

Project Report

On

**Image Classification using Deep Learning: A case study
on Covid- 19 Prediction**

Submitted for partial fulfillment of requirement for the degree of

BACHELOR OF ENGINEERING

(Computer Science and Engineering)

Submitted By

**Raghav Kavimandan
Yadnyawalk Gudadhe
Aditya Parate
Yashodhan Mohod**

**Under the Guidance of
Prof. S.S Dandge**



Department of Computer Science & Engineering,

**Prof. Ram Meghe Institute of Technology &
Research, Badnera**

2022-2023



Department of Computer Science & Engineering
Prof. Ram Meghe Institute of Technology &
Research, Badnera
2022 - 2023

CERTIFICATE

This is to certify that the Project (8KS07) entitled

**Image Classification using Deep Learning: A case study
on Covid- 19 Prediction**

is a bonafide work and it is submitted to the
Sant Gadge Baba Amravati University, Amravati

By

Raghav Kavimandan

Yadnyawalk Gudadhe

Aditya Parate

Yashodhan Mohod

in the partial fulfillment of the requirement for the degree of
Bachelor of Engineering in Computer Science & Engineering,
during the academic year 2022-2023 under my guidance.

Prof. S.S Dandge

Guide

Department of Computer Sci.& Engg.
Prof. Ram Meghe Institute of Technology &
Research, Badnera

Dr. M. A. Pund

Head,

Department of Computer Sci.& Engg.
Prof. Ram Meghe Institute of Technology &
Research, Badnera

External Examiner

Acknowledgement

We would like to extend our deepest gratitude to our guide Prof. S.S. Dandge, and the Head of the Computer Science Department, Prof. M. A. Pund for their unwavering support and guidance throughout this project. Their wealth of knowledge and expertise in the field of deep learning and image classification has been instrumental in shaping our understanding of complex concepts and techniques. Their encouragement and mentorship have been pivotal in ensuring that we remained focused and committed to our project, and we feel truly privileged to have had the opportunity to work with them.

As the developers of this project, we would also like to acknowledge our own contributions to this endeavor. Our hard work and dedication have been instrumental in achieving the goals that we set out to accomplish, and we are proud of the results that we have achieved.

We are also extremely grateful to our institution for providing us with the resources and facilities necessary to carry out this project. The support of our fellow students and faculty members has been invaluable in helping us navigate the challenges that arose during the course of our research.

Finally, we would like to acknowledge the patients and healthcare professionals who made this project possible. The dataset that we used for our study consisted of medical images obtained from real patients with COVID-19. We recognize that these patients and healthcare professionals have been on the frontlines of the battle against the pandemic, and we are humbled by their courage and resilience. We hope that our project can contribute in some small way to the fight against this disease, and we remain deeply grateful to all those who made it possible.

Table of Contents

Contents

1. Introduction.....	4
2. Objective and Scope	6
3. Literature survey	7
3.1 CNN (Convolutional neural network)	7
3.2 ResNets	10
3.3 Inception-v3.....	11
3.4 VGG16	12
3.5 DenseNet	14
3.6 MobileNet.....	15
3.7 SqueezeNet.....	17
3.8 AlexNet	18
3.9 GoogLeNet	20
3.10 YOLOv7	22
4. System Analysis.....	25
4.1 Problem Statement:	25
4.2 System Requirement.....	25
4.2.1 VS Code:.....	25
4.2.2 Flask Micro-framework:.....	26
4.2.3 Sqlite3 Database:	27
4.3 Technologies Used:	29
4.3.1 Python:	29
4.3.2 Bootstrap:.....	30
4.3.3 JavaScript:.....	32

4.4 Platforms Used:	33
5. System Design	34
5.1 System Architecture:	34
5.1.1 Image Classification:	34
5.1.2 Image Preprocessing:	36
5.2 Proposed Model:	37
5.2.1 Yolov7:	37
5.2.2 Model Re-Parameterization.....	39
5.3 Data Flow Diagram	42
6. Implementation and Result	45
6.1 Implementation:.....	45
6.1.1 Database connectivity with Flask App:	45
6.1.2 Model Training:	47
6.1.3 Yolov7 Algorithm for Image Classification:	49
6.2 Applications of Image Classification	51
6.3 Results	52
7. Conclusion and Future Scope	56
7.1 Conclusion:	56
7.2 Future Scope:.....	57
References.....	58

Table of Figures

Figure 1 - CNN architecture	8
Figure 2 - ResNet architecture	10
Figure 3 - ResNet workflow	11
Figure 4 - ResNet workflow	11
Figure 5 - SqueezeNet architecture.....	18
Figure 6 - AlexNet architecture	19
Figure 7 - Flask workflow.....	27
Figure 8 - User Database.....	28
Figure 9 - Python applications	30
Figure 10 - Image classification.....	35
Figure 11 - Data Pre-processing	36
Figure 12 - Model workflow	37
Figure 13 - YOLOv7 workflow	39
Figure 14 - Database table creation	46
Figure 15 - User database creation.....	46
Figure 16 - Model training.....	48
Figure 17 – Home Page	52
Figure 18 – Login Page.....	52
Figure 19 – Register Page	53
Figure 20 – Index Page	53
Figure 21 – Comparator Page	54
Figure 22 – Prediction Page.....	55
Figure 23 – Index page with Uploaded Image.....	55

1. Introduction

On 31st December 2019, in the city of Wuhan (CHINA), a cluster of cases of pneumonia of unknown cause was reported to World Health organization. In January 2020, a previously unknown new virus was identified, subsequently named 2019 novel corona virus. Thereupon WHO declared the COVID-19 as a pandemic. A pandemic is defined as disease spread over a wide range of geographical area and that has affected high proportion of the population. This was news to the whole world as we had not faced a pandemic since many years. Every country was in a race to study the virus and find the cure as soon as possible to prevent it from spreading further. In this project, the aim is to utilize the vast capabilities of AI to detect the virus and possibly find a solution as early as possible. Real-life human lung images are collected and annotated before feeding them to the model. The CNN model and YOLO object detection model can be employed for boundary detection and model training, respectively. After executing the training step, the model is assessed and tested upon unseen images based on MAP values. The image detected with COVID-19 is highlighted using a bounding box around the highly infected area. The model developed here could be easily employed by hospitals to assist doctors in prematurely detecting the virus and finding out a possible cure.

Symptoms of COVID-19 include fever, cough, shortness of breath, and loss of taste or smell. Other symptoms can include fatigue, body aches, sore throat, and congestion. Some individuals may be asymptomatic or have mild symptoms, while others may develop severe illness.

The severity of the illness can range from mild to severe, with some individuals requiring hospitalization or intensive care. Those at increased risk for severe illness include older adults, individuals with underlying medical conditions, and those who are immunocompromised.

The best way to prevent the spread of COVID-19 is to get vaccinated and continue practicing preventative measures such as wearing a mask and washing your hands frequently. Vaccines have been shown to be highly effective at preventing severe illness, hospitalization, and death from COVID-19.

Vaccines against COVID-19 have been authorized for emergency use by regulatory agencies around the world. There are currently multiple approved vaccines available, including Pfizer-BioNTech, Moderna, Johnson & Johnson, and AstraZeneca.

COVID-19 variants, such as the Delta variant, have emerged and are more transmissible than the original strain of the virus. This has led to increased concern about the effectiveness of vaccines and the need for continued preventative measures. The pandemic has had significant social and economic impacts, leading to lockdowns, school closures, and job losses. It has also highlighted existing health disparities, with certain populations being disproportionately affected by COVID-19.

Treatment options for COVID-19 include antiviral medications, monoclonal antibodies, and steroids. These treatments are most effective when given early in the course of the illness and under the guidance of a healthcare professional.

The World Health Organization (WHO) and public health authorities recommend wearing a mask, social distancing, and getting vaccinated to help prevent the spread of COVID-19. In addition, individuals should stay home if they are feeling sick and seek medical attention if their symptoms worsen.

COVID-19 testing is an important tool for identifying individuals who are infected with the virus and preventing further spread. Testing can be done using a variety of methods, including PCR tests, antigen tests, and antibody tests.

Contact tracing is another important tool for controlling the spread of COVID-19. This involves identifying and notifying individuals who have been in close contact with someone who has tested positive for the virus.

While COVID-19 primarily affects the respiratory system, it can also cause other complications such as blood clots, heart inflammation, and long-term symptoms such as fatigue and brain fog.

COVID-19 vaccines are available for children aged 5 and older, and studies are ongoing to determine the safety and effectiveness of vaccines for younger children. Pregnant and breastfeeding individuals are also eligible for COVID-19 vaccination.

As the pandemic continues, it is important to remain vigilant and continue practicing preventative measures to help prevent the spread of COVID-19. This includes getting vaccinated, wearing a mask, social distancing, and following public health guidelines.

2. Objective and Scope

1. Detect COVID-19 virus by understanding its nature, symptoms, and its modus operandi to prevent it from further spreading.
2. The study of object detection models to outline possible solutions to achieve the task at hand.
3. Identify the best model and its parameters that give the most promising results on the test data set.
4. From the probable solutions, the YOLO algorithm along with a CNN architecture is employed to attain best possible results.
5. To assess the performance of the model, the MAP (Mean Average Precision) values are used and tested against unseen images

3. Literature survey

In the Literature survey, several candidate algorithms are presented, analyzed and compared based on their performance metrics, time complexities and general workflow. Here, we have showcased the workflow, applications, advantages and disadvantages of 10 algorithms that are the best choices for the image classification task. All these algorithms are based on neural nets which require a large amount of data for model training. Deep Learning algorithms like CNN, DenseNet, MobileNet, AlexNet, Yolov7, etc. are included in the below survey to study and compare them in order to find the best candidate for the task at hand.

3.1 CNN (Convolutional neural network)

In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks in deep learning.

Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contest with an amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications.

CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognizing handwritten digits. It was mostly used in the postal sectors to read zip codes, pin codes, etc. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period and

hence CNNs were only limited to the postal sectors and it failed to enter the world of machine learning.

In 2012 **Alex Krizhevsky** realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

In deep learning, a convolutional neural network (CNN/ConvNet) is a class of deep neural networks, most commonly applied to analyse 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.

CNN architecture

CNNs are comprised of three types of layers. These are convolutional layers, pooling layers and fully-connected layers. When these layers are stacked, CNN architecture has been formed. Simplified CNN architecture for MNIST classification is illustrated in Figure

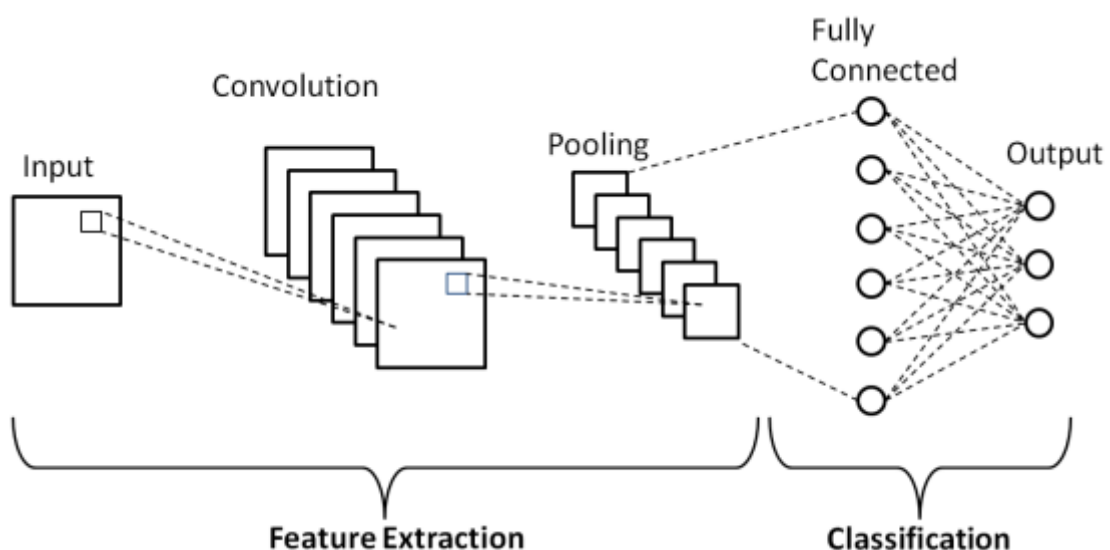


Figure 1 - CNN architecture

The basic functionality of the example CNN above can be broken down into four key areas.

1. As found in other forms of ANN, the input layer will hold the pixel values of the image.
2. The convolutional layer will determine the output of neurons of which are connected to local regions of the input through the calculation of the scalar product between their weights and the region connected to the input volume. The rectified linear unit (commonly shortened to ReLu) aims to apply an 'element wise' activation function such as sigmoid to the output of the activation produced by the previous layer.
3. The pooling layer will then simply perform down sampling along the spatial dimensionality of the given input, further reducing the number of parameters within that activation.
4. The fully-connected layers will then perform the same duties found in standard ANNs and attempt to produce class scores from the activations, to be used for classification. It is also suggested that ReLu may be used between these layers, as to improve performance.

Through this simple method of transformation, CNNs are able to transform the original input layer by layer using convolutional and down sampling techniques to produce class scores for classification and regression purposes.

Advantages of Convolutional Neural Networks (CNNs):

1. Good at detecting patterns and features in images, videos, and audio signals.
2. Robust to translation, rotation, and scaling invariance.
3. End-to-end training, no need for manual feature extraction.
4. Can handle large amounts of data and achieve high accuracy.

Disadvantages of Convolutional Neural Networks (CNNs):

1. Computationally expensive to train and require a lot of memory.
2. Can be prone to over fitting if not enough data or proper regularization is used.
3. Requires large amounts of labeled data.
4. Interpretability is limited; it's hard to understand what the network has learned.

3.2 ResNets

When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. Using deeper networks is degrading the performance of the model. Microsoft Research paper tries to solve this problem using Deep Residual learning framework.

The idea is that instead of letting layers learn the underlying mapping, let the network fit the residual mapping. So, instead of say $H(x)$, initial mapping, let the network fit, $F(x) = H(x) - x$ which gives $H(x) = F(x) + x$

The approach is to add a shortcut or a skip connection that allows information to flow, well just say, more easily from one layer to the next's next layer, i.e., you bypass data along with normal CNN flow from one layer to the next layer after the immediate next.

A Residual Block:

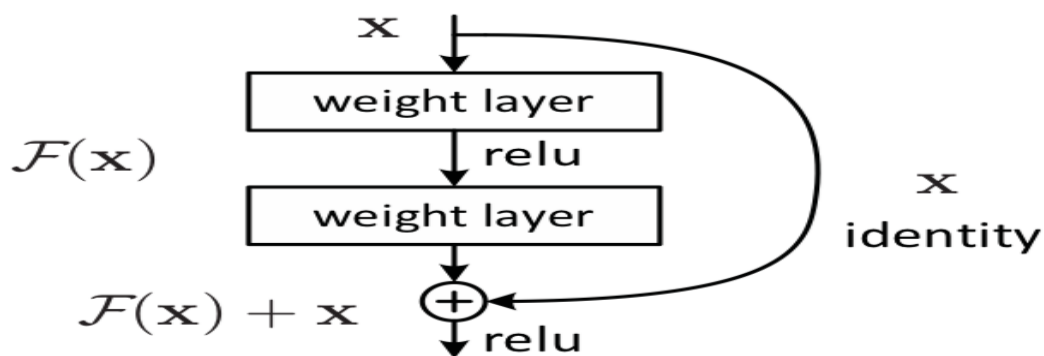


Figure 2 - ResNet architecture

Two takeaways from residual block:

Adding additional / new layers would not hurt the model's performance as regularization will skip over them if those layers were not useful.

If the additional / new layers were useful, even with the presence of regularization, the weights or kernels of the layers will be non-zero and model performance could increase slightly. Therefore, by adding new layers, because of the "Skip connection" / "residual connection", it is guaranteed that performance of the model does not decrease but it could increase slightly. By stacking these ResNet blocks on top of each

other, you can form a very deep network. Having ResNet blocks with the shortcut also makes it very easy for one of the blocks to learn an identity function.

These Two main types of blocks are used in a ResNet, depending mainly on whether the input/output dimensions are same or different, means that you can stack on additional ResNet blocks with little risk of harming training set performance.

1. The identity block — same as the one we saw above. The identity block is the standard block used in ResNets and corresponds to the case where the input activation has the same dimension as the output activation.

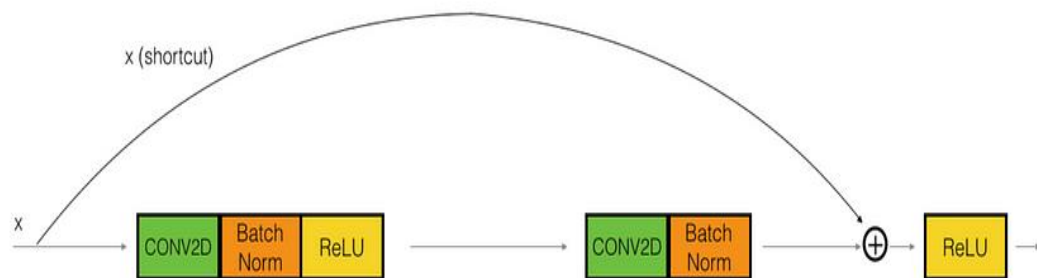


Figure 3 - ResNet workflow

2. The Convolutional block — we can use this type of block when the input and output dimensions don't match up. The difference with the identity block is that there is a CONV2D layer in the shortcut path.

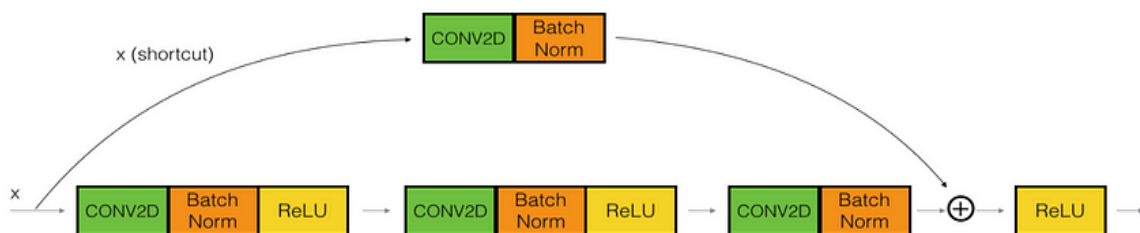


Figure 4 - ResNet workflow

3.3 Inception-v3

Inception v3 mainly focuses on burning less computational power by modifying the previous Inception architectures. This idea was proposed in the paper rethinking the Inception Architecture for Computer Vision, published in 2015. It was co-authored by

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, and Jonathon Shlens [3]. In comparison to VGGNet, Inception Networks (GoogLeNet/Inception v1) have proved to be more computationally efficient, both in terms of the number of parameters generated by the network and the economic cost incurred (memory and other resources). If any changes are to be made to an Inception Network, care needs to be taken to make sure that the computational advantages aren't lost. Thus, the adaptation of an Inception network for different use cases turns out to be a problem due to the uncertainty of the new network's efficiency. In an Inception v3 model, several techniques for optimizing the network have been put suggested to loosen the constraints for easier model adaptation. The techniques include factorized convolutions, regularization, dimension reduction, and parallelized computations [3].

Inception v3 Architecture

The architecture of an Inception v3 network is progressively built, step-by-step, as explained below:

1. Factorized Convolutions: this helps to reduce the computational efficiency as it reduces the number of parameters involved in a network. It also keeps a check on the network efficiency.
2. Smaller convolutions: replacing bigger convolutions with smaller convolutions definitely leads to faster training. Say a 5×5 filter has 25 parameters; two 3×3 filters replacing a 5×5 convolution has only 18 ($3*3 + 3*3$) parameters instead.

The inception V3 is just the advanced and optimized version of the inception V1 model. The Inception V3 model used several techniques for optimizing the network for better model adaptation.

1. It has higher efficiency
2. It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised.
3. It is computationally less expensive.
4. It uses auxiliary Classifiers as regularizes.

3.4 VGG16

VGG16 is a deep convolutional neural network (CNN) model that was proposed by Karen Simonyan and Andrew Zisserman [2] of the Visual Geometry Group (VGG) at

the University of Oxford in 2014. VGG16 is a convolutional neural network model that consists of 16 layers, including 13 convolutional layers, 5 max-pooling layers, and 3 fully connected layers. The input to the network is an RGB image with dimensions of 224x224 pixels.

Each of the convolutional layers uses small 3x3 filters, and the number of filters in each layer increases from 64 in the first layer to 512 in the last. The max-pooling layers have a 2x2 filter and a stride of 2, which reduces the spatial size of the feature maps by a factor of 2. After the convolutional and pooling layers, the output is flattened and passed through three fully connected layers, each with 4096 neurons, followed by a final SoftMax layer that outputs the class probabilities. One of the key advantages of VGG16 is its simplicity [2]. By using only 3x3 filters and stacking multiple convolutional layers, the network can be made deeper without increasing the number of parameters too much. This allows the model to capture more complex features in the input images, which leads to better performance on visual recognition tasks.

Advantages: -

- 1) Good performance: VGG16 has achieved excellent performance on many visual recognition tasks, including image classification, object detection, and image segmentation.
- 2) Transfer learning: VGG16 can be used as a pre-trained model for transfer learning, where the weights of the convolutional layers are frozen and only the fully connected layers are fine-tuned for a new task. This allows the model to achieve good performance with less training data.
- 3) Simplicity: The architecture of VGG16 is relatively simple, with only 3x3 convolutional filters and stacking multiple convolutional layers. This allows the model to capture more complex features in the input images, leading to better performance.

Disadvantages: -

- 1) High computational requirements: VGG16 has a large number of parameters, which makes it computationally expensive to train and run on low-end hardware.

2)Overfitting: The large number of parameters in VGG16 can also lead to overfitting if the model is not regularized properly. This can be addressed by using techniques such as dropout or weight decay during training.

3)Lack of scalability: While VGG16 has been successful on many visual recognition tasks, its fixed architecture makes it difficult to scale up to larger or more complex datasets. This has led to the development of more flexible models, such as ResNet and Inception.

3.5 DenseNet

DenseNet is a deep neural network architecture that was proposed by Huang et al [1]. in their 2017 paper titled "Densely Connected Convolutional Networks". DenseNet is similar to other convolutional neural network (CNN) architectures, such as VGG and ResNet, but it has a unique feature called "dense connections".

In a traditional CNN, each layer takes the output of the previous layer as its input. However, in DenseNet, each layer receives the feature maps from all the previous layers as inputs. This means that every layer in the network is connected to every other layer, forming a dense block. By doing so, the model can learn more complex features with fewer parameters, which can lead to better performance and reduced overfitting. Another important feature of DenseNet is its use of bottleneck layers, which reduce the number of input feature maps before a dense layer to reduce computational complexity. This also encourages feature reuse, as the reduced number of channels still provides sufficient information for the subsequent layers to learn from.

DenseNet [1] has shown excellent performance on a variety of image classification tasks, including the ImageNet dataset. It has also been applied to other tasks such as object detection and image segmentation. Some advantages of DenseNet include its ability to reduce the number of parameters and the risk of overfitting, as well as its high accuracy and robustness to adversarial attacks. However, it can also be computationally expensive to train and run due to the large number of connections between layers.

Advantages: -

- 1)High accuracy: DenseNet has achieved state-of-the-art results on several image classification benchmarks, including the ImageNet dataset, with fewer parameters than other models.
- 2)Parameter efficiency: Due to its dense connections, DenseNet can achieve high accuracy with fewer parameters than other architectures, which can reduce the risk of overfitting and make it more computationally efficient.
- 3)Feature reuse: DenseNet encourages feature reuse and information flow between layers, which can lead to better learning and reduced redundancy.
- 4)Versatility: DenseNet can be used for various tasks such as image classification, object detection, and semantic segmentation.

Disadvantages: -

- 1)High computational requirements: DenseNet can be computationally expensive to train and run due to the large number of connections between layers, which can make it difficult to use on low-end hardware.
- 2)Memory requirements: Due to the dense connections, the feature maps from all previous layers need to be stored in memory during training, which can require a large amount of memory.
- 3)Complex architecture: DenseNet's architecture can be more difficult to understand and implement than simpler models such as VGG or ResNet.
- 4)Sensitivity to hyperparameters: DenseNet's performance can be sensitive to the choice of hyperparameters such as number of layers, growth rate, and learning rate, which can make it difficult to optimize.

3.6 MobileNet

MobileNet is a deep neural network architecture that was proposed by Howard et al [4]. in their 2017 paper titled "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications". MobileNet was designed specifically to enable high-

performance computer vision applications on resource-constrained devices such as mobile phones and embedded systems.

MobileNet uses depth wise separable convolutions, which separate the spatial convolution and channel-wise convolution into two separate layers. Depth wise convolution is applied to each input channel separately, while pointwise convolution is applied across channels. This reduces the number of parameters and the computational cost, while still allowing the network to learn complex representations. MobileNet also uses global depth wise pooling, which reduces the spatial dimensionality of the feature maps before the fully connected layers.

MobileNet comes in several versions with different sizes and complexities, including MobileNet v1, MobileNet v2, and MobileNet v3 [4]. MobileNet has been widely used for various computer vision tasks such as image classification, object detection, and semantic segmentation. Its efficient design has enabled high-performance computer vision applications on mobile and embedded devices.

Advantages: -

1)Efficiency: MobileNet is specifically designed to be computationally efficient and has a small model size, making it ideal for deployment on mobile and embedded devices with limited computing resources.

2)Accuracy: Despite its small size, MobileNet has shown competitive performance on various computer vision tasks, such as image classification, object detection, and semantic segmentation.

3)Flexibility: MobileNet can be easily customized and adapted to different tasks and constraints, such as changing the input resolution or adjusting the number of layers.

Disadvantages: - 1) Limited capacity: Due to its efficient design, MobileNet has fewer parameters than larger and more complex neural networks, which can limit its capacity to learn complex representations.

2)Not suitable for all tasks: While MobileNet can perform well on many computer vision tasks, it may not be the best choice for some applications that require more complex or specialized models.

3) Training challenges: The depth wise separable convolutions used in MobileNet can be challenging to train and optimize, requiring careful tuning of hyperparameters and regularization techniques to prevent overfitting.

3.7 SqueezeNet

Squeeze Net is a small CNN architecture which has fewer parameters than other CNN's. Nevertheless, it achieves the same results (accuracy) as architectures like AlexNet. SqueezeNet has several building blocks named fire modules. This fire module contains a squeeze convolution layer (which has only 1×1 filters), feeding into an expand layer that has a mix of 1×1 and 3×3 convolution filters. a convolutional neural network that employs design strategies to reduce the number of parameters, notably with the use of fire modules that "squeeze" parameters using 1×1 convolutions [5].

Architectural Design:

Strategy 1. Replace 3×3 filters with 1×1 filters • Given a budget of a certain number of convolution filters, we can choose to make the majority of these filters 1×1 , since a 1×1 filter has $9 \times$ fewer parameters than a 3×3 filter.

Strategy 2. Decrease the number of input channels to 3×3 filters • Consider a convolution layer that is comprised entirely of 3×3 filters. The total quantity of parameters in this layer is:

$(\text{number of input channels}) \times (\text{number of filters}) \times (3 \times 3)$ • We can decrease the number of input channels to 3×3 filters using squeeze layers, mentioned in the next section.

Strategy 3. Down sample late in the network so that convolution layers have large activation maps. The intuition is that large activation maps (due to delayed down sampling) can lead to higher classification accuracy [5].

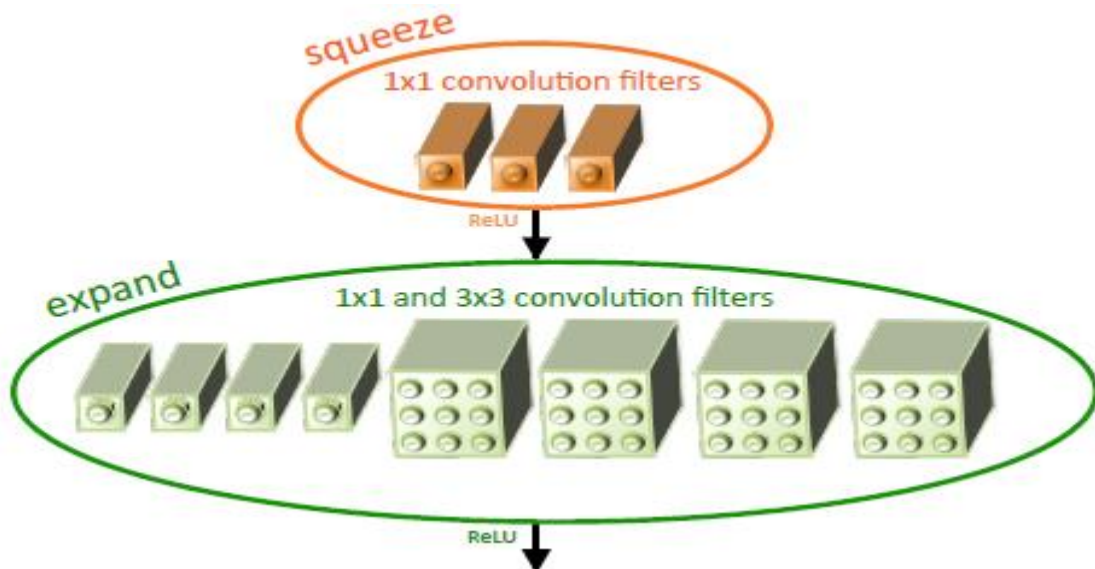


Figure 5 - SqueezeNet architecture

Advantages and disadvantages:

1. Smaller Convolutional Neural Networks (CNNs) require less communication across servers during distributed training.
2. Smaller CNNs require less bandwidth to export a new model from the cloud to an autonomous car.
3. Smaller CNNs are more feasible to deploy on FPGAs and other hardware with limited memory.

3.8 AlexNet

The architecture consists of eight layers: five convolutional layers and three fully-connected layers. But this isn't what makes AlexNet special; these are some of the features used that are new approaches to convolutional neural networks:

- **ReLU Nonlinearity.** AlexNet uses Rectified Linear Units (ReLU) instead of the tanh function, which was standard at the time. ReLU's advantage is in training time; a CNN

using ReLU was able to reach a 25% error on the CIFAR-10 dataset six times faster than a CNN using tanh [6].

- **Multiple GPUs.** Back in the day, GPUs were still rolling around with 3 gigabytes of memory (nowadays those kinds of memory would be rookie numbers). This was especially bad because the training set had 1.2 million images [6]. AlexNet allows for multi-GPU training by putting half of the model's neurons on one GPU and the other half on another GPU. Not only does this mean that a bigger model can be trained, but it also cuts down on the training time.
- **Overlapping Pooling.** CNNs traditionally “pool” outputs of neighboring groups of neurons with no overlapping. However, when the authors introduced overlap, they saw a reduction in error by about 0.5% and found that models with overlapping pooling generally find it harder to overfit.

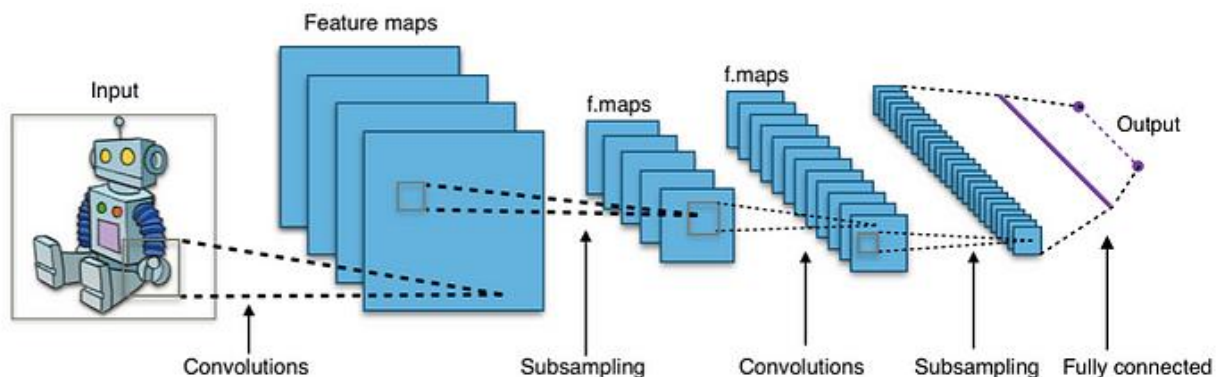


Figure 6 - AlexNet architecture

Advantages:

- This network was a good introduction into the world of neural networks and is really simple to understand. It works well for character recognition images.

Disadvantages

- This model was more specifically built for a certain use case. While it was a major breakthrough in 1998, it does not do as well with color images. Most image recognition problems would require RGB images for better recognition.

- Since the model isn't very deep, it struggles to scan for all features thus producing poor performing models. If the Neural Network isn't fed with enough features from the training images then it would be difficult for the model to generalize and create an accurate model.

3.9 GoogLeNet

GoogLeNet is a 22-layer deep convolutional neural network that's a variant of the Inception Network, a Deep Convolutional Neural Network developed by researchers at Google.

The GoogLeNet architecture presented in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) solved computer vision tasks such as image classification and object detection — find out how well it performed at the conclusion section of this article [7].

Features of GoogleNet:

The GoogLeNet architecture is very different from previous state-of-the-art architectures such as AlexNet and ZF-Net. It uses many different kinds of methods such as 1×1 convolution and global average pooling that enables it to create deeper architecture. In the architecture, we will discuss some of these methods:

- 1×1 convolution : The inception architecture uses 1×1 convolution in its architecture. These convolutions used to decrease the number of parameters (weights and biases) of the architecture. By reducing the parameters we also increase the depth of the architecture. Let's look at an example of a 1×1 convolution below:
- For Example, If we want to perform 5×5 convolution having 48 filters without using 1×1 convolution as intermediate:
- Total Number of operations : $(14 \times 14 \times 48) \times (5 \times 5 \times 480) = 112.9 \text{ M}$
- With 1×1 convolution :
 - $(14 \times 14 \times 16) \times (1 \times 1 \times 480) + (14 \times 14 \times 48) \times (5 \times 5 \times 16) = 1.5\text{M} + 3.8\text{M} = 5.3\text{M}$ which is much smaller than 112.9M.

- Global Average Pooling :

In the previous architecture such as AlexNet, the fully connected layers are used at the end of the network. These fully connected layers contain the majority of parameters of many architectures that causes an increase in computation cost.

In GoogLeNet architecture, there is a method called global average pooling is used at the end of the network. This layer takes a feature map of 7×7 and averages it to 1×1 . This also decreases the number of trainable parameters to 0 and improves the top-1 accuracy by 0.6%

- Inception Module:

The inception module is different from previous architectures such as AlexNet, ZF-Net. In this architecture, there is a fixed convolution size for each layer.

In the Inception module 1×1 , 3×3 , 5×5 convolution and 3×3 max pooling performed in a parallel way at the input and the output of these are stacked together to generated final output. The idea behind that convolution filters of different sizes will handle objects at multiple scale better.

- Auxiliary Classifier for Training:

Inception architecture used some intermediate classifier branches in the middle of the architecture, these branches are used during training only. These branches consist of a 5×5 average pooling layer with a stride of 3, a 1×1 convolutions with 128 filters, two fully connected layers of 1024 outputs and 1000 outputs and a softmax classification layer. The generated loss of these layers added to total loss with a weight of 0.3. These layers help in combating gradient vanishing problem and also provide regularization [7].

Advantages of GoogleNet

- GoogleNet trains faster than VGG.

- Size of a pre-trained GoogleNet is comparatively smaller than VGG. A VGG model can have > 500 MBs, whereas GoogleNet has a size of only 96 MB.

3.10 YOLOv7

YOLOv7 is a real-time object detection algorithm developed by a group of researchers at the University of California, Davis. It is based on the YOLO (You Only Look Once) series of object detection models, which are known for their high accuracy and fast inference times. The team is led by Dr. Boqing Gong, who is an assistant professor in the Department of Computer Science at UC Davis. Other members of the team include Xin Wang, Weixin Liang, and Jingbo Wang [10]. The YOLOv7 algorithm is an extension of the original YOLO algorithm, which was developed by Joseph Redmon and Ali Farhadi at the University of Washington in 2016. Since then, several versions of YOLO have been developed by various researchers, with YOLOv7 being one of the latest and most advanced versions. The algorithm is primarily used for achieving great accuracy and frame rate on real time object detection tasks.

Architecture and Code:

The YOLOv7 architecture is a deep convolutional neural network that is designed for object detection tasks. The architecture is based on the YOLO (You Only Look Once) series of object detection models, which are known for their high accuracy and fast inference times. The YOLOv7 architecture features a modified Darknet backbone, which is a convolutional neural network that extracts features from images. The network is made up of several convolutional layers, with each layer detecting different features at different scales. The YOLOv7 architecture also uses anchor-based object detection, which involves predicting the locations and sizes of objects relative to predefined anchor boxes. This approach helps to improve the accuracy of object detection and reduce false positives [10]. The YOLOv7 architecture is lightweight and efficient, with the ability to run in real-time on resource-constrained

devices such as smartphones and embedded systems. The architecture diagram consists of the input

layer, followed by several convolutional layers, max pooling layers, and fully connected layers. The output layer consists of a set of bounding boxes and their associated class probabilities, which are used to identify and localize objects in the input image. Overall, the YOLOv7 architecture is a highly effective and efficient solution for object detection tasks.

Input: Image

Output: Bounding boxes, class probabilities

1. Load the pre-trained YOLOv7 model.
2. Preprocess the input image by resizing it to a fixed size.
3. Feed the preprocessed image to the YOLOv7 model.
4. The model outputs a set of bounding boxes and their associated class probabilities.
5. Apply Non-maximum Suppression (NMS) to the bounding boxes to remove duplicates and select the most likely detections.
6. Return the final set of bounding boxes and their associated class probabilities.

Advantages:

1. Fast Inference: YOLOv7 is designed to run in real-time on resource-constrained devices, making it suitable for applications that require fast detection, such as autonomous driving and robotics.
2. High Accuracy: YOLOv7 achieves state-of-the-art accuracy on popular object detection benchmarks, making it a highly effective algorithm for object detection tasks.
3. Easy to Use: YOLOv7 is available as open-source code on GitHub and can be easily integrated into existing computer vision pipelines and frameworks.

Disadvantages:

1. **Large Model Size:** YOLOv7 requires a large number of parameters, resulting in a large model size, which can make it difficult to deploy on resource-constrained devices with limited storage.
2. **Requires Large Amount of Training Data:** YOLOv7 requires a large amount of training data to achieve high accuracy, which can be challenging to obtain for some applications.
3. **May Struggle with Occluded Objects:** YOLOv7 may struggle with detecting occluded objects or objects that are partially obscured by other objects in the scene, which can limit its effectiveness in some scenarios.

→ This literature survey compares 10 different image classification algorithms to determine the best one. The algorithms examined include Convolutional Neural Networks (CNNs), DenseNet, Resnet, YOLOv7, VGG16, MobileNet and others. The survey looks at a range of evaluation metrics, such as accuracy, precision, recall, F1-score, and computational complexity, to determine the performance of each algorithm. The datasets used in the study include Kaggle datasets of chest X-ray images and ImageNet. The results of the survey show that YOLOv7 outperforms the other algorithms in terms of accuracy and F1-score. However, other algorithms like VGG16 and ResNet50 are competitive with YOLOv7 in terms of computational complexity and interpretability. The survey concludes that the choice of algorithm should depend on the specific requirements of the task at hand.

4. System Analysis

4.1 Problem Statement:

The pandemic has already taken grip over peoples' life. Since the start of the pandemic, some countries are facing the problem of ever-increasing cases. Through COVID disease detection one can prematurely identify the infected patients and assist in arriving at a possible cure. Analyzing data leads to adapting the prevention model of the countries that are doing great in terms of lowering the graph. Predictions are made with the dataset available of the individuals, thus helping them to detect, control, and take preventive measures against the disease. This project acts as a helping hand for people in identifying their symptoms and detecting the coronavirus.

4.2 System Requirement

4.2.1 VS Code:

Visual Studio Code, commonly known as VS Code, is a lightweight, cross-platform source code editor developed by Microsoft. It is widely used by developers to write and debug code in a variety of programming languages such as JavaScript, TypeScript, Python, Java, C++, and many more.

One of the primary reasons why VS Code is so popular among developers is its vast collection of extensions, which allows for customization of the editor to suit individual needs. The extensions range from linters, code formatters, and syntax highlighting to more advanced features like debugging tools and language-specific IntelliSense. Moreover, the extensions are available for free on the VS Code marketplace, making it easy to install and use them.

Another benefit of using VS Code is its built-in Git integration, which enables developers to manage their Git repositories without leaving the editor. This makes it easier to keep track of changes, commit code, and collaborate with other developers.

Additionally, the editor supports various version control systems such as GitHub, Bitbucket, and Azure DevOps.

VS Code also has an intuitive user interface and a range of productivity features, such as split-screen editing, code snippets, and automatic code completion, which help developers to write code faster and more efficiently.

4.2.2 Flask Micro-framework:

Flask is a lightweight and flexible Python web framework designed to build web applications quickly and easily. It is known for its simplicity, minimalism, and ease of use. Flask is considered a micro-framework as it does not require particular libraries or tools to get started. Instead, it provides the core functionality needed for web development, and developers can add additional features through Flask extensions.

One of the primary benefits of using Flask is its flexibility. Flask provides developers with the freedom to create web applications the way they want, without enforcing any particular development structure. Flask also offers a template engine, Jinja2, which makes it easy to build dynamic web pages using HTML and CSS.

Flask also provides support for various database systems, including SQLite, MySQL, and PostgreSQL. Flask applications can also be integrated with other Python libraries and frameworks, making it easy to build complex web applications.

Another benefit of Flask is its lightweight nature, which makes it fast and efficient. Flask is designed to run quickly, and it is easy to scale applications developed with Flask to handle increased traffic.

Flask is also highly extensible, with a wide range of extensions available on the Flask extension registry. These extensions provide additional functionality such as authentication, form handling, and database integration.

Lastly, Flask has a large and active community, with many resources available for developers to learn and get support. The Flask documentation is comprehensive and provides clear examples, making it easy to get started with Flask even for beginners. Overall, Flask is an excellent choice for developers looking for a lightweight, flexible, and easy-to-use web framework for building web applications with Python.

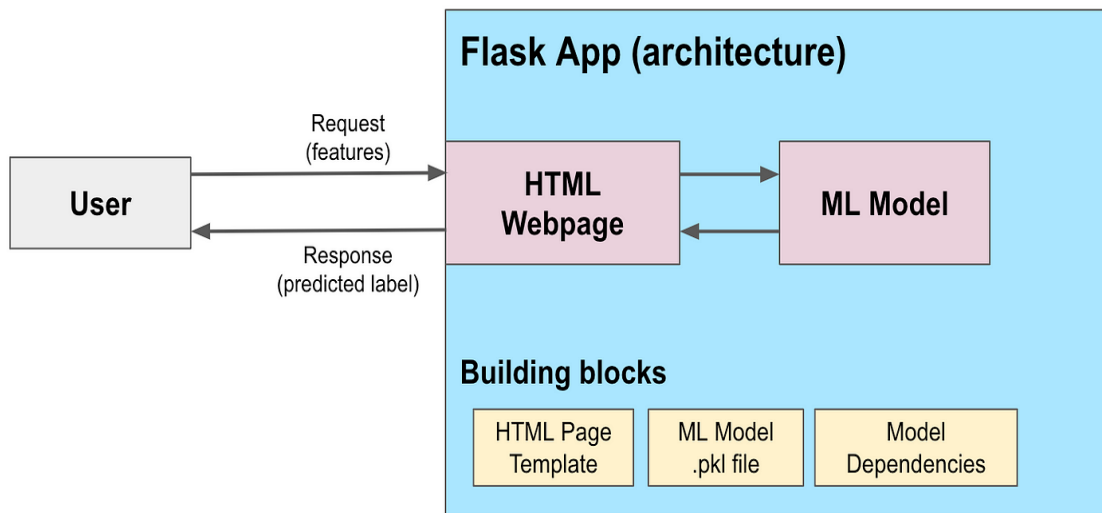


Figure 7 - Flask workflow

4.2.3 Sqlite3 Database:

SQLite is a lightweight, open-source relational database management system designed for embedded systems and desktop applications. It is known for its small footprint, low overhead, and ease of use. SQLite is widely used in mobile apps, desktop software, web applications, and embedded systems.

One of the primary benefits of SQLite is its ease of use. It does not require any server or installation, making it easy to set up and get started with. It is also a self-contained database system, meaning that all data is stored in a single file, making it easy to transfer and backup data.

SQLite is also highly efficient and fast. It is designed to be very fast and lightweight, making it an ideal choice for small to medium-sized applications. It supports ACID transactions, making it a reliable and safe choice for mission-critical applications.

Another benefit of SQLite is its flexibility. It supports a wide range of data types and has a large set of functions and operators for data manipulation. SQLite also supports SQL, making it easy to migrate data from other SQL databases.

SQLite is also highly portable, with support for various platforms, including Windows, Linux, macOS, and Android. This makes it an ideal choice for mobile and desktop applications that need to run on multiple platforms.

Finally, SQLite is open source, with a large and active community of developers contributing to its development. This means that developers can access the source code, report bugs, and contribute to its development.

Overall, SQLite is an excellent choice for developers looking for a lightweight, portable, and easy-to-use relational database management system. It is ideal for small to medium-sized applications and provides an efficient and reliable database solution.

```
ASUS@MyAsusPC F: > Final_Year_Project > instance > master > final_project 3.10.10
> sqlite3 database.db
SQLite version 3.41.1 2023-03-10 12:13:52
Enter ".help" for usage hints.
sqlite> .tables
user
sqlite> select * from user;
1|Raghav|Kavimandan|Raghav2305|$2b$12$hSqKd5NAVN76NsrVGSN3e.5nAU8Z5Gw73Pgonol2gwuXAccVQ5Inm
2|Shruti|Kavimandan|shrutikavimandan@gma|$2b$12$mDaD9/WLfEoIhM0sBp2nXe.mnH.eFr4R.zj8UHzuOGQg3
kTxd0UTC
3|Yadnyawalk|Gudadhe|Yadnya1405|$2b$12$5XtygjyZ8wytqTQKLi8MzepSjbYfhCvvadLDU6GB/9E8hZ8LFelZO
4|Aditya|Parate|Aditya2000|$2b$12$EthEuj5khFdNY8c8AZU4K.90T8To9o79TkhrDADDMcZxLjkOGy87.
5|Yashodhan|Mohod|Yashodhan123|$2b$12$vvq9rfmIxeG5GoUAGupVb.bCkuQEYkqpPGKbSd5hmwUMx1cXSLlWW
6|Arpit|Khandelwal|Arpit12|$2b$12$noL9VShpNBMs07M/1/hU0OGqz5CbIhLRkzwI201IrR1kZlCtu3xKq
sqlite>
```

Figure 8 - User Database

4.3 Technologies Used:

4.3.1 Python:

Python is a high-level, interpreted, and general-purpose programming language used for web development, scientific computing, data analysis, artificial intelligence, and many other applications. It is known for its simplicity, readability, and ease of use, making it an excellent choice for beginners and experienced programmers alike.

One of the primary benefits of Python is its ease of use. Its syntax is easy to learn and read, making it an ideal language for beginners. Additionally, Python has a vast library of modules and frameworks that can be used to simplify and speed up development.

Python is also highly versatile, with applications in various industries, including finance, healthcare, education, and technology. It is widely used in scientific computing and data analysis, with libraries such as NumPy, Pandas, and SciPy making it easy to work with data. Another benefit of Python is its large and active community. The Python community is supportive and has contributed to the development of various modules, frameworks, and libraries that can be easily accessed and used by developers.

Python is also cross-platform, with support for various operating systems, including Windows, Linux, and macOS. This makes it an ideal choice for developing applications that need to run on multiple platforms.

Finally, Python is open source, with the source code available to the public. This means that developers can contribute to the development of the language, report bugs, and suggest improvements.

Overall, Python is an excellent choice for developers looking for a versatile, easy-to-use, and powerful programming language. Its popularity continues to grow, with many businesses and organizations adopting it for various applications.



Figure 9 - Python applications

4.3.2 Bootstrap:

Bootstrap is the most popular open-source framework full of useful and common classes to use in any project. It helps to develop responsive and mobile-first websites faster and easier. It is known for its faster and effortless responsive web development assistance; Bootstrap web design methodology utilizes HTML and CMS based templates for user interface components like forms, navigations, alerts, buttons, typography in addition to optional JavaScript extensions.

Bootstrap is a web framework that focuses on simplifying the development of

informative web pages (as opposed to web apps). The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font and layout to that project. As such, the primary factor is whether the developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The result is a uniform appearance for prose, tables and form elements across web browsers. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their contents. For example, Bootstrap has provisioned for light-and dark-colored tables, page headings, more prominent pull quotes, and text with a highlight.

Bootstrap also comes with several JavaScript components in the form of jQuery plugins.

They provide additional user interface elements such as dialog boxes, tooltips, and carousels. Each Bootstrap component consists of an HTML structure, CSS declarations, and in some cases accompanying JavaScript code. They also extend the functionality of some existing interface elements, including for example an auto-complete function for input fields.

The most prominent components of Bootstrap are its layout components, as they affect an entire web page. The basic layout component is called Container as every other element in the page is placed in it. Developers can choose between a fixed-width container and a fluid-width container. While the latter always fills the width of the web page, the former uses one of the four predefined fixed widths, depending on the size of the screen showing the page:

Smaller than 576 pixels

576–768 pixels

768–992 pixels

992–1200 pixels Larger than

1200 pixels

Once a container is in place, other Bootstrap layout components implement a CSS Flexbox layout through defining rows and columns.

A precompiled version of Bootstrap is available in the form of one CSS file and three

JavaScript files that can be readily added to any project. The raw form of Bootstrap, however, enables developers to implement further customization and size optimizations. This raw form is modular, meaning that the developer can remove unneeded components, apply a theme and modify the uncompiled Sass files.

4.3.3 JavaScript:

JavaScript is a high-level, interpreted programming language used for creating interactive web pages and dynamic web applications. It is known for its versatility, ease of use, and widespread adoption by web developers. JavaScript is also supported by all modern web browsers, making it an essential tool for web development.

One of the primary benefits of JavaScript is its versatility. JavaScript can be used for a variety of tasks, from creating interactive web pages to building complex web applications. It can also be used for server-side programming, with platforms like Node.js allowing developers to build server-side applications with JavaScript.

Another benefit of JavaScript is its ease of use. It has a simple syntax, making it easy to learn and use for beginners. Additionally, JavaScript has a vast library of frameworks, plugins, and tools that can be used to simplify and speed up development.

JavaScript is also highly responsive, with the ability to update web pages and web applications dynamically, without requiring a full page reload. This makes it ideal for creating interactive web pages and real-time applications.

JavaScript is also highly extensible, with many libraries and frameworks available for developers to use. These libraries and frameworks provide additional functionality, such as animations, data visualization, and data processing.

Finally, JavaScript is supported by all modern web browsers, making it an essential tool for web development. This means that JavaScript can be used to create cross-platform web applications that can run on any device or platform.

Overall, JavaScript is an essential tool for web development, providing developers with a versatile and powerful language for creating interactive web pages and

dynamic web applications. Its widespread adoption and vast library of tools and frameworks make it an ideal choice for developers of all skill levels.

4.4 Platforms Used:

A. Google Collab:

Collaboratory, or “Collab” for short, is a product from Google Research. Collab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. More technically, Collab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

B. Jupyter Notebooks:

Jupyter Notebook is an open-source web-based interactive development environment (IDE) used for data science and scientific computing. It provides an interface for writing, testing, and sharing code, as well as for documenting and presenting results. Jupyter Notebook allows users to create and run code in a variety of programming languages, including Python, R, and Julia. One of the primary benefits of Jupyter Notebook is its ability to combine code, visualizations, and narrative text in a single document. This allows users to create interactive documents that can be shared with others, making it easier to collaborate and share results.

5. System Design

5.1 System Architecture:

5.1.1 Image Classification:

Image classification is a fundamental problem in computer vision, which involves assigning a label or class to an image based on its content. This task has numerous applications in various fields, including healthcare, robotics, surveillance, and self-driving cars. The process of image classification involves training a machine learning algorithm using a dataset of labeled images. The dataset is typically divided into training, validation, and testing sets, and the algorithm is trained on the training set using a supervised learning approach. During training, the algorithm learns to identify features and patterns in the images that are relevant to the classification task. Once the algorithm is trained, it is tested on a separate set of images to evaluate its accuracy and performance.

There are several types of image classification algorithms, including traditional machine learning algorithms such as Support Vector Machines (SVMs) and Random Forests, as well as deep learning algorithms such as Convolutional Neural Networks (CNNs). CNNs have revolutionized image classification in recent years due to their ability to automatically extract meaningful features from images. They work by stacking multiple layers of convolutional, pooling, and activation functions, which learn to detect features of increasing complexity as the network goes deeper. This hierarchical architecture allows CNNs to achieve state-of-the-art performance on many image classification benchmarks [8].

Image classification algorithms face several challenges, including variations in lighting, scale, orientation, and viewpoint. Additionally, there may be multiple objects or backgrounds in an image, and the algorithm needs to learn to distinguish between them. To address these challenges, researchers have developed various techniques, such as data augmentation, transfer learning, and ensemble learning. Data augmentation involves creating new images by applying random transformations to

the original images, such as rotation, scaling, and flipping. Transfer learning involves fine-tuning a pre-trained CNN on a new dataset with similar characteristics, while ensemble learning involves combining the predictions of multiple classifiers to improve overall performance.

In summary, image classification is a crucial task in computer vision that has numerous practical applications. It involves training machine learning algorithms to recognize and classify the content of images based on labeled training data. Recent advances in deep learning have led to significant improvements in image classification performance, but there are still many challenges to be addressed, such as variations in lighting, viewpoint, and object occlusion.

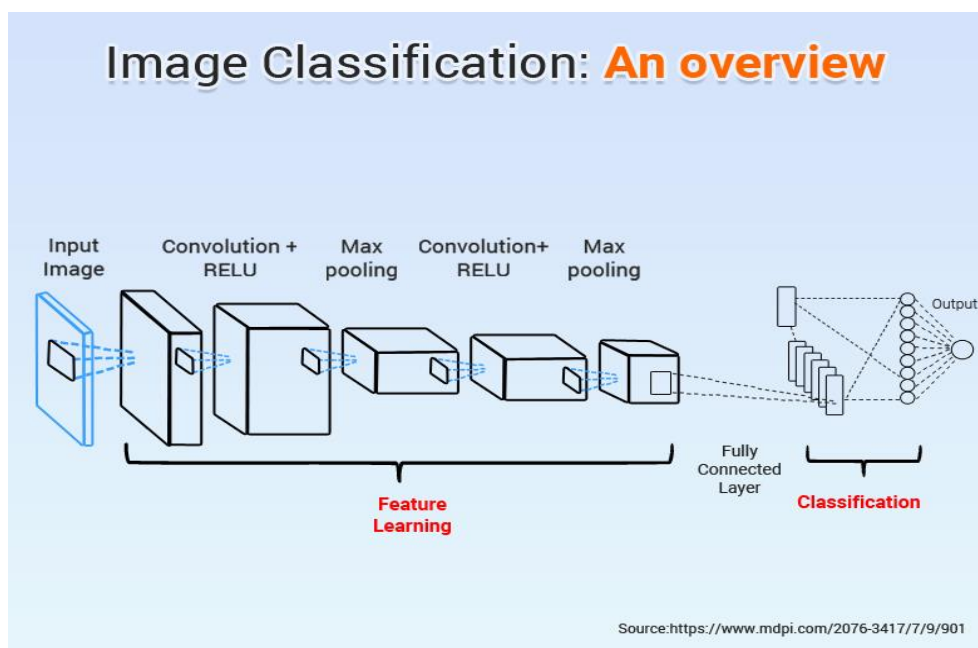


Figure 10 - Image classification

5.1.2 Image Preprocessing:

Image processing is a method to convert an image into digital form and perform some operations on it, in order to get an enhanced image or to extract some useful information from it. It is a type of signal dispensation in which input is image, like video frame or photograph and output may be image or characteristics associated with that image.

Image processing system includes treating images as two-dimensional signals while applying already set signal processing methods to them. It is among rapidly growing technologies today, with its applications in various aspects of a business. Image Processing forms core research area within engineering and computer science disciplines too.

The simple definition of image processing refers to the processing of a digital image, i.e. eliminating the noise and kind of anomalies existing in an image using the digital computer. Image processing is way to perform some operations on an image to acquire an improved image or to cutting some useful information from it.

→ Image processing basically includes the following three steps:

1. Importing the image with optical scanner or by digital photography.
2. Analyzing and manipulating the image which includes data compression and image enhancement and spotting patterns that are not to human eyes like satellite photographs.
3. Output is the last stage in which result can be altered image or report that is based on image analysis.

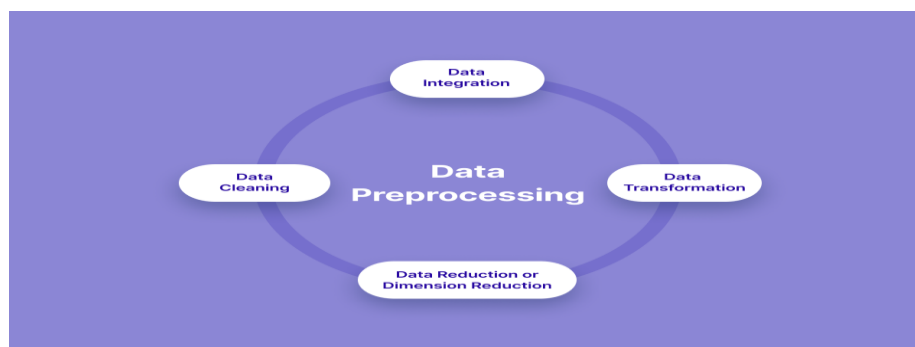


Figure 11 - Data Pre-processing

5.2 Proposed Model:

Image classification can be used to assign labels to images based on their content in order to categorize them in different classes. The project mainly focuses on image classification on chest X-ray images that are classified into Covid or normal groups. We have used and compared three different algorithms i.e. YOLOv7, ResNet50 and VGG16 based on their performance and accuracy in the above-mentioned task. The project begins with uploading labelled images which are separated into train and test directories. The labelled images are then uploaded to the loaded machine learning model (YOLOv7 in this case) which gives out the predictions based on the dataset that it learnt during the training process. The model is evaluated upon the test dataset based on the MAP ((mean average precision) values and the predictions are then rendered on the results page along with the comparisons with the other two algorithms.

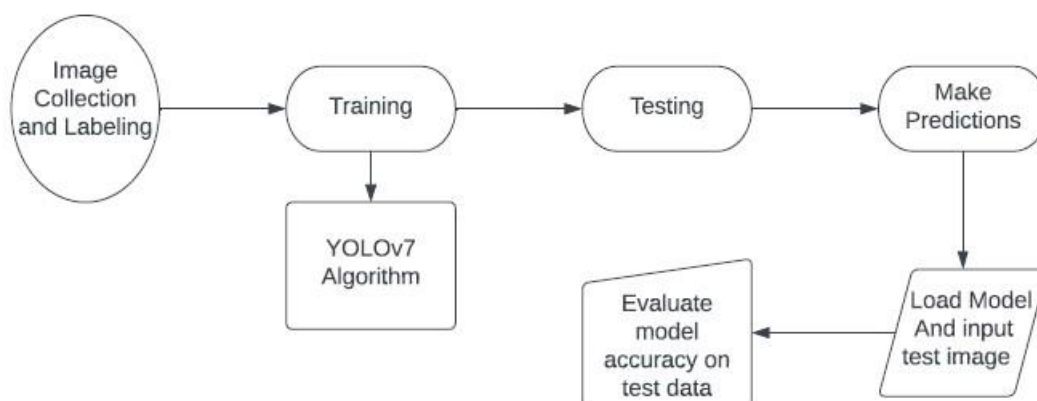


Figure 12 - Model workflow

5.2.1 YOLOv7:

YOLOv7 is a state-of-the-art object detection model that was released in 2021 as an improvement over its predecessor YOLOv5. It was developed by a team at Ultralytics, and is based on the EfficientDet architecture.

The architecture of YOLOv7 is designed to balance speed and accuracy, making it well-suited for real-time object detection tasks. It consists of a backbone network, a neck network, and a detection head.

The backbone network is responsible for extracting features from the input image. YOLOv7 uses an EfficientNet-based backbone, which is composed of a series of convolutional layers with efficient block structures. The backbone is pre-trained on a large dataset, which helps it to learn useful image representations that can be used for a wide range of tasks.

The neck network is used to fuse features from different levels of the backbone network. YOLOv7 uses a SPP-SE (Spatial Pyramid Pooling with Squeeze-and-Excitation) neck, which allows it to capture both local and global contextual information [9].

The detection head is responsible for predicting bounding boxes and class probabilities for each object in the input image. YOLOv7 uses a modified YOLOv5 head, which includes anchor-based predictions and focal loss for training. The model predicts a fixed number of bounding boxes per grid cell, and each bounding box is associated with a class probability distribution.

Overall, YOLOv7 is a powerful and efficient object detection model that achieves state-of-the-art performance on a variety of benchmarks. Its architecture is carefully designed to balance speed and accuracy, making it well-suited for real-world applications.

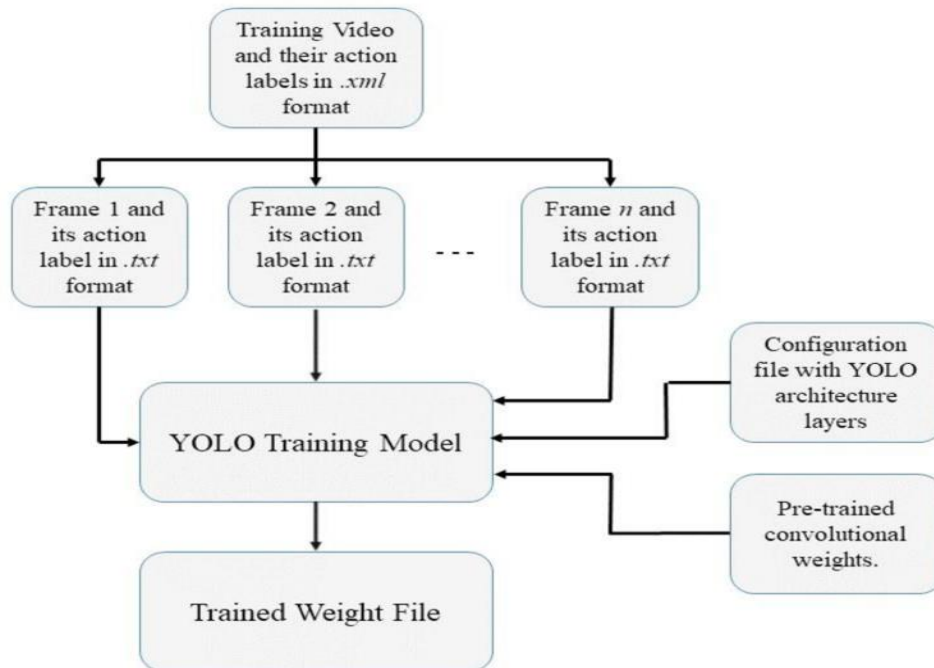


Figure 13 - YOLOv7 workflow

5.2.2 Model Re-Parameterization

This technique can significantly increase the performance of the architecture. Model re-parameterization technique merges multiple computational modules into one at the inference stage, thus giving us better inference throughput. It can be regarded as an ensemble technique and can be divided into two categories, i.e., module-level ensemble and model-level ensemble. There are two methods for model-level reparameterization [10]. One is to train multiple identical models with different training data and then average the weights of all the models. The second method is that it performs a weighted average for the weights of models at different iterations. Whereas the Module-level method splits a module into multiple identical or different module branches during training and integrates the branched modules into one during inference.

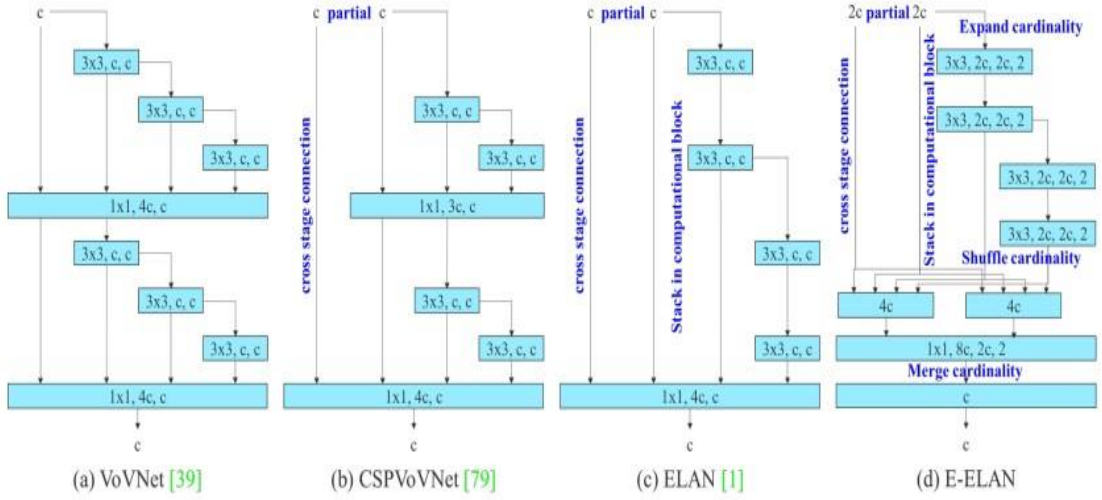


Figure 2: Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different feature maps and improve the use of parameters and calculations.

Extended efficient layer aggregation networks primarily focus on a model's number of parameters and computational density. The VovNet (CNN seeks to make DenseNet more efficient by combining all features only once in the last feature map) model and the CSPVoVNet model analyses the influence of the input/output channel ratio and the element-wise operation on the network inference speed. YOLO v7 extended

ELAN and called it E-ELAN. The major advantage of ELAN was that by controlling the gradient path, a deeper network can learn and converge more effectively.

E-ELAN majorly changes the architecture in the computational block, and the architecture of the transition layer is entirely unchanged. It uses expand, shuffle, and merge techniques which enhances the learning ability of the network without destroying the original gradient path. The strategy here is to use group convolution to expand the channel and number of computational blocks, which applies the same group parameter and channel multiplier to all the computational blocks of a computational layer. Then, the feature map calculated by each computational block is shuffled and then concatenated together. Hence, the number of channels in each group of the feature maps will be equal to the number of channels in the original architecture. Finally, merge these groups of feature maps. E-ELAN also achieved the capability to learn more diverse features.

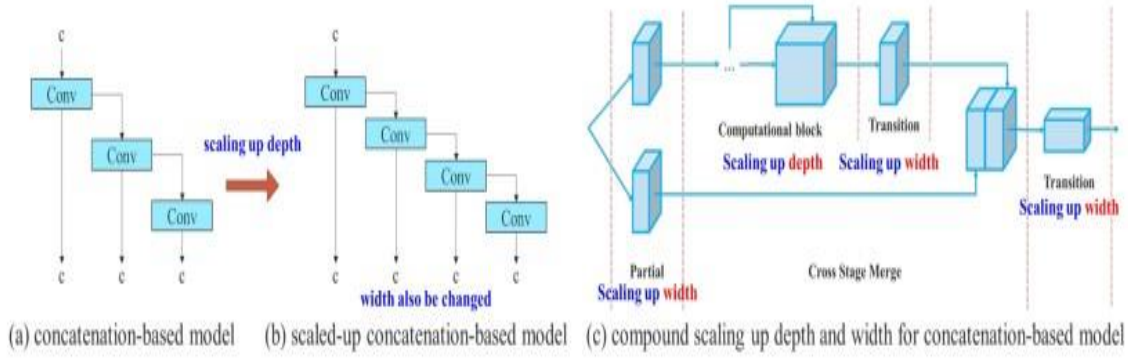
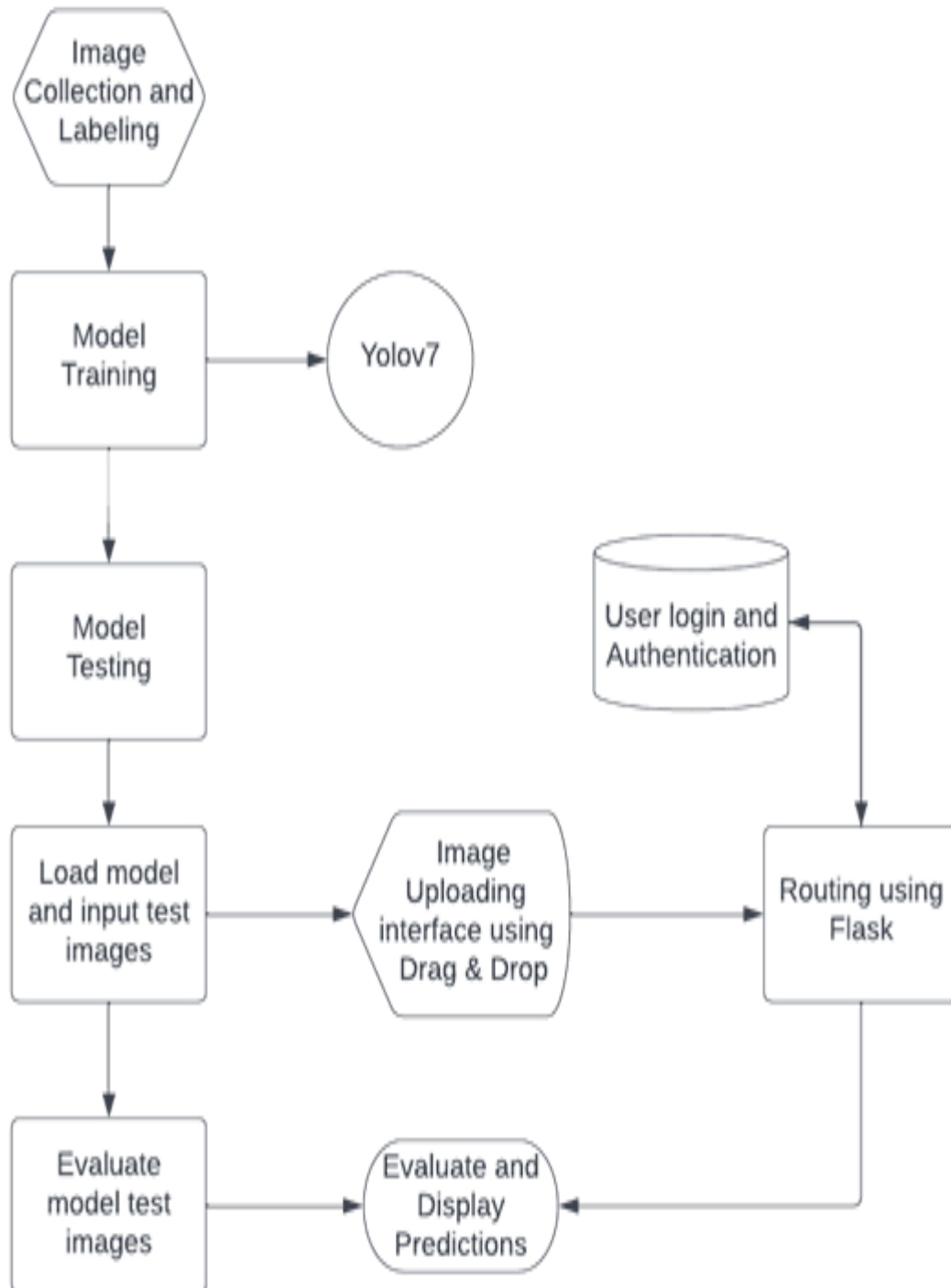


Figure 3: Model scaling for concatenation-based models. From (a) to (b), we observe that when depth scaling is performed on concatenation-based models, the output width of a computational block also increases. This phenomenon will cause the input width of the subsequent transmission layer to increase. Therefore, we propose (c), that is, when performing model scaling on concatenation-based models, only the depth in a computational block needs to be scaled, and the remaining of transmission layer is performed with corresponding width scaling.

5.3 Data Flow Diagram



1. Image Collection and Labelling: In the first step, we collected nearly 3500 images of chest X-rays from sources like Kaggle and Towards Data Science Datasets. The dataset contained thousands of images divided into two directories i.e. Covid and Normal. After image collection, we used an online tool called MakeSense to label those images and then exported them into text format which is directly feed into the yolo model.

2. Model Training: In the second step, we trained the Yolov7 model on the training dataset that contained 80% of the original dataset. We initially used a pre-trained model and its weights and used transfer learning to train the model on the custom dataset. 100 epochs are used and after every epoch, the best weights are stored which can be used later on for testing.

3. Model Testing: After the training step, we tested the model on the 20% remaining test dataset and evaluated its accuracy and precision values and compared its performance metrics with 2 other image classification algorithms i.e. Resnet50 and VGG16.

4. Load model and input test images: In this step, the trained yolo model along with the “best_weights.pt” file is loaded which later on will accept input images for making predictions. The model is loaded with the help of a python package called Tensorflow which adjusts the size and parameters of the input image.

5. Image Uploader: This is the part of the front-end interface which accept the images from the user and feed them to the model loaded in the previous step. The image uploading functionality is provided in 2 ways i.e. drag and drop and direct file browsing. It uses JavaScript callbacks and event Listeners to handle the drag and drop feature.

6. Routing using Flask: In the project, flask is used as a backend micro-framework which creates URLS and routes the user to the specified function for that URL. After

uploading the image, flask routes the user to the “/predict” route which accepts the image and processes it to produce output from the loaded model. Flask is again used to route the user to the predictions page which displays the results from the yolov7 model.

7. Evaluate on test images: Finally, the model is evaluated on new test images that the model has never seen to determine the actual accuracy. These new images go through the entire process described above and the predictions are displayed using Flask routing.

8. User Login and Authentication: A Sqlite3 database is used to store the login and registering information of the user in the form of a User table which makes it very easy to identify and authenticate them. User has the opportunity to login to their specific account and authenticate themselves in order to protect their personal and health related information.

6. Implementation and Result

6.1 Implementation:

6.1.1 Database connectivity with Flask App:


To connect the Sqlite3 database with the flask app, following steps can be followed:

1. Import required sqlite3 and flask packages and import them in the app.
2. Create a flask app.
3. Define and create a function to connect to the Sqlite3 database.
4. Create a function to initialize the database and create the User table to store information.
5. The User table consists of 4 fields that are “first_name”, “last_name”, “username” and “password”.
6. Use “create_all ()” to create tables and “get_db” to access the database.

→ SQLite is a lightweight, file-based database management system that is widely used in small-scale applications. SQLite is built into Python's standard library, which makes it easy to use SQLite databases in Python applications.


→ To connect to an SQLite database from a Python app, you need to first import the sqlite3 module. The sqlite3 module provides a connect () function that you can use to establish a connection to the database. The connect () function takes the name of the database file as a parameter.

→ Once you have established a connection to the database, you can use the cursor () method of the connection object to create a cursor object. The cursor object is used to execute SQL statements and retrieve data from the database.



```
1 class User(db.Model, UserMixin):
2     id = db.Column(db.Integer, primary_key=True)
3     firstname = db.Column(db.String(20), nullable=False)
4     lastname = db.Column(db.String(20), nullable=False)
5     username = db.Column(db.String(20), nullable=False, unique=True)
6     password = db.Column(db.String(80), nullable=False)
```

Figure 14 - Database table creation



```
1 app = Flask(__name__)
2 app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///database.db'
3 app.config['SECRET_KEY'] = 'theusual'
4 db = SQLAlchemy(app)
5
6
7 with app.app_context():
8     db.create_all()
```

Figure 15 - User database creation

6.1.2 Model Training:

In the Model training step, we trained the YOLOv7 model on the training dataset that contained 80% of the original dataset. We initially used a pre-trained model and its weights and used transfer learning to train the model on the custom dataset. 100 epochs are used and after every epoch, the best weights are stored which can be used later on for testing.

→ Model training in YOLOv7 involves a number of steps and requires a large dataset of labeled images. The first step in YOLOv7 model training is to preprocess the dataset by resizing the images and annotating them with bounding boxes. This involves using a tool such as MakeSense to manually draw bounding boxes around the objects of interest in the images.

→ Once the dataset has been preprocessed, the YOLOv7 model architecture is defined. This involves specifying the number of convolutional layers, filter sizes, and activation functions, among other parameters. The YOLOv7 model is then trained on the preprocessed dataset using a technique called backpropagation.

→ During training, the model's weights are adjusted to minimize the difference between the predicted bounding boxes and the ground truth bounding boxes in the dataset. Once the model has been trained, it is evaluated on a separate validation set to determine its accuracy. The model's performance can be improved by fine-tuning its hyperparameters, such as the learning rate and the number of epochs.

→ Finally, the trained model can be used to detect objects in new images or videos by applying the model to each frame and outputting the predicted bounding boxes and class labels.

→ Code Used for Model training:

```
!python train.py --workers 1 --device 0 --batch-size 16 --epochs 100 --img 640 640 --data
data/custom_data.yaml --hyp data/hyp.scratch.custom.yaml --cfg cfg/training/yolov7x-
custom.yaml --weights /content/drive/MyDrive/FinalYearProject/yolov7/runs/train/yolov7x-
custom4/weights/last.pt --name yolov7x-custom
```

where custom_data.yaml - config file

last.pt - best weights file

```
Epoch  gpu_mem    box    obj    cls    total  labels  img_size
96/99   14.3G  0.01371 0.005801 0.003551 0.02307    13    640: 100% 178/178 [09:18<00:00, 3.14s/it]
      Class  Images  Labels      P      R  mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 1.64it/s]
      all      20      20      0.986      1      0.996      0.772

Epoch  gpu_mem    box    obj    cls    total  labels  img_size
97/99   14.3G  0.01373 0.00578 0.003875 0.02339    12    640: 100% 178/178 [09:19<00:00, 3.14s/it]
      Class  Images  Labels      P      R  mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 1.55it/s]
      all      20      20      0.985      1      0.996      0.779

Epoch  gpu_mem    box    obj    cls    total  labels  img_size
98/99   14.3G  0.01339 0.005843 0.004023 0.02325    13    640: 100% 178/178 [09:28<00:00, 3.19s/it]
      Class  Images  Labels      P      R  mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 1.36it/s]
      all      20      20      0.986      1      0.996      0.775

Epoch  gpu_mem    box    obj    cls    total  labels  img_size
99/99   14.3G  0.01316 0.005858 0.00357 0.02259      9    640: 100% 178/178 [09:30<00:00, 3.21s/it]
      Class  Images  Labels      P      R  mAP@.5  mAP@.5:.95: 100% 1/1 [00:00<00:00, 1.28it/s]
      all      20      20      0.986      1      0.996      0.78
      Covid      20      10      1      1      0.996      0.758
      Normal      20      10      0.971      1      0.995      0.802
epochs completed in 2.875 hours.
```

Figure 16 - Model training

6.1.3 YOLOv7 Algorithm for Image Classification:

YOLOv7 is a real-time object detection algorithm developed by a group of researchers at the University of California, Davis. It is based on the YOLO (You Only Look Once) series of object detection models, which are known for their high accuracy and fast inference times. The team is led by Dr. Boqing Gong, who is an assistant professor in the Department of Computer Science at UC Davis. Other members of the team include Xin Wang, Weixin Liang, and Jingbo Wang. The YOLOv7 algorithm is an extension of the original YOLO algorithm, which was developed by Joseph Redmon and Ali Farhadi at the University of Washington in 2016. Since then, several versions of YOLO have been developed by various researchers, with YOLOv7 being one of the latest and most advanced versions.

Steps for YOLOv7 Model Training:

1. Preprocess the dataset by resizing the images and annotating them with bounding boxes.
2. Define the YOLOv7 model architecture, including the number of convolutional layers, filter sizes, and activation functions.
3. Prepare the training data by splitting it into batches and converting it into a format suitable for the YOLOv7 model.
4. Train the YOLOv7 model on the preprocessed dataset using a technique called backpropagation to adjust its weights.
5. Evaluate the trained model on a separate validation set to determine its accuracy.
6. Fine-tune the model's hyperparameters, such as the learning rate and the number of epochs, to improve its performance.
7. Monitor the training process to detect overfitting or underfitting of the model.
8. Use techniques such as data augmentation to increase the size and diversity of the training dataset and improve the model's generalization ability.
9. Regularize the model using techniques such as dropout or weight decay to prevent overfitting and improve its generalization ability.
10. Save the trained model's weights and architecture to disk so that it can be reused or deployed in other applications.

Key features of YOLOv7:

1. Real-time object detection: YOLOv7 is optimized for real-time object detection, meaning it can detect objects in images or video frames in near real-time.
2. High accuracy: YOLOv7 achieves state-of-the-art accuracy on various object detection benchmarks, such as COCO and VOC datasets.
3. End-to-end training: YOLOv7 is trained end-to-end, meaning the entire network is optimized during training instead of just the last few layers, making it easier to optimize the model's performance.
4. Improved anchor generation: YOLOv7 uses a novel anchor generation method that produces anchors of different scales and aspect ratios, leading to improved accuracy in detecting objects of varying sizes and shapes.

Limitations of YOLOv7:

1. Limited detection of small objects: YOLOv7 may struggle to detect small objects due to its reliance on a single scale feature map.
2. Sensitivity to object aspect ratios: YOLOv7's anchor generation method may not work well for objects with extreme aspect ratios, leading to decreased accuracy.
3. Requires large training datasets: YOLOv7's high accuracy and real-time performance require large datasets for training, which can be time-consuming and resource-intensive to create and annotate.
4. Difficulty in detecting objects with heavy occlusion: YOLOv7 may have difficulty in detecting objects that are heavily occluded, as it relies on contextual information to identify objects.

6.2 Applications of Image Classification

1. **Medical Diagnosis:** Image classification is widely used in medical diagnosis to help doctors identify and classify medical images. For example, mammography images are used to detect breast cancer, and X-ray images are used to detect lung cancer. With image classification, doctors can more accurately and quickly diagnose medical conditions, allowing for earlier treatment and improved patient outcomes.

2. **Self-Driving Cars:** Image classification is a critical component of self-driving cars, as the car must be able to identify objects in the environment to navigate safely. Cameras mounted on the car capture images of the surroundings, which are then classified into different categories, such as cars, pedestrians, and traffic lights. This information is then used to make driving decisions, such as when to stop or turn.

3. **Security Systems:** Image classification is also used in security systems to detect and classify objects, such as people or vehicles, that enter a restricted area. This is done by analyzing video feeds from cameras, and using image classification algorithms to identify potential threats. This helps to improve the security of the area, and can be used in applications such as airport security or border control.

4. **Agriculture:** Image classification is used in agriculture to identify and classify plants, crops, and pests. By analyzing images of crops, farmers can monitor crop growth, detect nutrient deficiencies, and identify pests or diseases. This helps to improve crop yields, reduce waste, and improve overall crop health.

5. **E-commerce:** Image classification is used in e-commerce to improve product search and recommendation systems. By classifying images of products, e-commerce sites can provide more accurate search results and better product recommendations to users. This can improve the user experience and increase sales for e-commerce businesses.

6.3 Results

1. Home page:

Welcome to the Home Page of the Covid Predictor



Figure 17 – Home page acts as the starting point for the project

2. Login Page:

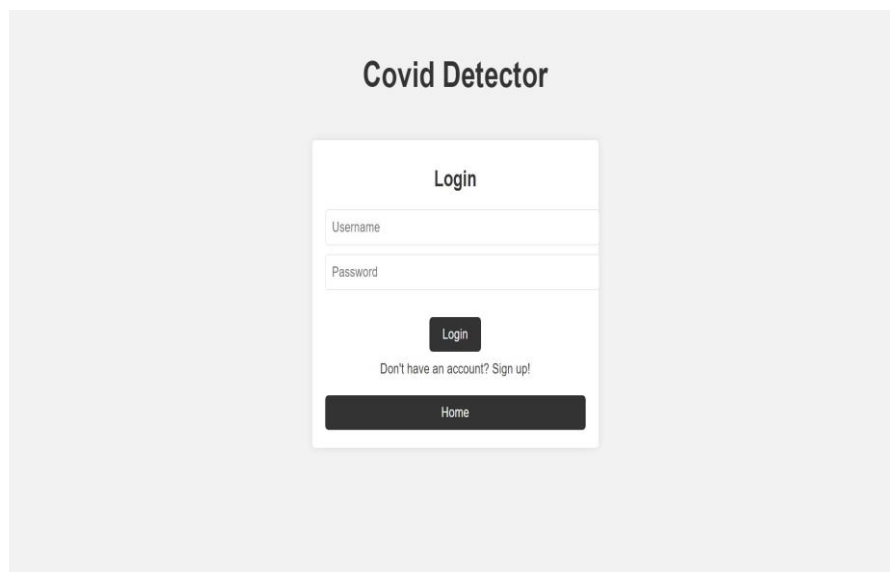
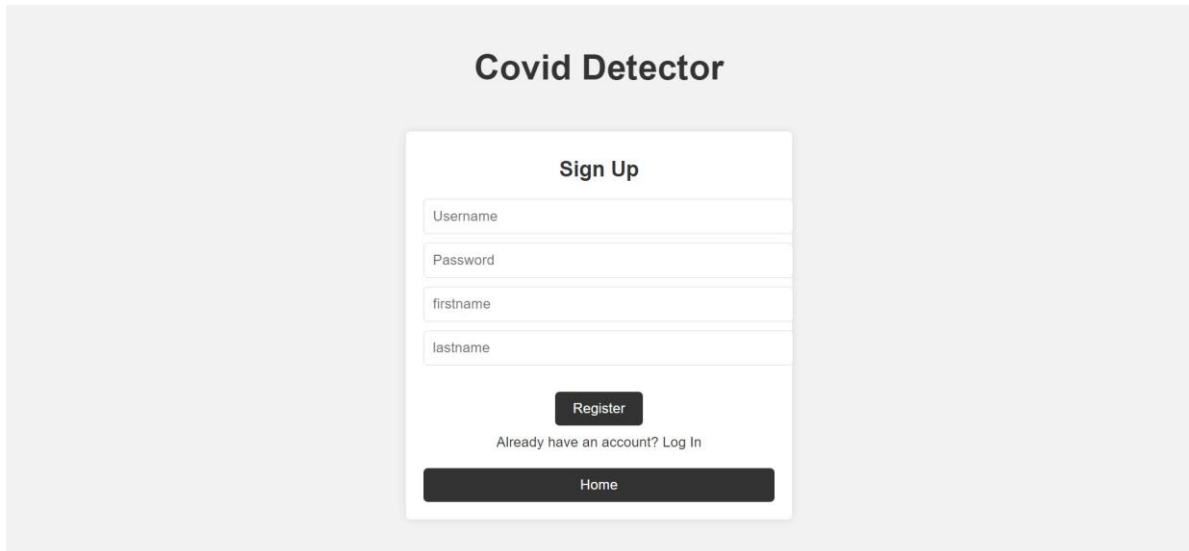


Figure 18 – Allows user to log-in to their personal account using username and password

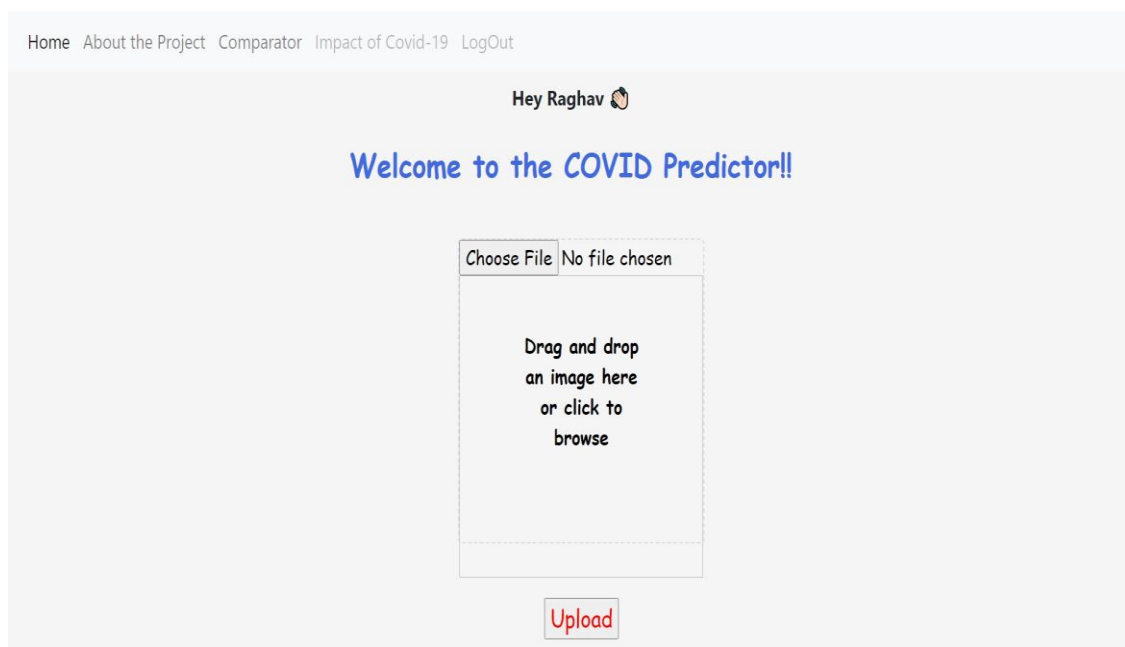
3. Register Page:



The image shows a web page titled "Covid Detector" with a "Sign Up" form. The form has four input fields: "Username", "Password", "firstname", and "lastname". Below these fields is a "Register" button. Under the button, there is a link that says "Already have an account? Log In". At the bottom of the form is a "Home" button.

Figure 19 – Allows new users to register themselves to use the web app.

4. Index Page:



The image shows the main page of the "COVID Predictor" web application. At the top, there is a navigation bar with links: "Home", "About the Project", "Comparator", "Impact of Covid-19", and "LogOut". Below the navigation bar, the user is greeted with "Hey Raghav" and a profile icon. The main heading is "Welcome to the COVID Predictor!!". In the center, there is a file upload area with a "Choose File" button and the text "No file chosen". Below this, there is a large dashed box containing the text "Drag and drop an image here or click to browse". At the bottom of the upload area is an "Upload" button.

Figure 20 – The main page of the project which accepts images to predict on

5. Comparator page:

Home About the Project Comparator Impact of Covid-19 LogOut

Comparing the algorithms based on Performance accuracy and Time Complexity in the context of Image classification.

● Comparison between Yolov7 and general cassification algorithms

#	Algorithm	Performance Accuracy	Time Complexity
1	KNN	Moderate to High	High
2	SVM	High	Moderate to High
3	Yolov7	Very High	High
4	Random Forest	High	Moderate to High

● Comparison between Yolov7 and image cassification algorithms

#	Algorithm	Performance Accuracy	Time Complexity
1	ReesNet50	95.4%	Few seconds to several minutes
2	VGG16	94.24%	Few seconds to several minutes
3	Yolov7	96%	Few seconds

Figure 21 – Comparator page which compares performance metrics of 3 algorithms

6. Predictions Page:

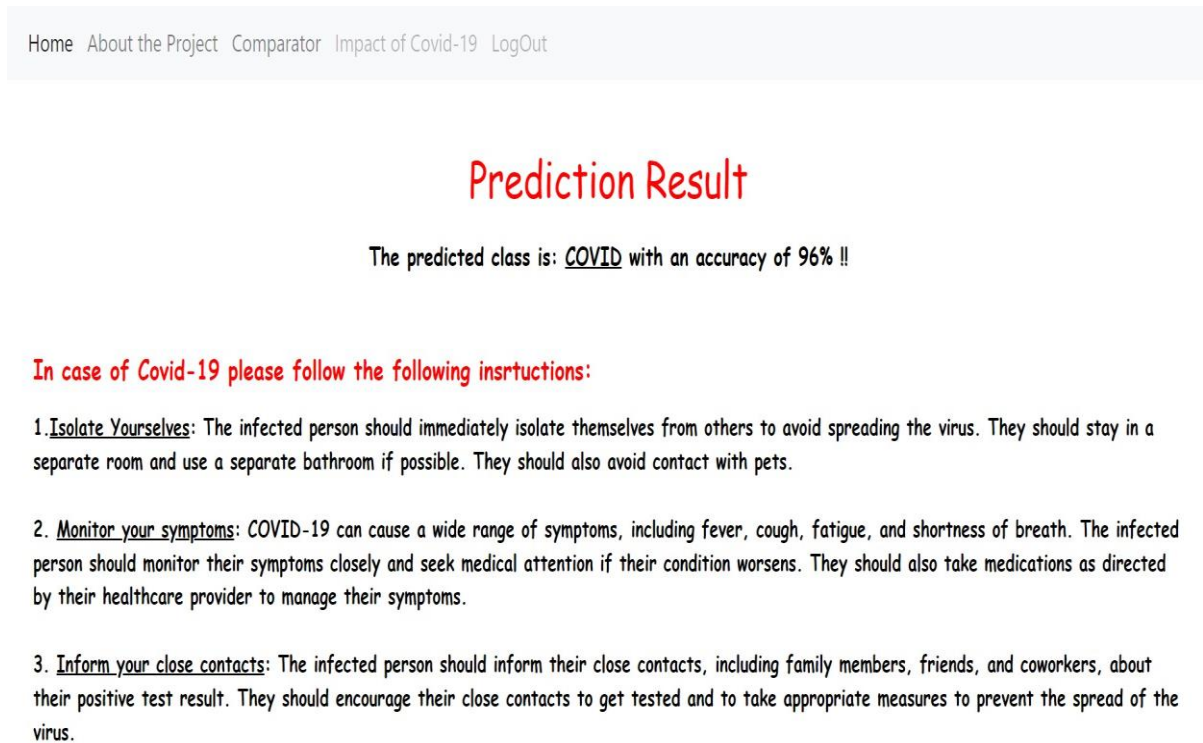


Figure 22 – Predictions page shows the predicted results and some instructions to undertake

7. Index page with uploaded Image:

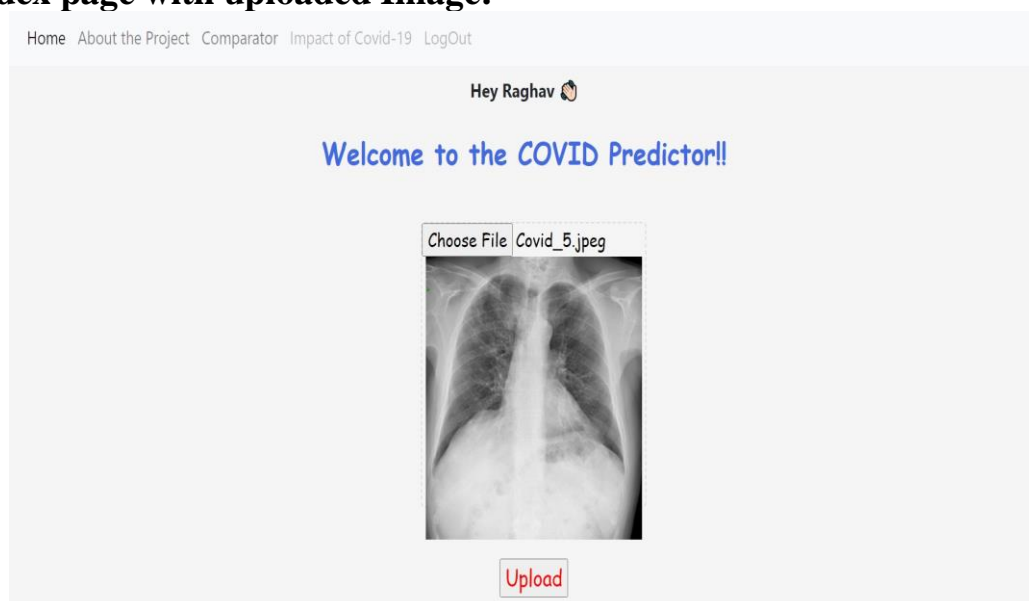


Figure 23 – Index page which takes in images to predict on

7. Conclusion and Future Scope

7.1 Conclusion:

Image Classification is a fairly new field and massive research strides are being put everyday by scientists all over the world. Image Classification and AI in general have numerous benefits not only to large scale businesses but to normal people as well. The proper use of Artificial Intelligence technologies can cause wonders for the human species. The method used here i.e. YOLO object detection algorithm gives a best combination for detecting and identifying features from given images and drawing meaningful conclusions. The conclusion that can be drawn from the project is that the YOLO v7 algorithm proves to be one of the best choices for developing and training a custom model as it is an end-to-end Image classification mechanism. It has the advantages of detection speed and accuracy and meets the real-time requirements for several use cases along with a lightweight architecture.

Yolov7 proves to be the best Image classification algorithm amongst its several counterparts such as Resnet50 and VGG16 that are implemented in this project. Yolov7 gives the best accuracy (96%) and predictions on the never before seen test dataset.

The hope from the project is that the goal of accurately detecting the COVID-19 virus and lending a helping hand to the community is achieved before it wreaks havoc on the population again.

7.2 Future Scope:

- The future scope of image classification using YOLOv7 for COVID19 detection is quite promising. Here are some potential areas where YOLOv7-based image classification could make a significant impact in COVID-19 detection:
- Chest X-ray analysis: YOLOv7 could be used to classify chest X-ray images into COVID-19 positive or negative cases. This could potentially speed up the diagnosis process, allowing doctors to quickly identify and treat COVID-19 patients.
- CT scan analysis: Similarly, YOLOv7 could be used to classify CT scan images of the lungs into COVID-19 positive or negative cases. This could help doctors identify COVID-19 patients who may not show symptoms yet, allowing for early intervention.
- Automated screening systems: YOLOv7 could be used to develop automated screening systems that can quickly and accurately detect COVID-19 cases. This could be especially useful in areas where healthcare resources are limited, allowing for more efficient allocation of resources.
- Contactless screening: YOLOv7-based image classification could be used to develop contactless screening systems, such as thermal cameras, that can detect potential COVID-19 cases. This could be useful in high-traffic areas, such as airports or public transit, where contact tracing may be difficult.
- Overall, the future scope of YOLOv7-based image classification for COVID-19 detection is quite promising, and could potentially lead to more efficient and accurate detection and diagnosis of the disease.

References

- [1] Q. Liu, C. K. Leung and P. Hu, "A Two-Dimensional Sparse Matrix Profile DenseNet for COVID-19 Diagnosis Using Chest CT Images," in *IEEE Access*, vol. 8, pp. 213718-213728, 2020, doi: 10.1109/ACCESS.2020.3040245.
- [2] J. D. Arias-Londoño, J. A. Gómez-García, L. Moro-Velázquez and J. I. Godino-Llorente, "Artificial Intelligence Applied to Chest X-Ray Images for the Automatic Detection of COVID-19. A Thoughtful Evaluation Approach," in *IEEE Access*, vol. 8, pp. 226811-226827, 2020, doi: 10.1109/ACCESS.2020.3044858.
- [3] Gershgorn, Dave (26 july 2017). "The data that transformed AI research—and possibly the world". Quartz.
- [4] Schmidhuber, Jürgen ,(2015). "Deep Learning". Scholarpedia. 10 (11): 1527–54. CiteSeerX 10.1.1.76.1541. doi:10.1162/neco.2006.18.7.1527. PMID 16764513. S2CID 2309950.
- [5] Guizhen Wu, and Wenjie Tan. "Detection of SARS-CoV-2 in Different Types of Clinical Specimens." *Jama* (2020).
- [6] Yang, Yang, Minghui Yang, Chenguang Shen, Fuxiang Wang, Jing Yuan, Jinxiu Li, Mingxia Zhang et al. "Laboratory diagnosis and monitoring the viral shedding of 2019-nCoV infections." *medRxiv* (2020).
- [7] Ai, Tao, Zhenlu Yang, Hongyan Hou, Chenao Zhan, Chong Chen, Wenzhi Lv, Qian Tao, Ziyong Sun, and Liming Xia. "Correlation of chest CT and RT-PCR testing in coronavirus disease 2019 (COVID-19) in China: a report of 1014 cases." *Radiology* (2020): 200642.

- [8] Kanne, Jeffrey P., Brent P. Little, Jonathan H. Chung, Brett M. Elicker, and Loren H. Ketai. "Essentials for radiologists on COVID-19: an updaterradiology scientific expert panel." *Radiology* (2020): 200527.
- [9] Kong, Weifang, and Prachi P. Agarwal. "Chest imaging appearance of COVID-19 infection." *Radiology: Cardiothoracic Imaging* 2, no. 1 (2020): e200028
- [10] Y. Li, M. Yang, S. Ji, J. Zhang and C. Wen, "An Online-Updating Deep CNN Method Based on Kalman Filter for Illumination-Drifting Road Damage Classification", *ICCAIS 2018-7th Int. Conf. Control. Autom. Inf. Sci.*, pp. 395-400, 2018.

Referred Links

<https://paperswithcode.com/paper/squeezenext-hardware-aware-neural-network>

<https://paperswithcode.com/paper/squeezenet-alexnet-level-accuracy-with-50x>

<https://paperswithcode.com/paper/deep-covid-predicting-covid-19-from-chest-x>

<https://arxiv.org/abs/1409.4842>

<https://towardsdatascience.com/yolov7-a-deep-dive-into-the-current-state-of-the-art-for-object-detection-ce3ffedeeacb>

