International Journal of Scientific & Engineering Research, Volume 13, Issue 11, November-2022
ISSN 2229-5518

Automatic Face Mask Detection and Recognition Using Deep Learning

Fida Hussain Dahri[1]
Department of Information Technology
Quaid-e-Awam University of Engineering, Science and Technology,
Nawabshah, Pakistan
Fidahussaindahri22@gmail.com

Ghulam Mustafa [2]
Department of Computer and Electronics Engineering
Sun Moon University, Asan-si, Chungnam, Republic of Korea
Ghulammustafa@sunmoon.ac.kr

Ubedullah Dahri [3]
Department of Information Technology
Quaid-e-Awam University of Engineering, Science and Technology,
Nawabshah, Pakistan
ubedullahdahri404@gmail.com

Abstract — The fast advancement of computer vision allows for human-computer interaction and has a broad range of applications. Since the first instance of COVID-19 was discovered, the worldwide battle against the pandemic has started. People's everyday actions, in addition to different research and conclusions by medical and healthcare professionals, have become critical in fighting the pandemic. In China, the government has adopted active and effective isolation and closure measures, as well as active public collaboration, such as making it unnecessary to remain inside and wear masks. China, the nation where the pandemic initially broke out, has now established itself as the world's model for epidemic prevention. Of course, people wearing masks deliberately isn't enough. Wearing masks in public areas still requires supervision. Real-world applications utilizing deep learning use deep learning as a critical component. Object detection is extremely important. currently, deep learning detection models and algorithms are using object recognition as their objective, which has achieved tremendous success in finding the object from an image. As this is the era of the COVID-19 virus, people frequently wear masks to cover themselves to minimize the transmission of the coronavirus. Because some portions of the face are concealed, this makes face identification is a very challenging job. In certain cases, traditional facial recognition technology is still inadequate, so it is very urgent to improve the recognition efficiency of the existing face recognition technology on masked faces, as masks are part of life from now for the next two to three years. This study proposes that, in this process, manual inspection be replaced with a deep learning method, and that YOLOV5, the most powerful objection detection algorithm currently available, be used to better apply it in the real world. For this study, First, we use the YOLO V5 to detect face masks. Using Face Net's trained model, we looked at the images to determine whether the subjects were wearing masks or not. Two separate medical face mask datasets have been brought together in one dataset for research purposes. Mean IoU has been utilized to determine the best number of anchor boxes, hence improving the object detection process. The results showed that the Adam optimizer got an average of 81% accuracy. Finally, a related conclusion is offered in the research as a comparison study. The new detector outperformed related work in terms of accuracy and precision.

index Terms — COVID-19, Face Mask Detection, Face Mask Recognition, Feature Extraction, Machine Learning, Convolutional Neural Network, YOLOV5, Confusion Metrix, Deep Learning.

I. Introduction

COVID-19 is now a major public health and economic concern owing to the virus's negative impacts on society, which include severe respiratory illnesses, mortality, and worldwide corporate problems (Rahmani & Mirmahaleh, 2020). "More than six million patients were infected by COVID-19 in more than 180 countries, with a mortality rate of 3%," according to WHO, 2020 [1]. COVID-19 spreads quickly in crowded places and through close touch. In many countries, governments face enormous problems and risks in protecting people from the coronavirus."[2].

There are numerous countries where citizens are legally required to wear face masks in public, which necessitates the use of face recognition in applications like object detection. [3]. Governments will require advice and surveillance of people in public places, particularly those that are extremely crowded, to enforce laws prohibiting the use of face masks, to combat and win the COVID-19 pandemic. The combined use of surveillance technologies and artificial intelligence models might enable this goal.

COVID-19 is a virus that affects the respiratory system and is transferred through close contact. As a result, the majority of individuals began to wear face masks for protection; as a result, facial recognition algorithms had difficulty identifying persons wearing medical masks [2].

A. Deep Learning

A neural network, unlike the human brain, which is astonishingly adept at making sense of what our eyes show us, takes a completely different approach to learning and seeing real-world objects. A neural network takes a new approach to the problem of visual pattern or image processing. The idea behind a neural network's operation is to take a large dataset of pictures, also known as training examples, and build a system that can learn from them. When a neural network is fed more training examples, it gets more efficient, increasing its accuracy in predicting the proper results without the need for human involvement [3].

Object detection. Artificial neural networks were influenced by how human neurons process information. Neurons are the most basic, critical computational components of the brain. The computational unit in an artificial neural network is known as a neuron, A subset of nodes and units. This measures or estimates an output by receiving input from an external source or another unit (node) with a weight, then calculating or estimating an output. The weights that have been allocated to the inputs have been determined by their relative significance to other inputs [4]. To calculate an output, the unit has a function called the Activation Function, Also known as the weighted sum or unbiased sum, it's the result of adding all the inputs together, as well as an additional input called bias, which is used with weight.

Because all real-world input is non-linear and the neural network is all about learning non-linear representations of the real world, the activation function is non-linear and is thus applied to the output to bring non-linearity to the output. Perceptions are another name for an artificial neuron. Artificial Neural Networks come in a variety of shapes and sizes, each with its own set of types and classes [5].

B. Face Recognition & Detection Introduction

To determine whether or whether there is a person in a photograph, face recognition technology has been created. It also recognizes a person's face and determines who they are. Face detection, face alignment, feature extraction, and face recognition are the four steps of the method [5].

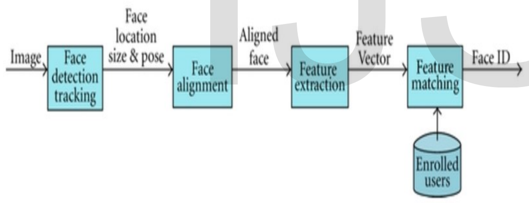


Figure 1. Facial Recognition System Structure [8]

The major goal of this initial phase is to identify the section of a photo or video that represents a face and to detect the position of these faces. The output of this step may then be used to convert the provided data into patches with each face as the input picture. By recognizing geometry and photometric, the system normalizes the face to become more consistent with the database in the second phase. In other words, the system takes a picture of a face and evaluates it. Most face recognition systems collect pictures in two dimensions rather than three dimensions to better match them to public photographs and databases. When the system scans an image, it looks for crucial details including the depth of an individual's eye sockets, the curve of their cheekbones, the distance between their eyes, the distance between their forehead and chin, and the form of their lips, chin, and ears [6].

Face patches are derived from pictures after the face has been normalized. The system's job is to transform a picture into data based on a person's facial characteristics. The algorithm prefers to extract the most meaningful data from these pictures, identifying the most crucial pieces of data while disregarding any noises [19]. Information packing, noise cleaning, dimension reduction, and salience extraction are all possible with facial recognition feature extractions. Each person's faceprint is calculated during feature extraction [7].

Figure 2. displays an example of the most important characteristics retrieved from a picture.

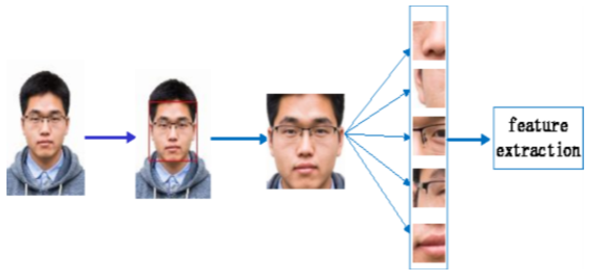


Figure 2. Feature Extraction [4]

The system is responsible for differentiating between the identities of various people's faces in the final phase. A face database may be constructed for the system to accomplish automated recognition by capturing numerous photos of each individual's face, then extracting and storing the characteristics of these images in the system database. The system then conducts face detection as well as feature extraction of the input image whenever it occurs. The system then compares the picture characteristics to each stored

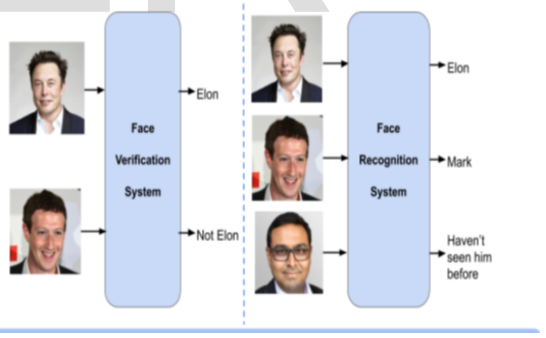


Figure 3. Identification and Verification in Face Recognition [4]

faceprint in the database to determine whether access is granted or denied.

II. FACE DETECTION MODELS

A. YOLO v5 Algorithm

The phrase "You Only Look Once" is abbreviated as YOLO. YOLO is a powerful object recognition system that works in real-time. This is an algorithm for detecting and recognizing different items in a photograph (in real-time). YOLO is performed as a regression problem, and the class probability of each of the identified photos is provided. A new member of the YOLO family was recently introduced: YOLOv5. YOLO (You Only Live Once) was the first model to have object detection prediction, object classification prediction, and object boundary detection prediction integrated into a single, differentiable network. The Darknet framework is used to build and manage it. The lightweight and accessible nature of YOLOv5 makes it one of the first YOLO models to be developed in the PyTorch framework, which makes it lighter and simpler to use. Despite these advances, however, the YOLOv5 network does not outperform the YOLOv4 network on the COCO dataset since it was unable to improve the architecture.

Object identification differs from the YOLO framework in that it addresses the identification process in a way unlike anyone else. Based on a single full-image instance, the predicted coordinates and class probabilities for these boxes are calculated. Processing 45 frames per second using YOLO is possible. While YOLO covers generic object representations, they are all permitted. [10].

The three strategies used by the YOLO algorithm are as follows:

- Residual blocks

To create the image, the grid divisions are extracted from the image first. Each grid has two equal sides: S units wide and S units long. Several identical-sized grid cells can be found in the image above. Grid cells will be able to detect any objects that appear inside them. When a certain grid cell emerges as a point of interest, that cell will bear the responsibility of detecting it. [10].

- Bounding box regression

When using a bounding box, you are drawing attention to something in a photograph by using an outline.
The image contains the following attributes:
Width (bw).
Height (bh).
social class (for example, person, car, traffic light, etc.)
The letter C stands for this..
Bounding box center
(bx,by).
The bounding box shown in the image is a good representation of what a bounding box looks like. The bounding box has been shown with a yellow outline.

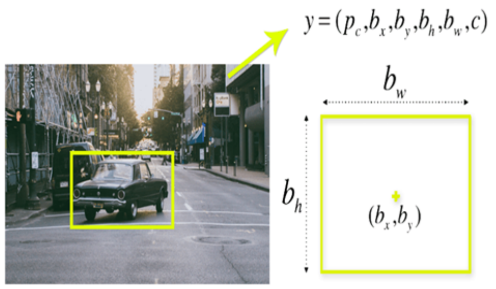


Figure 4. Illustrated a certain object in a picture [4]

[11].

- Intersection Over Union (IOU)

The concept of intersection over union (IOU) illustrates how boxes overlap in object detection. To wrap or

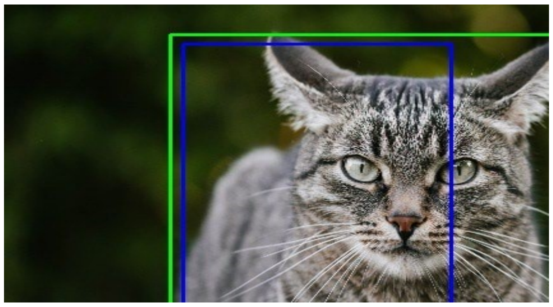


Figure 5. Intersection Over Union (IOU) [4]

encircle goods, IOU is utilized by YOLO.

Predicted grid cell confidence values for the bounding boxes and their bounding boxes are determined by each grid cell. The equation is thus equivalent to IOU = 1 if the anticipated and real bounding boxes are identical. This strategy eliminates unnecessary bounding boxes that are not identical in size. The graphic below is a simple example of how IOU works [12].

B. FaceNet Pretrained Mode

FaceNet was developed by Google and was said to be state-of-the-art in facial recognition in 2015 when it obtained top scores on a range of face identification benchmark datasets. As a result, The availability of pre-trained models, along with numerous third-party open-source implementations of the FaceNet system, make it possible to apply FaceNet broadly. FaceNet is good for training face recognition systems because it can provide accurate, accurate descriptions of facial traits. [12].

A face embedding is a system that, given a face image, predicts a 128-element vector representation of the face's properties. FaceNet is a machine learning method that uses facial photographs to learn a direct mapping to a compact Euclidean space, which is characterized by an exact relationship between the lengths of the distances in the space and a measurable degree of facial silliness. Vectors that have the same identity are allowed to grow closer together, but vectors with different identities are made to become further apart (larger distance). a crucial feature in this study was developing a model capable of producing embeddings without any intermediate steps [12].

C. Dataset For Detecting Face Masks

Training data represents the original data that is used to improve the model. The model makes use of the data for building and improving itself the quality of this data has a significant impact on the model's performance. Following the development, which aids in the establishment of a powerful model for any future applications that may utilize the same data for training. In machine learning, there is a human component to data training [13].

Individuals who wear masks and people who do not wear masks are divided into two groups in this dataset, which is accessible on Kaggle. There are a total of 7553 face pictures in the collection, including 3725 images of faces with masks and 3828 images of faces without masks. This dataset's pictures all have three color channels (RGB) [30]. Figure 1.9 depicts a variety of persons wearing face masks. Figure 1.10 depicts a group of persons who are not wearing face masks. People who wear face masks are labelled 0 in this dataset, whereas those who do not wear face masks are labelled 1. This dataset was first used to pre-train the model [14].

Figure 6. Dataset wearing face masks are not wearing face masks [4]

III. RESEARCH QUESTION

- What are the best approaches used to detect face masks?
- Which techniques are used to recognize whether people are wearing a proper mask or not?

IV. AIMS AND OBJECTIVES

As this is the era of the COVID-19 virus, people frequently wear masks to cover themselves to minimize the transmission of the coronavirus. Because some portions of the face are concealed, this makes face identification is a very challenging job. In certain cases, traditional facial recognition technology is still inadequate, so it is very urgent to improve the recognition efficiency of the existing face recognition technology on masked faces, as masks are part of life from now for the next two to three years.

In this study Medical face mask is the main attention of the study to lessen the spread of COVID-19 Virus.
This research is focused on locating and locating the face mask in a picture as shown in figure 6.

- To develop a face mask detection model, and then
- To develop a technique to recognize faces that are masked or unmasked.

V. RELATED WORKS

Preeti Nagrath, Rachna Jain, Agam Madan, and their SSDMNV2 team developed a method for detecting face masks in real-time using deep learning, Keras, TensorFlow, and OpenCV. This group was successful in locating an open-source dataset known as The Kaggle Medical Mask Dataset. The data compiled by Mikolaj Witkowski and Prajna Bhandary was an example of effective quantitative analysis. The researchers split the picture dataset into two groups in the proposed SSDMNV2 model, with the first category including persons wearing masks and the second category including those who were not wearing masks. Using the MobilenetV2 image classifier, they were able to categorize their pictures. The MobilenetV2 was chosen for its ease of real-time deployment. Mohamed Loey, Mohamed Hamed N. Taha, Gunasekaran Manogaran, and Nour Eldeen M. Khalifa developed a method for identifying and locating medical face mask items in real-world pictures. A feature extraction procedure utilizing the Their model incorporates ResNet-50 deep transfer learning model and a medical face mask detection that uses YOLO v2. Using the mean IoU, the authors were able to enhance the detection performance. This aided in determining the most appropriate number of anchor boxes. As a result, the authors were able to conclude that employing Adam optimizer yielded a maximum accuracy of 81 percent [14]. Ge S., Li J., Ye Q., and Luo Z. created a model to detect unmasked and masked faces using a dataset. Masked Faces (MAFA) was a dataset that comprised 35,806 pictures of people wearing masks. To propose their model, the authors employed a convolutional neural network with three separate modules: proposal, implementation, and authentication. Their work yielded accuracy results of 76.1 percent [15]. Md. S. Ejaz, Md. R. Islam, Md. R. Sifatullah, Md. Sifatullah, Md. Sifatullah, Md. Sifatullah, Md. Sif A used machine learning techniques to discriminate between those who used face masks and those who did not. Principal Component Analysis (PCA) was employed (PCA). In the Principal Component Analysis, the paper was successful in detecting persons who are not wearing masks, resulting in a higher recognition rate. The authors discovered that obtaining features from persons wearing face masks is less difficult than retrieving characteristics from those who are not wearing face masks. They also discovered that accuracy dropped dramatically after categorizing persons wearing a face mask, which had a 70% accuracy rate [16]. Researchers have suggested conducting research on real-time facemask recognition, which incorporates deep learning techniques and Convolutional Neural Networks, with both paper and digital designs by Sammy V. Militante and Nanette V. Dionisio (CNN). The findings of the authors' investigation of facemask detection are precise and quick. The goal of the study was to differentiate between persons who wore a facemask and those who did not. The authors were able to train their model using the CNN model, which enabled them to reach a performance accuracy of 96 percent. Furthermore, by identifying individuals who use or do not wear a facemask, the study proved a useful tool in combating The circulation of the COVID-19 virus. The findings might be used to assist set alerts to alert others if someone is not wearing a face mask [17]. Chethan et al, 2020 This study employs the YOLO algorithms v4 and v3 for object recognition in video and picture files, respectively. The model is trained on a dataset containing numerous objects and variable lighting. YOLOv3 extracts characteristics and predicts item class and position using darknet53 and CNN. The hyperparameters of the models are set using Batch Normalization, Leaky ReLU, Anchors, and Batch Norm. By creating n layers of classes for feature extraction, non-max suppression helps to avoid bounding box overlap. The time and density depend on the location., time, and density of the items, detecting numerous objects through video surveillance becomes difficult [18].

VI. RESEARCH DESIGNING

The study's design is based on the CRISP DM approach. The Cross-Industry Framework for Data Mining is a method for developing data mining research that is well-organized and systematic. Crisp DM is the most widely used and successful data analytics solution for resolving corporate issues. These six steps constitute six phases that constitute the entirety of a project: understanding, data preparation, modeling, evaluation, deployment, and post-project support CRISP DM framework. Each phase plays an essential function in its way. The phases of business understanding and data knowledge are critical to the overall framework. Understanding Data set: to determine the surveillance-based data on people who are with a mask or not. Data understanding entails delving into the massive amounts of data that are utilized in model construction. The pictures of the users' faces are the data in this study. For this study, a tailored dataset of 6 individuals' face data was produced. The raw picture data is labelled in the third phase, Data Preparation, to make it compatible with the YOLO V5 object recognition method. An image file with the same name as the image file is created, and then a text file containing the height, width, x, and y coordinates of the user's face is generated. The modeling step comprises the creation of an effective model. The YOLO object detection model is trained on a face picture dataset to create a multi-layered neural network in this study. A six-user custom face dataset is being created. In the Evaluation phase, the developed model, including the YOLO V5 utilized in this study, will be evaluated, with the best model accuracy for identifying masked faces being chosen for the next phase. In the deployment phase, the best model of the assessment phase will be used to identify whether or not a face mask is worn. The suggested detector model is shown in Figure 7. During the training, validation, and testing phases, the detector primarily employed YOLOv5 with FaceNet for feature extraction and detection. The following steps are taken to design, implement and test the model.

VII. DETECTION MODEL

For masked, unmasked, and incorrectly worn masks, BBOX

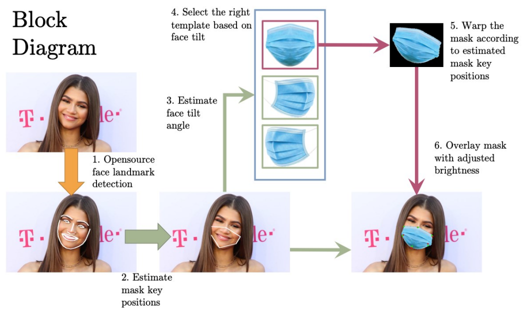

Block Diagram
4. Select the right template based on face tilt
5. Warp the mask according to estimated mask key positions
3. Estimate face tilt angle
6. Overlay mask with adjusted brightness
1. Opensource face landmark detection
2. Estimate mask key positions

Figure 7. YOLO object detection model

annotations in XML format were included in the dataset. After that, we used the YOLO v5 method as our mask and unmasked face detection model's basis model. As a starting point, we used the public implementation of YOLO v5 at https://github.com/ultralytics/yolov5. Then, to best fit our requirements, we made adjustments to the architecture and code source https://www.kaggle.com/andrewmvd/face-mask-detection/face-mask-detection/face-mask-detection/face-mask-detection/face-mask-detection/face-mask-detection/face-mask-dete . To train our model, we had to convert XML format annotations to YOLO v5 format annotations, which our YOLO v5 code could utilize and comprehend. It's done in the detector folder's InitializationNotebook.ipynb notebook. After that, we used YOLONotebook.ipynb to train our model. We utilized the transferred learning approach to transfer yolov5 coco weight into our base model and then trained our model on it to save time during training. To minimize overfitting, the algorithm employs augmentation and a variety of additional approaches. After the training, the model's weights and assessment results are saved in the results folder. Inferencing.ipynb notebook contains a demo inference on various pictures.

VIII.IDEA FOR FACE RECOGNITION

The FaceNet pretrained model was used to detect faces for face recognition. The problem was that this model had never been trained to compare a masked face to an unmasked face or a different colored masked face to another colored masked face during testing, thus it couldn't produce satisfactory results. We couldn't fine-tune the model for masked faces due to a lack of dataset (we couldn't find one in time) for training the face net model. During the test, we chose to apply masks of various colors and patterns on unmasked faces and compare the masked pictures to the input photos. This way we will compare masked face images with masked face pictures and unmasked face pictures with unmasked face images. How is a mask put on the face?

A. Recognition

FaceNet was built using an open-source Face Recognition Library that we customized to fit our needs. To begin, we create folders for each known individual and place four to five distinct photos of that person in each folder. Then we execute the code to cover these people's faces with masks. It generates a new directory in which each person's photo is stored, together with the masks we apply to them. Actual disguised photos of people can be placed in their appropriate files. Then we create embeddings for all of the faces (masked and unmasked) and save them. PrepareImagesAndGenerateEmbeddings.ipynb contains this code. To recognize faces, we start by looking at the output of our yolo Facemask detector. We then create an Embedding for yolo's output. We compared this output to known masked face embeddings if it was a masked face, and we compared it to known unmasked face embeddings if it was an unmasked face, and then we showed output. DetectMaskAndRecognizeFace.ipynb is a file in the DetectMaskAndRecognizeFace.ipynb directory.

International Journal of Scientific & Engineering Research, Volume 13, Issue 11, November-2022
ISSN 2229-5518

IX. DATA SOURCES AND DATA COLLECTION

The two publicly available medical face mask datasets from the studies in this study served as the basis for these studies. MMD (https://www.kaggle.com/vtech6/medical-masks-dataset) is the first dataset used in this competition. The MMD collection contains 682 images with over 3000 masked medical faces wearing masks. MMD picture examples are shown in Fig. 2. Facial Mask Dataset (FMD) at (https://www.kaggle.com/andrewmvd/face-mask-detection) is the second public masked face dataset. There are 853 pictures in the FMD dataset. In Figure 8, some FMD samples are presented. By merging MMD with FMD, we were able to generate a new dataset. By eliminating pictures of poor quality, the combined dataset now comprises 1415 images. The dataset selected to do the face detection model can be found at https://www.kaggle.com/andrewmvd/face-

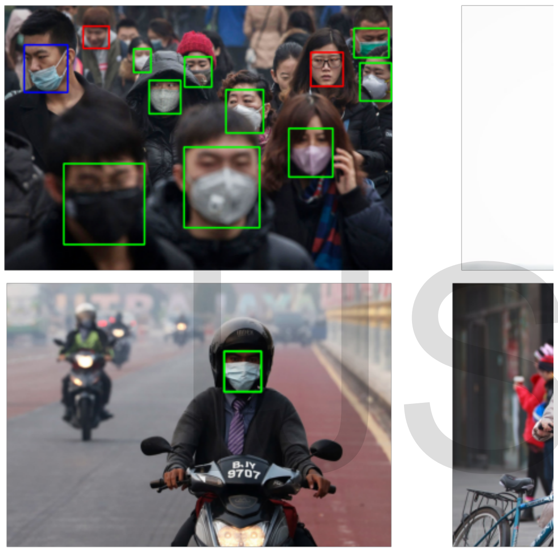


Figure 8. Sample images from Face Mask Dataset

mask-detection the dataset had BBOX annotations in XML format for masked, unmasked, and incorrectly wearing masks.

X. ENVIRONMENT

A free Google Integrated Development Environment (IDE) for artificial intelligence research and learning is known as Google Colab (Google Collaboratory) (AI). Collaboratory is a coding environment like Jupyter A notebook that lets you use the GPU and TPU for free (TPU). PyTorch, Tensor Flow, Keras, OpenCV and Deep Learning are a few of the prominent libraries that are installed in Google Colab by default. GPUs are not typically included in normal PCs because advanced machine learning and deep learning methods require a computer with much faster and more powerful computational capabilities. (Typically based on GPU). As a result, Colab provides GPU (Tesla V100) and TPU (TPUv2) on the cloud, one of the most powerful GPUs available at the time, to AI researchers.

The YOLOv5 architecture, like previous versions, was theoretically based and published through a GitHub repository. As previously stated, Ultralystic bases YOLOv5 on the PyTorch framework, which is one of the most widely used in the AI community. However, since this is just a prototype design, researchers may tweak it to get the best results for issues by adding layers, deleting blocks, integrating other image processing techniques, altering optimization methods or activation functions, and so on.

XI. PREPARING THE DATASET FOR TRAINING

The Mikolaj Witkowski's Medical Masks Dataset (MMD) (https://www.kaggle.com/vtech6/medical-masks-dataset) dataset has been made available for non-commercial usage. Because Colab is a Google service, it may be linked to a personal Google Drive account to access data from Drive for model training and to store the findings afterward. Following the download of the MMD dataset to a personal computer (PC), the data is published to Google Drive and connected to Colab. When you unzip the dataset in Drive, be sure to place it in the YOLOv5 folder. By allowing easy retrieval of input data as well as the ability to clear data (already contained in the YOLOv5 folder) to avoid unnecessary usage, we have made it easier to find information for use as input and to get rid of everything once the project is finished.

A. The MMD dataset contains

- train.zip – 980 outdoor images use for training.
- test.zip – 10 outdoor images use for the test.
- train.csv – training data.

Read the data from the train.cvs file. A bounding box's data is represented by each line. Because of the great density of face masks, each picture may include several face mask heads to identify, resulting in multiple bounding boxes. Each column of data corresponds to a unique picture id, image width, height, and bounding box data [xmin, ymin, width, height]. In the object detection dataset, the data file is a typical format for bounding boxes. The bounding box data in YOLO is structured as [class, xcenter, ycenter, width, height]. The centre point (xcenter, ycenter) may be determined using the minimum. point (xmin, ymin):

The authors of the MMD dataset do not identify the bounding boxes since this is an issue of single object detection, which is the face mask head. However, since YOLO requires a label argument, the bounding boxes must be labelled with the classes' names. Because class numbers are zero-indexed (beginning at 0), label '0' indicates the face mask head. Because all bounding boxes are in charge of identifying face mask, they are all labelled '0.' Box coordinates must be normalized in the range 0-1. As a result, xcenter and width are split by picture width, while ycenter and height are divided by image height. Now is the time to get rid of any redundant parameters. As seen in Figure 4.8, each column of data now corresponds to the unique picture id, class, xcenter, ycenter, width, and height of the bounding box.

B. Creating the label text files

Each line in data frame df constituted a bounding box, as previously stated. One picture may have many boxes

C. Changing the labels

This will be determined by how you want your model to behave, however as previously said, I've simplified the issue by restricting the labels to mask or no mask. Furthermore, since the label mask worn incorrectly occurs just a few times in the dataset, the model is unlikely to correctly categories it. this little script will do the following:

without mask	# label 0
with mask	# label 1

Remove duplicates from the picture id. The bounding box data is read from file text (.txt) rather than file.csv by YOLOv5 in PyTorch. As a result, the data of all bounding boxes in a single picture should be clustered and put in the same image-specific file text. After writing, all label text files will be in the MMD/label directory. Images contained in bounding boxes (e.g., items like cars, cell phones, bottles, hats, animals, and so on) are the only images written in CSV files in the MMD dataset, which has 800 images. On the other hand, photos of weeds may be useful in training the model because they keep the model from being misled by objects that resemble the face mask. Images that contain objects are positive, while images that do not contain objects are negative. For the YOLOv5 model to access and utilize the images during training, each negative image must have a label text file associated with it. None of these images include any objects, so there are no boundary boxes. Thus, their labeling text files will be blank.

D. Split Data into a training set and authentication set

To make a neural network model more accurate, a dataset that contains the same kind of item as the trained object is required. To gather a new dataset of images and text, you will need time and, in some cases, may be impossible to complete for an AI researcher. For instance, it is not always possible for researchers to collect an independent dataset for model validation when training is complete. This practice tends to lead to visually captured images from the dataset. On the usual training/assessment split, we find 80-20, with 80% of the dataset being used for training and 20% for assessment. In the absence of data from training, the model's performance will be evaluated using 20% of previously unseen data. This document contains an indisputable guarantee that the review will be fair. Training set: used in training; validation set used in the model assessment. Use 80% of the dataset to train, and 20% for validation. The file name of the picture and label image files are the same. This will ensure that labels and corresponding images are moved along with training and validation sets to their respective folders.

E. Creating the data.yaml file

All the images in the dataset are read in to PyTorch's YOLOv5 model, which is a neural network designed to recognize faces, through a yaml file that contains metadata about the entire dataset. This is the data's format. YOLO's model is based on a YAML configuration file. Populating the dataset is essential because the dataset does not include a data.yaml file. Data.yaml files are normally written in Notepad or Notepad++ and saved in YAML format. Then the data.yaml file is uploaded to Drive. However, the content will be created in Colab within this paragraph. The function must be imported from iPython.core.magic if the empty yaml file needs to be replaced.

XII. TRAINING PERIOD

Being Prepared for Architecture Many of Glenn Jocher's YOLOv5 model examples are also found in prior theories. YOLOv5 is trained on PyTorch with the help of the yaml file, and the built models are read from this file and used by YOLOv5. By implementing this, the architecture used for different object identification issues is simplified considerably.

A. Training model

Now it's time to feed the model with custom data. The train.py script is in the /yolov5 directory, so begin by locating it. A word of caution while scripting:

- The picture sizes must be specified in pixels (415 in our case). Because 415 is not a multiple of the specified maximum stride of 32, the model will stride each picture to 416 pixels.
- Specify the batch size. We'll keep it at 16 in this instance.
- The epoch count must be provided. The more epochs there are, the more certain the result will be. While the dataset is small, there is no way to avoid overfitting. As it stands, 30 epochs will be sufficient; otherwise, some overfitting will become evident.
- The —data option points to the. Yaml file that includes the dataset settings.
- To utilize the pretrained model in your training pipeline, you'll need to define it. YOLOv5s (yolov5s), YOLOv5m (yolov5m), YOLOv5l (yolov5l), and YOLOv5x (yolov5x) are the possible choices (yolov5x). In our situation, the first choice is the best since we need a model that is very light.
- Finally, —nosave just saves the final checkpoint, whereas —cache instructs the pipeline to cache pictures to speed up training. I won't use —nosave since I want to retain the best metric.
- Keep an eye on the mAP@.5 metric throughout training to get a sense of how well it's doing (mean average precision). If it's close to 1, the model is doing well!

Run the following commands to train the model:

This first line of code does some simple tests on our dataset, caches the images, and then applies any other parameters. They will generate a model architecture summary, and they will start training. During training, you should save two files at /content/yolov5/runs/train/exp/weights: last.pt and best.pt. Let's employ best.pt.

I recommend Tensor Board, which is a highly interactive visualization tool for looking at metrics that were captured during training.

The following reasons are in support of:

- img: provide the size of the input picture. The original picture size is 1024 by 1024 pixels; compressing the image to a smaller size speed up the training process. A lot of computer vision experts agreed that size 416 is the best option for handling a multitude of different input values without losing a significant amount of information.
- batch: estimate the batch size When there are thousands of pictures fed into the neural network at the same time, the number of weights the model learns in one time (one epoch) increases dramatically. To achieve the desired results, training data is typically partitioned into several training sets of n images, with each training set trained separately. After training all the batches, the results of each batch are stored in RAM, and then the aggregated results are written to disc. The weights that you've trained using the batches are saved in RAM, so as you train more batches, you'll need more memory.
- The training set includes 2738 pictures; with a batch size of 32, there will be 2738 32 = 86 batches.
- epochs: How many training epochs should we use? An epoch oversees all of the images it is being trained on or being trained on all of the images. The dataset is composed of multiple batches, so the training will be spread across epochs. When the model trains all the inputs and makes updates to the weights to come closer to the ground truth labels, it is measuring the number of epochs. You'll find that experience and intuition are commonly used to make decisions. Epochs are typically greater than 3000.
- data: The dataset summary is found in the data.yaml file, which is located at the relevant location. Since the model evaluation procedure runs immediately after each epoch, the model will access the validation directory through the location in the data.yaml file and utilize its contents for evaluation.
- cfg: Our model's configuration is found on the path to our model. In this command line, the train.py file is made available so that training input pictures based on the model yaml file's architecture can be constructed.
- weights: provide a strength training route. Using a pre-trained weight can help you save time. The name of the folder where the results are stored.
- name: The name of the folder where the results are stored. It will create a training results directory.
- cache: pictures are cached to make training go quicker.

B. *Testing the Model on Google Colab*

Let's have a look at how confident our model is right now. We may examine the confidence score of each label by plotting a validation batch acquired during training:
Python
Copy Code

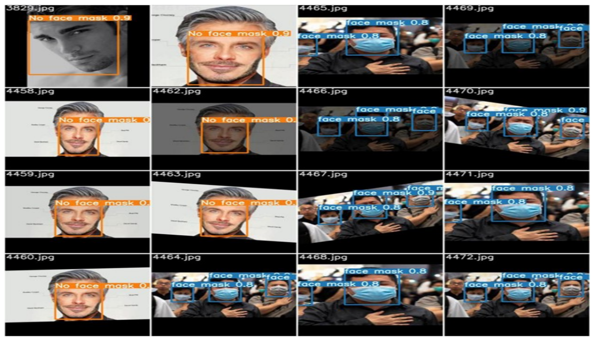


Figure 9. Testing the Model on Google Colab

Image(filename='runs/train/exp/test_batch0_pred.jpg', width=1000)
That will plot the following:
0.8 is a respectable score in these kinds of models, even if it isn't perfect. Keep in mind that YOLOv5 makes a trade-off between accuracy and detecting speed. We now have a model who has been properly trained. Create a new directory (for example, /content/test images), upload some images, and execute the following code to test it on fresh, unseen pictures:
Python
Copy Code
%load_ext tensorboard
%tensorboard --logdir runs
You'll use the detect.py script with the best.pt weights and picture size of 416x416 pixels (it's critical to follow these guidelines). The output will be stored in the directory runs/detect/exp. Run the following code to see the results:
Python
Copy Code
#Plotting the first image
Image(filename='runs/detect/exp/testimage1.jpg', width=415)

C. *The result of testing Model*

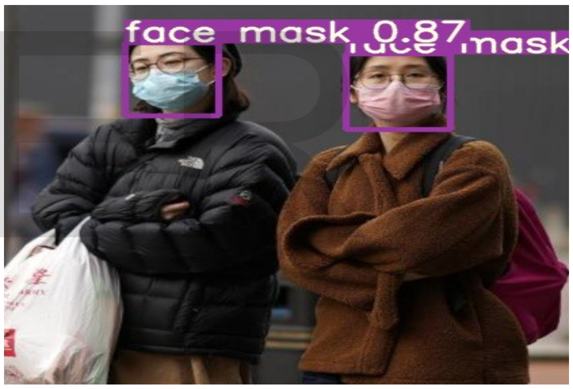


Figure 10. The result of testing Model

D. *Run YOLOv5 Inference on Test Images*

Our training model may now be used to understand examples from training photos. Once training is accomplished, recorded model weights will be referred to as weights/ Inference using these models uses those load and calibration settings coupled with a configuration (conf) that specifies confidence in the model (more confidence calls for fewer predictions) and an inference source. PICTURES (collective term for photos), MULTIMEDIA FILES (movies, songs, and video files), VIDEOS (videos), and the CAMERA PORT (devices only) are all permitted as sources. I've relocated our test/*jpg to test infer/ as a source. Any picture may be used to identify the face mask head using trained weights. If a face mask head is identified, a bounding box is created around the item to enclose it and show the likelihood that it is a face mask head. The MMD dataset contains 10 pictures of outdoor face masks that are non-duplicated with the 980 images used in the training. The ground truth bounding boxes are likewise not labeled on these pictures. Because of their appearance, these photos could be thought of as visual representations of a farmer in the field and evaluated using weights learned in the face mask head identification model earlier.

Using learned weights to conduct object detection is comparable to training the model. The detect.py file will be built using the command indicated, and it will recreate the architecture used in the training. With 93.5 percent accuracy, trained weights will anticipate items and limit boxes for them. Once the detection is complete, the expected bounding boxes that cover the objects (face mask heads) will be added to the picture. the training results will be placed in the same folder as the completed phases. Very good results have been obtained using the method for the first time. The model identifies that no matter how dense the face mask heads appear in the images, each picture reveals all the heads. However, due to the model's 93.5 percent accuracy, some face mask heads were still missed in detection. Although the face mask heads are accurately predicted, the chance of their occurring is low, as seen in Figure 4.25. As previously stated, the model forecasts an item based on its likelihood; if the probability is less than a specified threshold (here, 0.4), the model concludes that the object is not a face mask head. Face mask heads might have a lower probability of being face mask heads and be placed in the "no-face mask head" category if their prediction probability around the threshold is below the threshold. This is often the case in one of the ten anticipated images. Despite advances in computer vision, there is still much discussion surrounding the name and improvements to YOLOv5. Regardless of the terminology, YOLOv5 offers similar performance characteristics to YOLOv4. YOLOv5 will have a big future because it has a built-in Pytorch framework that is much more user-friendly and popular than the Darknet framework. The field of computer vision, especially object detection, has only recently seen a massive increase in research. The YOLO algorithm, which has undergone five generations of evolution, has not entirely met the company's expectations. Therefore, it is impossible to only design an AI system around an algorithm; for the AI system to succeed, other optimization techniques and cutting-edge concepts in computer vision should also be used.

E. RESULTs

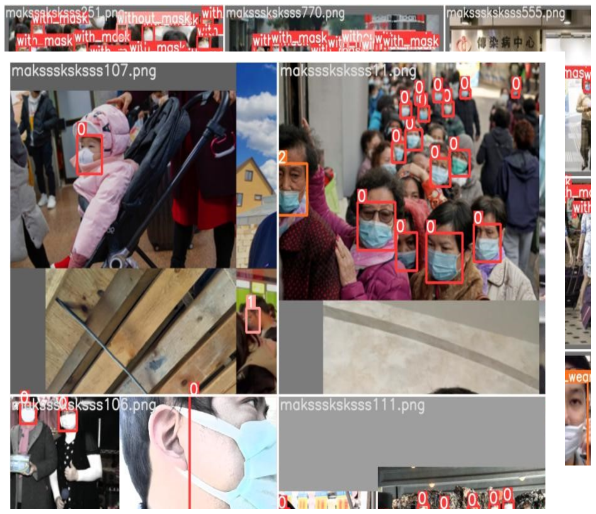


Figure 12. Recognize faces wearing mask or not

Examine the results to see what kinds of results have been produced. Let's keep things simple by first focusing on the developed model's accuracy. The classifier correctly predicted positive results is calculated by dividing the number of positive outcomes by the number of positive results it predicted. The model's performance is seen in the precision graph, as the precision value rises throughout the training process. The recall value of the created model is shown in the graph above. The recall is defined as the proportion of accurate positive findings to all samples determined to be positive.

Both YOLOv5s and YOLOv5x were used to test the model. Running the model via Google Colab finished the training process (An instance of the Google-developed virtual machine that is loaded with additional RAM and GPU memory for executing machine learning programs). Because my laptop is extremely old and does not have enough RAM or GPU memory to run this model, I had no choice but to use Google Colab. In terms of performance and speed, YOLOv5s outperforms YOLOv5x. In both methods, the mAP is quite comparable. When it comes to processing speed, the YOLOv5s is somewhat faster than the YOLOv5x.

XIII. FACENET-STYLE METHOD TO FACIAL RECOGNITION

To start with, I decided to use face recognition because of FaceNet. FaceNet is a face recognition pipeline that use face mapping algorithms to establish the spatial relationships between faces. "[29] FaceNet aims to locate an embedding for an image in a feature space such that the squared distance between all the face feature points of a particular identity is small, regardless of imaging conditions, while simultaneously locating two images of the same identity so that the squared distance between the pair of image feature points is large. FaceNet has a TensorFlow implementation accessible on GitHub, which enables the faces of one

International Journal of Scientific & Engineering Research, Volume 13, Issue 11, November-2022
ISSN 2229-5518

identity to reside on a manifold while still maintaining the distance and therefore discriminability to other identities."

Our approach is structured in three phases, building on prior work on FaceNet:

1. Pre-processing is a technique for converting a collection of pictures into a single, consistent format, in this instance, a square image featuring just a person's face. Because we have limited computing resources when utilizing the Edge TPU, a consistent dataset helps reduce variance while training. Our method is developed in three phases, building on prior work on FaceNet.

2. Embedding is a technique that learns representations of faces in a multidimensional space where distance correlates to a measure of face similarity. It is fundamental to how FaceNet works.

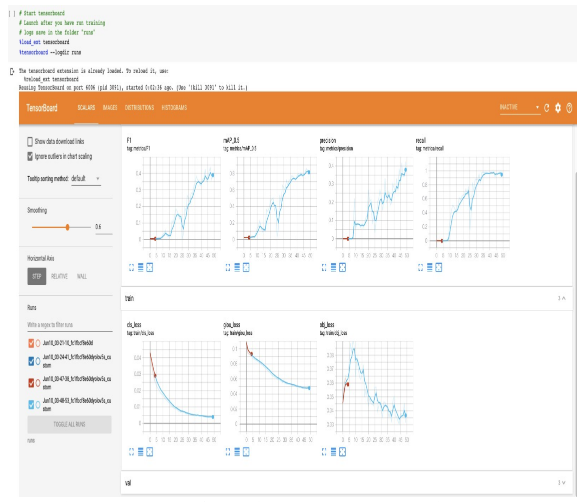
3. Classification is the last stage, which utilizes the information gathered throughout the embedding process to distinguish between different faces. Weight imprinting is another feature that we'd want to add. Weight imprinting is a process of training a model in two phases. In the first phase, the model is trained on a large dataset, which allows the incorporation of the first step's data. Then, the second phase is dedicated to training the last layer, exclusively using fresh data while relying on the data from the previous step. Learning with just a few sample photographs is possible with weight imprinting. It can memorize an unknown face's distinguishing features and compare them to those from a dataset known to it.

A. Results

Due to the lack of a huge sample size, we were not able to implement a complete model, but we achieved 50% accuracy ratio on face recognition.

XIV.EVALUATION

Analyze the performance of a custom YOLOv5 detector
Now that we've finished training, we can look at the validation metrics to see how well the training process worked. Tensorboard logs will be dropped in runs by the training script. Here's how we see them:



```
# Start tensorboard
# Launch after you have run training
# logs save in the folder "runs"
%load_ext tensorboard
%tensorboard --logdir runs
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
Reusing TensorBoard on port 6006 (pid 3091), started 0:02:36 ago. (Use '!kill 3091' to kill it.)
```

Figure 13. Analyze the performance of a custom YOLOv5 detector

The project provides the plot results() function, which enables you to assess your model's performance on the most recent training run:
1 from utils.utils import plot_results
2 plots_results();

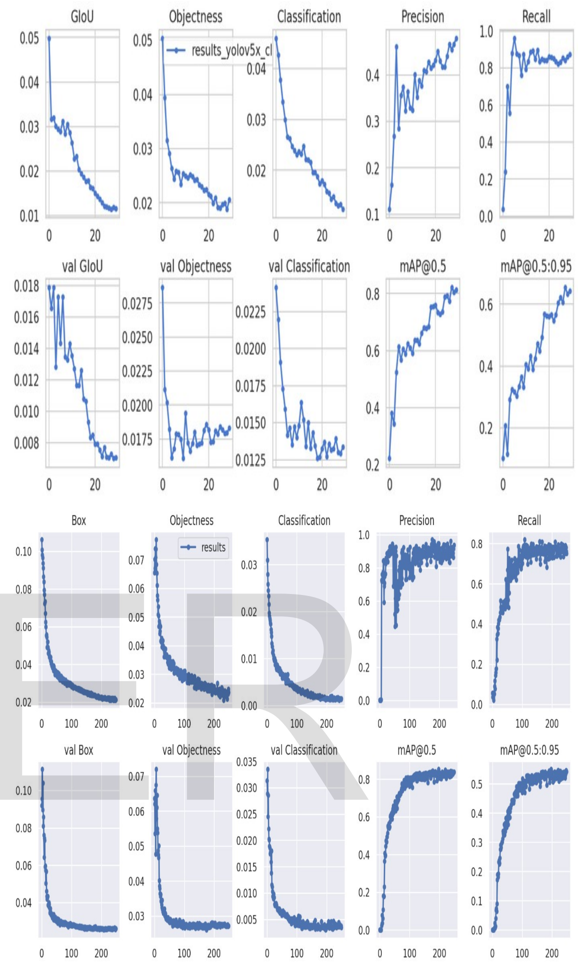
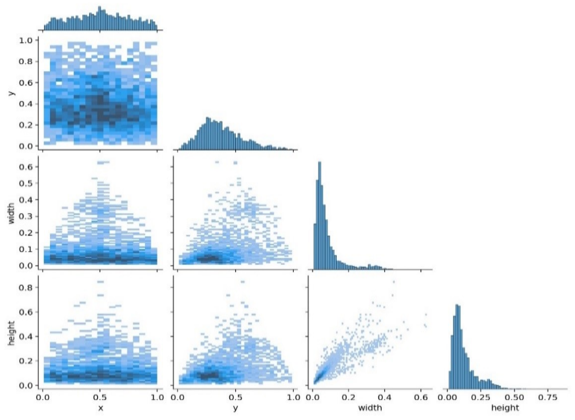


Figure 14. Performance of a YOLOv5 detector

XV. THE MEAN AVERAGE

Precision (mAP) seems to be improving during the training. More training for the model would be beneficial, but it is enough with a dataset comprising 980 pictures, the model takes approximately 20 seconds to complete one epoch, and the model's accuracy is about 93 percent with just 100 epochs, as illustrated in Figures 53 and 55. This demonstrates that, even without any optimization techniques, the YOLOv5 model is not only quick but also accurate. The model also stores two weighting results as a pt file. The weight at the final epoch is last.pt, while the weight at the last epoch with the greatest accuracy is best. pt, as illustrated in Figure 56. Both files are just 14MB in size, making it extremely easy to incorporate as a pretrained weight into AI systems (in online or mobile applications) while retaining 93.5 percent accuracy.

International Journal of Scientific & Engineering Research, Volume 13, Issue 11, November-2022
ISSN 2229-5518



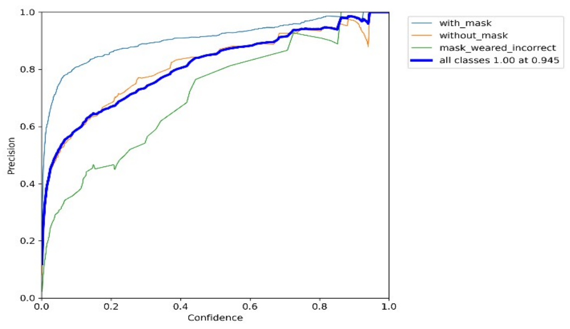


Figure 15. The Mean Average

XVI.COMPARE THE YOLOV5 MODEL'S PERFORMANCE WITH THE YOLOV5M MODEL'S PERFORMANCE

The two models have similar speeds. The YOLOv5m model performed better in terms of accuracy. The two models were run on both datasets to get the best possible score. Model differences in terms of mAP@0.5, mAP@0.5:0.9, precision, and recall are shown in Figure 5 for the dataset with 900 images. Model YOLOv5s has blue assigned to it, and model YOLOv5m has orange assigned to it. YOLOv5m shows to be more stable during the experiment while in motion. On the other hand, both models ultimately lead to the same destination.

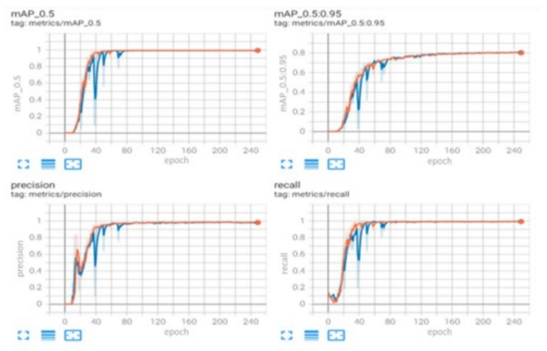


Figure 16. Performance Comparison between YoloV5s and Yolov5m

In figure 16, YOLOv5s (on the left) is compared to YOLOv5m (on the right). YOLOv5m are shown in orange and the YOLOv5s are shown in blue. To summaries, the plots demonstrate that as epochs progress, the models learn progressively, as the training performance increases. The loss function depicts the classification accuracy of a given predictor in a dataset. A better thinker is better at displaying relationships as the loss diminishes, there is less between input data and output targets. The loss that is shown in Figure 4.35 is differentiated into two categories. When calculating the loss at the top, it's calculated by considering both the predicted bounding box and the loss incurred due to an object in a particular cell being present during training. Val Box and Val Objectless' graphs are visual representations of their validation scores. Training loss is measured while performing each epoch of training, and validation loss is measured after the training has finished.

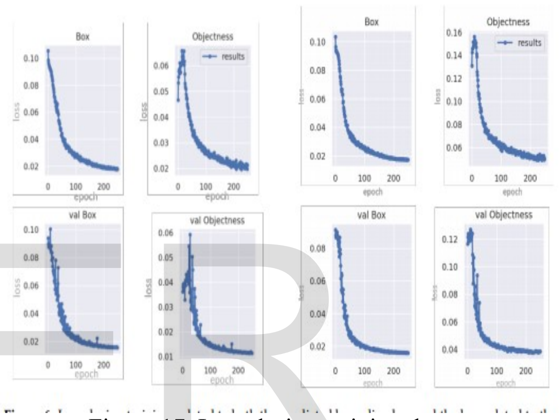


Figure 17. Loss during training data

The values of the dataset on the left show model YOLOv5s, and those on the right show model YOLOv5m. Based on the charts, it is evident that both models are virtually identical. Regarding stabilizing the learning process, the YOLOv5m is much faster than the YOLOv5s model. The figure illustrates the lost incurred due to the predicted bounding box (a cell containing an object and the class loss for the two models, YOLOv5s and YOLOv5m, on the dataset with 900 images), as well as YOLOv5s and YOLOv5performance m's on the dataset. The results for YOLOv5m are orange, and the results for YOLOv5s are blue. More importantly, these findings show that the evolution of loss is remarkably similar between the two models. The lines overlap almost completely. YOLO deep network for object detection and augmented reality glasses were proposed as a task assistant. Real-time use is possible due to the object detection model's

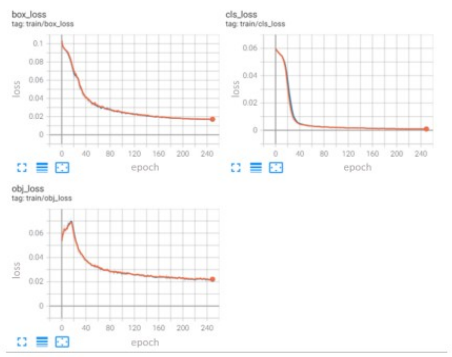
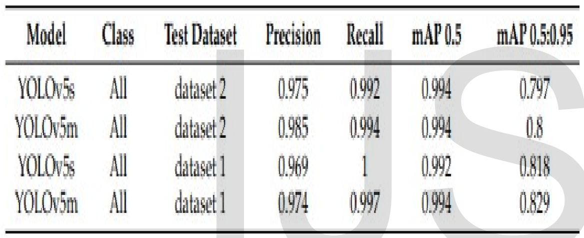


Figure 18. Loss functions for both the models YOLOv5s (blue) and YOLOv5m (orange).

capability of learning quickly and outputting a prediction in a fraction of a second. The precision of the two models matches other research on similar problems, which found a precision of 0.803, 0.89 for recall, and 0.905 for mAP 0.5. With the current prediction time of 0.007 seconds, you can get up to 140 frames per second on a TESLA P100 GPU. These results show that the system of detection integrated with the architecture, which is intended for use in real-time, has been successfully integrated into the maintenance work environment. In Table 6, both models, YOLOv5s and YOLOv5m are compared for detection for all classes on both datasets. Training and testing the models with the largest dataset yielded the first two lines. During training, 571 images were used, with 159 of them serving as test images. After training, a total of 571 images were used, with 159 of them used as test images. These are the smaller dataset results.

The main goal of the present work was to train a neural network for deep learning, which could be used as a foundation for implementing an augmented reality system to aid maintenance industry professionals. To train the model, two separate datasets were created: one using two different devices under different illumination conditions, and the other



Model	Class	Test Dataset	Precision	Recall	mAP 0.5	mAP 0.5:0.95
YOLOv5s	All	dataset 2	0.975	0.992	0.994	0.797
YOLOv5m	All	dataset 2	0.985	0.994	0.994	0.8
YOLOv5s	All	dataset 1	0.969	1	0.992	0.818
YOLOv5m	All	dataset 1	0.974	0.997	0.994	0.829

Figure 19. Tests with YOLOv5s YOLOv5m for both datasets

using two different devices in the same illumination conditions. Additionally, two different versions of YOLOv5 were also investigated, and it was discovered that YOLOv5 is enough for the problem the device is intended to resolve. YOLOv5s shows almost identical results in tests with the larger model and has shown to be able to recognize eight different mechanical parts in a car engine with high precision and recall always above 96.8 percent. All test results and analyses show that the network is sufficiently fast and stable to apply to the proposed system. Future work will involve the trained model being integrated into the CMMS so that the technician can be directed through the procedures of the working order while in the virtual reality environment.

XVII. CONCLUSION AND FUTURE WORK

Deep Learning is still evolving as a wonderful method that will help people not only recognize faces in photos, videos, speech, and audio translations but even create a new species with brains that are like humans. It's simply the most basic and important idea about which I've spoken and studied. Without a doubt, existing neural network models provide the basis for future deep learning success. Models such as CNN, Deep CNN, and other well-known training algorithms, as well as different noising or de-noising encoders and decoders, are used to describe deep learning methods. Deep learning has not only shown promising results in a variety of applications, but it also has a bright future in terms of holding success and wider features. Deep learning can still handle a variety of problems, including increasing face recognition accuracy, speech detection and translation, network training, and machine training, all of which will lead to more effective and theoretical deep learning techniques. We developed a technology that can identify whether people are wearing masks. It's appropriate for usage in stores and public spaces. This helps in the battle against the propagation of COVID-19 disease. Because holding a mask prevents the COVID-19 virus from spreading in the population. This may be used for several reasons, including ensuring that all customers are wearing facemasks. Using the YOLOv5 and TensorFlow technologies, the developed model processes images. The developed model, according to the results, can identify whether a person is wearing a facemask. Our research team presented a novel technique based on YOLOV5 enabling applications to identify faces wearing masks, as well as conventional machine learning models for comparison after testing, with a success rate of about 97.9%. The result is shown in the results section. There is also a recognizable picture of someone wearing a mask but not concealing their nose. We believe that this method may successfully reduce exposure distances and perform effective monitoring because of COVID-19's global impact. This technique has many limitations. It can sometimes detect whether someone is wearing a mask, for example. It comes in handy at places like supermarkets and airports. Improving the system's capacity to detect people's faces who aren't directly facing is one unresolved problem. Even though half of the faces are obscured by masks, this technique yielded accurate and quick results for facial recognition security systems. The test results show that identifying individuals who are wearing a face mask, is not wearing a face mask, or are wearing a face mask improperly is quite accurate. The model achieved a performance accuracy of 97.1 percent, which is an outstanding achievement. Furthermore, the study offered a useful tool in preventing the spread of the COVID-19 disease by allowing everyone to wear a face mask while completing biometric identification. Face recognition utilizing a face mask has grown increasingly essential in the past year because of the COVID-19 virus's appearance. The detection of social distance and notifying if someone is not wearing a face mask properly are two of our next initiatives. The research aims to come up with a technique for creating a masked facial recognition system. The algorithm will next trim the retrieved visual part using the FaceNet pre-trained model. There is still a lot of discussion in the computer vision industry about the nomenclature and improvements of YOLOv5 for advancements that haven't reached a breakthrough. Regardless of the terminology, YOLOv5 performs similarly to YOLOv4 in terms of speed and accuracy. There's little doubt that YOLOv5 will get more contributions and have a larger potential for development in the future. Computer vision, especially object identification, is a very new field. Therefore, the YOLO algorithm, while being one of the finest object identification algorithms and having evolved over five generations, is still flawed.

Therefore, an AI system cannot be built only based on an algorithm; it must also include other optimization methods as well as cutting-edge ideas in the field of computer vision to achieve optimum performance.

Acknowledgment

To my Beloved Parents who have supported me since the beginning of my studies with their endless love support and encouragement, they brought us to the level of excellence, where we all stand today, looking for the most promising future ahead, for which they sacrificed their most past. And to all my respected teachers who guided us throughout university life Finally, this thesis is dedicated to all those who believe in the richness of learning

References

[1] World Health Organization. (2020). Coronavirus disease 2019 (COVID-19): situation report, 72.
[2] Yang, Guanhao, et al. "Face Mask Recognition System with YOLOV5 Based on Image Recognition." 2020 IEEE 6th International Conference on Computer and Communications (ICCC). IEEE, 2020.
[3] Brosnahan, Shari B., et al. "COVID-19 and respiratory system disorders: current knowledge, future clinical and translational research questions." Arteriosclerosis, thrombosis, and vascular biology 40.11 (2020): 2586-2597.
[4] Feng, Shuo, et al. "Rational use of face masks in the COVID-19 pandemic." The Lancet Respiratory Medicine 8.5 (2020): 434-436.
[5] Ho, Hanley J., et al. "Use of surveillance technology to enhance exposure management for healthcare workers during the COVID-19 pandemic." The Journal of Hospital Infection 107 (2021): 101.
[6] Nishiura, H., Kobayashi, T., Miyama, T., Suzuki, A., Jung, S. M., Hayashi, K & Linton, N. M. (2020). Estimation of the asymptomatic ratio of novel coronavirus infections.
[7] Gupta, Neha. "Artificial neural network." Network and Complex Systems 3.1 (2013): 24-28.
[8] Loey, Mohamed, et al. "Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection." Sustainable cities and society 65 (2021): 102600.
[9] S. V. M. a. N. V. Dionisio, "Real-Time Facemask Recognition with Alarm System Using Deep Learning," IEEE Control and System Graduate Research Colloquium (ICSGRC),2020.
[10] Bruce, Vicki, and Andy Young. "Understanding face recognition." British journal of psychology 77.3 (1986): 305-327.
[11] Rowley, Henry A., Shumeet Baluja, and Takeo Kanade. "Neural network-based face detection." IEEE Transactions on pattern analysis and machine intelligence 20.1 (1998): 23-38.
[12] Boulos, Mira M. "Facial Recognition and Face Mask Detection Using Machine Learning Techniques." (2021).
[13] Benedict, Saranya R., and J. Satheesh Kumar. "Geometric shaped facial feature extraction for face recognition." 2016 IEEE International Conference on Advances in Computer Applications (ICACA). IEEE, 2016.
[14] V. Varshney, "CNNs: The Key to Computer Vision," Data Driven Investor, 14 February 2020.
[15] Mosavi, Amir, Sina Ardabili, and Annamaria R. Varkonyi-Koczy. "List of deep learning models." International Conference on Global Research and Education. Springer, Cham, 2019.
[16] Chen, W., Huang, H., Peng, S., Zhou, C., & Zhang, C. (2021). YOLO-face: a real-time face detector. The Visual Computer, 37(4), 805-813.
[17] Yang, W., & Jiachun, Z. (2018, July). Real-time face detection based on YOLO. In 2018 1st IEEE international conference on knowledge innovation and invention (ICKII) (pp. 221-224). IEEE.
[18] Ding, Yuchen, Zichen Li, and David Yastremsky. "Real-time Face Mask Detection in Video Data." arXiv preprint arXiv:2105.01816 (2021).
[19] Paszke, Adam, et al. "Pytorch: An imperative style, high-performance deep learning library." Advances in neural information processing systems 32 (2019): 8026-8037.
[20] Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.
[21] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proc. IEEE Conf. on computer vision and pattern recognition (CVPR), pp. 779-788. 2016.
[22] Smooth L1 Loss: https://github.com/rbgirshick/py-faster rcnn/files/764206/SmoothL1Loss.1.pdf.
[23] Mundial, Imran Qayyum, et al. "Towards facial recognition problem in COVID-19 pandemic." 2020 4th International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM). IEEE, 2020.
[24] Parkhi, Omkar M., Andrea Vedaldi, and Andrew Zisserman. "Deep face recognition." (2015).
[25] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.
[26] Pre-trainedfacenetmodel https://github.com/davidsandberg/facenet.
[27] Zhao, Z.; Zheng, P.; Xu, S.; Wu, X. Object Detection with Deep Learning: A Review. IEEE Trans. Neural Netw. Learn. Syst. 2019, 30, 3212–3232. [CrossRef] [PubMed]
[28] Jähne, Bernd, and Horst Haußecker. "Computer vision and applications." (2000): 111-151.
[29] William, Ivan, et al. "Face recognition using FaceNet (survey, performance test, and

comparison)." 2019 Fourth International Conference on Informatics and Computing (ICIC). IEEE, 2019.

[30] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[31] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

[32] Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). Fighting against COVID-19: A novel deep learning model based on YOLO-v2 with ResNet-50 for medical face mask detection. Sustainable cities and society, 65, 102600.

[33] Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. Sustainable cities and society, 66, 102692.

[34] Sanjaya, S. A., & Rakhmawan, S. A. (2020, October). Face Mask Detection Using MobileNetV2 in The Era of COVID-19 Pandemic. In 2020 International Conference on Data Analytics for Business and Industry: Way Towards a Sustainable Economy (ICDABI) (pp. 1-5). IEEE.

[35] Sethi, S., Kathuria, M., & Kaushik, T. (2021). Face mask detection using deep learning: An approach to reduce risk of Coronavirus spread. Journal of Biomedical Informatics, 120, 103848.

[36] Ge, S., Li, J., Ye, Q., & Luo, Z. (2017). Detecting masked faces in the wild with lle-cnns. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2682-2690).

[37] El-Bakry, H. M. (2006). Faster PCA for Face Detection Using Cross Correlation in the Frequency Domain. International Journal of Computer Science and Network Security, 6(2A), 69-74.

[38] Chavda, A., Dsouza, J., Badgujar, S., & Damani, A. (2021, April). Multi-stage cnn architecture for face mask detection. In 2021 6th International Conference for Convergence in Technology (I2CT) (pp. 1-8). IEEE.

[39] Girshick, R.B.; Donahue, J.; Darrell, T.; Malik, J. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 24–27 June 2014; pp. 580–587.

[40] Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. International journal of computer vision, 104(2), 154-171.

[41] Tompson, J., Goroshin, R., Jain, A., LeCun, Y., & Bregler, C. (2015). Efficient object localization using convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 648-656).

[42] Meuwese, J. D., van Loon, A. M., Lamme, V. A., & Fahrenfort, J. J. (2014). The subjective experience of object recognition: Comparing metacognition for object detection and object categorization. Attention, Perception, & Psychophysics, 76(4), 1057-1068.

[43] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. “Rich feature hierarchies for accurate object detection and semantic segmentation.” In Proc. IEEE Conf. on computer vision and pattern recognition (CVPR), pp. 580-587. 2014.

[44] Joseph Redmon, et al. “You only look once: Unified, real-time object detection.” CVPR 2016.

[45] Joseph Redmon and Ali Farhadi. “YOLO9000: Better, Faster, Stronger.” CVPR 2017.

[46] Huang, R., Pedoeem, J., & Chen, C. (2018, December). YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 2503-2510). IEEE.

[47] Qiao, S., & Ma, J. (2018). A Face Recognition System Based on Convolution Neural Network. In 2018 Chinese Automation Congress (CAC) (pp. 1923-1927). IEEE.

[48] CunLi, S., & Ji, S. (2021). An Unconstrained Face Recognition Method Based on Siamese Networks.

[49] Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

[50] Zhao, Zhong-Qiu, et al. "Object detection with deep learning: A review." IEEE transactions on neural networks and learning systems 30.11 (2019): 3212-3232.

[51] Jörgensen, E., Zach, C., & Kahl, F. (2019). Monocular 3d object detection and box fitting trained end-to-end using intersection-over-union loss. arXiv preprint arXiv:1906.08070.

[52] Chowdary, G. J., Punn, N. S., Sonbhadra, S. K., & Agarwal, S. (2020, December). Face mask detection using transfer learning of inceptionv3. In International Conference on Big Data Analytics (pp. 81-90). Springer, Cham.

[53] Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." Advances in neural information processing systems 28 (2015): 91-99.

[54] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards real-time object detection with region proposal networks.” In Advances in neural information processing systems (NIPS), pp. 91-99. 2015.

[55] Ross Girshick. “Fast R-CNN.” In Proc. IEEE Intl. Conf. on computer vision, pp. 1440-1448. 2015.

[56] Sharma, V. (2020). Face Mask Detection using YOLOv5 for COVID-19 (Doctoral dissertation, California State University San Marcos).

[57] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards real-time object detection with region proposal networks.”