

FACEMASK DETECTION USING JETSON NANO

A Project report submitted in partial fulfillment of the requirements

for the award of the degree of

BACHELOR OF

ENGINEERING IN

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

T.Pooja (317126512052)

D.Harika (3171726512017)

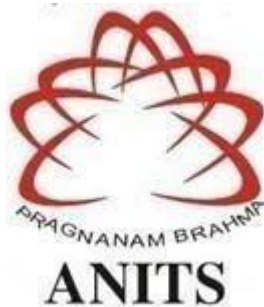
B.Yoganand (317126512010)

N.Hemanjali (317126512044)

Under the guidance of

Mr. B.Chandra Mouli

Assistant Professor, Department of ECE



DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA & NAAC with 'A' Grade)

Sangivalasa, Bheemili Mandal, Visakhapatnam Dist.(A.P) 2020-2021

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

ANIL NEERUKONDA INSTITUTE OF TECHNOLOGY AND SCIENCES

(Permanently Affiliated to AU, Approved by AICTE and Accredited by NBA
& NAAC with 'A' Grade)

Sangivalasa, Bheemili Mandal, Visakhapatnam Dist.(A.P)



ANITS

CERTIFICATE

This is to certify that the project report entitled "FACE MASK DETECTION USING JETSON NANO" submitted by T.Pooja(31712651052), D.Harika(317126512017), B.Yoganand(317126512010), N.Hemanjali(317126512044), in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Electronics & Communication Engineering of Andhra University, Visakhapatnam is a record of bonafide work carried out under my guidance and supervision.

HEAD OF THE DEPARTMENT

PROJECT GUIDE


Dr.V.Rajya Lakshmi


Mr.B Chandra Mouli

M.E., Ph.D., MIEEE, AMIETE, AMIE

M.Tech (Ph.D)

Professor
Department of ECE

Assistant Professor
Department of ECE

ANITS

ANITS
Assistant Professor
Department of E.C.E.

Head of the Department
Department of ECE

Anil Neerukonda

Anil Neerukonda Institute of Technology & Sciences
Sangivalasa - 531 162

Institute of Technology & Sciences
Sangivalasa, Visakhapatnam-531 162

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our project guide **Mr. B. Chandra Mouli**, Assistant Professor, Department of Electronics and Communication Engineering, ANITS, for his guidance with unsurpassed knowledge and immense encouragement. I Am grateful to **Dr. V. Rajyalakshmi**, Head of the Department, Electronics and Communication Engineering, for providing us with the required facilities for the completion of the project work.

I am very much thankful to the **Principal and Management, ANITS, Sangivalasa**, for their encouragement and cooperation to carry out this work.

I express my thanks to all **teaching faculty** of the Department of ECE, whose suggestions during reviews helped us in the accomplishment of our project. We would like to thank **all non-teaching staff** of the Department of ECE, ANITS for providing great assistance in the accomplishment of our project.

I would like to thank my parents, friends, and classmates for their encouragement throughout our project period. At last but not least, I thank everyone for supporting us directly or indirectly in completing this project successfully.

PROJECT STUDENTS

T.Pooja (31712651052)

D.Harika(317126512017)

B.Yoganand(317126512010)

N.Hemanjali (317126512044)

CONTENTS

ABSTRACT	i
LIST OF SYMBOLS	ii
LIST OF FIGURES	iii
LIST OF TABLES	iv
LIST OF ABBREVIATIONS	v
Chapter 1: Introduction	1-2
1.1 Project Objective	2
1.2 Project Outline	2
Chapter 2: Literature Survey	3-5
Chapter 3: Deep Learning-Based Detectors	6-23
3.1 Introduction	7
3.2 Types of Detectors	8
3.2.1 Viola-jones Detection Framework Based on Haar Features	9
3.2.2 Histogram of Oriented Gradients	13
3.2.3 Single Shot Detector	14
3.3 Object Detection Framework	15
3.3.1 ResNet	17
3.3.2 Feature Pyramid Network	19
3.3.3 Classifiers, Predictors, Estimators	20
3.4 Convolutional Neural Network	21
3.5 Context Attention Model	23

CHAPTER 4: HARDWARE TOOLS	24-35
4.1 Jetson Nano	25
4.1.1 Introduction	25
4.1.2 Overview	25
4.1.3 Carrier Board Layout	26
4.1.3.1 40 Pin Header	26
4.1.3.2 8 Pin Bottom Header	27
4.1.3.3 4 Pin Fan Header	28
4.1.4 Networking	29
4.1.5 Camera	29
4.1.6 Power	29
4.1.6.1 Power Consumption	30
4.1.7 Specification	30
4.1.8 Board Description	31
4.2 Logitech Camera Specifications and Features	33
4.3 USB Cable	34
4.4 Micro SD Card	34
CHAPTER 5 SOFTWARE TOOLS	36-51
5.1 Setup and Boosting	37
5.2 Virtual Environment	46
5.3 Tech Stack /Framework Used	48
CHAPTER 6 METHODOLOGY	52-57
CHAPTER 7 RESULTS	
CONCLUSIONS	
REFERENCES	
PAPER PUBLICATION DETAILS	

ABSTRACT

The World is suffering from the global pandemic Coronavirus. The infection is spreading quickly. One way to stop the spread by wearing a mask. Face Mask Detector detects whether a person is wearing a mask properly or not by image analysis. In this paper, we adopt a high efficient and accurate facemask detector, which is Retina Face Mask. To fuse high-level semantic information, we adopt a one-stage detector consisting of a feature pyramid network with many feature maps and we adopted a novel context attention module to detect facemasks. To reject the predictions with low confidence, we adopt a novel cross-class object removal algorithm. . To reject the predictions with low confidence, we adopt a novel cross-class object removal algorithm. We propose NVIDIA Processor, i.e NVIDIA Jetson Nano implementation to carry out the work. We can use this Face Mask detector at the entrances to alert people about the face mask.

LIST OF FIGURES:

Figure 1.1 Types of Haar features

Figure 1.2 Converting Original Image into Integral Image

Figure 1.3 Cascade Amplifier

Figure 1.4 Single Shot Detector

Figure 1.5 Image Recognition V/S Object Detection

Figure 1.6 Skip Connections

Figure 1.7 ResNet

Figure 1.8 Feature Pyramid Network

Figure 1.9 Feature Extraction in FPN

Figure 1.10 Convolutional Neural Network

Figure 1.11 Max Pooling

Figure 1.12 Fully Connected Layers

Figure 2.1 Front View of Jetson Nano

Figure 2.2 Rear View of Jetson Nano

Figure 2.3 Layout of Jetson Nano

Figure 2.4 40 Pin Header

Figure 2.5 8 Pin Button Header

Figure 2.6 4 Pin Header

Figure 2.7 Camera Lanes

Figure 2.8 Logitech Camera

Figure 2.9 USB Cable

Figure 2.10 Memory Card

Figure 3.1 Context Detection Head

Figure 3.2 Workflow of Facemask Detection

CHAPTER 1

INTRODUCTION

INTRODUCTION

1.1 Project Objective:

The spread of COVID-19 is increasing in the world. This virus can be affected from one human to another through the droplets and airborne. According to the instructions from WHO, to reduce the spread of COVID-19, every person needs to wear a facemask, do social distancing, evade the crowd area and, also always maintain the immune system. So everyone should wear the mask properly. To overcome this situation, robust facemask detection needs to be developed.

1.2 Project Outline:

The main goal of the project is to implement this system at colleges, airports, hospitals, and offices where chances of spread of COVID-19 through contagion are relatively higher. For efficient network for Face Mask Detection. We adopt an Object detection framework, which suggests a detection network with a backbone, neck, head. The novel Attention Model is to detect the facemask. To improve the detection by the rejection of low confidence prediction we use the concept of a Novel cross-class object removal algorithm. This algorithm has been carried out by using the NVIDIA Jetson Nano board.

CHAPTER 2

LITERATURE SURVEY

As people across the globe are combating the widespread COVID-19 pandemic, it becomes very essential to develop new technologies to analyze and fight against the disease spread. The most essential protection against coronavirus is Face Mask and as the day surpasses scientists and Doctors have recommended everyone wear the mask. At that moment World health organization (WHO) have recommended that people should have respiratory symptoms or people who are taking care of people who have symptoms. Furthermore, many public service providers require customers to use the service only if they wear masks correctly. As per our knowledge, there are only a few research studies about facemask detection based on image analysis. In 2019, Implementation on the Principal Component Analysis on Masked and Unmasked Face Recognition proposed by Md. Sabbir Ejaz and Rabiul Islam, here they broke down a concealed and non-covered face acknowledgment precision by utilizing a standard segment investigation. The dataset utilized is Olivetti and Oracle research lab (ORL) face information base. Here PCA is utilized for highlight extraction. The means used in this work incorporate Facial Element Extraction and Facial Picture Procurement utilizing PCA and EigenVector Estimation. Subsequently, it gives a high acknowledgment pace of the face without masks. In 2019, Facial Mask Detection using semantic segmentation, which was determined by Roshan Lal Meena Pal, Ashutosh Balakrishnan, and Amit Verma utilized a facemask detection using semantic division. Here the class names are referred to as face or non-face. The convolutional neural organization VGG-16 engineering followed by a completely convolutional network is utilized for division. Accordingly, it perceives numerous faces. This strategy is helpful for frontal appearances moreover as non-frontal countenances. Thus, it focuses on error predictions. In 2020, performance evaluation of intelligent mask detection systems with various deep learning classifiers proposed by C. Jagadeeswari, M.Uday Theja. Here the presentation of facemasks identification utilizing profound learning calculations like distinctive profound learning classifiers could likewise be investigated mobileNet V2, ResNet 50, VGG 16, ADAM, SGD. These are the classifiers utilized for it. For every classifier followed by [3]. Streamlining agents and assess the presentation. The enhancers are utilized here like ADAM, ADAGRAD, SGD (Stochastic Slope Plummet). Thus, ADAM analyzer execution is staggeringly acceptable and says that the MobileNet V2 classifier has the best outcomes with high exactness. In 2020, Retinal mask Detector [4] proposed by Mingjie Jiang, Xinqi fan, and Hong, presents a Retinal Mask Finder. It is a One-stage object identifier. The dataset contained 7959 pictures. The ResNet and versatile

Net are utilized as Spine. However, ResNet is contemplated as a standard spine. The location network comprises a spine, a neck, and head modules. Our idea RETINA FACE MASK DETECTION was implemented on a Jetson Nano board. The proposed system develops classification and predictive models that can account for accurate classification, grouping, and prediction of Facemasks on the face of a person. The proposed system will focus on enhancing the prediction by increasing its accuracy and detection probability.

CHAPTER 3

DEEP LEARNING-BASED DETECTORS

DEEP LEARNING-BASED DETECTORS

3.1 Introduction:

Deep learning: Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision-making. Deep learning is a subset of machine learning in artificial intelligence that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

- Deep learning is an AI function that mimics the workings of the human brain in processing data for use in detecting objects, recognizing speech, translating languages, and making decisions.
- Deep learning AI can learn without human supervision, drawing from data that is both unstructured and unlabeled.
- Deep learning, a form of machine learning used to help detect fraud or money laundering, among other functions.

How Deep Learning Works:

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through Fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

Deep Learning vs Machine Learning

One of the most common AI techniques used for processing big data is machine learning, a self-adaptive algorithm that gets increasingly better analysis and patterns with experience or with newly added data. If digital payments

The company wanted to detect the occurrence or potential for fraud in its system, so it could employ Machine Learning tools for this purpose. The computational algorithm built into a computer model will process all transactions happening on the digital platform, find patterns in the data set, and point out any detected by the pattern.

Deep learning, a subset of machine learning, utilizes a hierarchical level of artificial neural networks to carry out the process of machine learning. The artificial neural networks are built like the human brain, with neuron nodes connected like a web. While traditional programs build analysis with data in a linear way, the hierarchical function of deep learning systems enables machines to process data with a nonlinear approach.

3.2 Types of Detectors:

Different Object Detection Methods: Methods for object detection generally fall into neural network-based or non-neural approaches. For non-neural approaches, it becomes necessary to first define features using one of the methods below, then using a technique such as a Support vector machine (SVM) to do the classification. On the other hand, neural techniques can do end-to-end object detection without specifically defining features and are typically based on convolutional neural networks (CNN).

- Non-neural approaches:
 - Viola-jones object detection framework based on Haar features
 - Histogram of gradients
- Neural network approaches:
 - Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN, cascade R-CNN.)
 - Single Shot Detector (SSD)
 - You Only Look Once (YOLO)
 - Single-Shot Refinement Neural Network for Object Detection (Refine Det)
 - Retina-Net
 - Deformable convolutional networks

3.2.1 Viola-jones Detection Framework Based on Haar Features:

Viola-Jones Algorithm: Viola-Jones algorithm is named after two computer vision researchers who proposed the method in 2001, Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

Given an image(this algorithm works on grayscale images), the algorithm looks at many smaller subregions and tries to find a face by looking for specific features in each subregion. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

The Viola-Jones algorithm has four main steps, which we shall discuss in the sections to follow

1. Haar Like Features
2. Integral image
3. AdaBoost Training
4. Cascade Classifier

1. Haar Like Features: In the 19th century a Hungarian mathematician, Alfred Haar gave the concepts of Haar wavelets, which are a sequence of rescaled “square-shaped” functions, which together form a wavelet family or basis. Voila and Jones adapted the idea of using Haar wavelets and developed the so-called Haar-like features. Haar-like features are digital image features used in object recognition. All human faces share some universal properties of the human face like the eyes region is darker than its neighbor pixels, and the nose region is brighter than the eye region. A simple way to find out which region is lighter or darker is, to sum up, the pixel values of both regions and compare them. The sum of pixel values in the darker region will be smaller than the sum of pixels in the lighter region. If one side is lighter than the other, it may be an edge of an eyebrow or sometimes the middle portion may be shinier than the surrounding boxes, which can be interpreted as a nose This can be accomplished using Haar-like features and with the help of them, we can interpret the different parts of a face.

There are 3 types of Haar-like features that Viola and Jones identified in their research:

1. Edge features
2. Line-features
3. Four-sided features

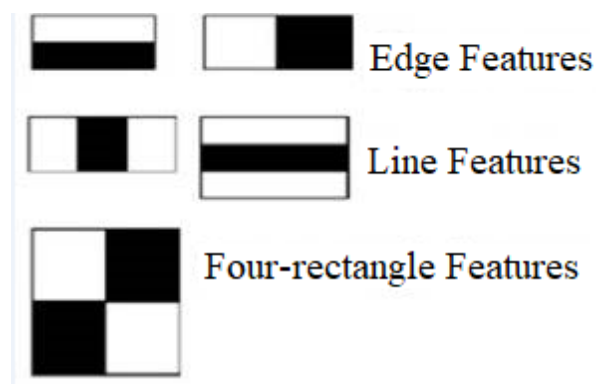


Fig 1.1 Types of Haar Features

Edge features and Line features are useful for detecting edges and lines respectively. The four-sided features are used for finding diagonal features.

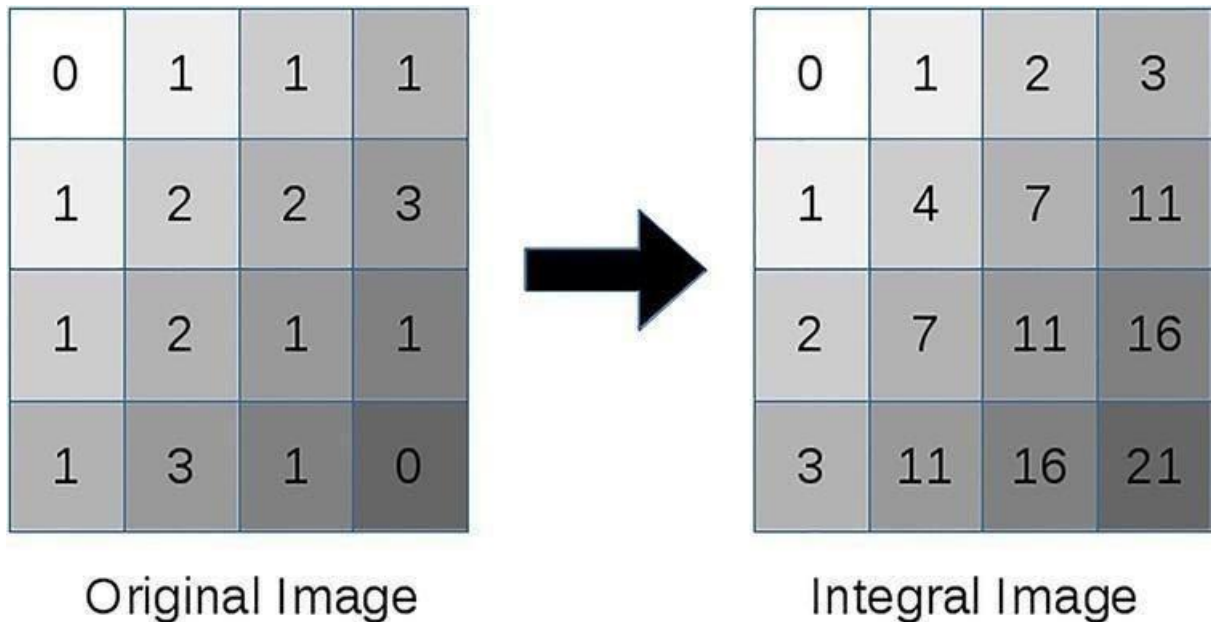
The value of the feature is calculated as a single number: the sum of pixel values in the black area minus the sum of pixel values in the white area. The value is zero for a plain surface in which all the pixels have the same value, and thus, provide no useful information. Since our faces are of complex shapes with darker and brighter spots, a Haar-like feature gives you a large number when the areas in the black and white rectangles are very different. Using this value, we get a piece of valid information out of the image. To be useful, a Haar-like feature needs to give you a large number, meaning that the areas in the black and white rectangles are very different.

2. Integral Image: The integral image plays its part in allowing us to perform these intensive calculations quickly so we can understand whether a feature of several features fits the criteria.

An integral image (also known as a summed-area table) is the name of both a data structure and an algorithm used to obtain this data structure. It is used as a quick and efficient way to calculate the sum of pixel values in an image or

the rectangular part of an image. In an integral image, the value of each point is the sum of all pixels above and to the left, including the target pixel:

Fig 1.2 Converting Original image to Integral image



Using these integral images, we save a lot of time calculating the summation of all the pixels in a rectangle, as we only have to perform calculations on four edges of the rectangle. See the example below to understand. When we add the pixels in the blue box, we get eight as the sum of all pixels, and here we had six elements involved in your calculation. Now to calculate the sum of these same pixels using the integral image, you just need to find the corners of the rectangle and then add the green vertices and subtract the vertices in the red boxes. Now doing that here

$$21 + 1 - 11 - 3 = 8$$

0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

We get the same answer and only four numbers are involved in calculations. No matter how many pixels are in the rectangle box, we will just need to compute on these 4 vertices. Now to calculate the value of any haar-like feature, you have a simple way to calculate the difference between the sums of pixel values of two rectangles.

3. AdaBoost Algorithm: The AdaBoost (Adaptive Boosting) Algorithm is a machine learning algorithm for selecting the best subset of features among all available features. The output of the algorithm is a classifier (a.k.a Prediction Function, Hypothesis Function) called a “Strong Classifier”. A Strong Classifier is made up of linear combinations of “Weak Classifiers” (best features). From a high level, to find these weak classifiers the algorithm runs for T iterations where T is the number of weak classifiers to find and it is set by you. In each iteration, the algorithm finds the error rate for all features and then chooses the feature with the lowest error rate for that iteration.

4. Cascade Classifier: A Cascade Classifier is a multi-stage classifier that can perform detection quickly and accurately. Each stage consists of a strong classifier produced by the AdaBoost Algorithm. From one stage to another, the number of weak classifiers in a strong classifier increases. An input is evaluated on a sequential (stage by stage) basis. If a classifier for a specific stage outputs a negative result, the input is discarded immediately. In case the output is positive, the input is forwarded onto the next stage. According to Viola & Jones (2001), this multi-stage approach allows for the construction of simpler classifiers which can then be used to reject most negative (non-face) input quickly while spending more time on positive (face) input.

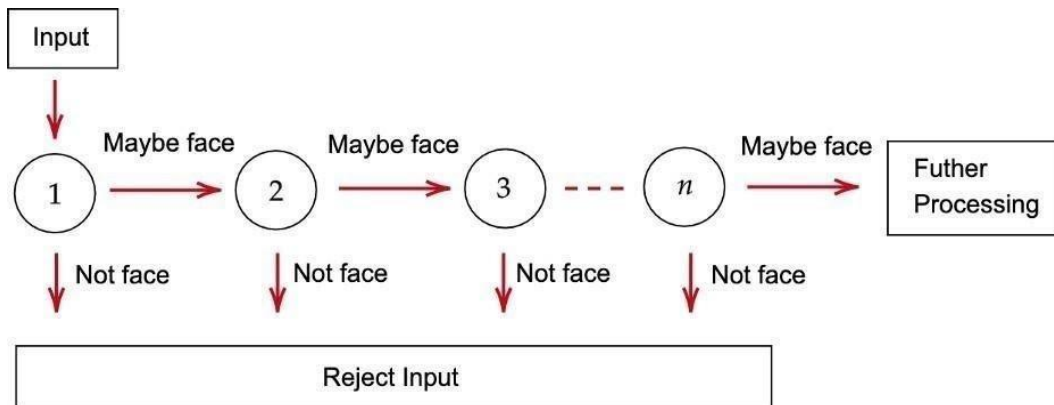


Fig 1.3 Cascade Amplifier

3.2.2 Histogram Of Oriented Gradients: The histogram of oriented gradients (HOG) is a feature descriptor used in computer vision and image processing for object detection. The technique counts occurrences of gradient orientation in localized portions of an image. This method is similar to that of edge orientation histograms, scale-invariant feature transform descriptors, and shape contexts, but differs in that it is computed on a dense grid of uniformly spaced cells and uses overlapping local contrast normalization for improved accuracy.

Algorithm overview:

Compute a Histogram of Oriented Gradients (HOG) by

1. global image normalization
2. computing the gradient image in x and y
3. computing gradient histograms
4. Normalizing across blocks
5. flattening into a feature vector

The first stage applies an optional global image normalization equalization that is designed to reduce the influence of illumination effects. In practice, we use gamma (power-law) compression, computing the square root or the log of each color channel. Image texture strength is typically proportional to the local

surface illumination so this compression helps to reduce the effects of local shadowing and illumination variations.

The second stage computes first-order image gradients. These capture contour, silhouette, and some texture information, while providing further resistance to illumination variations. The locally dominant color channel is used, which provides color invariance to a large extent. Variant methods may also include second-order image derivatives, which act as primitive bar detectors - a useful feature for capturing, e.g., bar-like structures in bicycles and limbs in humans.

The third stage aims to produce an encoding that is sensitive to local image content while remaining resistant to small changes in pose or appearance. The adopted method pools gradient orientation information locally in the same way as the SIFT feature. The image window is divided into small spatial regions, called "cells". For each cell, we accumulate a local 1-D histogram of gradient or edge orientations over all the pixels in the cell. This combined cell-level 1-D histogram forms the basic "orientation histogram" representation. Each orientation histogram divides the gradient angle range into a fixed number of predetermined bins. The gradient magnitudes of the pixels in the cell are used to vote into the orientation histogram.

The fourth stage computes normalization, which takes local groups of cells and contrasts their overall responses before passing to the next stage. Normalization introduces better invariance to illumination, shadowing, and edge contrast. It is performed by accumulating a measure of local histogram "energy" over local groups of cells that we call "blocks". The result is used to normalize each cell in the block. Typically each cell is shared between several blocks, but its normalizations are blocked dependent and thus different. The cell thus appears several times in the final output vector with different normalizations. This may seem redundant but it improves the performance. We refer to the normalized block descriptors as Histogram of Oriented Gradient (HOG) descriptors.

3.2.3 Single-Shot Detector(SSD):

SSD has two components: a backbone model and an SSD head. The backbone model usually is a pre-trained image classification network as a feature extractor. This is typically a network like ResNet trained on ImageNet from which the final fully connected classification layer has been removed. We are thus left with a deep neural network that can extract semantic meaning from the input image while preserving the spatial structure of the image albeit at a lower resolution. For ResNet34, the backbone results in a 256 7x7 feature map for an input image. The SSD Head is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the activations of the final layers.

In the figure below, the first few layers (white boxes) are the backbone, the last few layers (blue boxes) represent the SSD head.

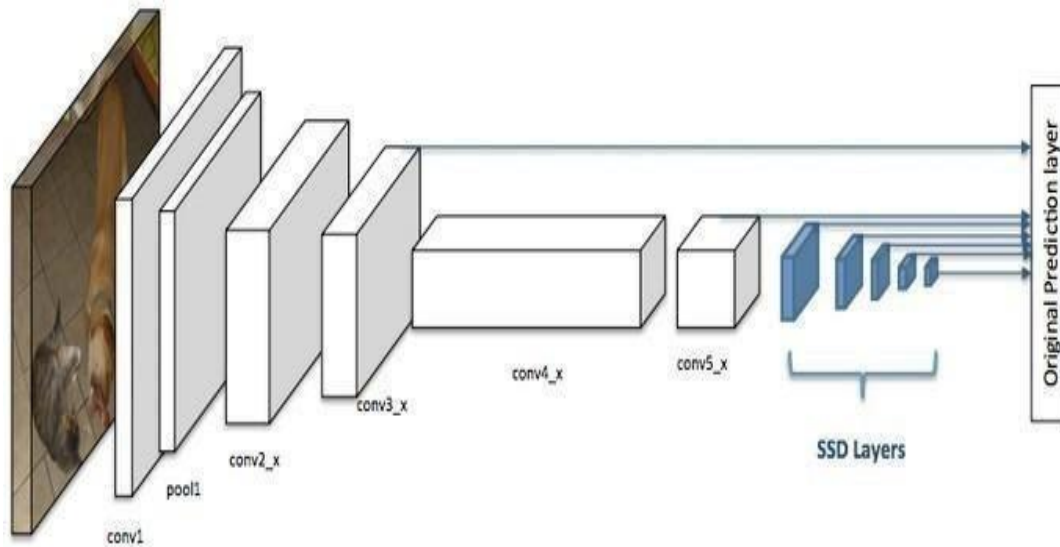


Fig 1.4 Single Shot Detector

3.3 Object Detection Framework:

Object Detection: Object detection is a computer vision technique that works to identify and locate objects within an image or video. Specifically, object detection draws bounding boxes around these detected objects, which allow us to locate where said objects are in (or how they move through) a given scene.

Object detection is commonly confused with image recognition, so before we proceed, we must clarify the distinctions between them. Image recognition assigns labels to an image. A picture of a dog receives the label “dog”. A picture of two dogs still receives the label “dog”. Object detection, on the other hand, draws a box around each dog and labels the box “dog”. The model predicts where each object is and what label should be applied. In that way, an object provides more information about an image than recognition.

Here's an example of how this distinct look in practice:

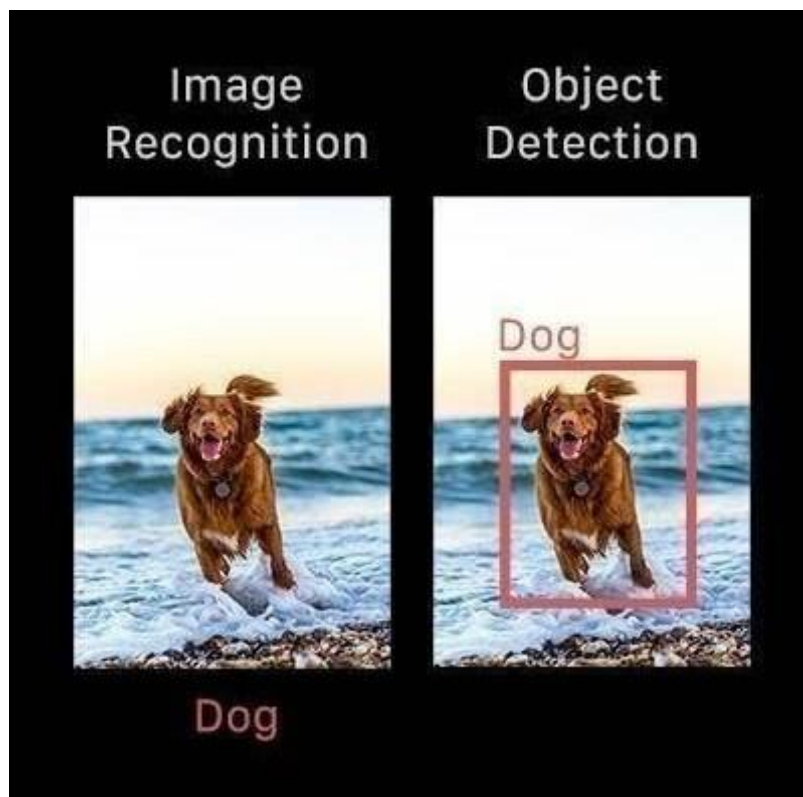


Fig 1.5 Image recognition v/s Object detection

Traditional object detection uses a multi-step process. A well-known detector is the Viola-Jones detector, which can achieve real-time detection. The algorithm extracts feature by Haar feature descriptor with an integral image method, selects useful features, and detects objects through a cascaded detector. Although it utilizes integral images to facilitate the algorithm, it is still very computationally expensive. Rather than using handcrafted features, deep learning-based detectors demonstrated excellent performance recently, due to their robustness and high feature extraction capability. There are two popular categories, one-stage object detectors, and two-stage object detectors. On one hand, we have two-stage detectors, such as Faster R-CNN (Region-based Convolutional Neural Networks) or Mask R-CNN, that (i) use a Region Proposal Network to generate regions of interests in the first stage and (ii) send the region proposals down the pipeline for object classification and bounding-box regression. Such models reach the highest accuracy rates but are typically slower. On the other hand, we have single-stage detectors, such as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector), that

treat object detection as a simple regression problem by taking an input image and learning the class probabilities and bounding box coordinates. Such models reach lower accuracy rates but are much faster than two-stage object detectors. The researchers found that a one-stage detector does not perform well by using the last feature output only, because the last feature map has fixed receptive fields, which can only observe certain areas on original images. To design an effective network for face mask detection, we adopt the object detector framework, which suggests a detection network with a backbone, a neck, and heads. The backbone refers to a general feature extractor made up of convolutional neural networks to extract information in images to feature maps. In RetinaFaceMask, we adopt ResNet as a standard backbone, In terms of the neck, it is an intermediate component between a backbone and heads, and it can enhance or refine original feature maps. In RetinaFaceMask, FPN is applied as a neck, which can extract high-level semantic information and then fuse this information into previous layers' feature maps by adding operation with a coefficient. Finally, heads stand for classifiers, predictors, estimators, etc., which can achieve the final objectives of the network.

3.3.1 ResNet: A residual neural network (ResNet) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramid cells in the cerebral cortex. Residual neural networks do this by utilizing *skip connections*, or shortcuts to jump over some layers. Typical ResNet models are implemented with double- or triple-layer skips that contain nonlinearities and batch normalization in between. An additional weight matrix may be used to learn the skip weights; these models are known as Highway nets. Models with several parallel skips are referred to as DenseNets. In the context of residual neural networks, a non-residual network may be described as a *plain network*.

There are two main reasons to add skip connections: to avoid the problem of vanishing gradients, or to mitigate the Degradation (accuracy saturation) problem; where adding more layers to a suitably deep model leads to higher training error. During training, the weights adapt to mute the upstream layer and amplify the previously skipped layer. In the simplest case, only the weights for the adjacent layer's connection are adapted, with no explicit weights for the upstream layer. This works best when a single nonlinear layer is stepped over, or when the intermediate layers are all linear. If not, then an explicit weight matrix should be learned for the skipped connection (a Highwaynet should be used).

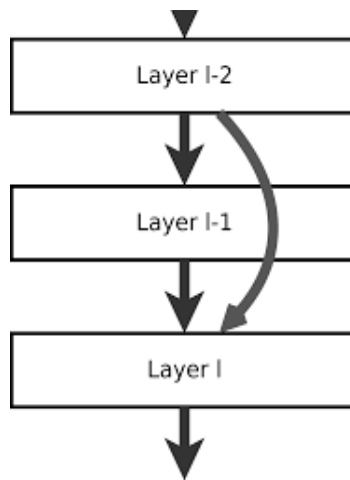


Fig 1.6 Skip Connections

Skipping effectively simplifies the network, using fewer layers in the initial training stages. This speeds learning by reducing the impact of vanishing gradients, as there are fewer layers to propagate through. The network then gradually restores the skipped layers as it learns the feature space. Towards the end of the training, when all layers are expanded, it stays closer to the manifold and thus learns faster. A neural network without residual parts explores more of the feature space. This makes it more vulnerable to perturbations that cause it to leave the manifold and necessitates extra training data to recover.

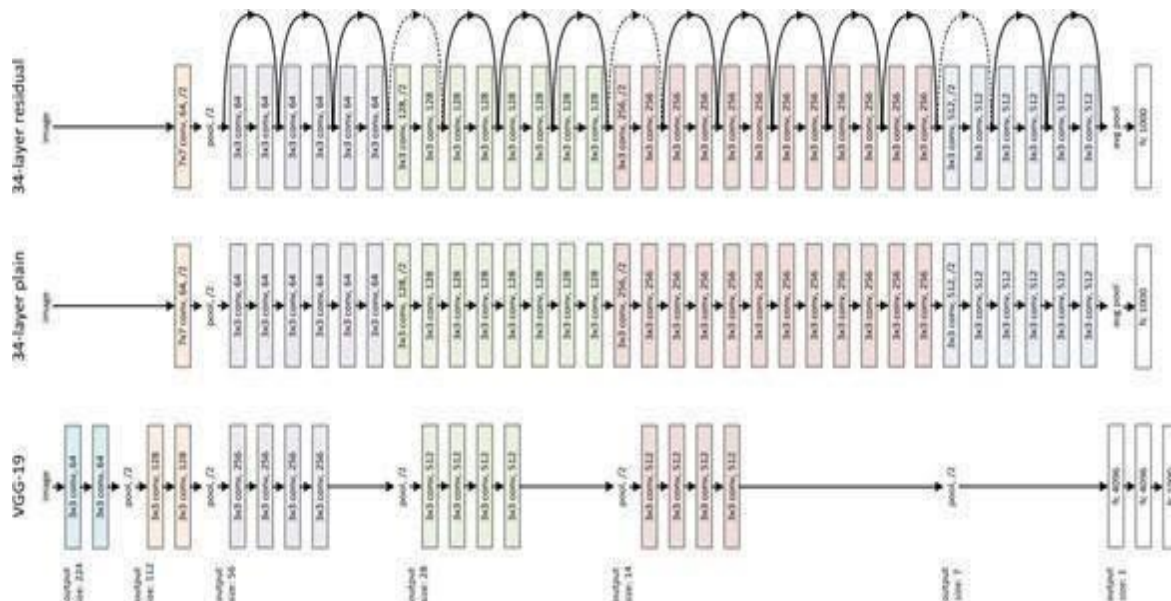


Fig 1.7 ResNet

3.3.2 Feature Pyramid Network: Feature Pyramid Network (FPN) is a feature extractor designed for such pyramid concepts with accuracy and speed in mind. It replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection.

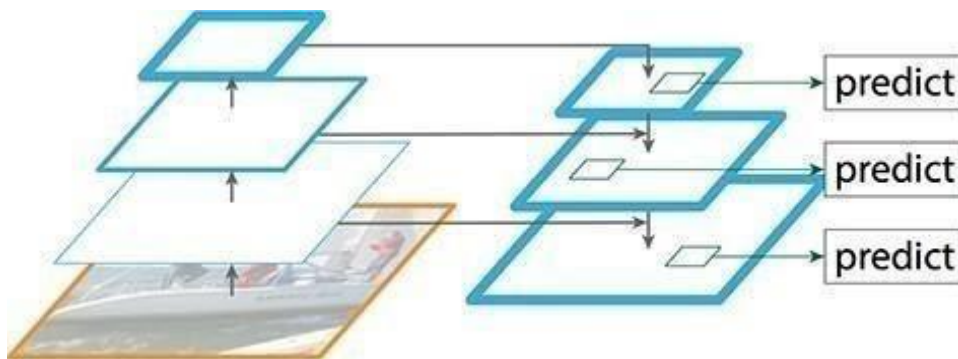


Fig 1.8 Feature pyramid Network

FPN consists of a bottom-up and a top-down pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the semantic value for each layer increases.

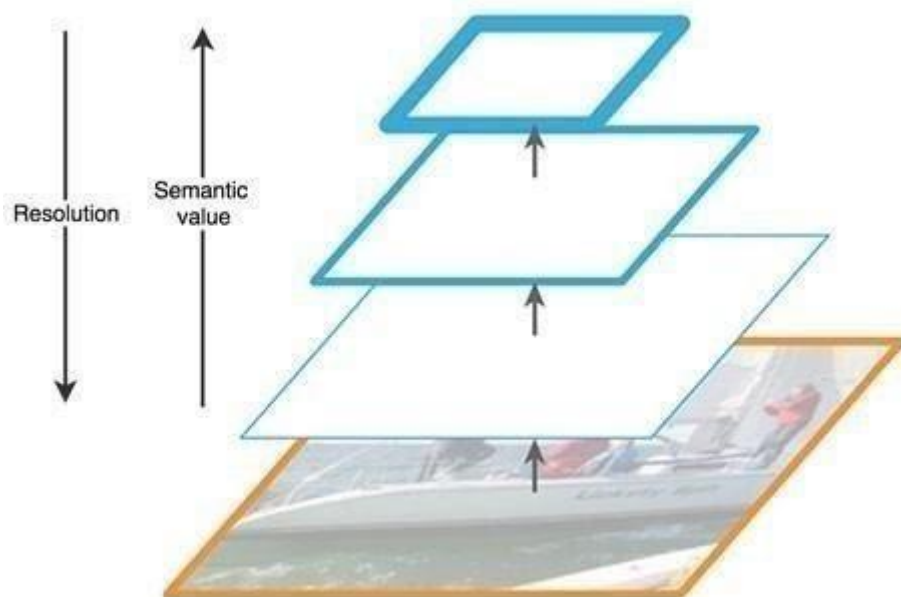


Fig 1.9 Feature extraction in FPN

SSD makes detection from multiple feature maps. However, the bottom layers are not selected for object detection. They are in high resolution but the semantic value is not high enough to justify its use as the speed slow-down is significant. So SSD only uses upper layers for detection and therefore performs much worse for small objects. FPN provides a top-down pathway to construct higher resolution layers from a semantic-rich layer. While the reconstructed layers are semantic strong but the locations of objects are not precise after all the downsampling and upsampling. We add lateral connections between reconstructed layers and the corresponding feature maps to help the detector to predict the location better. It also acts as skip connections to make training easier.

3.3.3 Classifiers, Predictors, Estimators:

Classifiers: A classifier is an algorithm that sorts data into labeled classes or categories of information. A simple practical example is spam filters that scan incoming “raw” emails and classify them as either "spam" or "not spam." Classifiers are a concrete implementation of pattern recognition in many forms of machine learning.

Classifiers are where high-end machine theory meets practical application. These algorithms are more than a simple sorting device to organize or "map" unlabeled data instances into discrete classes. Classifiers have a specific set of dynamic rules, which includes an interpretation procedure to handle vague or unknown values, all tailored to the type of inputs being examined. Most classifiers also employ probability estimates that allow end-users to manipulate data classification with utility functions.

predictors: Predictor variables in the machine learning context the input data or the variables that are mapped to the target variable through an empirical relationship usually determined through the data. In statistics, you refer to them as predictors. Each set of predictors may be called an observation.

Estimators: In machine learning, an estimator is an equation for picking the "best," or most likely accurate, data model based upon observations in reality. Not to be confused with estimation in general, the estimator is the formula that evaluates a given quantity (the estimand) and generates an estimate. This estimate is then inserted into the deep learning classifier system to determine what action to take.

Estimators come in two broad categories -point and interval. Point equations generate single value results, such as standard deviation, that can be plugged into a deep learning algorithm’s classifier functions. Interval equations generate a range of likely values, such as confidence intervals.

3.4 Convolutional Neural Network: In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyze visual imagery. Now when we think of a neural network we think about matrix multiplications but that is not the case with ConvNet. It uses a special technique called Convolution. Now in mathematics convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other.

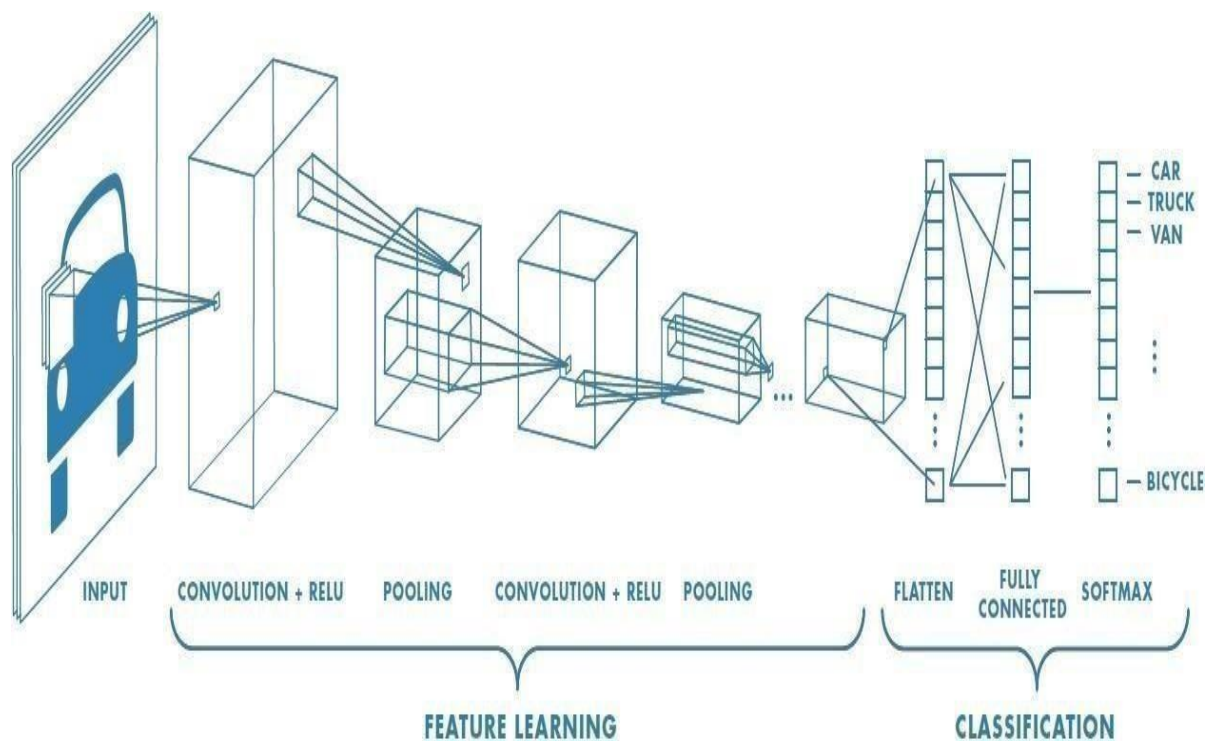


Fig 1.10 Convolutional neural Network

Convolutional layer: Convolutional layers are the major building blocks used in convolutional neural networks. Convolution is the simple application of a filter to an input that results in inactivation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

Pooling: Pooling is nothing other than the downsampling of an image. The most common pooling layer filter is of size 2×2 , which discards three fourth of the activations. The role of the pooling layer is to reduce the resolution of the feature map but retain features of the map required for classification through translational and rotational invariants. In addition to spatial invariance robustness, pooling will reduce the computation cost by a great deal.

- Backpropagation is used for training of pooling operation
- It again helps the processor to process things faster.

Max pooling: where we take the largest of the pixel values of a segment.

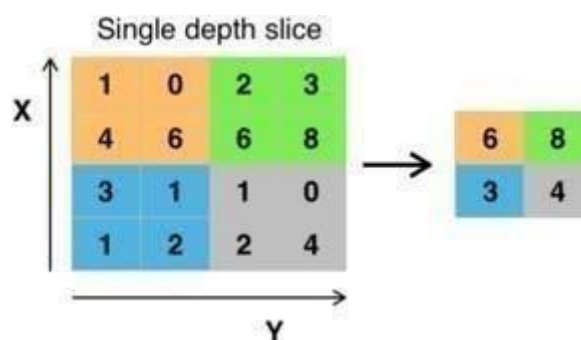


Fig 1.11 Max Pooling

Padding: When we augment the $5 \times 5 \times 1$ image into a $6 \times 6 \times 1$ image and then apply the $3 \times 3 \times 1$ kernel over it, we find that the convolved matrix turns out to be of dimensions $5 \times 5 \times 1$. Hence the name — Same Padding.

Fully Connected layer: Fully Connected layers in a neural network are those layers where all the inputs from one layer are connected to every activation unit of the next layer. In most popular machine learning models, the last few layers are full connected layers that compile the data extracted by previous layers to form the final output. It is the second most time-consuming layer second to the

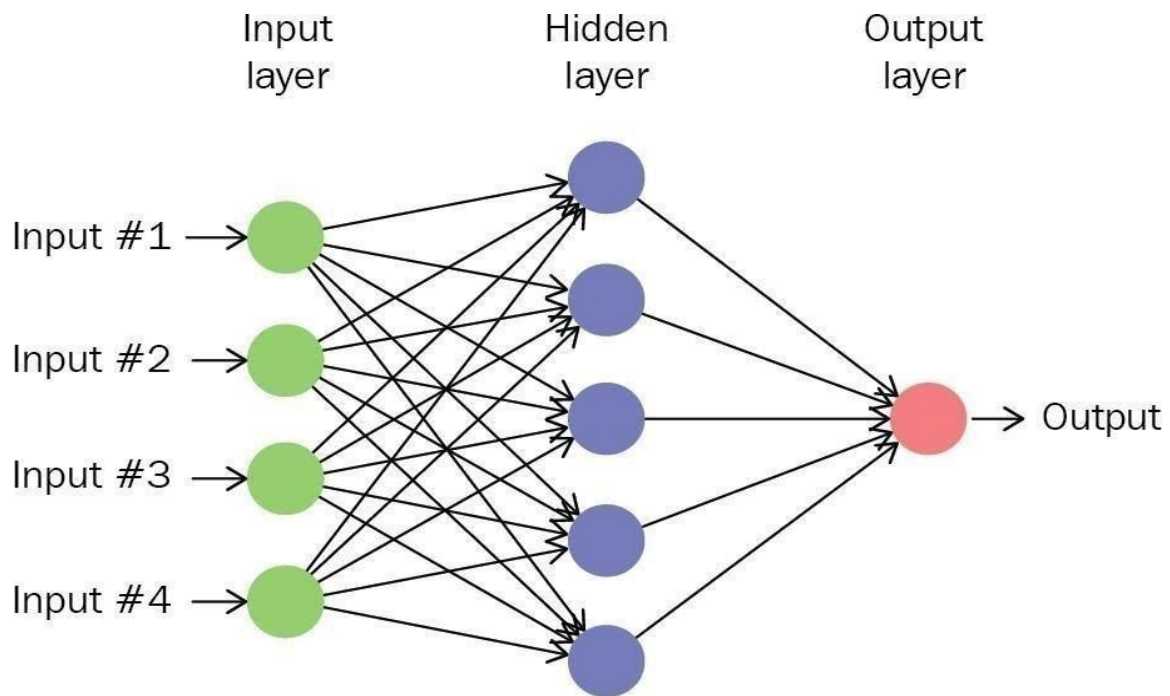


Fig 1.12 Fully Connected layers

3.5 Context Attention Model: To improve the detection performance for face masks, RetinaFaceMask proposes a novel context attention module as its detection heads. Similar to the context module in SSH, we utilize different sizes of kernels to form an Inception-like block. It can obtain different receptive fields from the same feature map, so it would be able to incorporate more different sizes of objects through concatenation operations. However, the original context module does not take faces or masks into account, so we simply cascade an attention module CBAM after the original context module to allow RetinaFaceMask to focus on face and mask features. The context-aware part has three sub-branches, including one 3×3 , two 3×3 , and three 3×3 kernels in each branch individually. Then, the concatenated feature maps are fed into CBAM through channel attention to select useful channels by a multi-layer perceptron and then spatial attention to focus on important regions.

CHAPTER 4

HARDWARE TOOLS

HARDWARE TOOLS:

4.1 Jetson Nano:

4.1.1 Introduction:

1. NVIDIA Jetson Nano developer kit is an AI computer. It delivers the compute performance to run modern AI workloads at an unprecedented size. It is incredibly power-efficient, consuming as little as 5 watts.
2. GPU128-core Maxwell CPU Quad-core ARM AS7GHz.
3. NVIDIA India Memory 4 GB 64-bit LPDDR4 25.6 GB/s.
4. It is supported by NVIDIA Jetpack, used across the entire NVIDIA Jetson family of products, reducing complexity and overall effort for developers, learners, and makers.

4.1.2 overview:

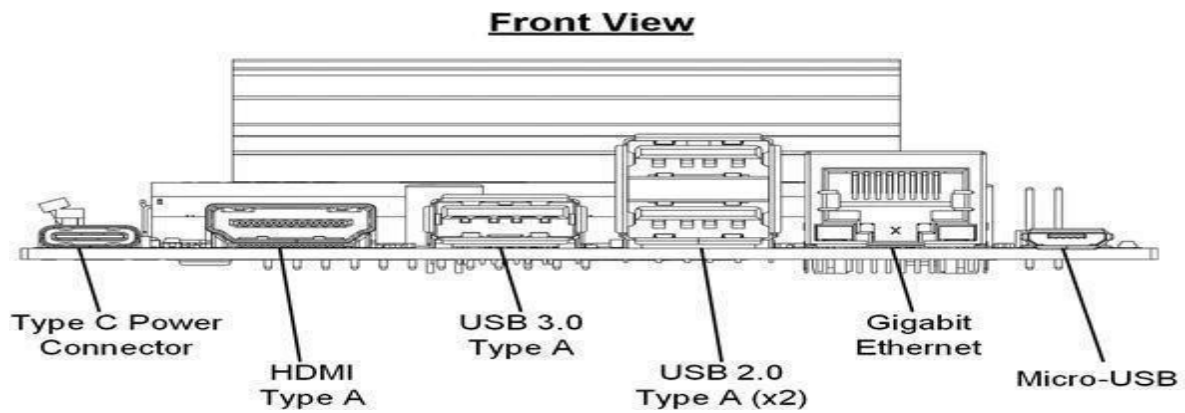


Fig 2.1 Front view of Jetson Nano

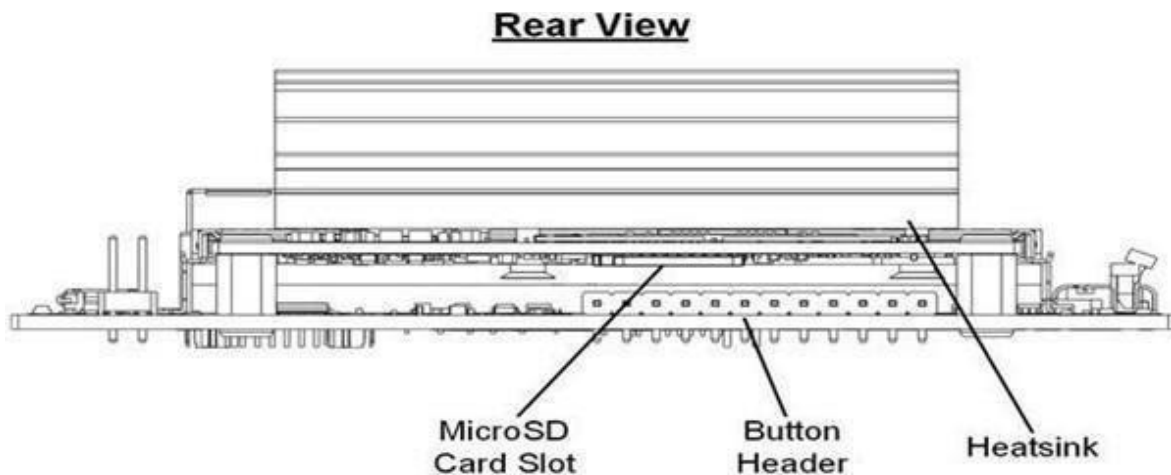


Fig 2.2 Rear view of Jetson Nano

4.3.1 carrier board layout:

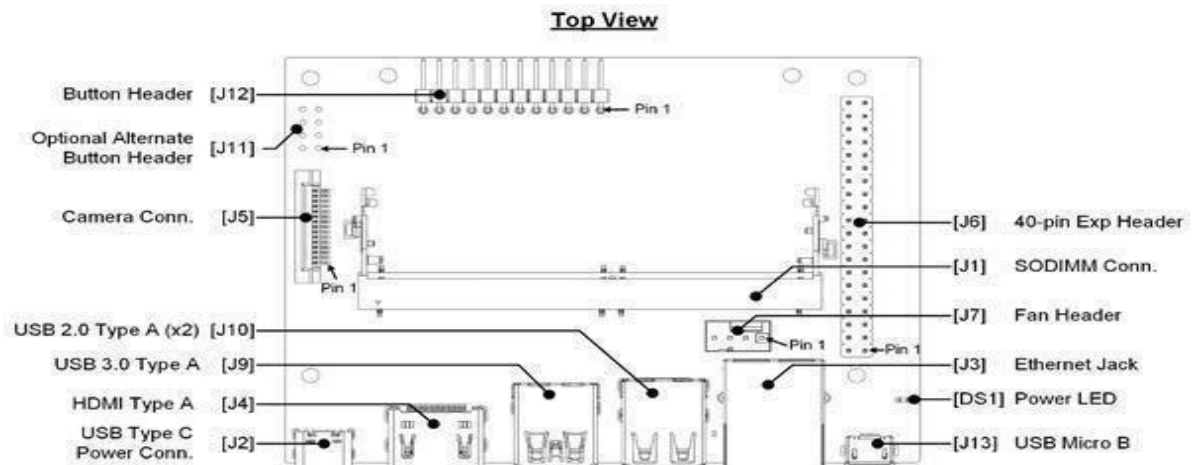


Fig 2.3 Layout of Jetson Nano

[DS1] Power LED; lights when the developer kit is powered on

[J1] - SO-DIMM connector for Jetson module. The module is pre-assembled on the developer kit

[J2] - USB Type C power connector for 5V=3A power supply.

[J3] - RJ45 ethernet connector. See networking section

[J4] - HDMI connector

[J5] - Camera connector for MIPI-CSI2 camera. See camera section

[J6] - 40-pin header: Includes power pins (+5V/+3.3V) and interface signal pins for I2C (2x), UART, SPI (2x), I2S, and GPIOs

[J7] - 4-pin fan control header. Pulse Width Modulation (PWM) output and tachometer input are supported

[J8] - Optional coin-cell socket

[J9] - USB 3.0 type A connector. Limited to 1A total power delivery

[J10] - Stack of two USB 2.0 type A connectors

[J11] - Optional button header (2x4); Includes connections for Reset/Force Recovery/Power Buttons, and Auto-power-on disable

[J12] - Button header (1x12); Includes connections for power LED, Reset/Force Recovery/Power Buttons, UART, and Auto-power-on disable

[J13] - Micro-USB 2.0 connector supporting Recovery Mode and Device Mode.

4.3.1 40-Pin header:

The 40-pin header provides access to power, ground, and interface signal pins.

Power pins

o There are two 3.3V power pins and two 5V power pins. These are not switchable; power is always available when the developer kit is connected to power.

o The two 5V pins can be used to power the developer kit at 2.5A each. (Do not power the developer kit via these pins and USB-C connector at the same time.)

Interface signal pins

o All signals use 3.3V levels

o By default, all interface signal pins are configured as GPIOs, except those supporting I2C and UART.

SoC GPIO	Linux GPIO #	Alternate Function	Default Function			Default Function	Alternate Function	Linux GPIO #	SoC GPIO
			3.3 VDC	①	②	5 VDC			
PJ.03	75	GPIO	I2C1_SDA	③	④	5 VDC			
PJ.02	74	GPIO	I2C1_SCL	⑤	⑥	GND			
PBB.00	216	AUD_CLK	GPIO	⑦	⑧	UART1_TXD	GPIO	48	PG.00
			GND	⑨	⑩	UART1_RXD	GPIO	49	PG.01
PG.02	50	UART1_RTS	GPIO	⑪	⑫	GPIO	I2S0_SCLK	79	PJ.07
PB.06	14	SPI1_SCK	GPIO	⑬	⑭	GND			
PY.02	194		GPIO	⑮	⑯	GPIO	SPI1_CS1	232	PDD.00
			3.3 VDC	⑰	⑱	GPIO	SPI1_CS0	15	PB.07
PC.00	16	SPI0_MOSI	GPIO	⑲	⑳	GND			
PC.01	17	SPI0_MISO	GPIO	㉑	㉒	GPIO	SPI1_MISO	13	PB.05
PC.02	18	SPI0_SCK	GPIO	㉓	㉔	GPIO	SPI0_CS0	19	PC.03
			GND	㉕	㉖	GPIO	SPI0_CS1	20	PC.04
PB.05	13	GPIO	I2C0_SDA	㉗	㉘	I2C0_CLK	GPIO	18	PC.02
PS.05	149	CAM_MCLK	GPIO	㉙	㉚	GND			
PZ.00	200	CAM_MCLK	GPIO	㉛	㉜	GPIO	PWM	168	PV.00
PE.06	38	PWM	GPIO	㉝	㉞	GND			
PJ.04	76	I2S0_FS	GPIO	㉟	㊱	GPIO	UART1_CTS	51	PG.03
PB.04	12	SPI1_MOSI	GPIO	㊲	㊳	GPIO	I2S0_DIN	77	PJ.05
			GND	㊴	㊵	GPIO	I2S0_DOUT	78	PJ.06

Fig 2.4 40 Pin Header

4.1.3.2 8-Pin Button Header :

This is an alternate 8-Pin (2x4) button header that can be soldered on the carrier board in location J11 and used in place of the main button header. Header details (examples):

Total pins/positions 8

2 rows of 4 pins

Pitch is 2.54mm

Unshrouded

Through-hole vertical



Fig 2.5 8 Pin Header

4.1.3.3 4-Pin Fan Header

The pinout of the 4-pin fan control header at location J7 is shown below.

The header can support either a 3-pin fan connection (GND, PWR, and TACH) or a 4-pin fan connection (GND, PWR, TACH, and PWM). Using a fan with PWM capability allows the software to adjust the speed of the fan as needed.



Fig 2.6 4 Pin Header

4.1.4 Networking:

The developer kit supports wired and wireless networking:-

Wired - Ethernet will be available as soon as a cable with a network connection is plugged into the RJ45 port

WLAN - Wireless networks will be available after plugging in a supported USB wireless networking adapter

WPAN - Bluetooth will be available after plugging in a supported USB Bluetooth adapter.

4.1.5 Camera:

Supports 2-Lanes CSI Camera

Upgraded 2-Lanes CSI, Instead Of The Previous 1-Lane, Easily Play Around With Binocular Vision.

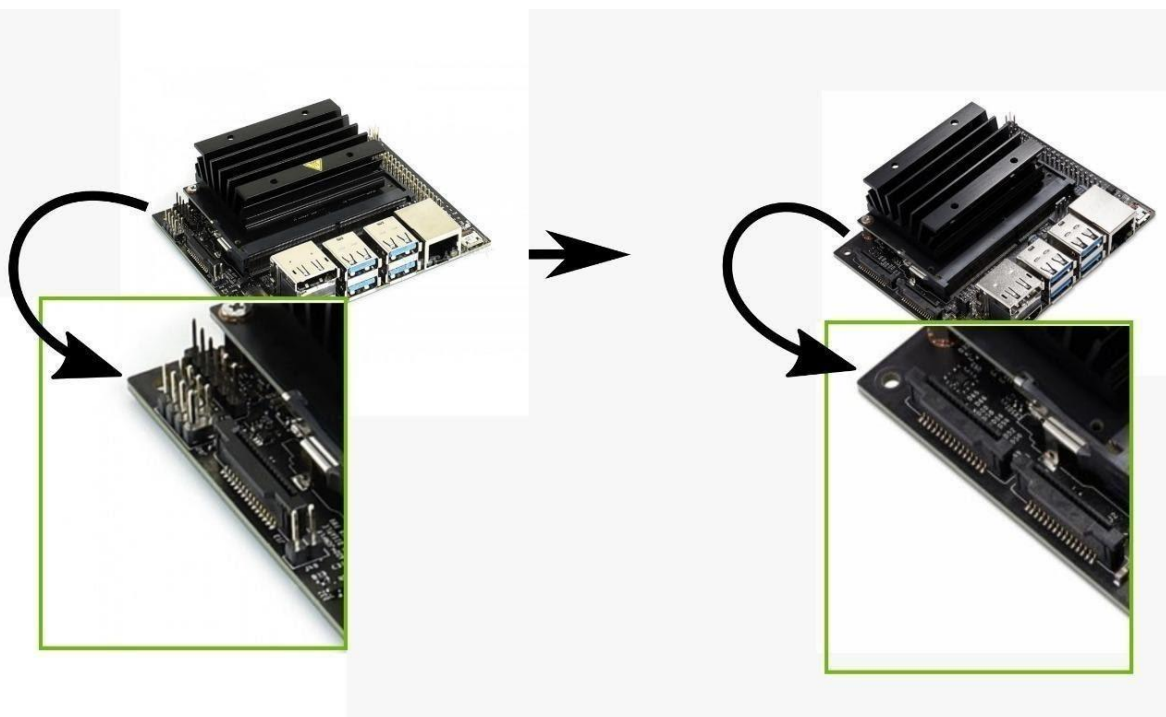


Fig 2.7 Camera Lanes

The developer kit supports USB-C power supplies of $5V \pm 5\%$, 3A. If your phone uses a USB-C power supply, there is a chance that it is enough to power the device. Check its specifications.

If the voltage drops below 4.25V, the system will shut down.

4.1.6 Power consumption:

The developer kit's total power usage is the sum of carrier board, module, and peripheral power usage, as determined by your particular use case.

There are two software-defined power modes for the Jetson module. The two-module power modes are:

10W - default mode for more performance

5W - suggested for less energy use

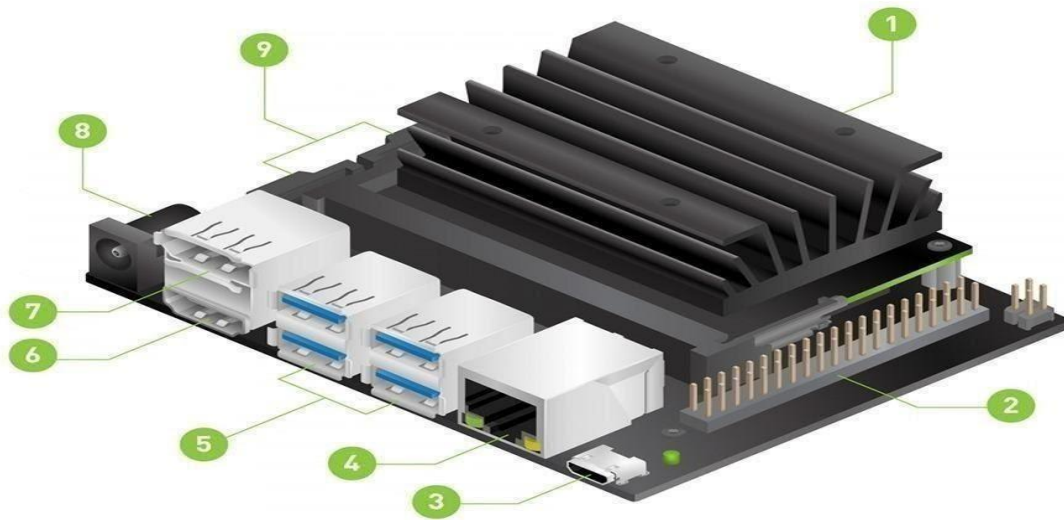
Power via 40-pin Header

The developer kit can be powered by connecting the 40-pin header.

4.1.7 Specifications:

1.	GPU 128-core Maxwell
2.	CPU Quad-core ARM A57 @ 1.43 GHz
3.	MEMORY 4 GB 64-bit LPDDR4 25.6 GB/s
4.	STORAGE micro SD card (NOT included.)
5.	VIDEO ENCODER 4K @ 30 4x 1080p @ 30 9x 720p @ 30 (H.264/H.265)
6.	VIDEO DECODER 4K @ 60 2x 4K @ 30 8x 1080p @ 30 18x 720p @ 30 (H.264/H.265)
7.	CAMERA 2x MIPI CSI-2 D'PHY lanes
8.	CONNECTIVITY Gigabit Ethernet, M.2 Key E expansion connector
9.	DISPLAY HDMI and DP
10.	USB 4x USB 3.0, USB 2.0 Micro-B

4.1.8 Board description:



1.	<p>MICRO SD CARD SLOT</p> <p>Insert a 16GB or larger TF card for main storage and writing system image. Jetson nano developer kit uses a microSD card as a boot device and for main storage</p>
2.	<p>40 PIN EXPANSION HEADER</p> <p>The 40 pin expansion header lets you connect a jetson developer kit to off-the-shelf-raspberry pi HATs such as seed grove modules, Sparkfun qwiic products, and others. Many of the pins can be used either as GPIO or as “special function I/O”(SFIO) such as 12C,12S, etc.</p>
3.	<p>MICRO USB PORT</p> <p>The micro USB port is used for 5v power input or USB data transmission.</p>
4.	<p>Gigabit Ethernet port</p>

	Jetson nano uses 10/100/1000 base-t auto-negotiation
5.	<p>Gigabit Ethernet port</p> <p>Ethernet ports can also be referred to as sockets or jacks. The main function of an ethernet port is to create an ethernet connection.</p> <p>Ethernet connections can be created between computers, servers, switches, hubs, routers, modems, gaming consoles, printers, and much more. Jetson nano uses 10/100/1000 base-t auto-negotiation.</p>
6.	<p>USB 3.0 port</p> <p>Its main function is to store, receive, and transfer data across computers and electronic devices. The current USB 3.0 transfer speed is at a whopping 5 Gigabits per second. jetson nano has 4 x USB 3.0 ports</p>
7.	<p>HDMI OUTPUT PORT</p> <p>The HDMI interface allows a port to send high-resolution digital video, theatre-quality sound, and device commands through a connector and down a single cable. There are several types of HDMI cable, each designed to support a video resolution and features in the HDMI specification.</p>
8.	<p>DISPLAYPORT CONNECTOR</p> <p>DisplayPort is a digital interface designed to deliver video and audio over a singular cable. Much like HDMI, it can connect a monitor to a data source, like a graphics card, and deliver the video and sound that it's outputting to the display screen.</p>
9.	<p>DC JACK</p> <p>A DC jack is a component used in many electronic devices that allows a steady power source to be plugged in. nano uses a dc jack for 5v power input</p>
10.	NANO HAS 2 X MIPI CSI CAMERA CONNECTOR

4.2 Logitech camera specifications and features:

Specifications:

Max video :1280 x 720 pixels

Interface: USB 2.0 port

Bundled software Pan, tilt, and zoom controls. Video and photo capture. Face tracking. Motion detection.

Features:

Full HD widescreen video calling: Logitech C270 lets you make widescreen video calls in HD 720p at 30fps. The lense with a 60-degree field of view covers all of the action

HD lighting adjustment: Automatically improves the warmth and balance of your image for whatever setting you are in, so you look your best, even in dim environments

- Built-in noise-reducing mic: Enjoy clear conversation even in busy surroundings and streaming over wifi with noise reducing mic
- Universal clip: Attaches securely to your screen or works as a stand on a shelf or desk, the clip mounts at different angles to bring your friends and family all the details around you.
- Ideal for laptop or tablet: Compatible with Windows 10 or later Windows 8, Windows 7, Mac OS 10.10 or later, and Chrome OS via the usb port.



Fig 2.8 Logitech Camera

USB Cable:

USB cable is an interface developed for connecting compact and mobile devices such as your smartphones, MP3 players, GPS devices, photo printers, and digital cameras.

There are several different USB cables each of which has different benefits and it's suited to a different task.

Some of the types of USB cables are

USB-a

USB-b Mini-USB

Micro-usb

Usb-c

USB-3



Fig 2.9 USB Cable

4.4 Micro SD card:

A memory card or memory cartridge is an electronic data storage device used for storing digital information, typically using flash memory. These are commonly used in portable electronic devices, such as digital cameras, mobile phones, laptop computers, tablets, PDAs, portable media players, video game consoles, synthesizers, electronic keyboards, and digital pianos, and allow adding memory to such devices without compromising ergonomics, as the card is usually contained within the device rather than protruding like USB flash drives.



Fig 2.10 Memory Card

CHAPTER 5

SOFTWARE TOOLS

5.1 Set up and booting :

Write Image to the MicroSD Card:

To prepare a microSD card, we need a computer with an Internet connection and the ability to read and write SD cards, either via a built-in SD card slot or adapter.

1. Download the *Jetson Nano Developer Kit SD Card Image*, and note where it was saved on the computer.
2. Write the image to the microSD card

After the microSD card is ready, proceed to set up the developer kit.

Setup and First Boot:

There are two ways to interact with the developer kit:

- 1) With display, keyboard, and mouse attached
- 2) In “headless mode” via a connection from another computer. We can conduct the initial setup either way.

	Initial setup with display attached	Initial setup in headless mode
Monitor, keyboard and mouse	Required	Not required
Extra compute	Not required	Required
Power options	Either Micro-USB or DC the power supply can be used	DC power supply is needed

Figure 3.1.: Requirements for Initial setup with a display attached and Initial setup in headless mode

1. Initial Setup with Display Attached Setup Steps:

- Insert the microSD card (with the system image already written to it) into the slot on the underside of the Jetson Nano module.

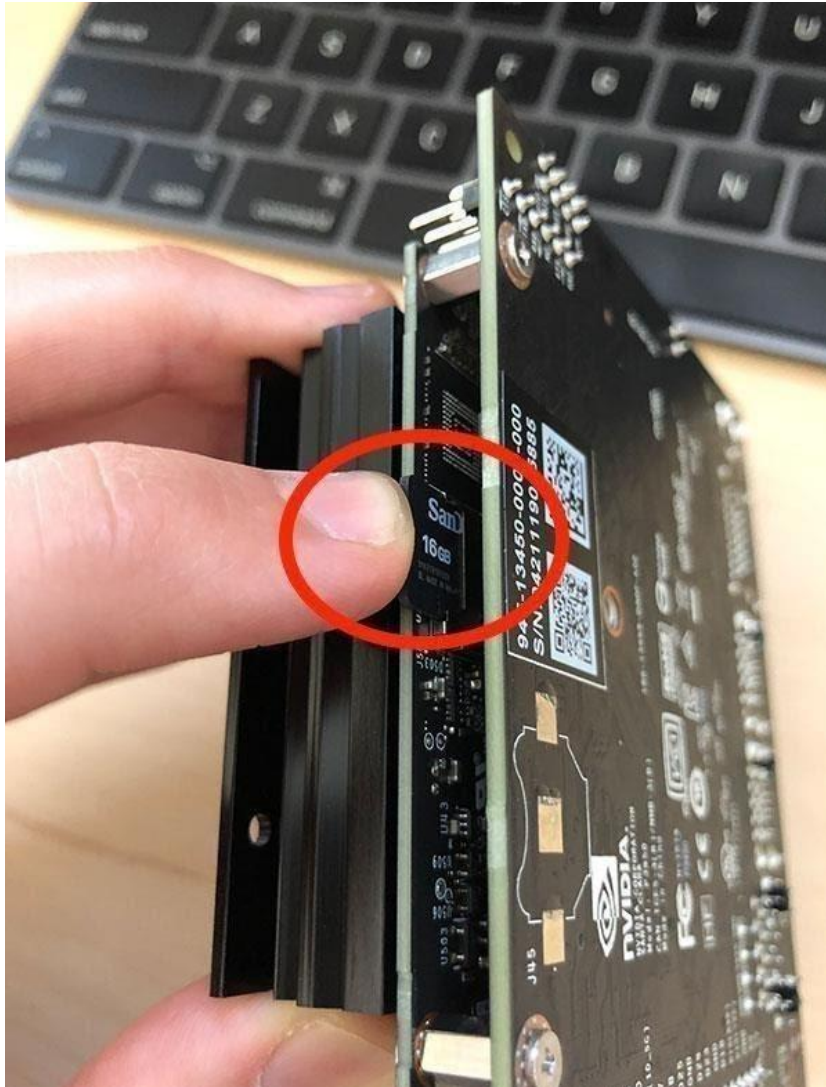


Figure 3.2: To insert Jetpack-flashed microSD after it has been flashed, find the microSD slot as shown by the red circle in the image. Insert the microSD until it clicks into place.

- Set the developer kit on top of the paper stand.
- Power on your computer display and connect it.
- Connect the USB keyboard and mouse.
- Connect the Micro-USB power supply. The developer kit will power on and boot automatically.

First Boot:

A green LED next to the Micro-USB connector will light as soon as the developer kit powers on. When you boot the first time, the developer kit will take you through some initial setup, including:

- Review and accept NVIDIA Jetson software EULA
- Select system language, keyboard layout, and time zone
- Create username, password, and computer name
- Select APP partition size—it is recommended to use the max size suggested

After Logging In:

We will see this screen. Congratulations!



Figure 3.3: Successful completion of Initial setup with Display Attached.

2. Initial Setup Headless Mode:

To complete setup when no display is attached to the developer kit, we'll need to connect the developer kit to another computer and then communicate with it via a terminal application (e.g., PuTTY) to handle the USB serial communication on that other computer.

Note: Headless initial configuration requires the developer kit to be powered by DC power supply with barrel jack connector since the Micro-USB port is required to access the initial configuration prompts

Setup Steps:

- Unfold the paper stand and place it inside the developer kit box.
- Insert the microSD card (with the system image already written to it into the slot on the underside of the Jetson Nano module.

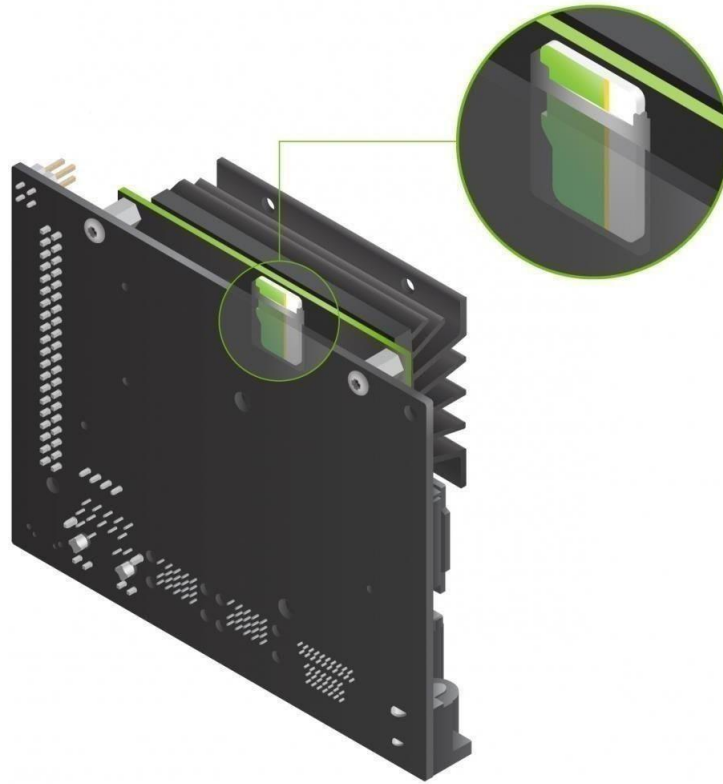


Figure 3.4: To insert Jetpack-flashed microSD after it has been flashed, find the MicroSD slot and insert microSD until it clicks into place.

- Set the developer kit on top of the paper stand.
- Jumper the J48 Power Select Header pins.
- Connect your other computer to the developer kit's Micro-USB port.
- Connect a DC power supply to the J25 Power Jack. The developer kit will power on automatically.
- Allow 1 minute for the developer kit to boot.
- On your other computer, use the serial terminal application to connect via host serial port to the developer kit

Instructions for Windows:

Locate the correct COM port: Assuming you have already connected your Windows PC to the developer kit's Micro-USB port, right-click the Windows Start icon and select "Device Manager."

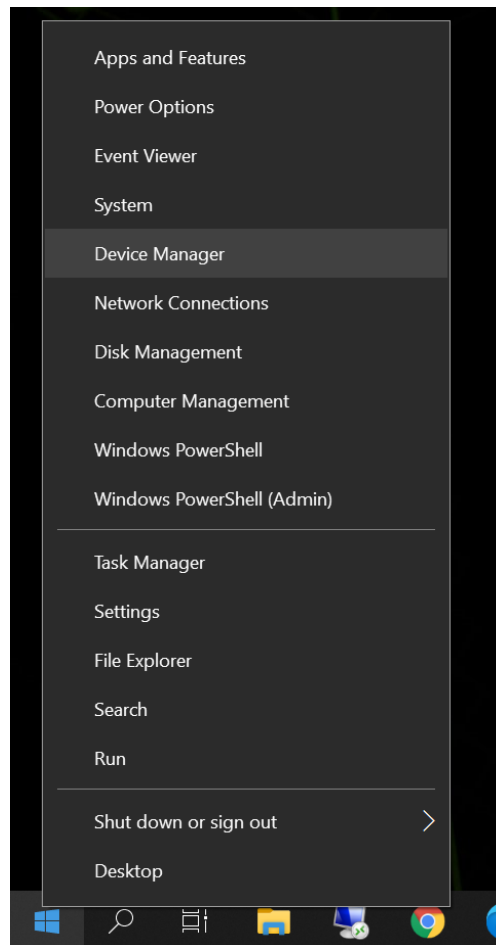


Figure 3.5: Selecting Device Manager

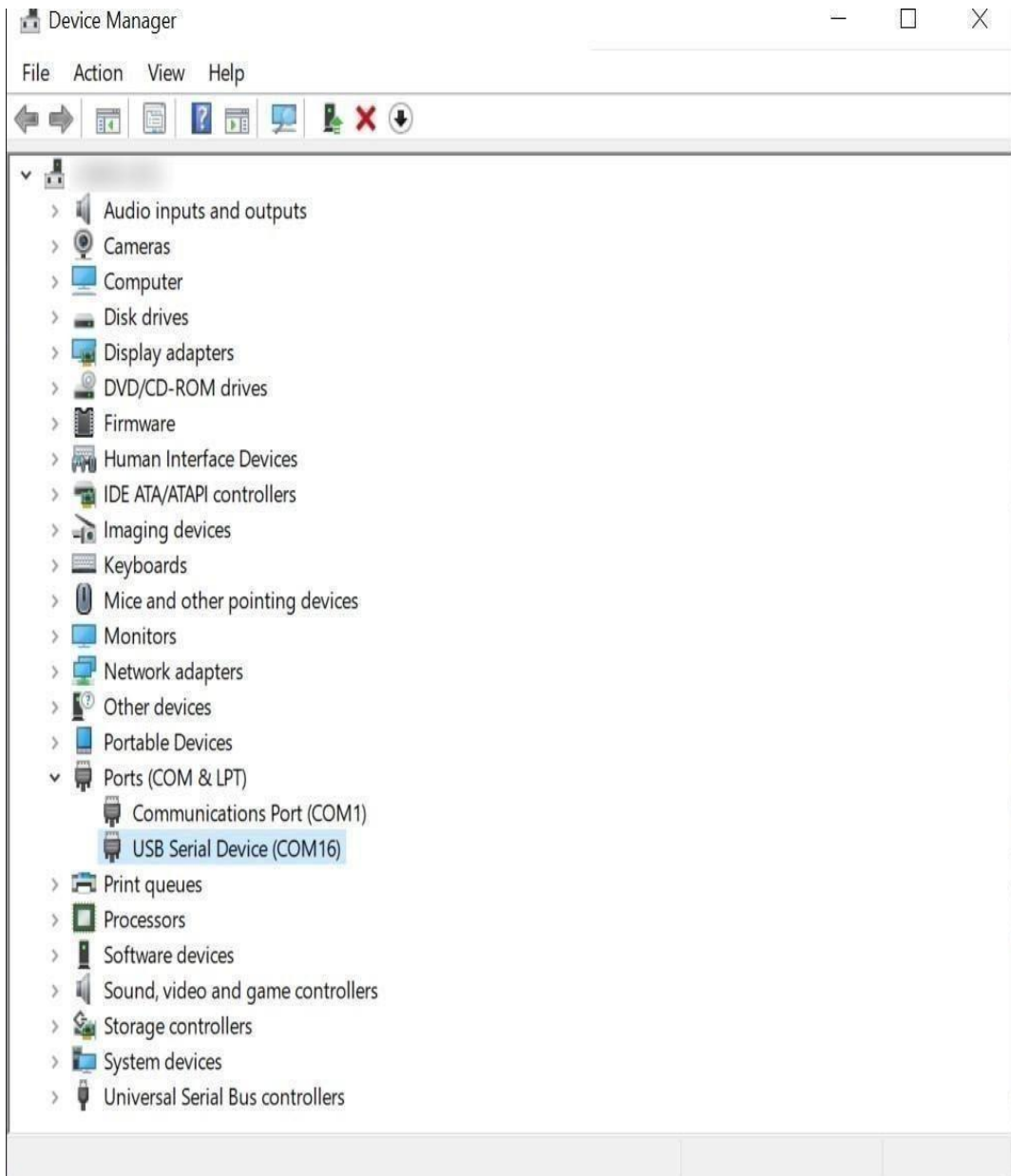


Figure 3.6: Open the “Ports (COM & LPT)” to find the COM port number for “USB Serial Device” (in this case “COM 16”)

Double click each USB Serial Device entry so you can check its properties. Go to the “Details” tab, and select “Hardware Ids”. If you see VID 0955 and PID 7020, that USB Serial Device for your Jetson developer kit. Note the COM port name (COM16 in this example) for later use.

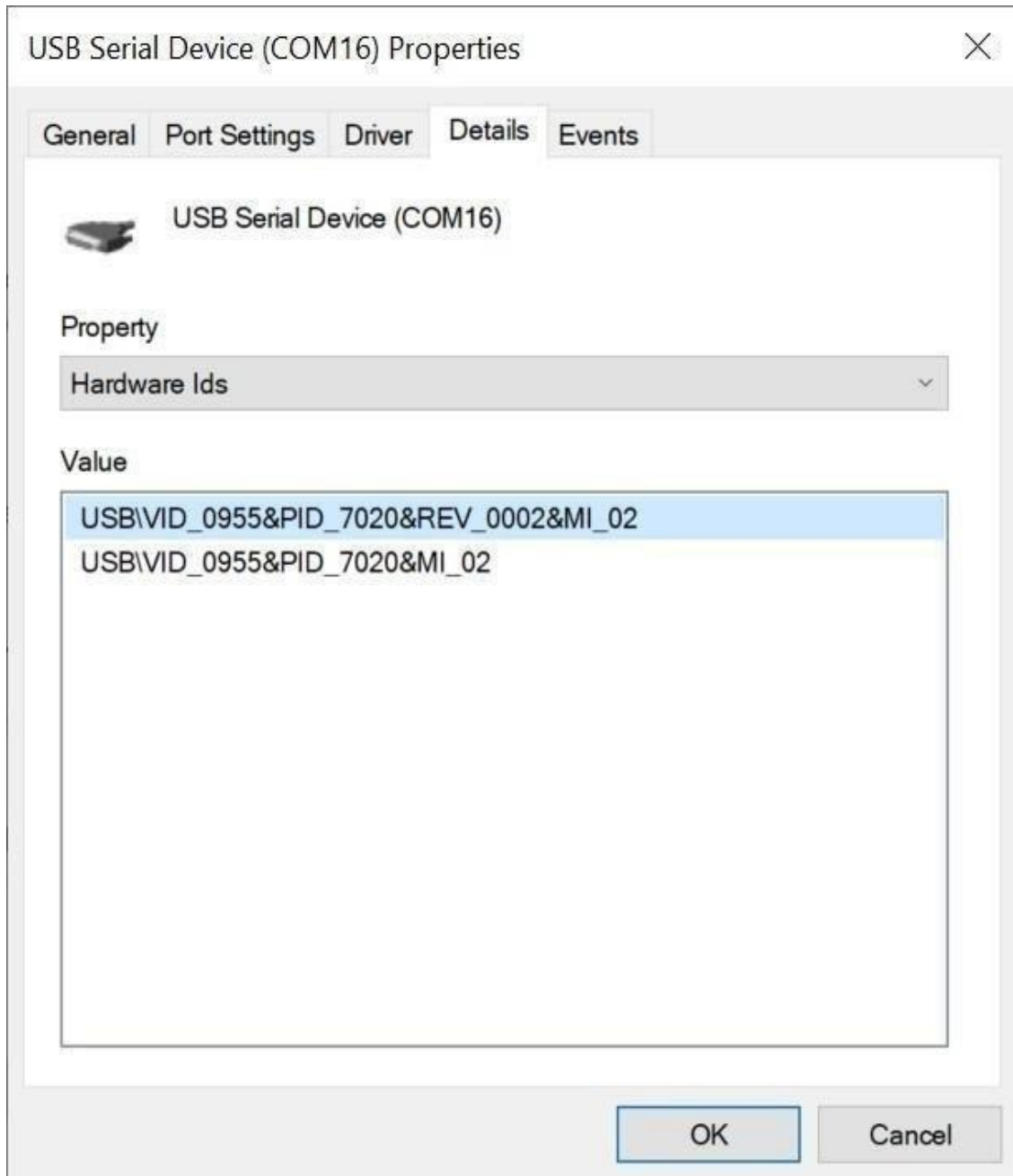


Figure 3.7: USB Serial Device (COM16) Properties

Open the COM port on PuTTY:

PuTTY is one of the most widely used terminal applications for accessing serial consoles. You can use other terminal applications, but if you don't have any on your Windows PC, you can download PuTTY

Open the PuTTY application. When "Session" is selected in the left "Category" pane, input the COM port name for "Serial line" and "115200" for "Speed".

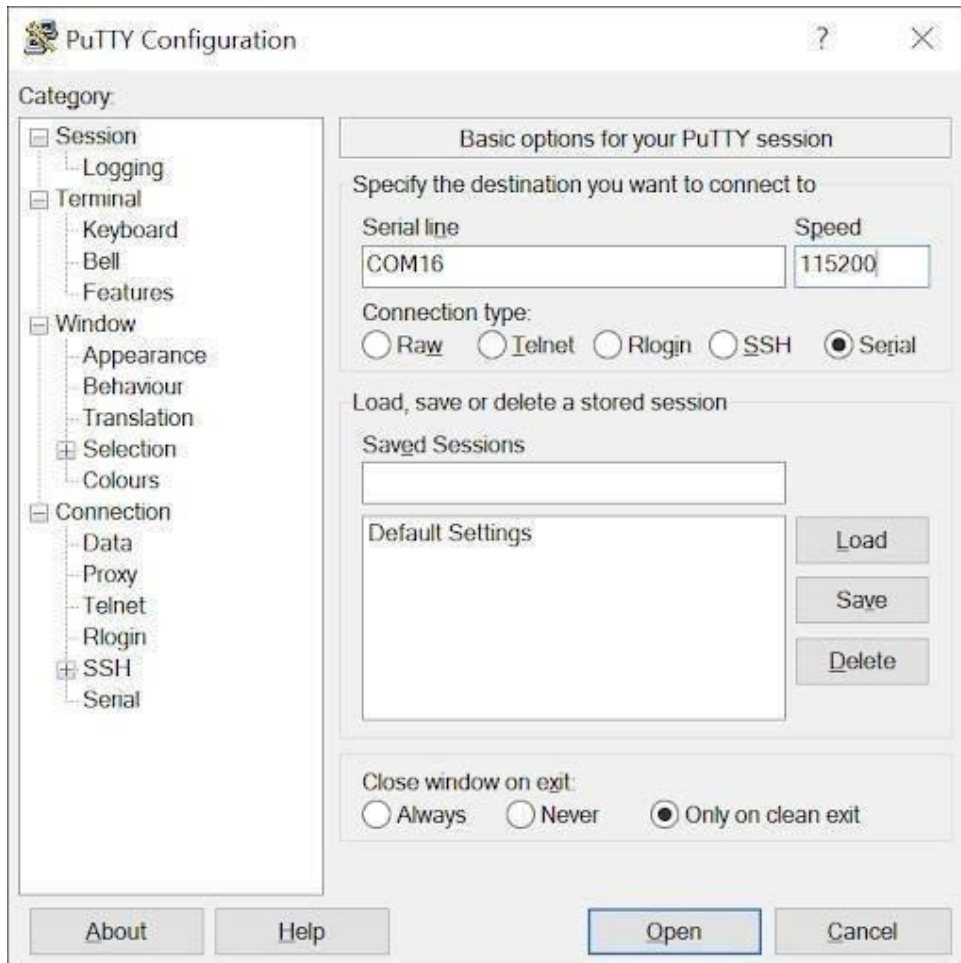


Figure 3.8: PUTTY Configuration

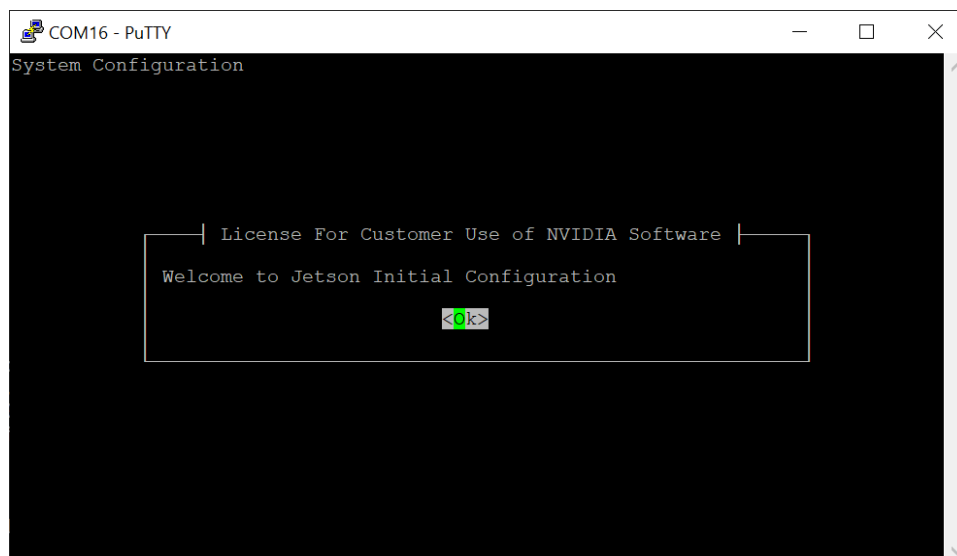


Figure 3.9: Click “Open” to connect to the console.

Once connected to the developer kit, hit <space> if the initial setup screen does not appear automatically.

First Boot:

A green LED next to the Micro-USB connector will light as soon as the developer kit powers on. When you boot the first time, the developer kit will take you through some initial setup, including:

- Review and accept NVIDIA Jetson software EULA
- Select system language, keyboard layout, and time zone
- Create username, password, and computer name
- Select APP partition size—it is recommended to use the max size suggested

After Logging In: You will see a standard Linux command line prompt in your serial terminal application. Congratulations!

Troubleshooting:

1. Power: If you cannot boot your Jetson Nano Developer Kit, the problem may be with your USB power supply. Please use a good quality power supply. It's also important to have a good quality cord connecting your power supply to the developer kit:

- It's good to use a power supply with a permanently attached cord.
- Shorter cables will drop less voltage.

2. Display: HDMI to DVI adaptors is not supported. Please use a display that accepts HDMI or DP input

5.2 Virtual environment:

Python virtual environments on Jetson Nano: Python virtual environments are a best practice when both developing and deploying Python software projects.

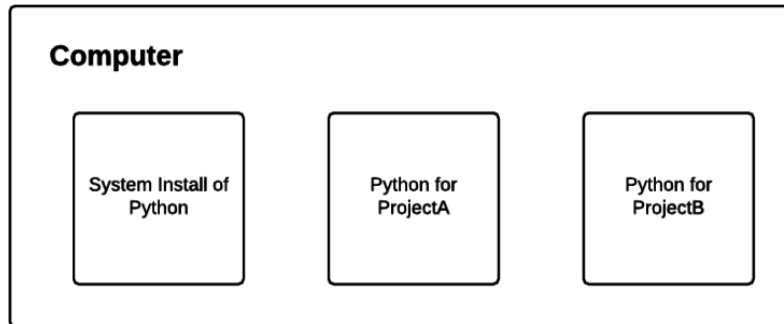


Figure 3.10: Each Python virtual environment you create on your NVIDIA Jetson Nano is separate and independent from the others.

Virtual environments allow for isolated installs of different Python packages. When we use them, we could have one version of a Python library in one environment and another version in a separate, sequestered environment

```

pyimagesearch@pyimagesearch-nano: ~
pyimagesearch@pyimagesearch-nano:~$ mkvirtualenv py3cv4 -p python3
created virtual environment CPython3.6.9.final.0-64 in 579ms
  creator CPython3Posix(dest=/home/pyimagesearch/.virtualenvs/py3cv4, clear=False,
  global=False)
  seeder FromAppData(download=False, pip=latest, setuptools=latest, wheel=latest
  , via=copy, app_data_dir=/home/pyimagesearch/.local/share/virtualenv/seed-app-da
  ta/v1)
  activators BashActivator,CShellActivator,FishActivator,PowerShellActivator,Pyt
  honActivator,XonshActivator
virtualenvwrapper.user_scripts creating /home/pyimagesearch/.virtualenvs/py3cv4/
bin/predeactivate
virtualenvwrapper.user_scripts creating /home/pyimagesearch/.virtualenvs/py3cv4/
bin/postdeactivate
virtualenvwrapper.user_scripts creating /home/pyimagesearch/.virtualenvs/py3cv4/
bin/preactivate
virtualenvwrapper.user_scripts creating /home/pyimagesearch/.virtualenvs/py3cv4/
bin/postactivate
virtualenvwrapper.user_scripts creating /home/pyimagesearch/.virtualenvs/py3cv4/
bin/get_env_details
(py3cv4) pyimagesearch@pyimagesearch-nano:~$

```

Figure 3.11: Terminal output from the virtualenvwrapper setup installation indicates that there are no errors. We now have a virtual environment management system in place so we can create computer vision and deep learning virtual environments on our NVIDIA Jetson Nano.

‘py3cv4’ virtual environment: The Virtualenvwrapper tool provides the following commands to work with virtual environments:

- `mkvirtualenv`: Create a Python virtual environment
- `lsvirtualenv`: List virtual environments installed on your system
- `Rmvirtualenv`: Remove a virtual environment
- `work on`: Activate a Python virtual environment
- `deactivate`: Exits the virtual environment taking you back to your system environment

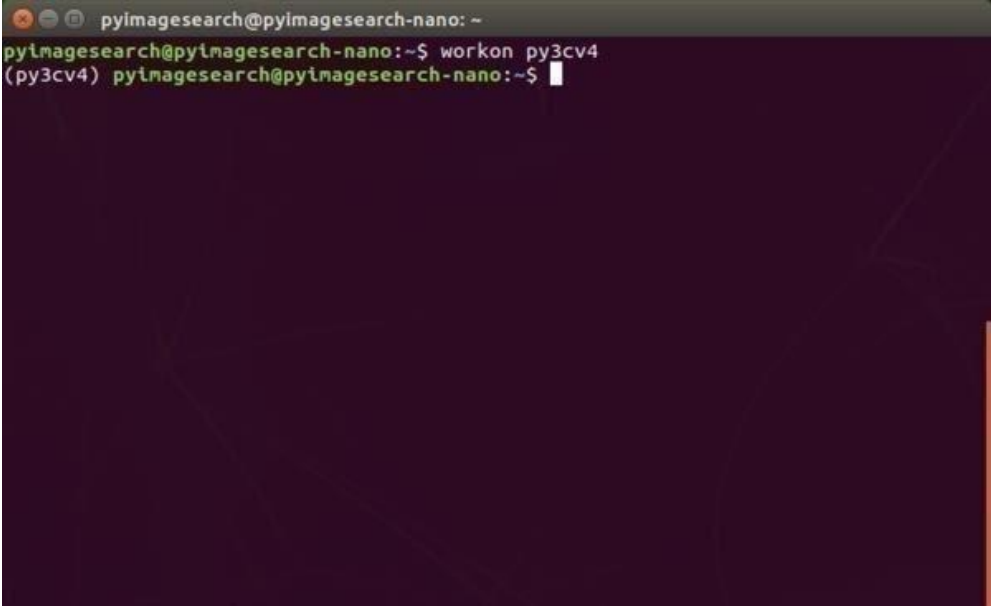
A terminal window screenshot from a Jetson Nano Kit. The window title is 'pyimagesearch@pyimagesearch-nano: ~'. The prompt is 'pyimagesearch@pyimagesearch-nano:~\$'. The user enters the command 'workon py3cv4'. The prompt changes to '(py3cv4) pyimagesearch@pyimagesearch-nano:~\$' with a cursor at the end.

Figure 3.12: Created py3cv4 virtual environment on Jetson Nano Kit.

5.3 TechStack/framework used:

OpenCV: OpenCV is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture, and analysis including features like face detection and object detection.

Keras: Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only

TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

TensorFlow: TensorFlow is an end-to-end open-source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and developers easily build and deploy ML-powered applications.

Why TensorFlow:

- **Easy model building:** TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.
- **Robust ML production anywhere:** TensorFlow has always provided a direct path to production. Whether it's on servers, edge devices, or the web, TensorFlow lets you train and deploy your model easily, no matter what language or platform you use
- **Powerful experimentation for research:** Build and train state-of-the-art models without sacrificing speed or performance. TensorFlow gives you flexibility and control with features like the Keras Functional API and Model Subclassing API for the creation of complex topologies. For easy prototyping and fast debugging, use eager execution

Resnet: A residual neural network (**ResNet**) is an artificial neural network (ANN) of a kind that builds on constructs known from pyramidal cells in the cerebral cortex. Residual neural networks do this by utilizing skip connections, or shortcuts to jump over some layers.

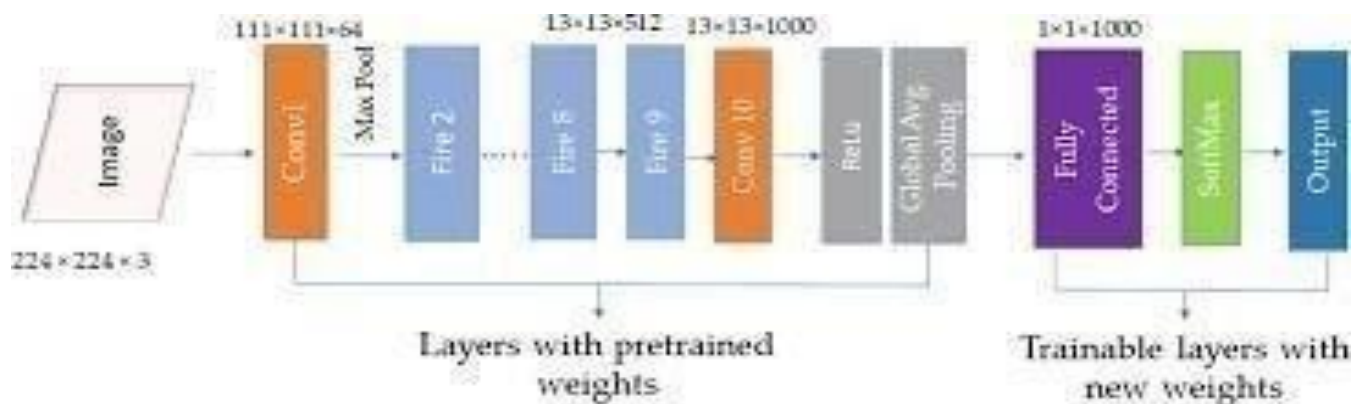


Figure 3.13: Resnet50 architecture

1. MobileNet V2: **MobileNet V2** is a convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers.

The intermediate expansion layer uses lightweight depthwise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

2. CMake: CMake is an open-source, cross-platform family of tools designed to build, test, and package software. CMake is used to control the software compilation process using simple platform and compiler independent configuration files, and generate native makefiles and workspaces that can be used in the compiler environment of your choice. The suite of CMake tools was created by Kitware in response to the need for a powerful, cross-platform build environment for open-source projects such as ITK and VTK.

3. Protobuf Compiler: Protocol Buffers (Protobuf) is a free and open-source cross-platform library used to serialize structured data. It is useful in developing programs to communicate with each other over a network or for storing data. The method involves an interface description language that describes the structure of some data and a program that generates source code from that description for generating or parsing a stream of bytes that represents the structured data.

4. NumPy: **NumPy** is a Python library used for working with arrays. It also has functions for working in the domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open-source project and you can use it freely.

5. SciPy: **SciPy** is a scientific computation library that uses NumPy underneath. SciPy stands for Scientific Python. It provides more utility functions for optimization, stats, and signal processing.

6. VNC Viewer: **In** computing, Virtual Network Computing (VNC) is a graphical desktop sharing system that uses the Remote FrameBuffer protocol (RFB) to remotely control another computer. It transmits the keyboard and mouse input from one computer to another, relaying the graphical-screen updates, over a network.

Multiple clients may connect to a VNC server at the same time. Popular uses for this technology include remote technical support and accessing files on one's work computer from one's home computer, or vice versa.

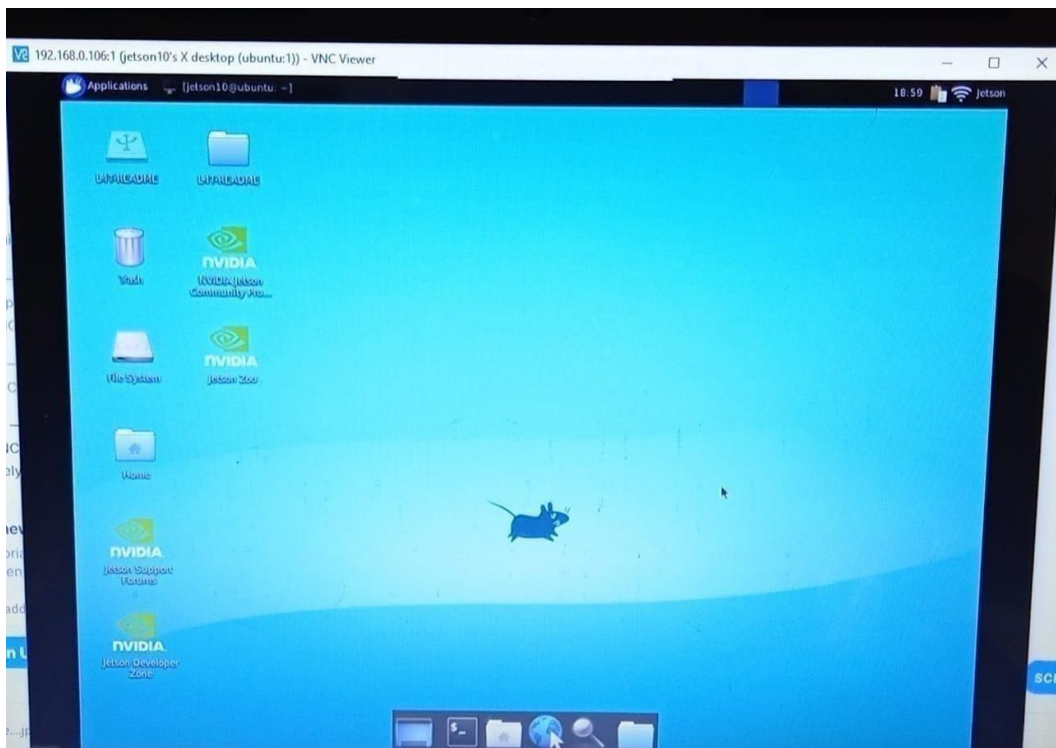


Figure 3.14: set up a VNC server on jetson nano developer kit

CHAPTER 6

METHODOLOGY

METHODOLOGY:

We adopt the object detector framework, which suggests a detection network with a backbone, neck, and head. Object detection is a computer vision technique that allows us to identify and locate objects in an image or video. The backbone network is essentially a truncated high-quality image classifier that is used to extract features that are to be used for prediction in the later stages of the network. Backbone plays an important role in object detectors. The performance of object detectors highly relies on features extracted by the backbone. The backbone takes images as input and extracts the feature maps. In Face Mask Detection, we adopt Residual Networks as a standard backbone. Residual Network is a classic neural network used as a backbone for many computers' vision tasks. The fundamental breakthrough with ResNet was it allows us to train extremely deep neural networks with 150+layers successfully. Also, include MobileNet as a backbone. The MobileNet network architecture is a special class of convolutional neural models that are built using depth-wise separable convolutions and are therefore more lightweight in terms of their parameter count and computational complexity.

The neck is an intermediate component between a backbone and head, and it can enhance or refine original feature maps. In Face Mask Detection, FPN is applied as a neck. Feature Pyramid Network (FPN) is a feature extractor designed for such a pyramid concept with accuracy and speed in mind. It replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (multi-scale feature maps) with better quality information than the regular feature pyramid for object detection. We use a pyramid of the same image at different scales for detection.

Finally, headstands for classifiers, predictors, estimators, etc., can achieve the final objectives of the network. We add a context attention module for neural networks that allows the network to focus on specific aspects of a complex input, one at a time until the entire dataset is categorized. The output of the detection head is through a fully convolutional network rather than a fully connected network to further reduce the number of parameters in the network.

Face Mask Detection consists of an SSD and an FPN and can detect small face masks as well. SSD was the first deep neural architecture that did not use region proposals and featured an End-to-End approach to detecting objects in an image using a single deep neural network that was just as accurate as methods that did. Moreover, with the removal of the region proposal steps, the SSD method was capable of delivering faster execution times.

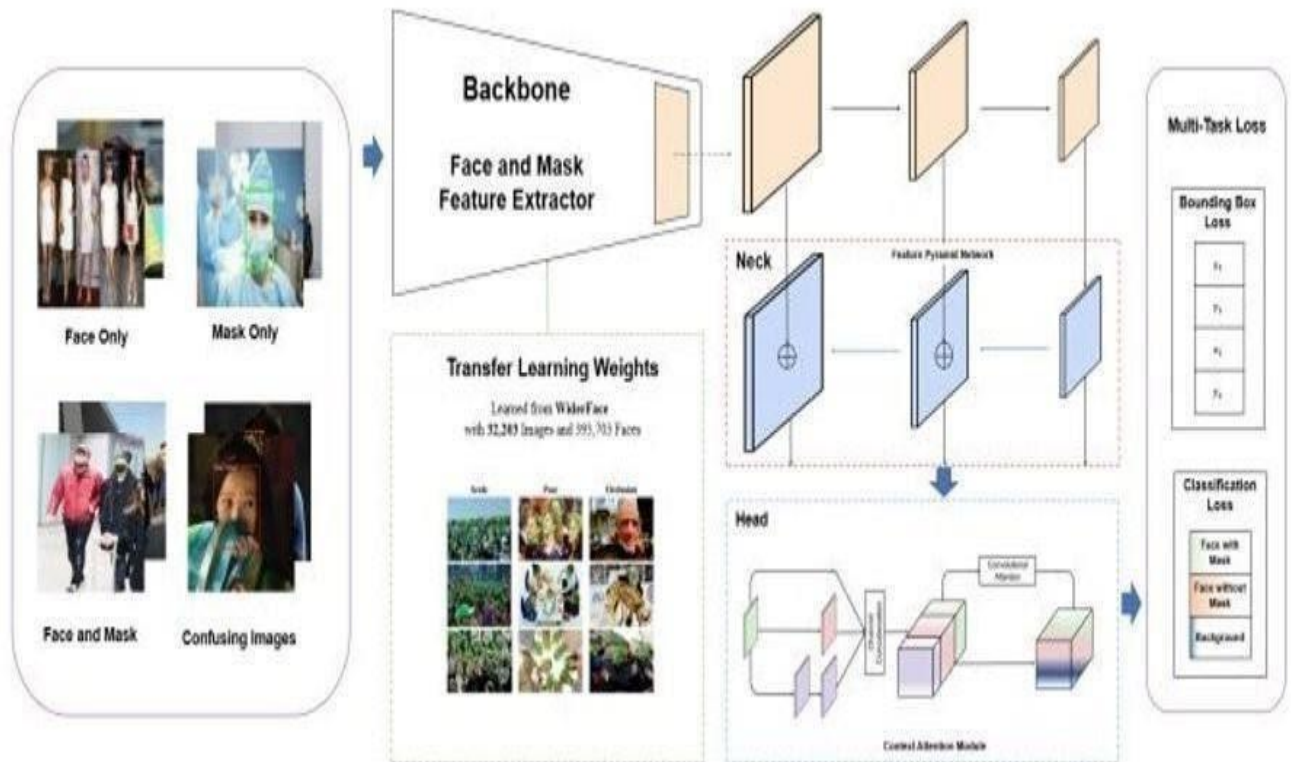


Figure 4.1: Architecture of Face Mask Detection

As deep learning-based methods often require larger datasets, transfer learning is proposed to transfer learned knowledge from a source task to a related target task. To improve the detection performance for face masks, Face mask detection proposes a novel context attention module as its detection heads. We are using the NVIDIA Jetson Nano Developer Kit, which is a small, powerful computer that lets us run multiple neural networks in parallel for applications like image classification, object detection, segmentation, and speech processing. We capture images and video using a Logitech camera of 8MP.

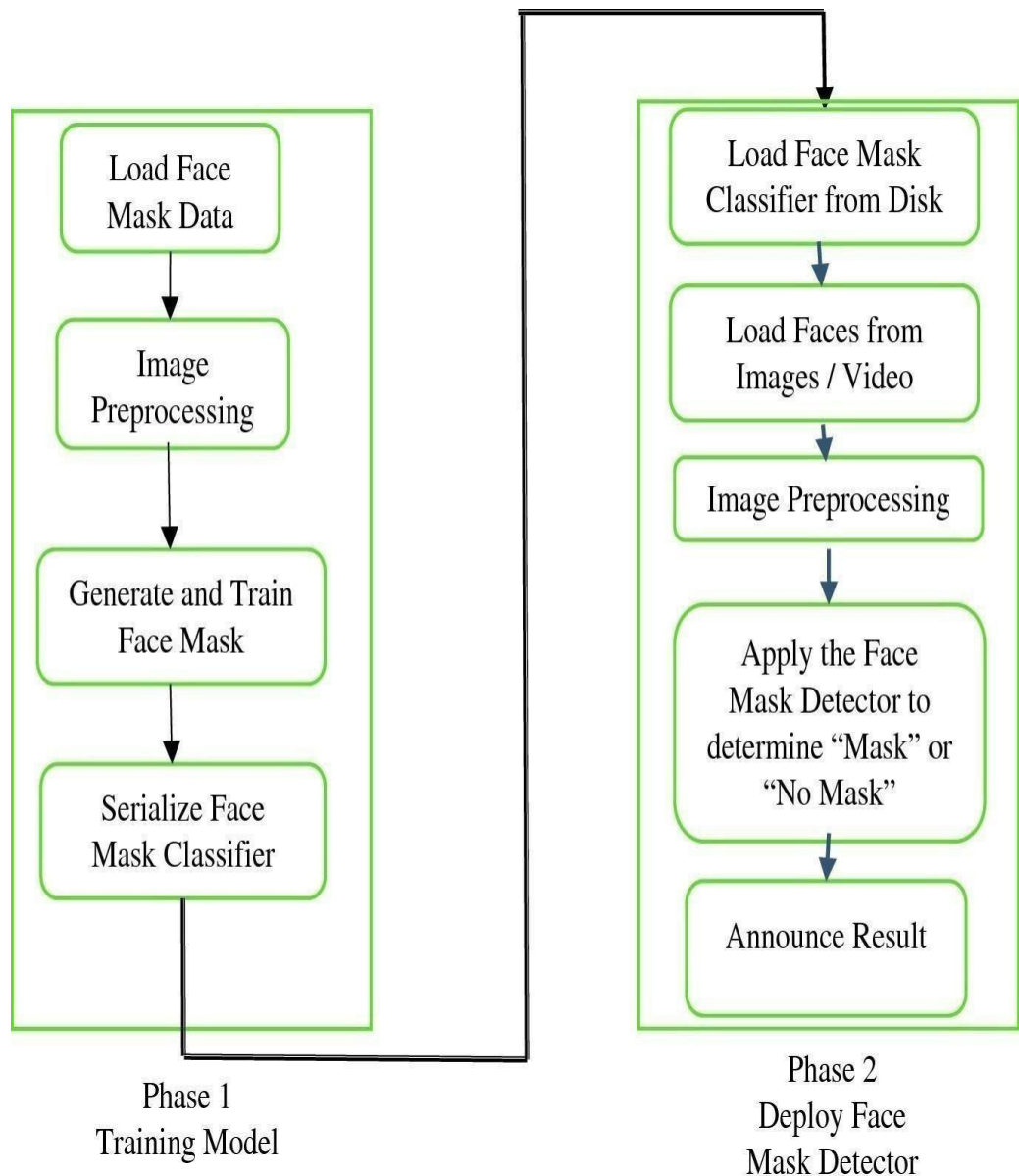


Figure 4.2: Workflow of Face Mask Detection

In the first step, we load data from the disk which consists of different images. Then we process this data using image processing algorithms. We divide data into training data and testing data. In training data, we generate and train to face masks using CNN and SSD algorithms. The training data result in serializing the Face Mask Classifier. Now we apply a Face Mask Classifier to the testing data. Finally, we have a Face Mask Detector which can detect "Mask" or "No Mask" on real-time objects.

CHAPTER 8

RESULTS

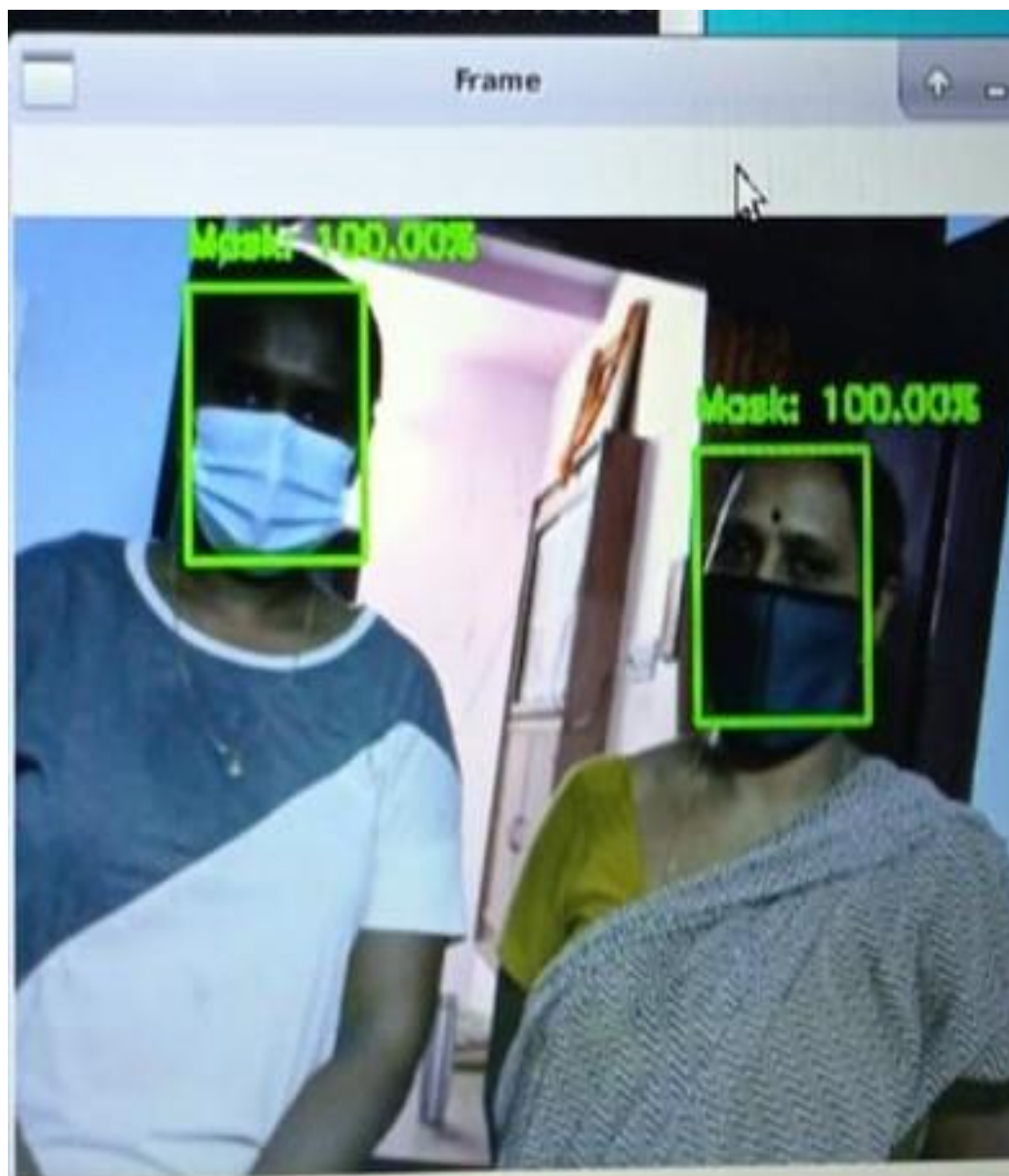
Faces with No Mask



Confusing Face without Mask



Faces with Mask



CONCLUSION

This work is done with help of (NVIDIA Jetson Nano and HD Logitech Camera), ResNet , VNC server. The aim of our project is to detect the facemasks of an individual and help public health . This can be used at shop entrances and allow the persons who wear facemasks perfectly. By using this we can protect the employees at the stores who verify face masks by replacing them. It can also be implemented in our college premises and can prevent spread to some extent.

REFERENCES

- [1] Mingjie Jiang, Xinqi Fan and Hong Yan ,”Retina face mask : A face mask detector ,” arXiv:2005.03950v1, City University of Hong Kong,May 11, 2020.
.com/chandrikadeb7/Face-Mask-Detection
- [2] Chandrikadeb7, “Face Mask Detection.” [Online]. Available : <https://github.com/chandrikadeb7/Face-Mask-Detection>
- [3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015
- [4] Yamashita, R., Nishio, M., Do, R.K.G. *et al.* Convolutional neural networks: an overview and application in radiology. *Insights Imaging* **9**, 611–629 (2018).
- [5]] Z. Zou, Z. Shi, Y. Guo, and J. Ye, “Object detection in 20 years: A survey,” arXiv preprint arXiv:1905.05055, 2019.
- [6] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu et al., “Mmdetection: Open mmlab detection toolbox and benchmark,” arXiv preprint arXiv:1906.07155, 2019
- [7] M. Najibi, P. Samangouei, R. Chellappa, and L. S. Davis, “Ssh: Single stage headless face detector,” in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 4875–4884.
- [8] R. Girshick, “Fast r-cnn,” in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1440–1448.
- [9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009, pp. 248–255.
- [10] A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way -3bd2b1164a53. Sumit Saha , Dec 15, 2018.[Online]. Available : <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>
- [11] 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, 2014, pp. 580–587.
- [12] Dasiopoulou, Stamatia, et al. "[Knowledge-assisted semantic video object detection.](#)" IEEE Transactions on Circuits and Systems for Video Technology 15.10 (2005): 1210–1224).