face mask. The current findings revealed that faster RCNN was efficient and precise, giving 97% accuracy. The overall loss after 200,000 epochs is 0.0503, with a trend to decrease. While the loss is decreasing, we are getting more accurate results. As a result, the faster RCNN technique effectively identifies whether a person is wearing face masks or not, and the training period was decreased with better accuracy. In the future, Deep Neural Network (DNN) might be used first to train the data and then compress the dimensions of the input to run it on low-powered devices, resulting in a lower computational cost. The results achieved from the proposed approach reflect significant accuracy as compared to the other commonly used approaches, i.e., Table 2.

Data Availability

No such private data were used to support the findings of the study. Only publicly available data has been used.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] T. Greenhalgh, M. B. Schmid, T. Czypionka, D. Bassler, and L. Gruer, "Face masks for the public during the covid-19 crisis," *BMJ*, vol. 369, p. 1435, 2020.

[2] S. Feng, C. Shen, N. Xia, W. Song, M. Fan, and B. J. Cowling, "Rational use of face masks in the COVID-19 pandemic," *The Lancet Respiratory Medicine*, vol. 8, no. 5, pp. 434–436, 2020.

[3] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: a review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[4] A. Kumar, Z. J. Zhang, and H. Lyu, "Object detection in real time based on improved single shot multi-box detector algorithm," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, 204 pages, 2020.

[5] L. Jiao, F. Zhang, F. Liu et al., "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, Article ID 128837, 2019.

[6] Z. Li and Wu, "Efficient object detection framework and hardware architecture for remote sensing images," *Remote Sensing*, vol. 11, no. 20, p. 2376, 2019.

[7] H. Schneiderman and T. Kanade, "A statistical method for 3D object detection applied to faces and cars,"vol. 1, pp. 746–751, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 1, pp. 746–751, IEEE, Hilton Head, SC, USA, June 2002.

[8] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings of the International Conference on Image Processing*, vol. 1, September 2003.

[9] A. Torralba, K. P. Murphy, and W. T. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, June 2004.

[10] K. Hotta, "Robust face detection under partial occlusion," *Systems and Computers in Japan*, vol. 38, no. 13, pp. 39–48, 2007.

[11] P. F. Felzenszwalb and D. P. Huttenlocher, "Pictorial structures for object recognition," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 55–79, 2005.

[12] Y.-Y. Lin and T.-L. Liu, "Robust face detection with multiclass boosting,"vol. 1, pp. 680–687, in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 680–687, IEEE, San Diego, CA, USA, June 2005.

[13] L. Goldmann, U. J. Monich, and T. Sikora, "Components and their topology for robust face detection in the presence of partial occlusions," *IEEE Transactions on Information Forensics and Security*, vol. 2, no. 3, pp. 559–569, 2007.

[14] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.

[15] V. B. Subburaman and S. Marcel, *Fast bounding box estimation based face detection,*In ECCV, Workshop on Face Detection: Where We Are, and What Next? Lausanne, Switzerland, 2010.

[16] G. Mesnil, Y. Dauphin, G. Xavier et al., "Unsupervised and transfer learning challenge: a deep learning approach," in *Proceedings of the ICML Workshop on Unsupervised and Transfer Learning*, vol. 27, pp. 97–110, Washington, WA, USA, 2012.

[17] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2879–2886, IEEE, Providence, RI, USA, June 2012.

[18] B. Yang, J. Yan, Z. Lei, and S. Z. Li, "Aggregate channel features for multi-view face detection," in *Proceedings of the IEEE International Joint Conference on Biometrics*, pp. 1–8, Sheraton Sand Key Resort Clearwater, FL, USA, 2014.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580–587, Oregon, OR, USA, 2013.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[21] S. S. Farfade, M. J. Saberian, and L.-J. Li, "Multi-view face detection using deep convolutional neural networks," in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval - ICMR*, vol. 15, China, 2015.

[22] B. F. Klare, B. Klein, E. Taborsky et al., "Pushing the frontiers of unconstrained face detection and recognition: IARPA Janus Benchmark A," in *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1931–1939, Boston, MA, USA, June 2015.

[23] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7263–7271, Hawaii, HA, USA, 2016.

[24] Z. Dong, J. Wei, X. Chen, and P. Zheng, "Face detection in security monitoring based on artificial intelligence video retrieval technology," *IEEE Access*, vol. 8, Article ID 63421, undefined 2020.

[25] X. Sun, P. Wu, and S. C. H. Hoi, "Face detection using deep learning: an improved faster RCNN approach," *Neurocomputing*, vol. 299, pp. 42–50, 2018.

[26] M. S. Ejaz, M. R. Islam, M. Sifatullah, and A. Sarker, "Implementation of principal component analysis on masked and non-masked face recognition," in *Proceedings of the 2019 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)*, pp. 1–5, IEEE, Dhaka, Bangladesh, May 2019.

[27] Z. Wang, "Masked face recognition dataset and application," 2020, https://arxiv.org/abs/2003.09093.

[28] X. Fan and M. Jiang, "RetinaFaceMask: a single stage face mask detector for assisting control of the COVID-19 pandemic," 2020, https://arxiv.org/abs/2005.03950.

[29] T. V. Janahiraman and M. S. M. Subuhan, "Traffic Light Detection Using Tensorflow Object Detection Framework," in *Proceedings of the 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET)*, pp. 108–113, IEEE, Shah Alam, Malaysia, October 2019.

[30] S. Yin, H. Li, and L. Teng, "Airport detection based on improved faster RCNN in large scale remote sensing images," *Sensing and Imaging*, vol. 21, no. 1, p. 49, 2020.

[31] J. Zhang, G. Ye, Z. Tu et al., "A spatial attentive and temporal dilated (SATD) GCN for skeleton-based action recognition," *CAAI Transactions on Intelligence Technology*, vol. 7, no. 1, pp. 46–55, 2022.

[32] K. Jafarbigloo, H. Danyali, Sanaz, and H. Danyali, "Nuclear atypia grading in breast cancer histopathological images based on CNN feature extraction and LSTM classification," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 4, pp. 426–439, 2021.

[33] S. Fatima, N. Aiman Aslam, I. Tariq, and N. Ali, "Home security and automation based on internet of things: a comprehensive review," *IOP Conference Series: Materials Science and Engineering*, vol. 899, no. 1, Article ID 012011, 2020.

[34] Q. Zou, K. Xiong, Q. Fang, and B. Jiang, "Deep imitation reinforcement learning for self driving by vision," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 4, pp. 493–503, 2021.

[35] M. A. Aslam, M. Naveed Salik, F. Chughtai, N. Ali, S. Hanif Dar, and T. Khalil, "Image classification based on mid-level feature fusion," in *Proceedings of the 2019 15th International Conference on Emerging Technologies (ICET)*, pp. 1–6, IEEE, Peshawar, Pakistan, December 2019.

[36] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448, IEEE, Santiago, Chile, December 2015.

[37] R. L. Galvez, A. A. Bandala, E. P. Dadios, R. R. P. Vicerra, and J. M. Z. Maningo, "Object detection using convolutional neural networks," in *Proceedings of the TENCON 2018 - 2018 IEEE Region 10 Conference*, pp. 2023–2027, IEEE, Jeju, Republic of the Korea, October 2018.

[38] J. Krpec and M. Němec, "Face detection cuda accelerating," in *Proceedings of the ACHI 2012, The Fifth International Conference on Advances in Computer-Human Interactions*, pp. 155–160, Citeseer, New Jersey, NJ, USA, 2012.

[39] Y. Ito, R. Matsumiya, and T. Endo, "ooc_cuDNN: accommodating convolutional neural networks over GPU memory capacity," in *Proceedings of the 2017 IEEE International Conference on Big Data*, pp. 183–192, IEEE, Boston, MA, USA, December 2017.

[40] A. Kadiyala and A. Kumar, "Applications of Python to evaluate environmental data science problems," *Environmental Progress & Sustainable Energy*, vol. 36, no. 6, pp. 1580–1586, 2017.