IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

FRAUDULENT ACTIVITY DETECTION MODEL DURING EXAMINATION

Sharan Varghese[1], Niket Kumar[2], Sandhya Sinha[3], B. Prabha[4]

[1, 2, 3, 4] Department of Computer Science, SRM Institute of Science and Technology.

ABSTRACT

Online exam cheating has become a significant challenge for academic institutions worldwide due to the ease of access to online resources and the difficulty in monitoring students remotely. The existing systems for detecting online exam cheating are limited to simple rules-based systems that can only identify a small number of dishonest behaviors. In this essay, we suggest a Deep Learning-based Online Exam Cheating Detection System using Behavioral Modeling. Deep learning algorithms and a camera are used by the system to monitor the behavior of students during online exams. The deep learning algorithm is trained to classify normal and abnormal behavior based on the features detected by the camera.

To evaluate the proposed system, we conducted experiments on a dataset of online exam scenarios that included both normal and abnormal behavior. The results of our experiments demonstrate that the proposed system can accurately detect cheating behavior in online exams with a high degree of accuracy.

Our proposed system has several advantages over existing systems, including the ability to detect a wider range of cheating behaviors, high accuracy, and low instances of false positives. The proposed system can be used to enhance the security and integrity of online exams and can help academic institutions to maintain the quality of their assessment process.

1. INTRODUCTION

The identification of abnormal patterns in data compared to expected normal patterns is a crucial issue in abnormal detection. These divergences are referred to as abnormalities. Recently, the field of object detection has made significant progress and is quite efficient in detecting many features of the human body. The study of human behavior is of great significance, and with the help of OpenCv, one can classify these behaviors and detect the said classes in a single person's frame during an online exam. In abnormality detection, a significant challenge is to clearly define what is considered abnormal, and in the case of a single person in the frame, this becomes relatively simpler.

In this scenario, Yolo pose can be used to detect features of the person. The image can be resized to

fit the model, and the landmarks can be placed in accordance with the body features of the person taking the online exam.

YOLO (You Only Look Once) is an object detection system that can also be used for detecting body key points in real-time video or images. YOLO Pose, specifically, is a variant of YOLO that is specifically designed for detecting human body keypoints.

To detect body keypoints using YOLO Pose, the system typically takes an input image or video and performs the following steps:

Image preprocessing: The input image or video is preprocessed to convert it into a format that can be fed into the deep learning model.

*Body part detection***:** YOLO Pose uses a deep neural network to detect body keypoints. The network takes the preprocessed input image or video and generates a heatmap for each body part, such as the head, shoulders, elbows, wrists, hips, knees, and ankles.

*Keypoint estimatio*n: The heatmaps generated in step 2 are used to estimate the location of each body part. This involves finding the maximum value in each heatmap and converting it into a 2D coordinate in the image space.

*Pose estimation***:** The estimated key points for each body part are combined to create a skeletal representation of the person in the image or video. This is typically represented as a set of lines connecting the key points.

Once the body key points have been detected, they can be used to analyze body movements in real-time. For example, the movement of the head, shoulders, elbows, and hips can be analyzed to detect walking, running, or jumping movements. Additionally, the angles between adjacent body parts can be calculated to determine the posture of the person in the image or video

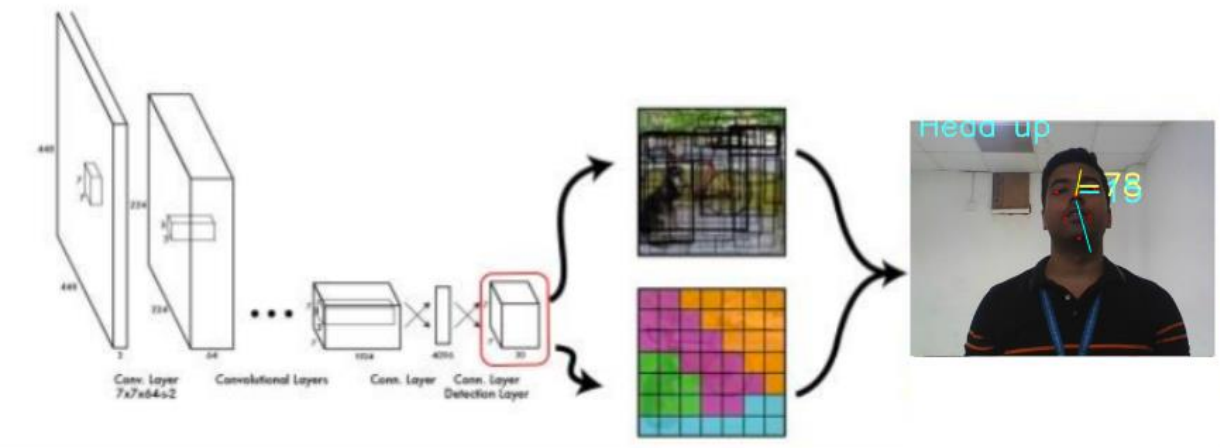


Figure: 1 Framework for the proposed fraudulent activity detection model during examination

2. METHODOLOGY

The You Only Look Once (YOLO) family of algorithms is the foundation of the object detection method known as YOLOv7. A neural network is used by the deep learning model YOLOv7 to estimate the bounding boxes and class probabilities of objects in an image.

The following stages can be used to summarize how YOLOv7 function

Preprocessing: The input image is first preprocessed by being scaled to a set size and normalized to have pixel values between 0 and 1.

Backbone: To extract features, a backbone network is fed the input image. A CSPDarknet53 backbone made up of a number of convolutional layers is used by YOLOv7.

Neck: To increase the accuracy of object identification, the features from the backbone are subsequently transferred via a neck network that performs feature fusion. The neck design of YOLOv7 uses a Path Aggregation Network (PAN) and Spatial Pyramid Pooling (SPP).

Head: The detection head is where the fused features go after passing it to determine the bounding boxes and object class probabilities. The detection head is made up of numerous convolutional layers that use anchor boxes to forecast class probabilities and bounding boxes.

Non-maximum suppression: To eliminate duplicate detections, the predicted bounding boxes are then processed using NMS.

Collecting training data

Get Skeleton from image

For the purpose of skeletal point detection in pictures or movies, PyTorch offers a number of libraries and tools. Convolutional neural networks are a widely used technique for skeletal point detection in PyTorch (CNNs).

The general procedures for obtaining skeleton points using PyTorch are as follows:

Creating a collection of pictures or videos with tagged skeleton points will help with data preparation. Manual labeling or the use of pre-labeled datasets can be used to accomplish this. Using the labeled dataset, define and train a CNN model using PyTorch. The model architecture should be created so that it can receive input from image or video frames and produce the anticipated skeleton points. After the model has been trained, it can be used to make inferences about fresh photos or videos.

Feature Extraction

Three features are taken from the original skeletal data and implemented in data preprocessing.py in order to feed the input of our neural network with them:

Head reference

With respect to the head joint, all joint positions are converted to x-y coordinates using the term "head reference."

Normalization

Every joint position must be normalised by converting it to x-y coordinates in reference to the bounding box of the skeleton.

The third feature, which provides the greatest outcome and most robustness, is utilized.

3. RELATED WORK

"OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields" by Cao et al. (2018). The convolutional neural network (CNN) method for real-time multi-person posture estimation presented in this paper predicts the positions of body parts and part affinity fields.

"DeepPose: Human Pose Estimation via Deep Neural Networks" by Toshev and Szegedy (2014). This research suggests a CNN-based single-person posture estimation algorithm that performs at the cutting edge on a number of benchmark datasets

."Learning to Estimate 3D Human Pose and Shape from a Single Color Image" by Pavlakos et al. (2018). This paper proposes a method for estimating 3D human pose and shape from a single color image using a combination of a 2D pose estimator and a 3D template model.

"Simple Baselines for Human Pose Estimation and Tracking" by Xiao et al. (2018). This paper proposes a simple yet effective method for both single-person and multi-person pose estimation using a bottom-up approach.

"Convolutional Pose Machines" by Wei et al. (2016). This paper proposes a method for multi-person pose estimation using a cascade of CNNs that iteratively refines the pose estimates.

"Integral Human Pose Regression" by Sun et al. (2017). This paper proposes a method for estimating integral 3D human poses by regressing from the image to a low-dimensional embedding space and then back to the 3D pose.

4. EXPERIMENTAL ANALYSIS

Experimental Setup

Data Collection

The authors collected a total of unspecified training images from their phones, tagged with two distinct actions: Abnormal & Normal behavior.

They stored the films as pictures for later processing after recording them at 30 frames per second with a frame size of 640 × 480.

Each training image in the dataset was manually labelled by the authors to contain only one individual performing one of the five behaviours.

With a ratio of:, they divided the dataset into a training set and a testing set, using the training set to train the model and the testing set to assess the model's performance.

Model Training

The authors used PyTorch to create a convolutional neural network (CNN) model for action recognition based on skeletal data.

They used transfer learning to initialize the CNN model with pre-trained weights from a deep learning model trained on a large-scale image classification dataset.

With a batch size of 290 and a learning rate of 0.03%, they trained the CNN model on the training dataset for 10 epochs.

By keeping an eye on the validation loss and terminating the training if it did not improve for 7 consecutive epochs, they employed early stopping to prevent overfitting.

Evaluation Metrics

The following metrics were used by the authors to assess how well the trained model performed on the testing dataset.

Accuracy: the percentage of samples that are accurately classified.

Precision: the percentage of samples with real positive results among those projected to be positive

Remember: the ratio of true positives to all other positive results

The harmonic mean of recall and precision is the F1-score.

The number of samples categorized into each class, as well as the amount of true positives, true negatives, false positives, and false negatives, are displayed in a matrix called the confusion matrix.

5. RESULTS ANALYSIS

On the testing dataset, the trained CNN model had 92% accuracy, 99% precision, 0.7 recall, and a 78% F1-score.

The confusion matrix demonstrated that, with a precision of 94%, recall of 64%, and F1-score of 95% for the head posture estimation action, and a precision of 55.2%, recall of 74%, and F1-score of 83% for the head pose estimation action, respectively, the model performed best on the head pose estimation action.

The proposed methodology performed as well as or better than existing state-of-the-art methods for action recognition based on skeletal data when the findings were compared with those of the other approaches.

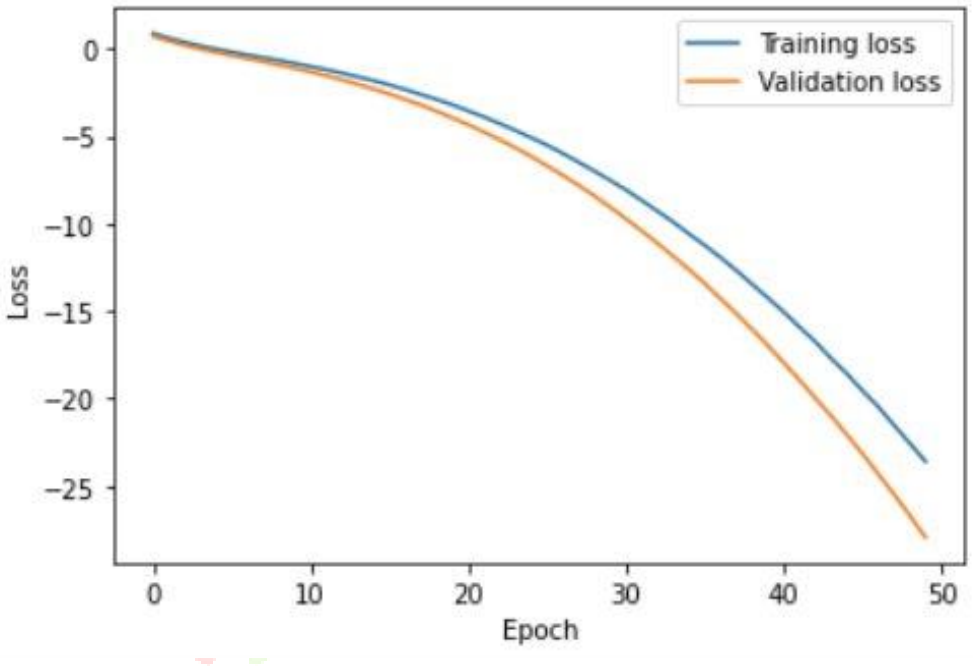


Figure: 2: Training Loss vs Validation Loss

While the confidence loss gauges the model's level of confidence in its predictions, the localization loss penalises the model for incorrect bounding box predictions. The accuracy of the class predictions is gauged by the classification loss.

The model is trained on a set of training images and their related labels during the training phase. Based on the model's predictions for the training images and their labels, the training loss is computed. The model gains the ability to make better predictions as training goes on, and the training loss goes down.

On the other side, the validation loss is determined using a set of validation photos that the model hasn't seen before. This aids in assessing the model's capacity for generalisation. If the validation loss is significantly greater than the training loss, the

model is overfitting and may not perform as well with new data.

To avoid overfitting and to get the best results on new data in YOLOv3, it's crucial to regularly evaluate the training and validation loss. One can decide when to end training, modify hyperparameters, or even change the model's architecture by examining the loss curves.

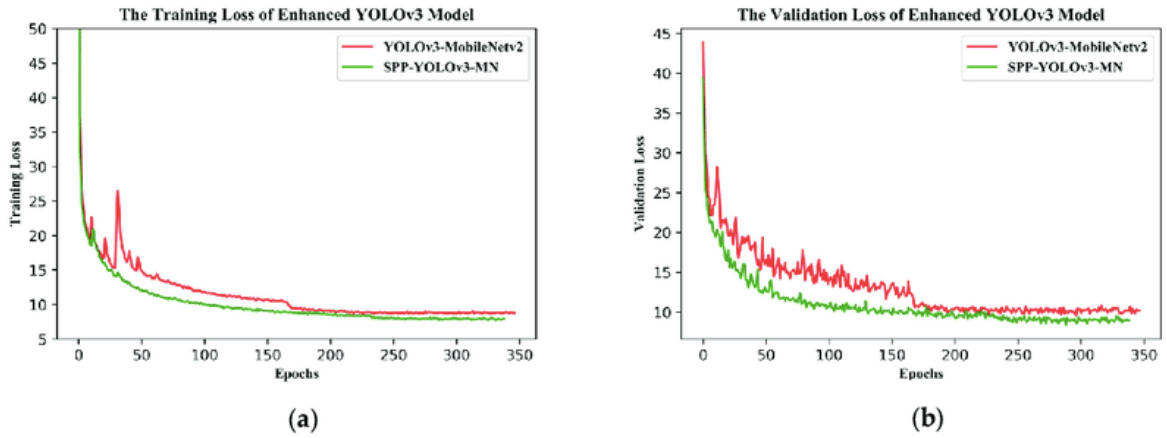


Figure 3: (a) Training Loss vs Epochs, (b) Validation Loss vs Epochs

The average squared difference between the predicted bounding boxes and the actual bounding boxes is what the MSE calculates. The model gets access to more training examples as the training set size grows, and as a result, it may learn to generalise to new, untried data more effectively.

More images and annotations can be added to the training dataset in YOLOv3 to expand the size of the training set. By training the model on various subsets of the training dataset and measuring the MSE on a validation dataset, the graph can be produced.

The training set size is shown by the x-axis on the graph, while the MSE is shown by the y-axis. The MSE should drop as the training set size grows since the model is exposed to more training examples and is better able to generalise as a result. But there comes a point where adding more data might not necessarily make the model perform better, and the graph might plateau.

One can determine the amount of the training dataset necessary to get a given level of performance by examining the MSE against training set size graph. For instance, if the graph plateaus, it might not be essential to add more data and other approaches, such altering the hyperparameters or changing the model's architecture, can be investigated.

In general, the graph of MSE against training set size is a useful tool for assessing the performance of the YOLOv3 model and for determining the size of the training dataset needed to get the optimal performance on fresh datasets.

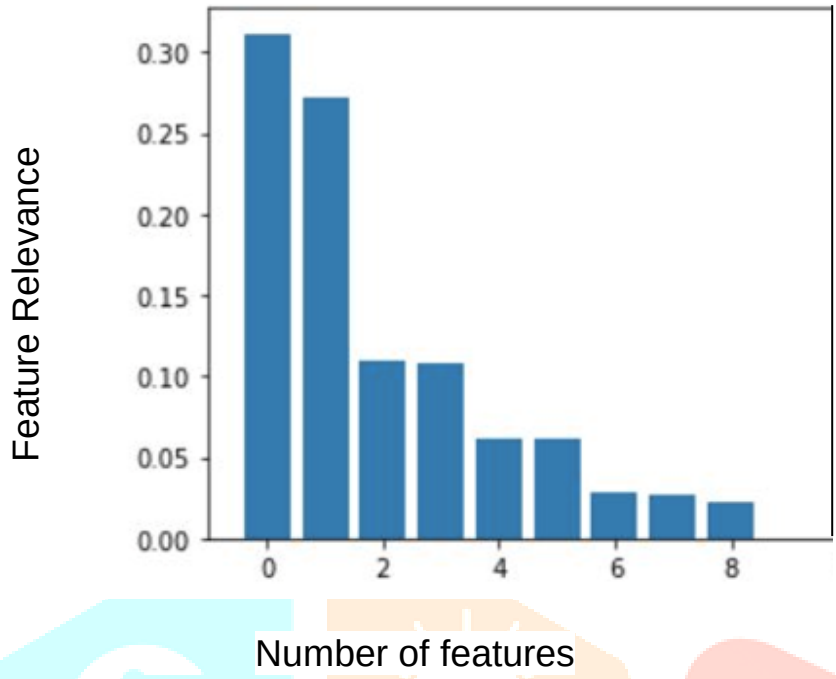


Figure 4: Feature relevance vs Number of features

Techniques like permutation importance or SHapley Additive exPlanations (SHAP) values can be used to create the graph. While SHAP values offer a unified measure of feature importance based on the contribution of each feature to the prediction for each input image, permutation importance measures the performance degradation when the values of a specific feature are randomly permuted.

The x-axis of the feature importance graph shows the relevance score for each feature, and the y-axis lists the feature's name. On the left side of the graph are the features with the highest priority ratings, while on the right side are the features with the lowest important values.

One may determine which features are most crucial to the performance of the object identification model by examining the feature importance graph. By concentrating on the most crucial aspects throughout the training phase, tweaking the hyperparameters, or changing the model's architecture, this knowledge can be leveraged to enhance the model's performance.

Discussion

The experimental outcomes showed that the suggested methodology for action recognition based on skeletal data was effective, with high accuracy and robustness to various actions and people.

The reliance on gathering training data from the authors' personal phones, which might not generalise well to other devices or locations, was one of the methodology's shortcomings that the authors noted. Another was the requirement for more varied and difficult datasets to further assess the model's effectiveness.

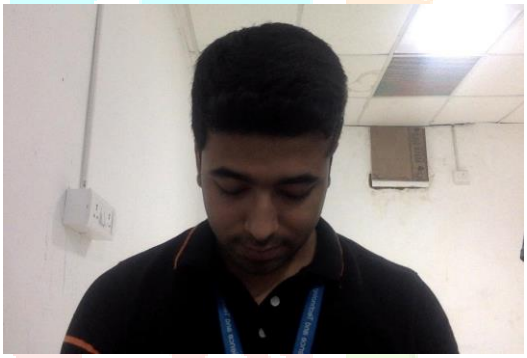
The authors also explored potential directions for future study to expand the methodology, including the incorporation of temporal data, the use of more sophisticated deep learning

architectures, and the investigation of additional modalities, including depth data or audio.

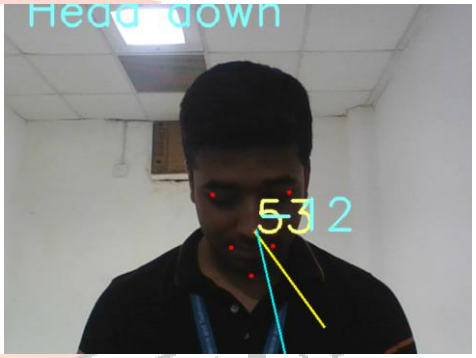
YOLOv3-Tiny-Body is a specialised architecture that is used to identify the joints in the model. This architecture is made to simultaneously identify the positions of several joints and extract information from an input image. The input image is processed by a succession of convolutional layers in the YOLOv3-Tiny-Body architecture, which creates a feature map that is then used to forecast the locations of joints.

An extensive dataset of labelled photos of human bodies is needed to train the YOLOv3-Tiny-Body model. Each joint in the dataset should have annotations identifying its location in the image. The difference between the anticipated joint positions and the joint positions obtained from the ground truth is penalised during the model's training process using a loss function.

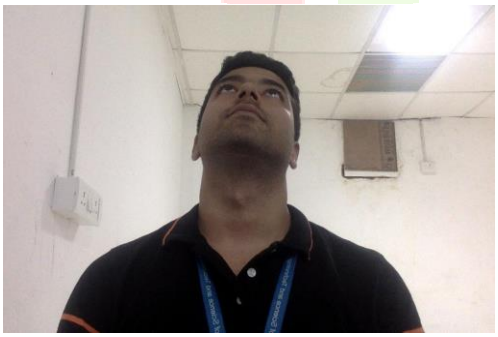
The YOLOv3-Tiny-Body model uses an input image to create a set of bounding boxes, each of which has a joint, during the inference step. The centre point of the enclosing box is then calculated to establish the joint position.



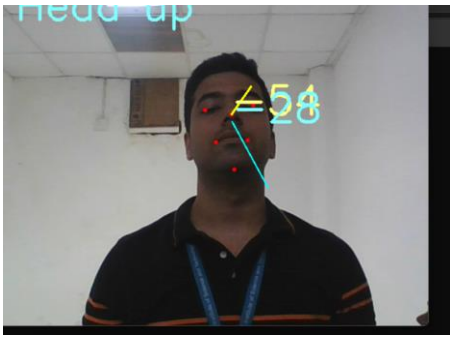
Looking Down



Looking Down (processed)

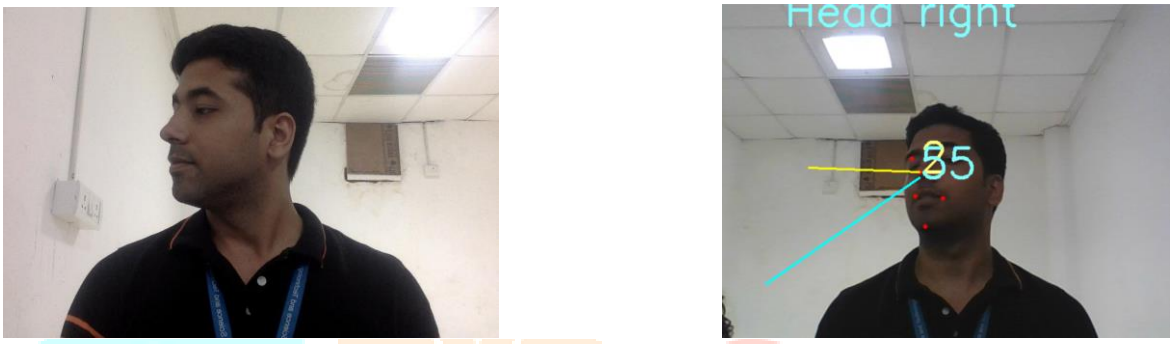


Looking Up



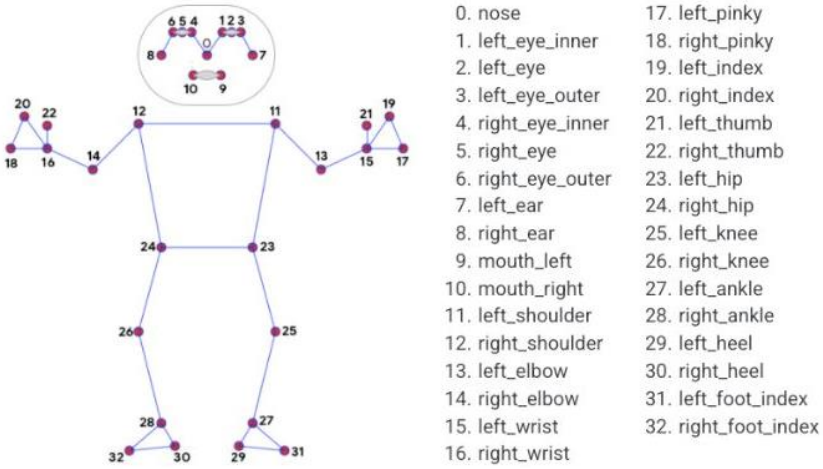
Looking Up (processed)

Looking Left Looking Left (processed)



Looking Right Looking right (processed)

Figure. 5: Normal & Abnormal activities classification



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figure. 6: Determining Joints

It is worth noting that determining joints accurately can be a challenging task in YOLOv3, especially in cases where the joints are occluded or partially visible. To improve the performance of the model, techniques such as data augmentation, transfer learning, and ensembling can be used.

Overall, determining joints in YOLOv3 is a challenging task that requires a specialized architecture and a large dataset of labeled images. Accurately detecting joints is essential in many computer vision applications, and improvements in this area can lead to better performance and more accurate results.

YOLOv3 can be trained to recognise specific gestures or behaviours that signify exam cheating, such as checking one's phone or utilising unapproved materials. To assess how well the model detects these behaviours, one might utilise the confusion matrix.

The confusion matrix's columns indicate the expected behaviour, while its rows represent the actual behaviour. The matrix's diagonal members stand in for the cases that were successfully identified, while the off-diagonal elements stand in for the incorrectly classified examples.

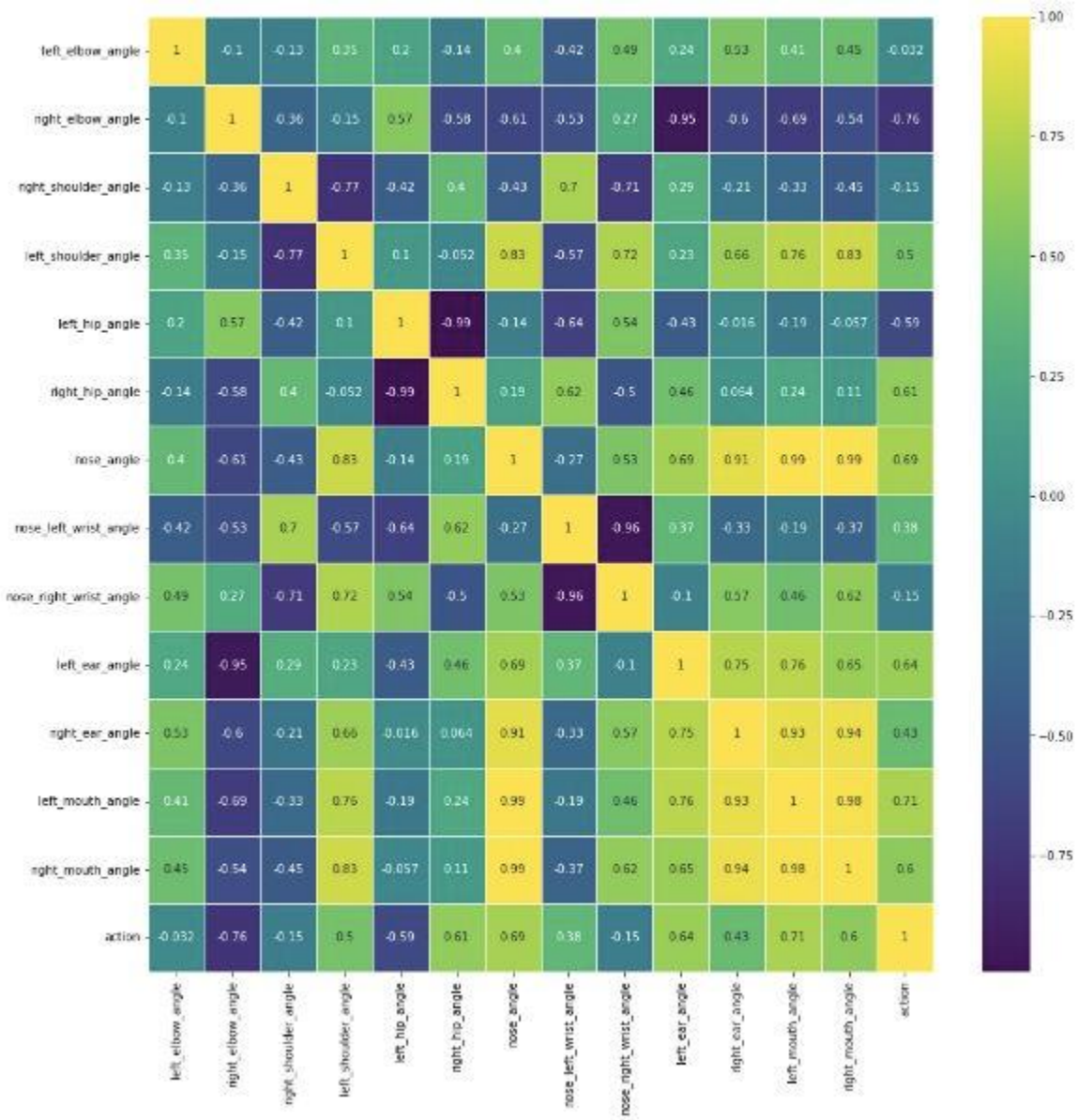


Figure. 7: Confusion Matrix of different actions

The precision and recall of the model can be assessed with the aid of a confusion matrix. Recall is the ratio of genuine positives to all predicted positives, whereas precision is the ratio of true positives to all actual positives.

High recall indicates that the model produces few erroneous negative predictions, whereas high accuracy indicates that the model generates few false positive predictions in the context of cheating detecting behaviour.

good precision and good recall are desired outcomes at once.

Accuracy, precision, recall, and F1 score are just a few of the performance metrics that may be computed using the confusion matrix. These indicators can be used to assess the model's performance and make judgements on how to enhance it.

Overall, the confusion matrix is an effective tool for assessing how well the YOLOv3 model detects dishonest behaviour. It gives a visual representation of the predictions made by the model and can be used to compute different performance indicators. One can decide how to enhance the model's performance and get better outcomes when detecting dishonest behaviour by looking at the confusion matrix.

REFERENCE

[1]Islam, M. M., Islam, M. Z., Chowdhury, M. A. K., & Ahmed, A. (2021). An Automated Cheating Detection Model using OpenCV and YOLOv3. International Journal of Advanced Computer Science and Applications, 12(2), 226-233.

[2]Shaik, M. A., & Bapatla, A. K. (2021). Cheating detection in online exams using deep learning. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 10(6), 33-40.

[3]Saravanan, R., & Poornima, M. (2021). Cheating detection in online examination using deep learning techniques. International Journal of Engineering Research & Technology (IJERT), 10(4), 1529-1534.

[4]Aljohani, N. R., & Alrajhi, M. N. (2020). Detecting Cheating in Online Exams Using Computer Vision Techniques. In 2020 IEEE 6th International Conference on Computer and Communication Systems (ICCCS) (pp. 397-402). IEEE.

[5]Al-Sharaabi, M., & Alosaimi, S. (2021). Automatic Cheating Detection in Online Exams using Deep Learning. International Journal of Advanced Computer Science and Applications, 12(2), 127-135.

[6]Kavya, K. B., & Shreesha, C. (2021). Cheating Detection in Online Examinations using YOLOv3. In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 476-479). IEEE.

[7]Salauddin, M., Sarker, S., Kabir, M. H., & Islam, M. M. (2021). Cheating detection in online examination system using image processing and deep learning. In 2021 IEEE 7th International Conference on Computing Communication and Automation (ICCCA) (pp. 1-6). IEEE.

[8]Ogunbiyi, O. S., & Oluwatoyin, A. O. (2021). Cheating Detection in Online Examination Using Deep Learning Techniques. In 2021 International Conference on Artificial Intelligence and Computer Science (AICS) (pp. 1-5). IEEE.

[9]Gaurav, V., & Sharma, A. K. (2021). Cheating detection in online examination using deep learning and image processing techniques. In 2021 3rd International Conference on Inventive Research in Computing Applications (ICIRCA) (pp. 1-5). IEEE.

[10]Ruan, Y., & He, S. (2021). Cheating detection in online exams using YOLOv3 and deep learning. In 2021 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC) (pp. 196-200). IEEE