COMP 6721

Project Phase II

TEAM FL_07

Name	Student Id	Specialization
Amritpreet Singh	40150231	Data Specialist
Nikhil Nikhil	40151391	Training Specialist
Pushpa Gautam Ojha	40151892	Evaluation Specialist

Table of Contents

Dataset

For the dataset creation Images were collected from various open-source datasets available online and those images were manually classified into the classes for the training and testing of the model. For detailed information of the image sources please check the ***datasourceInfo.pdf*** file.

Since images were gathered from the various open sources available online because of which image sizes were inconsistent to resolve. Hence, before loading the dataset we fixed the image dimensions to 100X100. So once image size was fixed, we were able to process images and create our training and test datasets.

We have a total of 3544 images which are divided into training and test data sets. For training we have kept 2898 images across all classes and for testing total comprises 646 images across all classes. Further details are as follows:

Class	Training Dataset (Image Count)	Test Dataset (Image Count)
Class 0 (Unmasked)	1092	220
Class 1 (Masked)	1177	252
Class 2 (Not a person)	629	164
Total	2898	646

Data Preprocessing

The system pre-processes the raw images before feeding them to the CNN model. At first, we resized all the input images to 100 X 100 resolution. In addition to this, in phase II we are converting the images to grayscale to achieve a better performance of the system. Once, all the raw images are converted to the specified size and into grayscale we are storing them to a new location marking them ready to be feed to the CNN model. We are reusing existing python library OpenCV for resizing and converting the input images to grayscale.

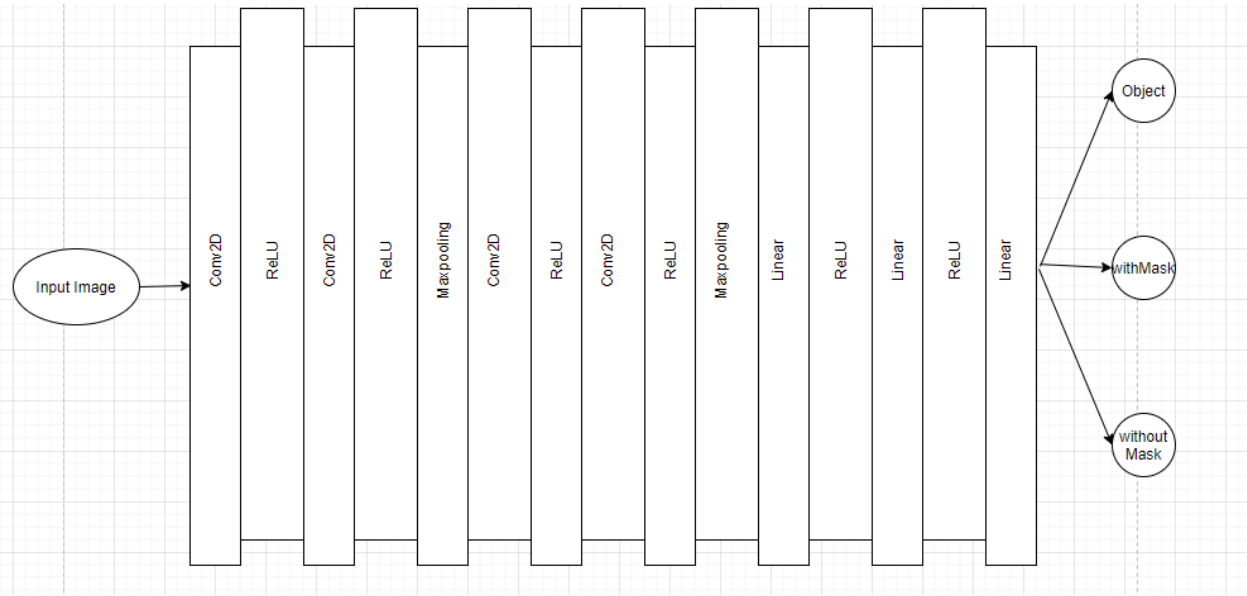
CNN Architecture

We have created CNN model which has total 9 major layers considering convolution layer, pooling layer and classification layer along with the activation function. We use ReLU as an activation function because of its faster computability and fewer vanishing gradients. Below is the brief description of each of these layer and function used in our CNN architecture.

- **Conv2d**: This is the convolution layer we convolute the input the weight matrix to create activation maps.
- **ReLU (Rectified Linear Unit):** This is an activation function which use the activation maps to give the outputs to the next layer. The activation function defined as f(x) =max (0, x).

- **Pooling**: We use max pooling to reduce the number of activation maps.
- **Linear**: This layer basically maps the input to output linearly (n inputs to m output). it can be described as Ax = b function.

The structure of the model:



Below is the detail information about the image shape transformation, when these pass into each of these layers:

Layers		Input channel	Filter matrix size	Stride	Padding	Pool size	Output channel	Output Image shape
Layer 1	Convolution Layer	3	3*3	1	1	-	100	100*100
Layer 2	Convolution Layer	100	3*3	1	1	-	128	100*100
Layer 3	Max Pooling Layer	-	-	-	-	2*2	128	50*50
Layer 4	Convolution Layer	128	3*3	1	1	-	256	50*50
Layer 5	Convolution Layer	256	3*3	1	1	-	256	50*50
Layer 6	Max Pooling Layer	-	-	-	-	2*2	256	25*25

Once these layers are trained with the data, then the image has to be converted to 1D form for mapping them into the specified class labels. Hence, the output from layer 6 is feed to the classification layer, i.e., Layer 7 to flatten the data into 1D array. In our project we have 3 classes and the output of last layer of size 3.

Layers		Input	Output
Layer 7	Linear Layer	256*25*25	512
Layer 8	Linear Layer	512	256
Layer 9	Linear Layer	256	3

We also use cross Entropy loss or log loss that mesures the difference between the actual and predicted values. This loss is then back propagated through the network to update the weights.

We use SGD (Stochastic gradient descent) to calculate gradients to update the weights with a learning rate of 0.005 and a momentum of .9.

We train the model for 30 epochs to make it more robust. At the 28th epoch, the model shows the training accuracy as 100%. Below is the image how the accuracy for the training data increases in each epoch.

```
C:\Users\Admin\anaconda3\python.exe C:/Users/Admin/PycharmProjects/AI-project-1/Code/main.py
Accuracy of the network on Epoch 0 : 57 %
Accuracy of the network on Epoch 1 : 86 %
Accuracy of the network on Epoch 2 : 89 %
Accuracy of the network on Epoch 3 : 91 %
Accuracy of the network on Epoch 4 : 92 %
Accuracy of the network on Epoch 5 : 93 %
Accuracy of the network on Epoch 6 : 94 %
Accuracy of the network on Epoch 7 : 94 %
Accuracy of the network on Epoch 8 : 95 %
Accuracy of the network on Epoch 9 : 96 %
Accuracy of the network on Epoch 10 : 96 %
Accuracy of the network on Epoch 11 : 97 %
Accuracy of the network on Epoch 12 : 98 %
Accuracy of the network on Epoch 13 : 97 %
Accuracy of the network on Epoch 14 : 98 %
Accuracy of the network on Epoch 15 : 98 %
Accuracy of the network on Epoch 16 : 99 %
Accuracy of the network on Epoch 17 : 99 %
Accuracy of the network on Epoch 18 : 98 %
Accuracy of the network on Epoch 19 : 99 %
Accuracy of the network on Epoch 20 : 97 %
Accuracy of the network on Epoch 21 : 99 %
Accuracy of the network on Epoch 22 : 99 %
Accuracy of the network on Epoch 23 : 98 %
Accuracy of the network on Epoch 24 : 99 %
Accuracy of the network on Epoch 25 : 99 %
```

```
Accuracy of the network on Epoch 26 : 99 %
Accuracy of the network on Epoch 27 : 100 %
Accuracy of the network on Epoch 28 : 100 %
Accuracy of the network on Epoch 29 : 100 %
```

Evaluation:

The model designed for this project has been evaluated using mainly 5 metrics, which are described below:

1. **Accuracy:** For computing accuracy, we are comparing the output labels, class 0,1 or 2, representing classes *Person With Mask*, *Person Without Mask* and *Not a Person* respectively, from CNN model with the actual test data labels. We then compute the sum of the test data that has been evaluated correctly by the model and divide it with the total test data size to get accuracy rate.
2. **Recall:** This is the ratio of true positive results and sum of true positive and false negatives result from CNN model. For our model it is computed by using *sklearn.metrics.recall_score*
3. **Precision**: This is the ratio of true positive results and sum of true and false positives result from CNN model. For our model it is computed by using *sklearn.metrics.precision_score*.3.4 F1 Measure: This is the ratio of twice the product of recall and precision over sum of recall and precision. For our model it is computed by using *sklearn.metrics.f1_score*.3.5
4. **Confusion Matrix**: This provides the accuracy of classification of each of the results by the CNN model. For our model it is computed by *sklearn.metrics.confusion_matri*x

For Computing above four metrics, i.e., *precision, recall, f1 measure* and *computation metrics*, we are first preparing the list of the *test data*, that have the labels represented by 0,1 or 2 (*0 for People Without Mask* class, *1* for *People With Mask* class *and 2* for *Not a person* class). Then we prepare the list of predicted results by the CNN model against these test data, which also have elements representing class by 0,1 or 2. We are then using *sklearn.metrics* library to compute these metrics for each of the classes, by sending labels as input i.e. *labels=[0,1,2].*

Accuracy for 630 tested images as predicted by model was 86%. We are planning to work on the various aspects of the model in Project phase II for improving model's performance. Below is the CNN model output for 3 classes of the test data for each evaluation metrics as specified above.

```
C:\Users\Admin\anaconda3\python.exe C:/Users/Admin/PycharmProjects/AI-project-1/Code/main.py
Accuracy of the network on  630  test images: 86 %
****Confusion Metrics****
[[267   0   8]
 [ 39 140  30]
 [  4   4 138]]
****Precision Metrics****
Person Without Mask:: 0.8612903225806452
Person With Mask:: 0.9722222222222222
Not a Person:: 0.7840909090909091
****Recall Metrics****
Person Without Mask:: 0.9709090909090909
Person With Mask:: 0.6698564593301436
Not a Person:: 0.9452054794520548
****F1 Measure Metrics****
Person Without Mask:: 0.9128205128205129
Person With Mask:: 0.793201133144476
Not a Person:: 0.8571428571428572
```

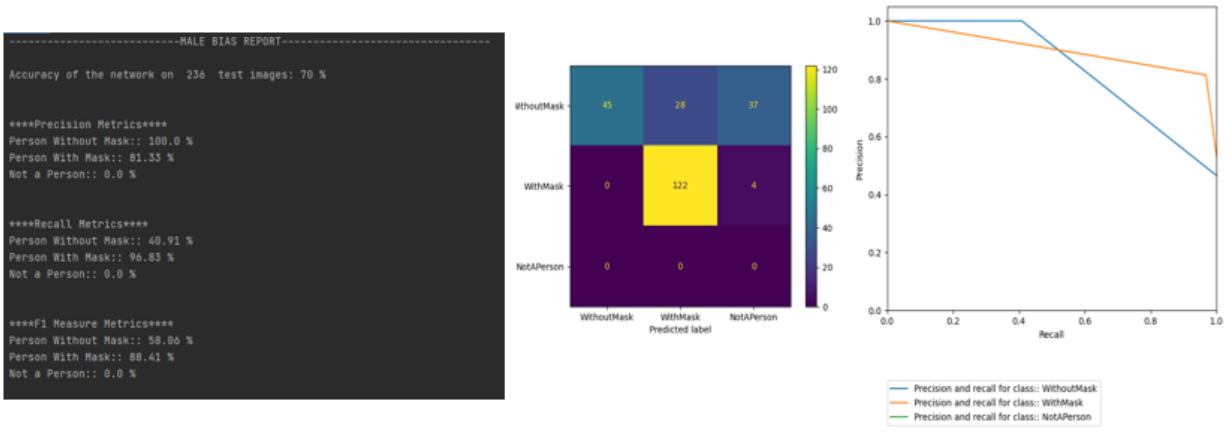
Figure 1: Accuracy, Precision, Recall, F1-Measures, Confusion Matrix and Graph for Precision and Recall for Test results of Original Model

BIAS

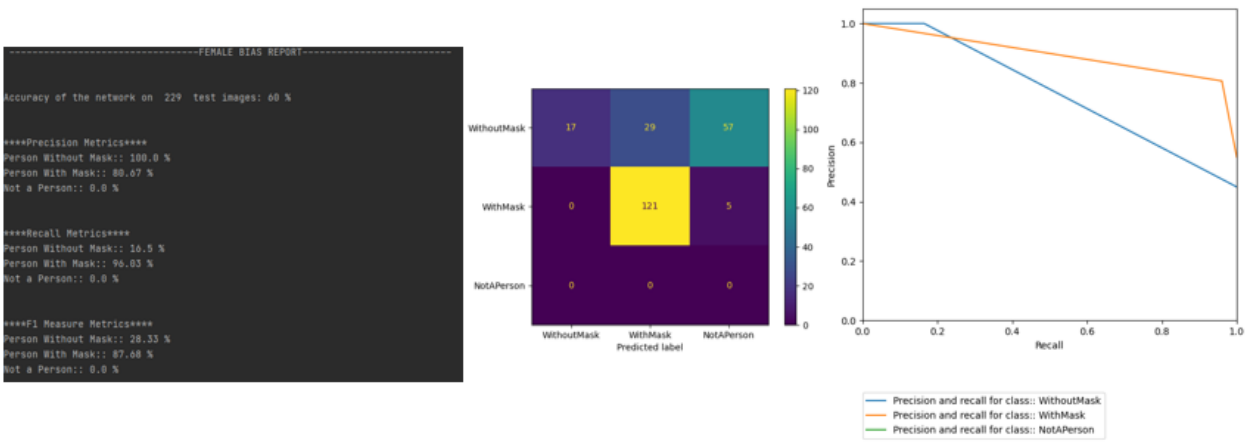
For Bias Testing, we considered the category of gender. Initially we separated the test dataset into two type.

1. Male Test Dataset – Contains only Male images in masked and unmasked category.
2. Female Test Dataset – Contains only Female images in masked and unmasked category.

We completely removed the “Not a Person” folder to get accurate bias results for just male and female images. When we tested our model on these datasets, the results are given below.

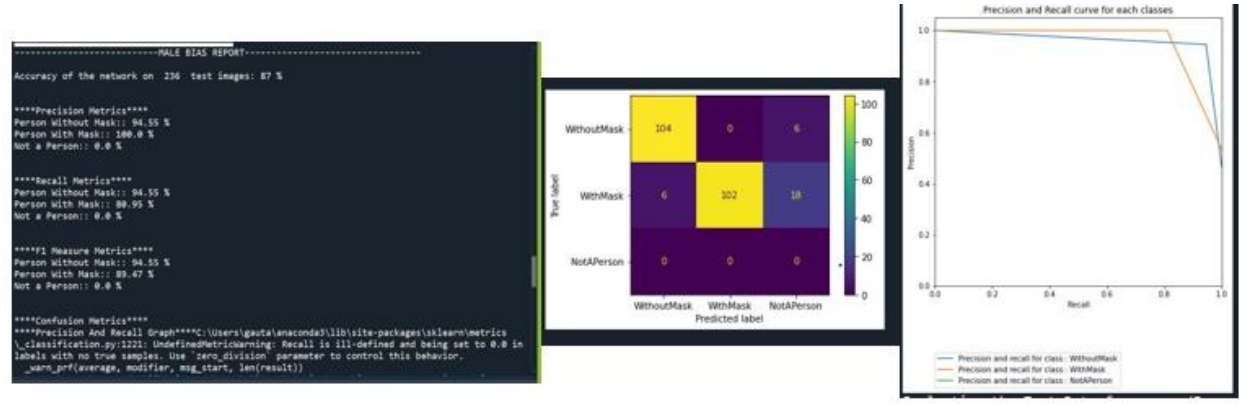


Above figure depicts the metrics for male data set run by old model.

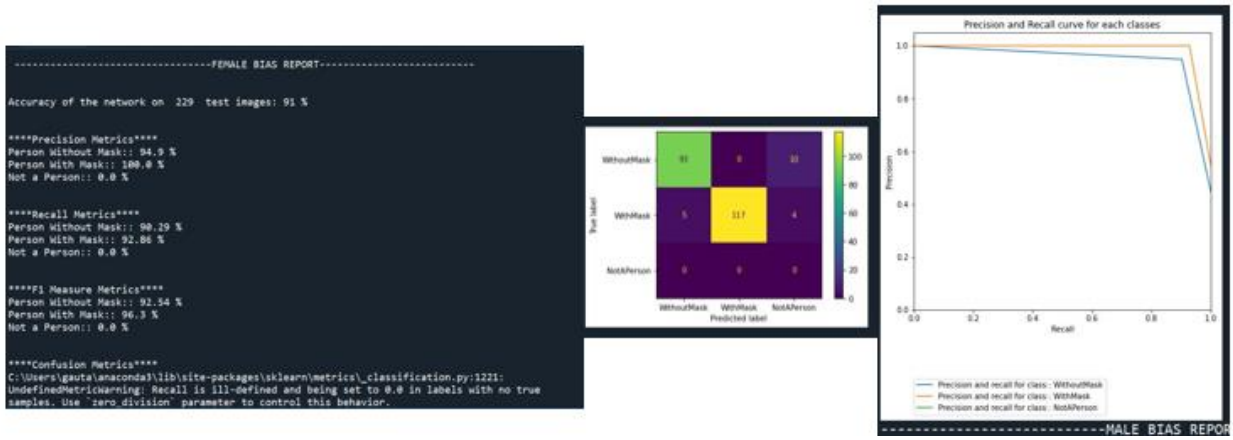


Above figure depicts the metrics for Female data set run by old model.

We rebalanced our data to equal number of images for both the categories in the training set and trained the model again. This new model showed better performance. The results are displayed below.



Above figure depicts the metrics for Male data set run by New model.



Above figure depicts the metrics for Female data set run by New model

We had an accuracy of 70% for male and 60% for female for old model. After the rebalancing of training datasets, we have an accuracy of 88% for male and 91% for female. Below is the detail statistical metrics evaluation for separate Male and Female test dataset with original biased dataset and Bias eliminated New datasets.

			Test Data			
			Old Model Male metrics	Old Model Female metrics	New Model Female metrics	New Model Male metrics
Test Image Count			236	229	236	229
Accuracy (in percentage)			70	60	89	249
Precision (in Percentage)	Without Mask		100	100	94.9	94.55
	With Mask		81.33	80.67	100	100
	Not A Person		0	0	0	0
Recall (in Percentage)	Without Mask		40.91	16.5	90.29	94.55
	With Mask		96.83	96.03	92.86	80.95
	Not A Person		0	0	0	100
F1-Measure (in Percentage	Without Mask		58.06	28.33	92.54	94.55
	With Mask		88.41	87.86	96.3	89.47
	Not A Person		0	0	0	100
Confusuin Matrix (Image Count)	Without M	True Positive	45	17	93	104
		False Positiv	0	0	5	6
		True Negative	126	133	128	120
		False Negati	65	86	10	6
	With Mas	True Positive	122	121	117	102
		False Positiv	28	29	0	0
		True Negative	82	81	110	110
		False Negati	4	5	9	24
	Not A Pers	True Positive	0	0	0	0
		False Positiv	0	0	0	0
		True Negative	195	452	224	214
		False Negati	41	50	14	24

Summary of the bias results.

K-Fold Cross Validation

Instead of the fixed set of train and test dataset as in Phase I, we implemented the K-fold cross validation with 10 folds for the train dataset, during the model training. To achieve this, We utilized the existing python library ***sklearn.model_selection.StratifiedKFold.*** First, we maintain two separate datasets i.e., the original dataset from Phase I and a new dataset maintained after the elimination of Bias. Then, we trained the model with these two different datasets and stored their trained model separately for performing further testing.

We captured the test data splitted by K-Fold cross-validation(10-folds) in each fold. Then for the test /validation data in each of the fold, we evaluated the accuracy, precision, recall, f1-measure and confusion matrix. Below are the detail results obtained by training the model with original datasets and new dataset with 10-fold cross validation:

K-Fold Cross validation on Original Dataset

We applied the K-Fold cross validation on Original Dataset with Train Images Count -> 2898 and Test Images Count ->646. For original dataset, we used the same old CNN model, that use to process 100*100 resolution of image as RGB (3-input/output channels). The statistical evaluation results for test data splitted by K-Fold cross-validation in each fold is provided below in the tabular format:

			Test Images splitted in each fold from Train Dataset										Average Statistics for 10 Folds
			Fold0	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	
Test Image Count			290	580	870	1160	1450	1740	2030	2320	2609	2898	-
Accuracy (in percentage)			86	85	70	97	93	89	95	93	68	66	84.2
Precision (in Percentage)	Without Mask		96.7	95	95.85	96.04	95.92	92.99	93.84	93.9	89.04	86.33	93.56
	With Mask		80.14	82.11	75.21	80.07	81.87	83.77	85.04	86.4	87.02	87.36	82.9
	Not A Person		86.79	82.61	77.84	82.45	84.93	85.76	87.24	86.9	82.21	77.18	83.39
Recall (in Percentage)	Without Mask		80	77.73	63.22	71.92	77.33	80.95	83.66	85.1	86.78	87.19	79.39
	With Mask		100	100	100	100	99.83	98.73	98.67	98.1	91.7	85.73	97.27
	Not A Person		73.02	75.4	76.19	80.16	78.73	76.46	77.55	79.2	77.56	77.42	77.16
F1-Measure (in Percentage)	Without Mask		87.56	85.5	76.19	82.25	85.63	86.55	88.46	89.3	87.89	87.11	85.64
	With Mask		88.97	90.17	85.5	88.93	89.96	90.64	91.35	91.9	89.3	86.54	89.32
	Not A Person		79.31	78.84	77.01	81.29	81.71	80.84	82.11	82.9	79.82	77.3	80.11
Confusion Matrix (Image Count)	Without Mask	True Positive	88	171	208	315	423	531	640	744	853	960	-
		False Positive	3	9	9	13	18	40	42	48	105	152	-
		True Negative	177	351	532	709	885	1044	1223	1398	1521	1654	-
		False Negative	22	49	121	123	124	125	125	130	130	132	-
	With Mask	True Positive	117	234	352	470	587	697	813	924	972	1009	-
		False Positive	29	51	116	117	130	135	143	145	145	146	-
		True Negative	144	295	402	573	732	899	1063	1233	1404	1575	-
		False Negative	0	0	0	0	1	9	11	18	88	168	-
	Not A Person	True Positive	46	95	144	202	248	289	342	399	439	487	-
		False Positive	7	20	41	43	44	48	50	60	95	144	-

		True Negative	220	434	640	865	1091	1314	1539	1756	1948	2125	-
		False Negative	17	31	45	50	67	89	99	105	127	142	-

Table 1: K-fold Cross-validation on Original Dataset

We have attached screenshots for the first and last Fold of the K-fold cross-validation on training data from the python console:

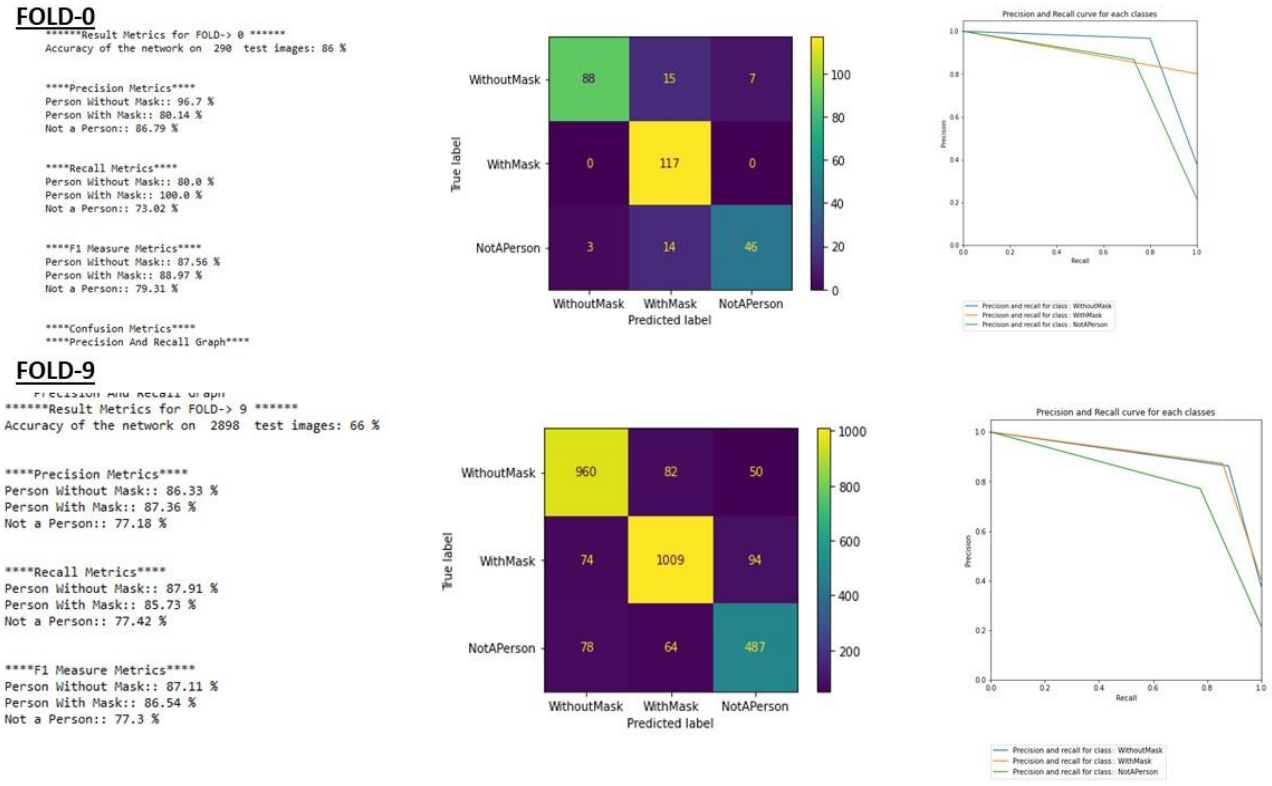


Fig: Fold 0 and Fold1 Result metrics in python console

K-Fold Cross validation on Bias Eliminated Dataset

The count of train images in the new dataset after the bias elimination was kept same as the original dataset to maintain the consistency in the obtained result i.e., Train Images Count -> 2898. However, the count of new test image dataset count was 639.

For this new dataset, we used the new CNN model, that process 100*100 resolution of grayscale image as 1-input/output channel. The statistical evaluation results for test data splitted by K-Fold cross-validation in each fold is provided below in the tabular format.

			Test Images splitted ineach fold from Train Dataset										Average Statistics for 10 Folds
			Fold0	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	
Test Image Count			290	580	870	1160	1450	1740	2030	2320	2609	2898	-
Accuracy (in percentage)			67	72	69	88	92	88	91	85	71	62	78.5
Precision (in Percentage)	Without Mask		86.21	92.31	93.14	92.45	92.64	91.87	90.53	89	85.5	83.15	89.67
	With Mask		67.84	73.19	71.87	74.64	77.45	79.24	81.4	82	81.95	82.28	77.16
	Not A Person		60	57.07	57.3	61.68	66.22	68.24	70.8	72.4	71.4	67.16	65.222
Recall (in Percentage)	Without Mask		22.73	27.7	28.88	44.75	55.21	62.04	67.45	71.2	73.75	75.92	52.96
	With Mask		99.15	99.15	99.43	99.57	99.32	97.88	97.21	96.7	92.08	85.22	96.52
	Not A Person		85.71	89.68	85.19	78.57	77.78	76.72	76.42	73.2	71.91	72.81	78.8
F1-Measure (in Percentage)	Without Mask		35.97	42.11	44.08	60.31	69.19	74.07	77.3	79.1	79.19	79.37	64.06
	With Mask		80.56	84.21	83.43	85.32	87.03	87.58	88.61	88.8	86.72	83.72	85.59
	Not A Person		70.59	69.75	68.51	69.11	71.53	72.23	73.5	72.8	71.65	69.87	70.95
Confusion Matrix (Image Count)	Without Mask	True Positive	25	60	95	196	302	407	516	622	725	829	-
		False Positive	4	5	7	16	24	36	54	77	123	168	-
		True Negative	176	355	534	706	879	1048	1211	1369	1503	1638	-
		False Negative	85	160	234	242	245	249	249	252	258	263	-
	With Mask	True Positive	116	232	350	468	584	691	801	911	976	1003	-
		False Positive	55	85	2	159	170	181	183	200	215	216	-
		True Negative	118	261	381	531	692	853	1023	1178	1334	1505	-
		False Negative	1	2	137	2	4	15	23	31	84	174	-
	Not A Person	True Positive	54	113	161	198	245	290	337	369	407	458	-
		False Positive	36	85	120	123	125	135	139	141	163	224	-
		True Negative	191	369	561	785	1010	1227	1450	1675	1880	2045	-
		False Negative	9	13	28	54	70	88	104	135	159	171	-

Table 2: K-fold Cross-validation Bias Eliminated Dataset

We have attached screenshots for only first and last folds results of k-fold cross-validation on training data from the python console.

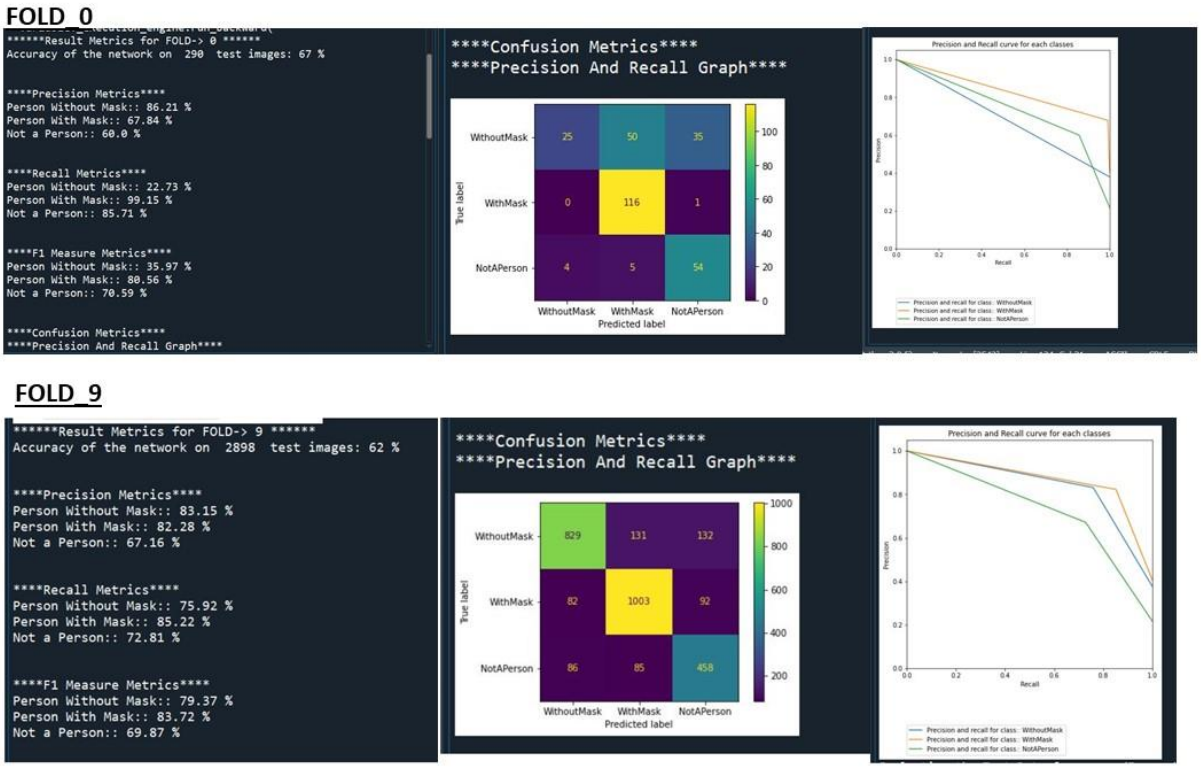


Fig: Fold 0 and Fold1 Result metrics in python console

Result Comparison (K-fold vs Fixed Datasets)

We observed slight increase in the performance of system after we added k-fold cross validation with 10 folds during the model training. The accuracy of the system with fixed dataset train/test split was 86%. However, when we trained the model of the same original dataset along with k-fold cross validation, the accuracy was improved by 1%. We also observed a higher accuracy of 10-fold implementation in the new dataset after the Bias elimination. In addition to this, the new CNN model, that converts image to grayscale instead of RGB seems to have improvement in the computation time and slightly with the performance too.

Below is the statistical comparison of testing on the original model, K-fold cross validation model trained with original dataset and K-fold cross validation model trained with new dataset (bias elimination).

			Test Data		
			Original Train/Test Split Model	K-Fold Cross validation in Original Trained Model	K-Fold Cross validation in New Trained Model
Test Image Count			630	646	639
Accuracy (in percentage)			86	87	89
Precision (in Percentage)	Without Mask		86.12	84.71	87.82
	With Mask		97.22	90.69	97.35
	Not A Person		78.4	86.11	82.94
Recall (in Percentage)	Without Mask		97	98.18	95.87
	With Mask		66	88.89	87.3
	Not A Person		94	71.26	85.98
F1-Measure (in Percentage)	Without Mask		91	90.95	91.67
	With Mask		79	89.78	92.05
	Not A Person		85	77.99	84.43
Confusion Matrix (Image Count)	Without Mask	True Positive	267	216	209
		False Positive	43	39	29
		True Negative	312	387	392
		False Negative	8	4	9
	With Mask	True Positive	140	224	220
		False Positive	4	23	6
		True Negative	417	371	381
		False Negative	69	28	32
	Not A Person	True Positive	138	124	141
		False Positive	38	20	29
		True Negative	446	452	446
		False Negative	8	50	23

Table 3: Evaluation Metrics for original model, K-fold cross validation on Original Trained model and K-fold cross validation on New Trained Model

```
Loading Existing Train Model from .... trained_cnn_Model.pt
Evaluating the Test Data from :: ../Images/test
Accuracy of the network on  646  test images: 87 %


****Precision Metrics****
Person Without Mask:: 84.71 %
Person With Mask:: 90.69 %
Not a Person:: 86.11 %


****Recall Metrics****
Person Without Mask:: 98.18 %
Person With Mask:: 88.89 %
Not a Person:: 71.26 %


****F1 Measure Metrics****
Person Without Mask:: 90.95 %
Person With Mask:: 89.78 %
Not a Person:: 77.99 %


****Confusion Metrics****
****Precision And Recall Graph****
```

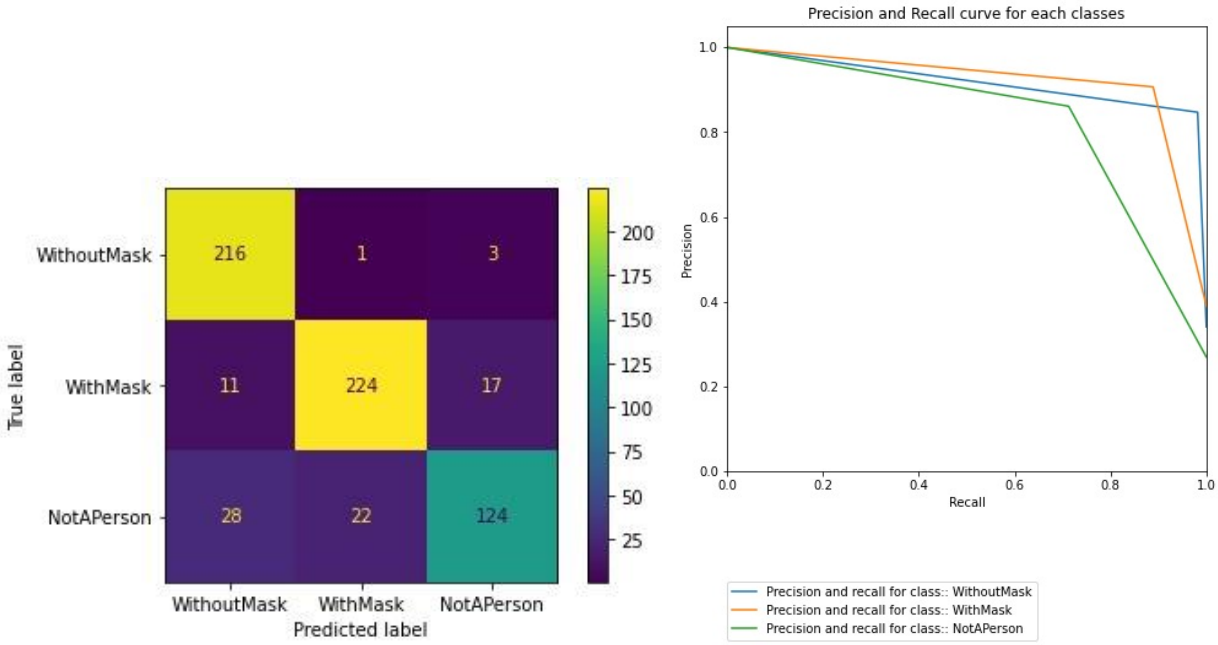


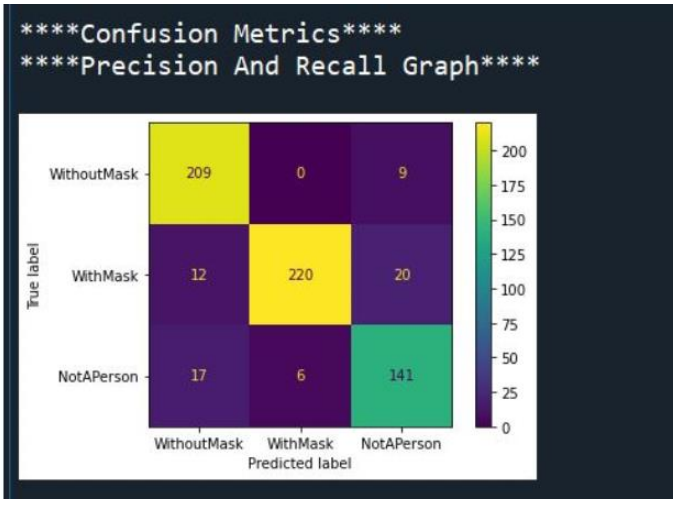
Figure 1: Accuracy, Precision, Recall, F1-Measures, Confusion Matrix and Graph for Precision and Recall for Test results of K-fold cross-validation on Original Model

```
In [20]: runfile('D:/Concordia Notes/ArtificialIntelligence_COMP6721/Project/AI-
Project-2/AI-project-1/Code/main.py', wdir='D:/Concordia Notes/
ArtificialIntelligence_COMP6721/Project/AI-Project-2/AI-project-1/Code')
Loading Existing Train Model from .... trained_cnn_Model.pt
Evaluating the Test Data from :: ../Images/test
Accuracy of the network on  634  test images: 89 %


****Precision Metrics****
Person Without Mask:: 87.82 %
Person With Mask:: 97.35 %
Not a Person:: 82.94 %


****Recall Metrics****
Person Without Mask:: 95.87 %
Person With Mask:: 87.3 %
Not a Person:: 85.98 %


****F1 Measure Metrics****
Person Without Mask:: 91.67 %
Person With Mask:: 92.05 %
Not a Person:: 84.43 %
```



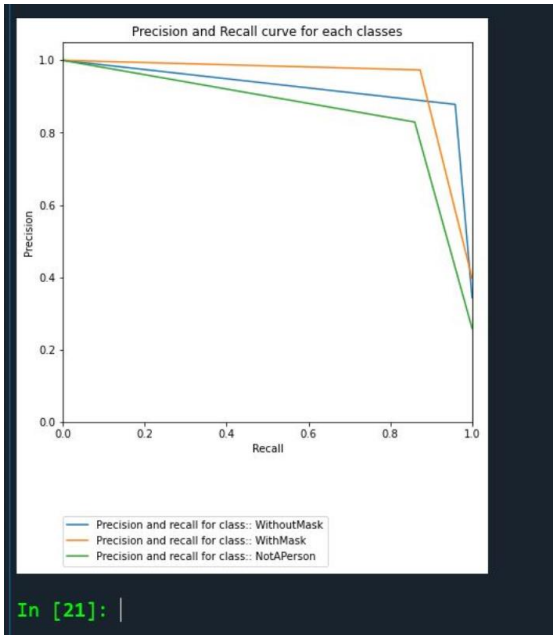


Figure 2: Accuracy, Precision, Recall, F1-Measures, Confusion Matrix and Graph for Precision and Recall for Test results of K-fold cross-validation model (After Bias)

GIT Link

https://github.com/nikpat9/AI-project-1

References:

- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://stats.stackexchange.com/questions/360157/epoch-vs-iteration-in-cnn-training
- https://www.analyticsvidhya.com/blog/2020/07/how-to-train-an-image-classification-model-in-pytorch-and-tensorflow/
- https://medium.com/jovianml/how-i-created-a-simple-mask-detector-using-gpu-in-pytorch-bd13f3542f46
- https://medium.com/@i2i-blog/facial-mask-detection-using-deep-learning-and-computer-vision-c0966e14dd94
- https://towardsdatascience.com/understanding-pytorch-with-an-example-a-step-by-step-tutorial-81fc5f8c4e8e
- https://www.researchgate.net/publication/267269542_SWU-OFDB_A_database_for_occluded_face_detection_research
- https://deeplizard.com/learn/video/MasG7tZj-hw
- https://www.tutorialspoint.com/pytorch/pytorch_convolutional_neural_network.htm
- https://scikit-learn.org/0.15/modules/generated/sklearn.cross_validation.StratifiedKFold.html
- https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G
- https://machinelearningmastery.com/k-fold-cross-validation/
- https://scikit-learn.org/stable/modules/cross_validation.html