Malpractice Detection in Examinations

A Project Report in partial fulfillment of the degree

Bachelor of Technology

in

Electronics & Communication Engineering/Computer Science & Engineering

By

19K41A04B6	S. Amulya
19K41A05A8	P. Rithika Reddy
19K41A05B1	S. Shiva Keerthi

Under the Guidance of
Dr. V. Venkataramana,
Assistant Professor, EEE Dept.

Submitted to

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
S R ENGINEERING COLLEGE(A), ANANTHASAGAR, WARANGAL
(Affiliated to JNTUH, Accredited by NBA)

Dec-2021

SR Engineering College

Innovation . Creativity . Entrepreneurship

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Project Report entitled "Malpractice Detection in Examinations" is a record of bonafide work carried out by the student(s) S. Amulya, P. Rithika Reddy, S. Shiva Keerthi bearing Roll No(s) 19K41A04B6, 19K41A05A8, 19K41A05B1 during the academic year 2021-2021 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Electronics & Communication/Computer Science Engineering** by the Jawaharlal Nehru Technological University, Hyderabad.

.

Supervisor **Head of the Department**

External Examiner

ABSTRACT

Online education has become more prevalent these days due to the covid-19 pandemic. Students have gotten used to taking online tests with manual proctoring through video communication platforms such as zoom, Ms. Teams, etc. But as the times change, students are getting more and more creative while cheating during examinations. As the manual invigilation is tougher in this scenario, taking advantage of it, malpractice in online exams is increasing. To overcome this, we developed a Neural Network model which proctors each student through their own camera. This model detects if a candidate taking the exam is involved in Malpractice or not depending on their posture, and other factors like multiple people appearing in camera or detection of electronic gadgets being used like mobile phones, laptops, etc.

Table of Contents

1. INTRODUCTION

The Artificial Intelligence sector is thriving, and sequentially, mankind is taking huge strides towards the progress of technology. Several AI algorithms are being written for different problem statements in daily life to make human life more comfortable and easier. To contribute our own little share to this magnificent community, we found a problem which is in dire need of a solution, and hence, we tried to resolve it using Deep learning model.

The covid-19 pandemic has compelled the students to study from home for a significant period of time. As a consequence, online exams are now very common in every educational institution. Students can easily take the test at their home very conveniently. In the present year, even after going back to offline classes, the online tests are still being utilized due to the time constraint of in-person classes.

Online tests are advantageous for both the students and teachers, because they find these tests more comfortable and time saving. One more advantage is that, most of the online tests are self-evaluated after the exam is finished, so it reduces burden of teachers and students can immediately see their results. The grass is not greener on the other side, as the online tests highly increases the risks of malpractice. Even though the management conducts online proctoring, it is difficult to keep an eye on each and every student throughout the exam. This is where we step in with our deep learning model. We use Artificial Intelligence to build a Neural Networking model which constantly observes all the students through their individual cameras and alerts the teacher whenever malpractice is detected. The prediction is based on the posture of student while taking the test. Through this model, teachers can be hugely benefited as proctoring has never become so easy.

2. LITERATURE REVIEW

In this paper the authors have developed an AI Based proctoring system. Here they have proposed a web-based system to identify malpractice during online examinations. At First, students should register on a portal for the first time. Then using the web cam if there are multiple persons or if a mobile phone is detected then it is saved in the log as malpractice. They have also considered voice detection and browser switching as malpractice. For MCQ based questions as there is no need for pen and paper the head position of students is analyzed and if the student is looking away from the screen even it is treated as malpractice [1].

In this paper, they used a face recognition algorithm to analyze and detect faces of multiple people. While writing an exam, they consider real-time recording scenes by using reatime video capture [2].

Here, they have developed a model to detect malpractice using we-cam to capture the video and audio. Along with that, they also considered active window capture. They detect the examinee's face feature points and head pose to detect malpractice.[3]

3. DESIGN

3.1 Requirement Specifications (S/W & H/W)

Hardware Requirements

✓ **Processor**	: Intel(R) Core (TM) i7-10700 CPU @ 2.90GHz 2.90 GHz
✓ **RAM**	: 64.0 GB (63.8 GB usable)
✓ **Hard Disk**	: 1TB
✓ **Input**	: Keyboard and Mouse
✓ **Output**	: Desktop Screen

Software Requirements

✓ **OS**	: Windows 11
✓ **Platform**	: Visual Studio Code
✓ **Deployment software**	: Stream lit
✓ **Programing Language**	: Python

4. DATASET INSIGHTS

The model we build will predict whether or not a person is involved in malpractice during examination based on their posture by taking an RGB image input. Accordingly, we have collected the dataset by gathering some students and conducting a mock test to understand the postures of the person during an exam. We noted that the most common posture for a person not involved in malpractice is to sit straight and focus on the screen. On the other hand, the malpractice postures usually involve looking sideways and away from the screen. So, from this, we have collected a total of 330 images, out of which 174 samples are for students engaged in malpractice and 156 samples for students not engaged in malpractice during the online exam. A few samples from our dataset are as shown below.

Accepted Data Samples

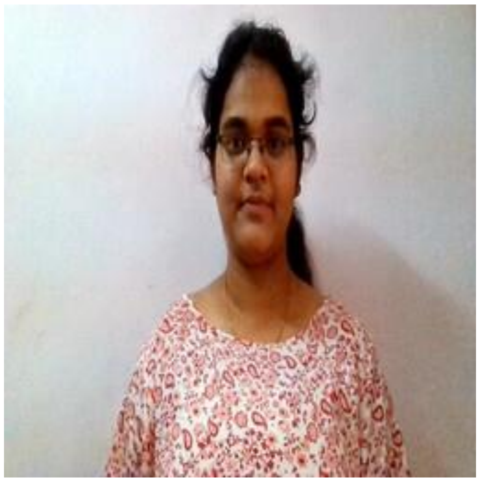

Fig 4.1 *Looking into the screen*

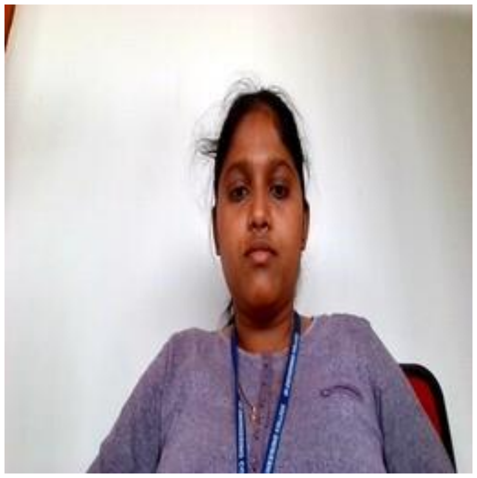

Fig 4.2 *Normal behavior during exam*

Not Accepted Data Samples

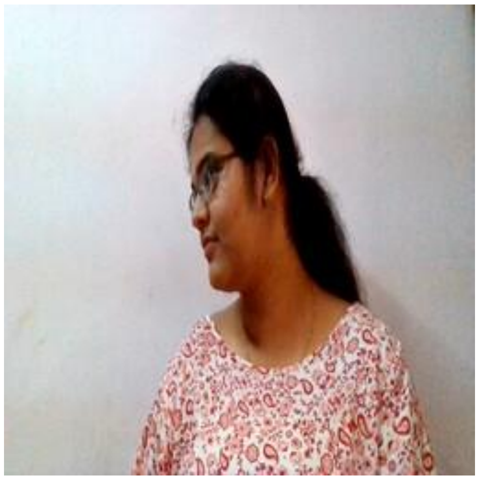

Fig 4.3 *Looking sideways*

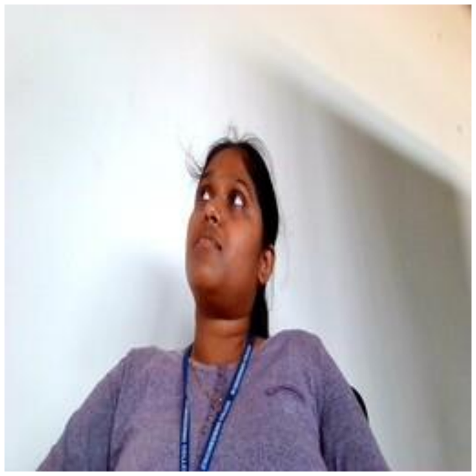

Fig 4.4 *Looking upward*

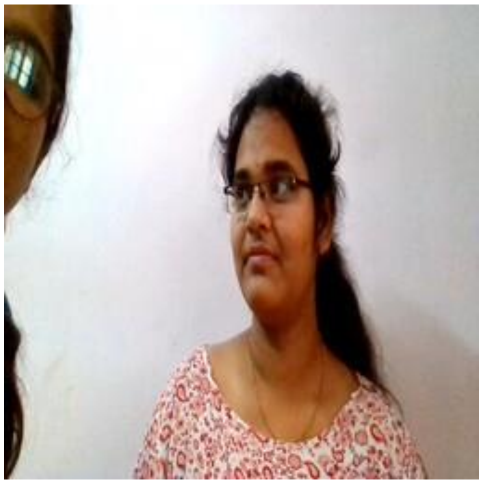

Fig 4.5 *Another person*

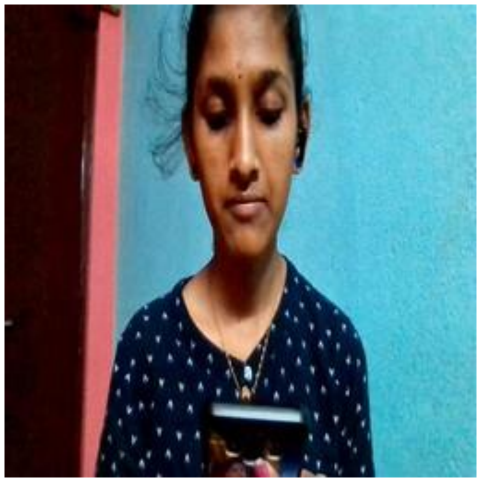

Fig 4.6 *Looking in mobile*

5. DATA PREPROCESSING

5.1 Normalization

This is also known as Data re-scaling [4]. Generally, the pixels are between 0 to 255. As a result, some pixel values can act as a bias to the model. So here we scale the pixels so that the lie between 0 to 1. This can also be used to tackle the propagating gradients problem [4]. As shown below there are multiple ways to normalize the images. Here we chosen the first technique to normalize the images.

#way1-this is common technique followed in case of RGB images

norm1_image = image/255

#way2-in case of medical Images/non natural images

norm2_image = image - np.min(image)/np.max(image) - np.min(image)

#way3-in case of medical Images/non natural images

norm3_image = image - np.percentile(image,5)/ np.percentile(image,95) - np.percentile(image,5)

plot_image([image, norm1_image, norm2_image, norm3_image], cmap='gray')

5.2 Resizing Images

The images of our dataset are resized from (1280*720) original resolution of the image to a compressed size of (256*256) resolution. The image resizing is done to make our dataset compatible to the deep learning models. Additionally, high image resolution images require more computation power, which in turn increases the model complexity. So, to avoid this, we pre-process our dataset by resizing the images.

5.3 Data Split

To train any model, it is essential to perform data splitting. It is the process of dividing the dataset into training and testing data. The training dataset is used to train the model. After the model is trained, we use the testing dataset to crosscheck the accuracy of our model. Data splitting allows us to assess the performance of the model. In this model, we use an ideal ratio 80:20 split. That is, 80% of data is used for training and the remaining 20% is used for the testing purpose.

6. METHODOLOGY

6.1 Image net

It is an image database that has 14,197,122 images. These images were arranged according to the WordNet hierarchy. It was developed to help researchers and students in image research. It also hosts contests such as ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). In these contests researchers were challenged to come up with solutions with least error rate. They were asked to classify the images into 1000 classes. As a result of this competition some models such as Inception, VGG, ResNet etc. were selected as the winners. In this project, we have used Transfer Learning to train our CNN Model. From the available algorithms we have chosen four different deep learning models like VGG19, InceptionV3, Xception and MobilenetV2.

6.2 VGG19

It is an image recognition model that takes RGB Images of size (224*224) as input. The input matrix has a shape if (224,224,3) [5]. The pre-processing they applied was to subtract the mean RGB value from each pixel over the training dataset [5]. The filters used are of size (3*3) with a stride of 1 pixel. In order to decrease the possibility of losing the data in the corner pixels spatial padding is used here. In addition, Max Pooling is performed with a filter size of (2*2) and stride 2. In the hidden layers the activation used is Rectified Linear Unit (ReLu) [5]. In the output layer softmax activation function is used with 1000 neurons to classify image into 1000 classes.

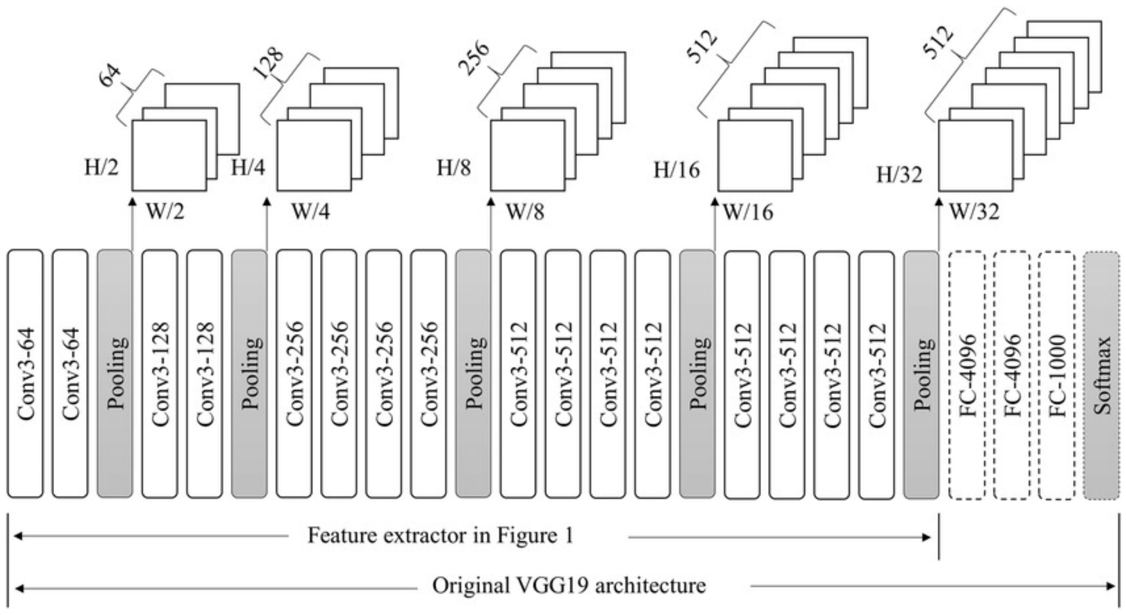

Fig 6.2.1 *VGG19 Architecture [6]*

6.3 InceptionV3

This model was developed by Google. It is has a total of 42 layers. This is a higher than the older versions InceptionV1, InceptionV2 models [7]. This model gives an accuracy of 78.1% on the image net dataset [8]. The default input size of this model is (299*299). But the image size in the imagenet dataset was (224*224). Batch Normalization is widely used in this model. Softmax activation function is used in the output layer.

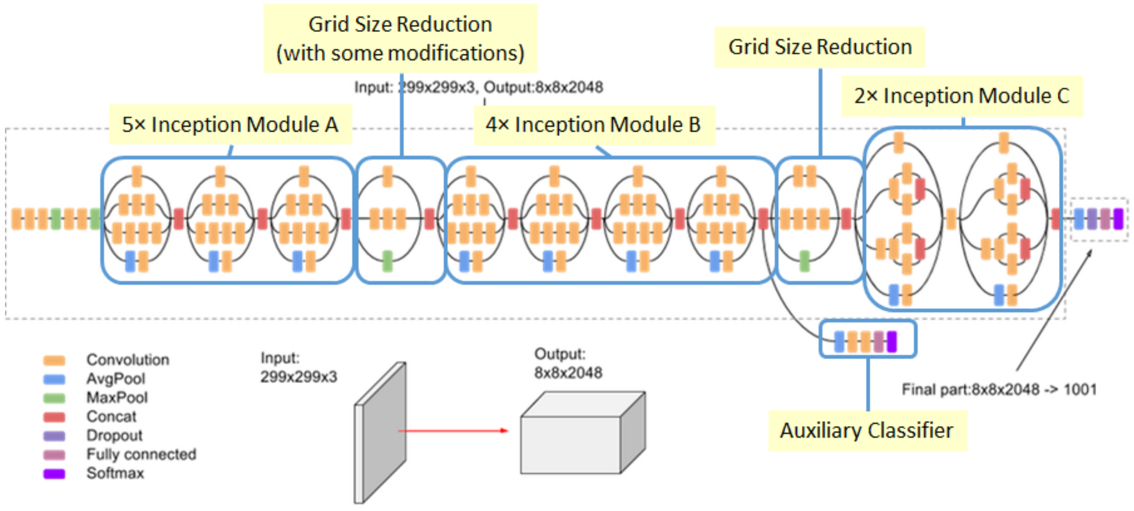

Fig 6.3.1 InceptionV3 Architecture [8]

6.4 Xception

This model is an extreme version of InceptionV3 developed by google. This model is much better than InceptionV3. In InceptionV3 Depth wise Convolution is performed first and then Point wise convolution [9]. But in the Xception model they first performed Pointwise convolution and then depth wise convolution. Also, there is no intermediate Relu non-linearity. This model achieved highest accuracy when they have not used any intermediate activation when compared with the ones using ELU or ReLu [9].

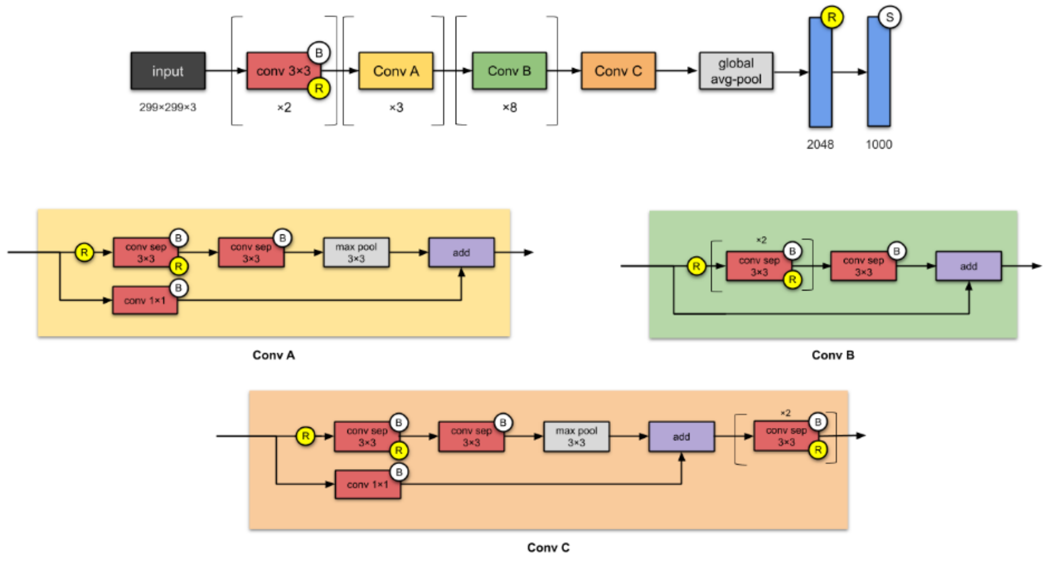

Fig 6.4.1 Xception Architecture [10]

6.5 MobilenetV2

In the previous version MobilenetV1 Depth wise Separable Convolution is used which reduced the complexity cost and model size of the network. This makes the model appropriate for mobile devices. In MobilenetV2 inverted residual structure in introduced. With this model non-linearities in narrow layers are removed.

This model comprises of two blocks. One is a residual block with stride 1 and the other is with stride 2 for downsizing [11]. In both the blocks there are 3 layers. The first layer has 1*1 convolution with activation function as ReLu6. The second layer is depth wise convolution and the third layer is with 1*1 convolution and with no linearity [11]. It is understood that is ReLu is used again then the neural network has the power of linear classification on the non-zero part in the output layer. Also, there is an expansion factor t. For all the main experiments t value is 6. That is if the input has 64 channels the output internally would be 64*6 = 384 channels [11].

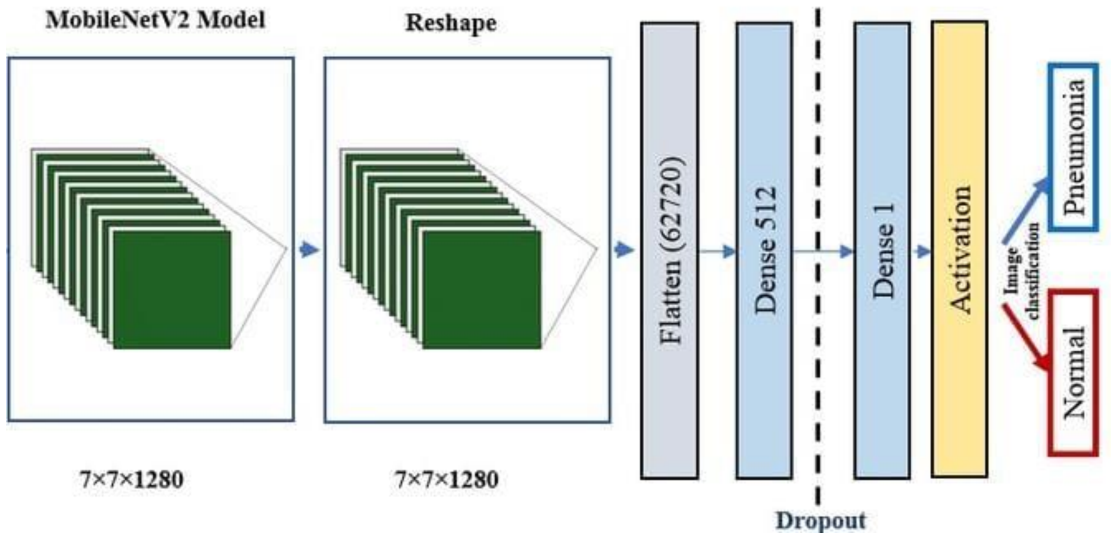

Fig 6.5.1 *MobilenetV2 Architecture [12]*

7. RESULTS

To find the best model with highest accuracy, we worked with different deep learning models and noted down all the accuracies we compiled. We tried to find out the finest model by varying number of neurons in hidden layer, activation functions in hidden layer, activation functions in output layer for different deep learning models (i.e., VGG19, Xception, InceptionV3 and MobileNetV2). The accuracies we gathered can also be used for other research purposes in other works for future.

7.1 VGG19

- As shown in Table 1 we have tabulated the accuracies by using sigmoid as an activation function in the output layer. Here we have increased the neurons and tested the model with other activation functions in hidden layers.

Table 1 *Accuracies using sigmoid as activation function in output layer*

Single layer	**Sigmoid**	**TanH**	**ReLu**	**Model parameters**
20	68%	68%	70.2%	6,55,401
40	68%	68%	68%	13,10,801
60	68%	68%	68%	19,66,201
80	68%	68%	72.3%	26,21,601
100	68%	68%	70.2%	32,27,001

- As shown in Table 2 below we have used softmax as activation function in output layer by varying the neurons and activation functions in hidden layers.

Table 2 *Accuracies using softmax as activation function in output layer*

Single layer	**Sigmoid**	**TanH**	**ReLu**	**Model Parameters**
20	68%	68%	72.3%	6,55,422
40	68%	68%	68%	13,10,842
60	68%	68%	68%	19,66,262
80	68%	68%	72.3%	26,21,682
100	68%	68%	74.4%	32,77,102
120	68%	68%	70.2%	39,32,522

- We have observed that the model is showing good accuracy of 74.4% when the input neurons are 100, activation function in hidden layer is ReLu and activation function in output layer is softmax. We fixed these parameters and changed hidden layers as below.

Table 3 *Accuarcies after increasing the hidden layers*

No. of hidden layers	**Relu**	**Model Parameters**
1	74.4%	32,77,102
2	76.6%	32,87,202
3	72.3%	32,97,302
4	70.2%	33,07,402

- The highest accuracy that we have achieved for VGG19 model is 76.6% with 2 hidden layers each with 100 neurons, activation function in hidden layers is ReLu and the activation function in output layer is softmax.

7.2 Xception

- As shown in Table 4 we have tabulated the accuracies by using sigmoid as an activation function in the output layer and by varying the neurons in hidden layer.

Table 4 *Accuracies using sigmoid as activation function in output layer*

Single layer	**Sigmoid**	**TanH**	**ReLu**	**Model Parameters**
10	68%	68%	72.3%	13,10,741
20	68%	68%	68%	26,21,481
40	68%	68%	70.2%	52,42,961
60	68%	68%	68%	78,64,441
80	68%	68%	68%	1,04,85,921

- As shown in Table 5 we evaluated the performance of the model by using softmax as activation function in output layer by varying the neurons and activation functions in hidden layers.

Table 5 *Accuracies using softmax as activation function in output layer*

Single layer	**Sigmoid**	**TanH**	**ReLu**	**Model Parameters**
10	68%	68%	68%	13,10,752
20	68%	68%	68%	26,21,481
40	68%	68%	68%	52,43,002
60	68%	68%	68%	78,64,502
80	68%	68%	65.9%	1,04,85,921
100	68%	68%	68%	1,31,07,502

- We noticed that the best accuracy the model attained is 72.3% with 10 neurons in the hidden layer, activation function in hidden layer is ReLu and activation function in output layer is sigmoid. So, as displayed in Table 6 we have increased the hidden layers for more accuracy.

Table 6 *Accuracies after increasing the hidden layers*

No. of hidden layers	ReLu	Model Parameters
1	72.3%	13,10,741
2	70.2%	13,10,851
3	68%	13,10,961
4	68%	13,11,071

- The best accuracy that the model achieved is 72.3% with 1 hidden layer of 10 neurons, activation function in hidden layer is ReLu and the activation function in output layer is sigmoid.

7.3 InceptionV3

- As shown in Table 7 we tabulated the accuracies by using sigmoid as an activation function in the output layer and by varying the neurons in hidden layer.

Table 7 *Accuracies with sigmoid as activation function in output layer*

Single layer	Sigmoid	TanH	ReLu	Model Parameters
20	66.1%	66.1%	69.2%	20,041
40	66.1%	66.1%	66.1%	40,081
60	66.1%	66.1%	66.1%	60,121
80	66.1%	66.1%	66.1%	80,161
100	66.1%	66.1%	66.1%	100,201

- As displayed in Table 8 we also calculated the accuracies when we use softmax as activation function in output layer by varying the number of neurons in a layer.

Table 8 *Accuracies with softmax as activation function in output layer*

Single layer	Sigmoid	TanH	ReLu	Model Parameters
20	66.1%	66.1%	66.1%	20,062
40	66.1%	66.1%	66.1%	40,122
60	66.1%	66.1%	66.1%	60,182
80	66.1%	66.1%	66.1%	80,242
100	66.1%	66.1%	67.6%	100,302

- We noted that the best model with one hidden layer is with using 20 neurons in hidden layer, activation function in hidden layer as ReLu and the output layer activation function as sigmoid. As shown in Table 9 we increased the hidden layers to get good accuracy.

Table 9 *Accuracies after increasing the hidden layers*

No. of hidden layers	ReLu	Model Parameters
1	69.2%	20,041
2	66.1%	20,461
3	67.6%	20,881
4	66.1%	21,301

- Finally, the model that got the highest accuracy of 69.2% is with 20 neurons in hidden layer, one hidden layer. The activation functions in hidden layers and output layers are ReLu and Sigmoid respectively.

7.4 MobilenetV2

- Finally, among all the models that we have implemented MobilenetV2 has given the highest accuracy of 92.4%. We have tabulated the model and its accuracies below.

Table 10 *Model Accuracies*

Single layer	Sigmoid	TanH	ReLu
20	56%	57.5%	63.6%
40	68.7%	70.6	72.7%
60	68.9%	69.6%	76.1%
80	68.4%	73.4%	83.6%
100	69.3%	73.8%	89.5%
120	73%	84.5%	92.4%

- The overall model accuracies are tabulated below.

Table 11 *Model Accuracies*

Model	Accuracy
VGG 19	76.60%
Xception	72.30%
InceptionV3	69.20%
MobilenetV2	92.40%

- Based on this data, the best model is chosen as MobilenetV2 with 92.40% accuracy.

- The loss and accuracies for training and testing data of MobilenetV2 model are plotted as shown in Fig 7.1.

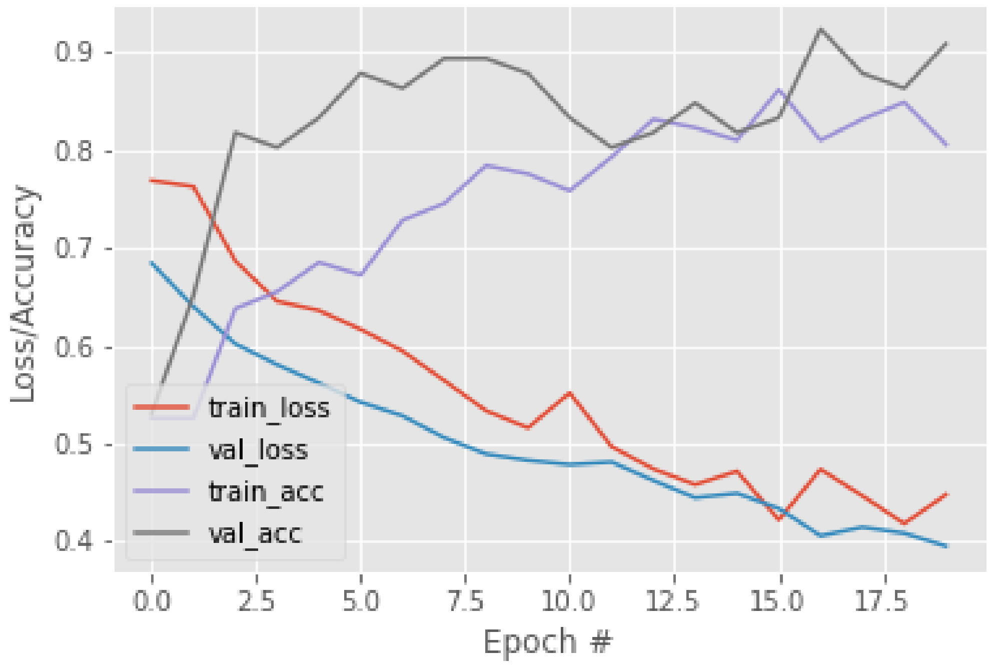

Fig 7.1 *Training Loss and Accuracy*

8. APPLICATION

In real time, to make the users understand that the model detects malpractice based on posture, candidates must be well informed about this. It is better if the examiner gives instructions to the users before starting the exam as mentioned below.

Instructions to students:

1. Please be seated in an isolated place with plain background.
2. Focus on your screen while taking the exam.
3. Malpractice will be detected based on your posture, so restrain from moving sideways.
4. If the camera detects another person or any electronic gadgets like laptop, mobile phone, etc., it is considered as malpractice.
5. Press the start button to start the camera and begin the exam.

   When the camera starts, continuous image inputs will be taken and fed to the model to get output as malpractice or not. The testcases are as shown below.

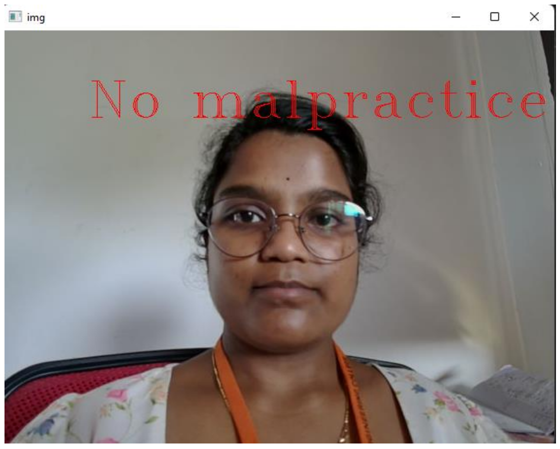

Fig 8.1 *No Malpractice*

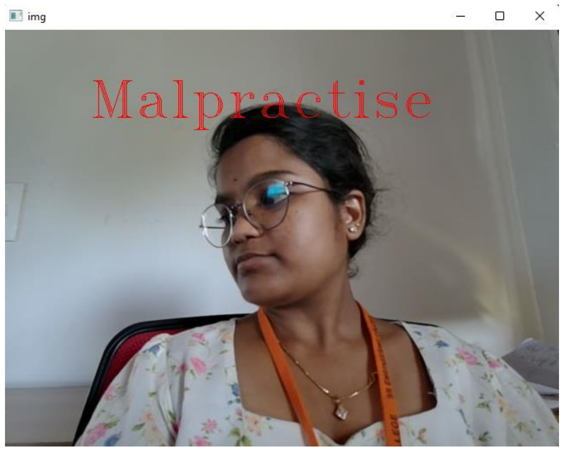

Fig 8.2 *Looking Side ways*

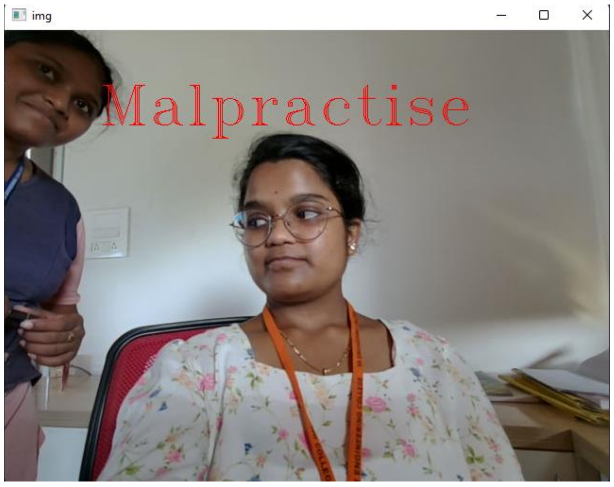

Fig 8.3 *Another person detected*

9. CONCLUSION

Through this project, we created a Malpractice detection deep learning model which can be applied to online exams in real time. This model provides online proctoring based on the posture of students, and is helpful for the examiners.

10. FUTURE SCOPE

Our model can be further improvised by adding these features in future.

1. Build a model with same concept to detect malpractice offline examinations.
2. Use Eye-tracking software to make the model more efficient
3. Prepare a full-fledged cloud application which can be accessed by all.

REFERENCES

[1] Motwani, S., Nagpal, C., Motwani, M., Nagdev, N., & Yeole, A. (2021). AI-Based Proctoring System for Online Tests. *Available at SSRN 3866446*.

[2] Kumar, S. J., Spandan, G., Kumar, N. N., Mamatha, G. R., & Roopesh, R. (2020). Online examination and malpractice detection. *J. Xi'an Univ. Archit. Technol*.

[3] Prathish, S., & Bijlani, K. (2016, August). An intelligent system for online exam monitoring. In *2016 International Conference on Information Science (ICIS)* (pp. 138-143). IEEE.

[4] https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-15624ef3a28b

[5] https://iq.opengenus.org/vgg19-architecture/

[6] https://www.researchgate.net/figure/Feature-extractor-architecture-using-VGG19-network_fig1_344440237

[7] https://iq.opengenus.org/inception-v3-model-architecture/

[8] https://cloud.google.com/tpu/docs/inception-v3-advanced

[9] https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

[10] https://miro.medium.com/max/2000/1*NNsznNjzULwqvSER8k5eDg.png

[11] https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c

[12] https://www.researchgate.net/profile/Dina-Hussein-4/publication/344058411/figure/fig6/AS:931406553284608@1599076035154/MobileNetV2-architecture.ppm