Object Detection: Face Mask Detection

Vedant Parnaik 50487478
Vrashi Shrivastava 50479413
Abhishek Paharia 50481502
Group Number 5
Github link : vedantparnaik/Pattern-Recognition-Project

Abstract

The objective of the project is to take a given dataset and run it through machine-learning models to detect an object (face mask) in the images of individuals. The project involves training and evaluating three different models on this dataset to identify the most effective approach for face mask detection. The technologies we have used are PyTorch, Python, and other libraries. We have used Git and GitHub for code management. We are running the data on 3 different models to compare the accuracy and loss in each model. The architecture for each model is defined and discussed in detail further in the report. The dataset is split into training, validation, and test sets. The models are trained on the training set, and their hyperparameters are optimized on the validation set. The effectiveness and accuracy of the models in detecting face masks are then assessed based on their performance on the test set. A comparison of the three models will be presented together with the experiment's outcomes and findings.

Dataset

The dataset includes pictures of people wearing and not wearing face masks, coupled with labels indicating whether the masks are present or not. It contains 853 images which can be three categorized into 3 categories:
1. With a mask
2. Without mask
3. With a mask worn incorrectly
Dataset link: face-mask-detection

Method Selection (CNNs over LSTM, GRU)

LSTM (Long Short-Term Memory) and GRU (Gated Recurrent Unit) are types of recurrent neural networks (RNNs) that excel in sequence modeling tasks such as natural language processing and time series analysis. However, they are less suitable for object detection tasks due to:

- Sequential Processing: These models process data sequentially, which isn't fully utilized in object detection where entire images or regions are analyzed at once.
- Spatial Information: Object detection demands understanding spatial relationships within images. LSTMs and GRUs lack the inherent capacity to capture such spatial layouts effectively.
- Computational Complexity: LSTMs and GRUs are computationally expensive, leading to slower inference times, especially with large, high-resolution images.
- Architectural Limitations: Standard LSTM and GRU architectures struggle to capture long-range dependencies in images, crucial for understanding object contexts.

Instead, convolutional neural networks (CNNs), designed for grid-like data, are preferred for object detection. Modern systems like YOLO and Faster R-CNN utilize CNNs, with their ability to process images in parallel, capture spatial information, and handle large-scale datasets efficiently, making them more suitable for object detection tasks.

1 Traditional CNN

A Convolutional Neural Network (CNN) is a type of deep learning model that excels at processing grid-like data, like images, where spatial relationships between pixels are crucial. These networks use convolutional, pooling, and fully connected layers to learn and make decisions from complex, multi-dimensional datasets.

1.1 Model

This CNN model follows a sequential structure with alternating convolutional (with an increasing number of filters) and max pooling layers for feature extraction from 35x35x3 input images. Dropout layers are used intermittently for regularization, reducing overfitting. After flattening the tensor output, fully connected layers classify the data, with a final 'softmax' layer outputting the class probabilities.

1.2 Results

The best accuracy achieved by the model was 93.66%

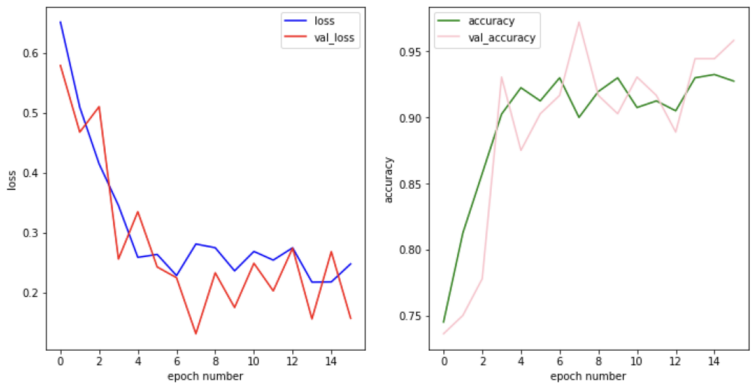

Figure 1: CNN: Learning Curve

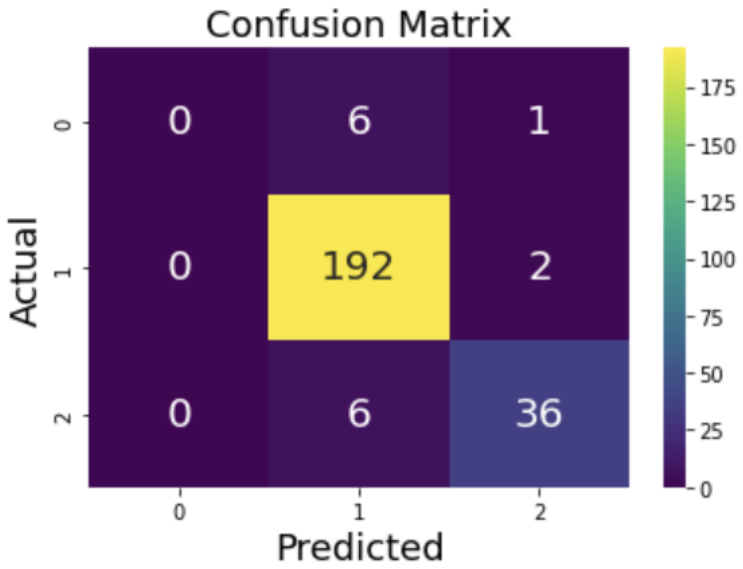

Figure 2: CNN: Confusion Matrix

1.3 Summary

It can be concluded that traditional CNN methods are very useful in object detection and classification. The pictures were accurately classified on unseen data too. They are very efficient.

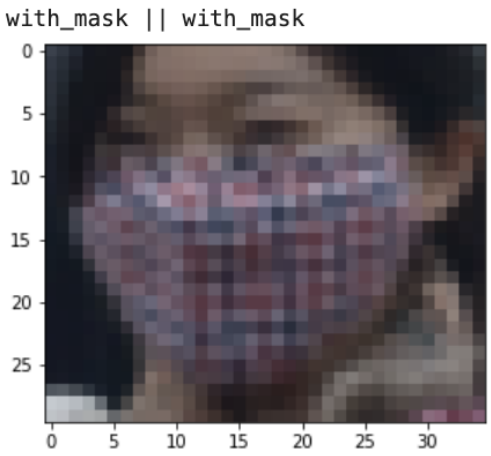

Figure 3: CNN: Model Prediction

2 Faster R-CNN

Faster R-CNN is a cutting-edge object detection architecture that identifies and localizes objects in an image, combining the strength of both region proposals and CNNs. This network improves the speed and accuracy of object detection by replacing selective search in the traditional R-CNN with a more efficient Region Proposal Network (RPN).

2.1 Model

This Faster R-CNN model employs a ResNet50 backbone integrated with a Feature Pyramid Network (FPN), pre-trained on the COCO dataset for rich feature extraction. The Region of Interest (RoI) heads house a box predictor, which, after receiving the number of classifier input features, is replaced with a new one that forecasts bounding boxes for a specified number of classes. The combined strength of the deep ResNet50, FPN, and an adaptable box predictor make this an efficient and versatile object detection architecture.

2.2 Results

The best accuracy of the model is 95.3%.

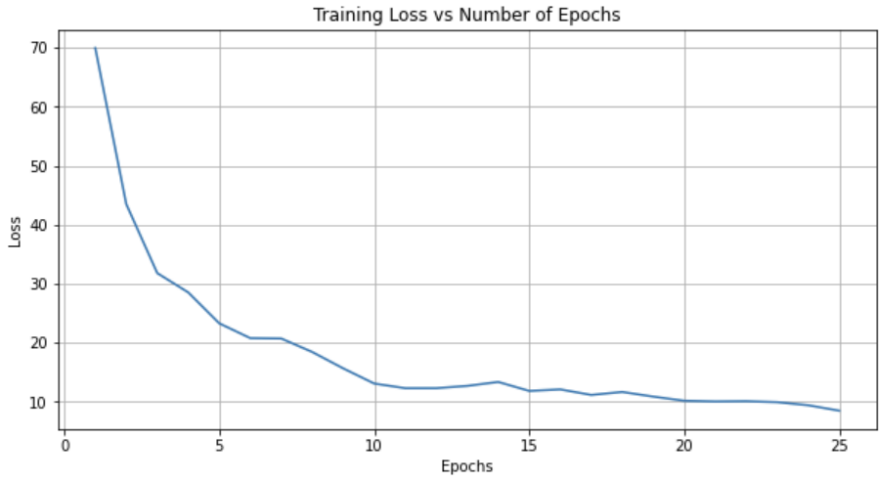

Figure 4: FasterRCNN: Learning Curve

2.3 Summary

It can be concluded that Faster R-CNN method is very useful in object detection and classification. The model provided more efficient and accurate object detection compared to CNNs which just classify or localize objects.

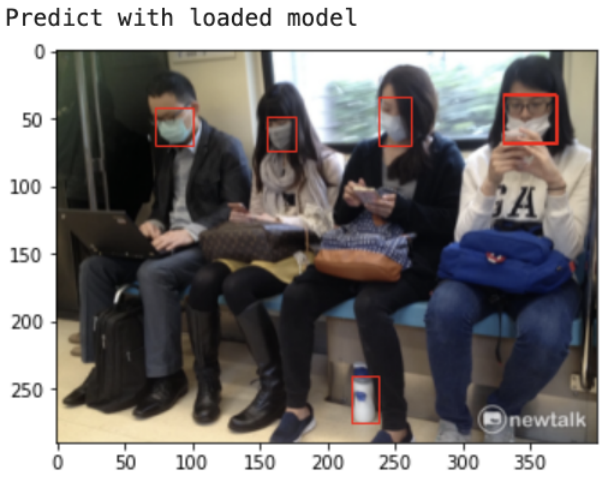

Figure 5: FasterRCNN: Model Prediction

The pictures were accurately classified on unseen data too.

3 ResNet-34

Residual network (ResNet) is a family of modern Convolutional networks. ResNet consists of residual blocks with normal convolutional layers and fully connected layers. Fig. 6 is diagram of the residual blocks. A residual block

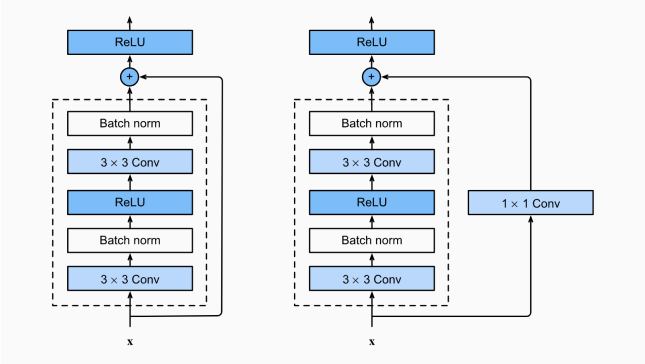

Figure 6: Residual block

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 7: ResNet Architecture details

consists of 3X3 convolutional layer followed by batch normalisation and ReLu activation function. there is again one more 3X3 convolutional layer followed by batch normalisation. the output of dashed block is added to the input X and resultant output is passed to ReLu activation layer.

The residual network alters the weights until the output is equivalent to the identity function. In the process the outcome of residual function eventually becomes 0 and X gets mapped onto the hidden layers. Therefore, the error correction is not required. In turn the identity function helps in building a deeper network. The residual function then maps the identity, weights and biases to fit the actual value.

3.1 Model

Fig. 8 shows architecture of the ResNet-34. Fig. 7 describes various layers in the model. Layers in sequential order are as follows.

1. 7X7 convolutional (conv) layer with 64 channel (ch.) with stride 2
2. 3X3 maxpool, stride 2
3. 3 residual blocks. each residual block has two 3X3, 64 ch. conv. layers.
4. 4 residual blocks. each residual block has two 3X3, 128 ch. conv. layers.
5. 6 residual blocks. each residual block has two 3X3, 256 ch. conv. layers.
6. 3 residual blocks. each residual block has two 3X3, 513 ch. conv. layers.
7. average pool layer
8. 1000-d fully connected layer

We used pre-trained ResNet-34 model and modified the model based on problem requirement. Since we have 3 output classes so we replace the last linear layer with 1000 output features with linear layer which outputs 3 features each for a output class.

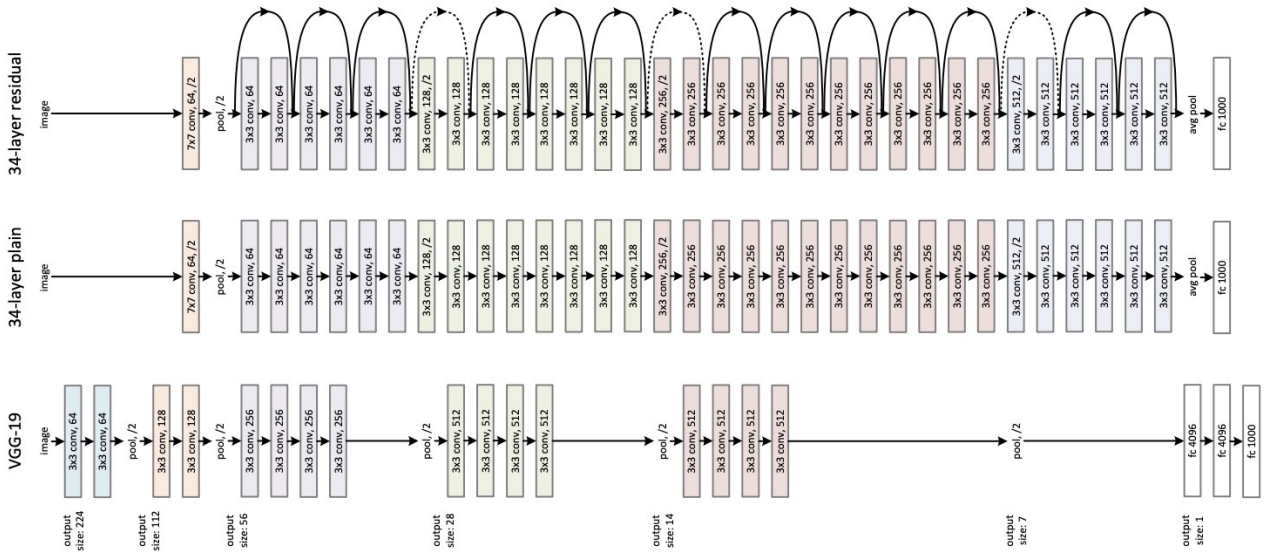

Figure 8: ResNet-34 architecture

3.2 Loss function & Optimisation algo.

We used the scholastic gradient descent (SGD) optimization algorithm along with the pre-trained ResNet-34 network. Cross entropy loss function is used in the model as it is a classification problem.

3.3 Metrics and Results

The metric to access the model is accuracy which is the ratio of correctly predicted samples with all samples. Fig. 9 shows the results of the experiment. the left chart shows training and resting accuracy versus epochs the right chart shows the loss of training and testing phases.

The best training accuracy is 0.93 and the testing accuracy is 0.92.

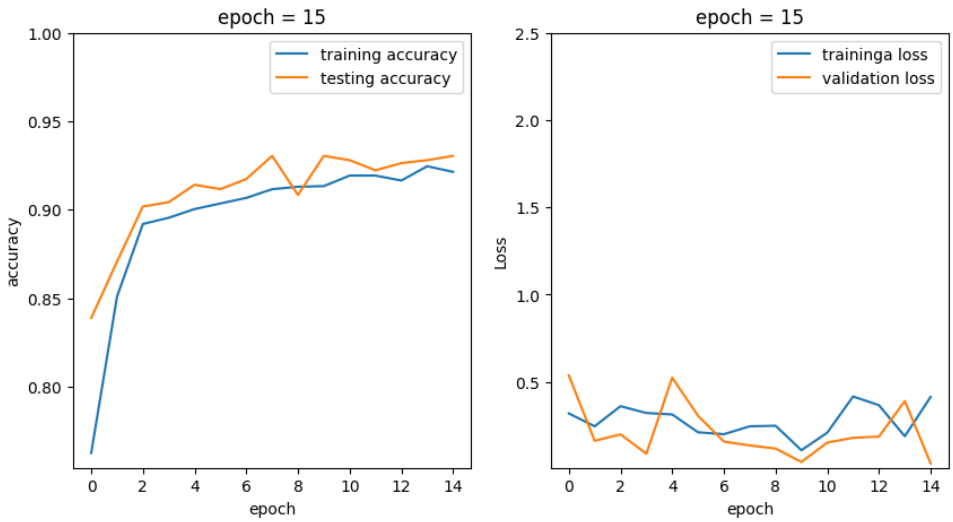

Figure 9: resNet-34 exp. results

4 YOLO

YOLOv5 (You Only Look Once version 5) is a state-of-the-art object detection system that operates in real-time by dividing an image into a grid and predicting bounding boxes and class probabilities simultaneously. It is recognized for its efficiency, being faster and requiring fewer computational resources.
We tried working on YOLO for the task. YOLO was one of the easiest methods compared to the other methods we applied.

4.1 Model

It uses the CSPDarknet53 backbone network for feature extraction, uses PANet (Path Aggregation Network) for feature fusion. Bounding box coordinates, object class probabilities, and objectness ratings are all predicted by the detecting head. YOLOv5 anticipates that boxes will manage objects of various sizes at three different scales. Anchor boxes, which are preconfigured boxes representing different item sizes and aspect ratios, are used by each detecting head to anticipate bounding box coordinates. Additionally anticipated are object class probabilities and objectness ratings, which express the likelihood of the presence of an object. The network predicts more than one bounding box for each object. Then, depending on the confidence scores of the redundant detections, non-maximum suppression is used to eliminate them.

4.2 Results

The best accuracy of the model is 95.1%.

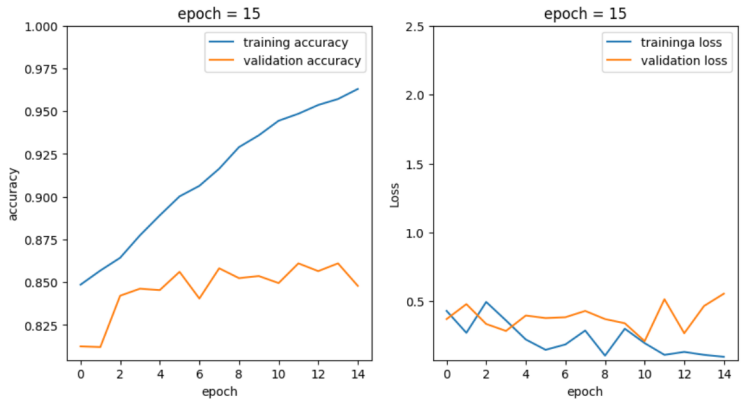

Figure 10: Yolo Results

4.3 Resource

using this GitHub Repository as the resource for this model. We are yrt to make custom changes to the model and test it. **Github link:** yolov5/models

5 Comparison

Model	Test Accuracy
Faster-RCNN	0.95
Traditional CNN	0.93
ResNet	0.92
Yolo V5	0.95

Table 1: Model Comparision

6 Conclusion

Of the three methods, the Faster-RCNN model proved the most efficient with the highest accuracy. The model is very robust. Later, ResNet showed good results, and the Traditional CNN method performed well too.

6.1 Contribution

Name	Work	Percentage
Vedant Parnaik	CNN, Faster RCNN	33 %
Vrashi Shrivastava	YOLO-v5	33 %
Abhishek Paharia	ResNet-34	33 %

Table 2: Contribution Table

7 References

1. https://www.kaggle.com/datasets/andrewmvd/face-mask-detection
2. https://d2l.ai/chapter_convolutional-modern/resnet.html#
3. https://arxiv.org/pdf/1512.03385.pdf