Automated Proctoring in Physical Exams by implementing a Machine Learning Algorithm

Meeuwis, Wouter	Melaet, Robbie	Rutten, Niek	Thijssen, Yorn	Tiemens, Ronald
1338390	1228955	1240193	1342320	1633775
Eindhoven University of Technology	Eindhoven University of Technology	Eindhoven University of Technology	Eindhoven University of Technology	Eindhoven University of Technology
w.p.a.meeuwis@student.tue.nl	r.melaet1@student.tue.nl	n.j.e.rutten@student.tue.nl	y.j.thijssen@student.tue.nl	r.tiemens@student.tue.nl

ABSTRACT

In the past decades the occurrence of academic cheating has increased enormously. Academic dishonesty decreases the trustworthiness of academic diplomas, which makes it important to tackle this problem. In this paper we propose a solution for supporting invigilators with their task of supervising students during physical exams. We propose an automated proctoring tool by implementing a supervised machine learning algorithm with intuitive textual and visual communication to the invigilator (XAI). A test/train evaluation shows that the proposed system has an accuracy of 94,4%. Therefore, this might serve as a realistic and helpful solution for academic cheating in physical exam situations.

Keywords

Academic Dishonesty; Supervised Learning Algorithm; Machine Learning, Explainable AI (XAI); Supervising; Invigilators.

1. INTRODUCTION

1.1 The Problem

A university learning experience is not just about having a qualification, it is also about the journey of the student, their acquisition of skills, expertise and development of competencies (Baijnath & Singh, 2019). However, the outcomes and expertise of a student are most of the times measured by some form of examination or assessment, this provides the assessors the opportunity to attain a quantitative variable of the knowledge, skills and competencies of the student (Baijnath & Singh, 2019). During these assessment moments the pressure on the students is rather high to achieve a good result, since the graduation criteria are most of the times solely focused on results and learning outcomes reflected through this examination.

Previous research by Baldwing et al. conducted a survey with almost 2500 medical students in 31 different schools and showed that 39 percent of the people has witnessed some type of cheating during examination among classmates, in the first two years of their medical education (Baldwin et al., 1996). This paper concludes that this is an alarming amount of people and can possibly undermine the trustworthiness of graduation certificates. This conclusion is supported by the research performed by Mollie K. Galloway (Galloway, 2012). In her research she gathered data from 4316 high school students. She concludes that 93% (4013) reported cheating at least once, and 26% (1122) of the students reported that they have cheated 7 or more times during high school.

Psychological research into academic dishonesty tries to identify motivational factors that people must decide to cheat. However, the results from these studies are widely ranging. Research by Wang et al. identified three most common reasons as: time restrictions for the preparation, people wanting to help each other and the effect of ‘others cheat, so I should cheat’ (Wang et al., 2015). Although the academic dishonesty rates are already high during examination moments, the problem is getting even worse due to the increase in smartphones in the world (Morgan & Whitley, 2008). During physical examinations invigilators are being used to keep an eye on the students and punish them when they are cheating. This humanized way of cheating prevention definitely has drawbacks, especially when looking at the high occurrence of cheating in current day school. Therefore, in our project we aim to use technology to support the academic integrity during physical examination moments.

In this paper, we will propose a method to support exam invigilators during physical exams. We designed a system that implements a supervised learning algorithm that is able to inform the supervisor when a student behaves abnormally and suspects cheating. The algorithm makes this decision based on visual eye recognition and tracking. A calculation is being made on how often the user looks away from their test-material. This information is communicated to the supervisor through an intuitive digital user interface as an heatmap. This interface also gives additional explanation on the decisions that are being made by the system through textual and visual explanation.

The system proposed in this paper contributes design-relevant knowledge for developing AI systems that are able to support supervisors during physical exams. Through the implementation of our system, we hope to decrease the occurrence of academic dishonesty during physical tests.

1.2 Related work

Many studies have tried to tackle academic dishonesty by developing proof-of-concepts for online test environments. Especially during the COVID-19 pandemic the majority of schools and universities had to switch to remote teaching and therefore online tests were the main manner of examination (Kamalov et al., 2021). Most of these studies developed software that is able to track the computer usage of people during tests (Cavalcanti et al., 2012; Diedenhofen & Musch, 2017; Tiong & Lee, 2021). For example, Diedenhofen and Musch (2017) developed a script that detects when participants abandon test

pages by switching to another window or browser tab. However, now that the COVID-19 pandemic is coming to an end, online examination is decreasing and shifting to on-campus examination again. However, the area of technological proctoring in on-campus examination remains underexplored.

Research by Bancud and Palconit focuses on camera detection of human pose in relation to cheating in physical, on-campus examination (Bancud & Palconit, 2021). Their system is being trained by images which are labeled by proctors. The limitation of this study is that these images need to be updated and evaluated often in order to keep the system at high accuracy. The cheating detection accuracy of the designed system might decrease when students find ways to cheat without changing their posture, which is the biggest drawback of their research. We hope to develop a proof-of-concept that is able to support the cheating assessment of a proctor in a smaller scale, and therefore decreasing the chance of cheating the anti-cheating system.

Furthermore, Justin Thomas and Adam Jeffers developed a proof-of-concept study on mobile eye tracking in order to support academic integrity (Thomas & Jeffers, 2020). Using smart glasses with an integrated camera they were able to identify what a person was looking at during examination. They deployed the camera at three volunteering students during an examination. After the exam, the authors assessed the recordings on whether cheating has occurred. Whereas this probably is a trustworthy method of assessing whether students cheated, the process of watching all videos is very time consuming if the student quantity increases. Therefore, automated classification of the looking behavior is desired. Besides, the authors identify that the glasses are expensive and fragile (Thomas & Jeffers, 2020). Therefore, an alternative, more stable and firm solution is desired.

2. Methods and materials

2.1 Explain your approach/specific methods or theory.

The challenges that educational organizations face to prevent cheating during physical exams on campus take place in a multi stakeholder environment. Given that the number of invigilators that can reasonably be employed and placed in exam halls is limited, other options have to be considered to enable invigilators to work more efficiently and effectively.

Automating the invigilation process can be considered with the use of intelligent systems that watch, analyze and determine if cheating behavior has taken place. However, the use of AI empowered systems to take on the role of invigilators faces a number of concerns and technical limitations. The desirability of an automated making binding decisions and judgements on students taking a test is questionable. A system might make incorrect decisions, identifying regular behavior as suspicious or cheating, with serious consequence for the students involved. Furthermore, to make it possible for students to dispute a decision made by an automated system the data that the system uses needs to be logged somewhere to be reviewed. This possess issues of privacy and ownership of personal data. Lastly the knowledge that a system is constantly watching and making binding decisions based on its observations can be very obtrusive.

The question that we then aim to answer is how can we create a system that can help reduce cheating taking into account these limitations. We aim to answer this question by designing a system that takes into account the needs of the various people involved in the on-campus examination. We note that the automated classification of looking behavior is desired, that invigilators have to be the main actors in decision making and that an AI empowered agent for automation of tasks cannot store personal information. The system formed in this design space is system that classifies looking behavior and relays this information to invigilators to notify them of places in the exam hall that deserve extra attention.

We approach the design of these two elements, classifying and relaying information, with the aforementioned limitations in mind. For the design of the classifying element, we use a supervised learning algorithm to create a model that can determine where a student is looking by analyzing a video feed. Because collection of personal data is not desired the algorithm does not learn during the interaction when the model is deployed. The model as such uses offline learning. For the design of the relaying of information to the invigilator we employ explainable artificial intelligence (XAI). XAI helps invigilators not only seeing the output of the classification of suspicious behavior but provides the invigilator with the reason why this classification was made. The use of the algorithm and design of the system is further elaborated in this paper.

2.2 Learning algorithm

The application of machine learning in our system is twofold. Firstly, the system utilizes the OpenCV (Intel, 2021) and dLib (King, 2021) libraries. In our case these libraries make use of a Support Vector Machine Algorithm (a type of supervised learning algorithm) to detect faces and 68 facial landmarks within those faces in an image. The specific algorithm applied can be found in the 2001 paper by Felzenszwalb et al. included in the references. Using the camera feed of a webcam as an input, the libraries output the coordinates of the 68 facial landmarks using code based on code by Canu (2019). This data is used as an input of a second supervised learning algorithm. For efficiency's sake we only use the coordinates of the landmark associated with the tip of the nose as input data, as this was enough to acquire accurate results. The landmarks associated with the left and right sides of the jaw are however used to rescale the landmark coordinates to one universal size. Regardless of how large the face appears on the image; the system resizes the landmark data such that the distance between the leftmost and rightmost jaw points is fixed. The system also repositions the landmarks so the rightmost jaw point is at the window's origin (0,0) (figure 1). The x and y coordinates of the nose landmark are then passed on to the second learning algorithm.

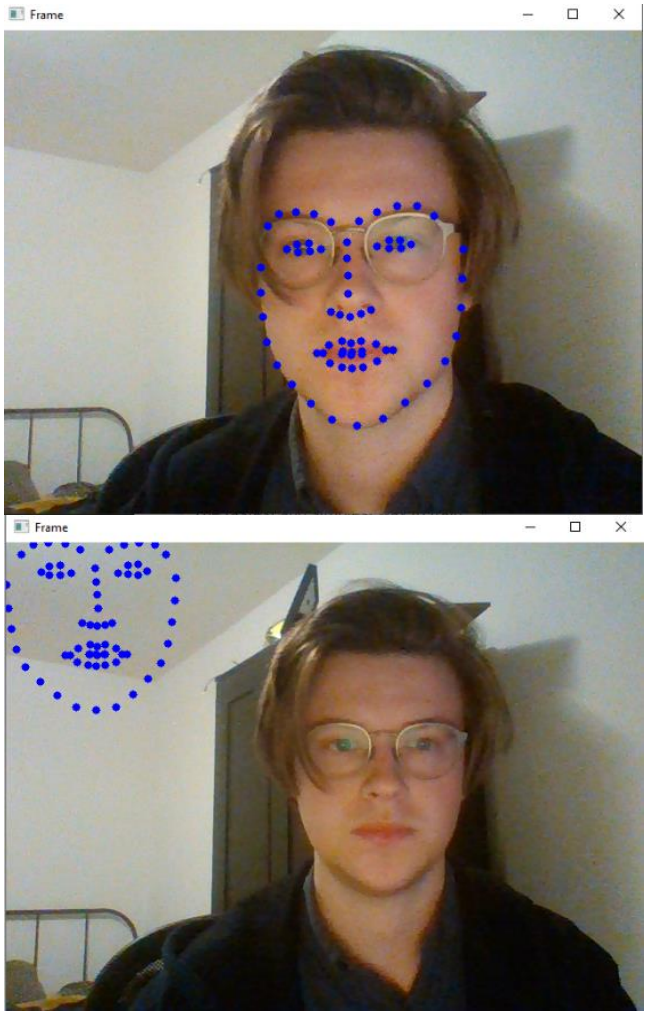

Figure 1. facial landmarks before (left) and after resizing and repositioning (right)

This algorithm is another supervised learning algorithm, specifically a Random Forest Classification algorithm. This algorithm was chosen since detecting the direction of the head is a classification problem and the Random Forest Classifier came out of our tests, based on code by Tsiakas (2021), as the most accurate. The algorithm was trained with a dataset which was generated using the first algorithm. A camera feed was provided to the SVM algorithm as well as a label: either right, center or left in numerical form (1, 2 and 3 respectively). A person facing the webcam then looked in the direction of the currently selected label and using a keypress the coordinates of the nose landmarks are saved into a pandas dataframe. A bunch of datapoints are recorded for all three directions, with the person facing the webcam slightly varying their gaze for each datapoint. The dataframe is then exported into a CSV which the final system uses to train its Random Forest Classifier. This final system uses the real time output of the first algorithm (which uses the real time camera feed from the webcam) to estimate which direction a person is looking at, thus providing our system with the data we desire (figure 2).

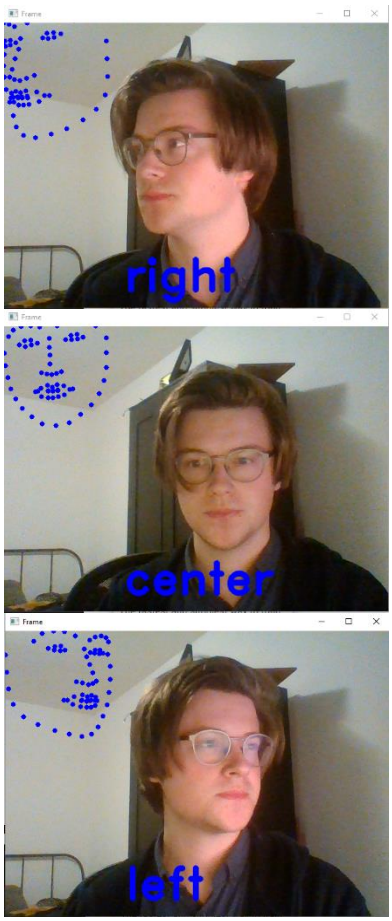

Figure 2. The three looking directions

3. Describe your design or solution

3.1 Design of the interaction

The system is developed to reduce the number of cheating students during physical exams. To achieve this, a specific goal of interaction had to be designed. This goal of interaction is to let the system communicate when students act suspiciously. Even though the system is developed to stop students from cheating, the invigilators are the actual agents that interact with the system during exams and thus are the agents where the system should communicate with.

The communication between the system and the invigilators happens through a UI (User Interface). The UI communicates towards the invigilator when students are acting suspicious. Students are classified as suspicious when their face is not facing the test material. Several XAI (Explainable Artificial Intelligence) features are used within this communication. Through both visual and textual ways, the invigilator is shown who and specifically why a specific student is classified as suspicious.

The exam room, where the physical exam takes place, is visualized on the UI (figure 3). This means that all the desks in the exam room are visible on the UI. At the beginning of the exam, all the desks are colored white. The moment a student looks away from the test material, the correlated desk of that student starts coloring red (figure 4). The more a student looks away from the test material, the redder the desk of that student becomes on the UI. The invigilator can hoover over the desks with his cursor.

When this happens, the UI shows the number of times the student of that specific desk looked away from the test material in a textual way (figure 5).

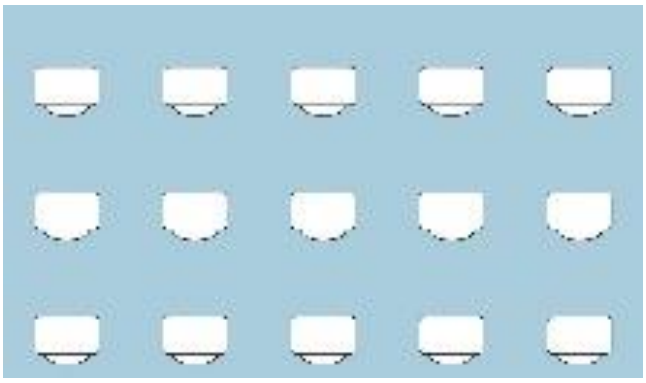

Figure 3. Visualization of classroom on the User Interface (UI).

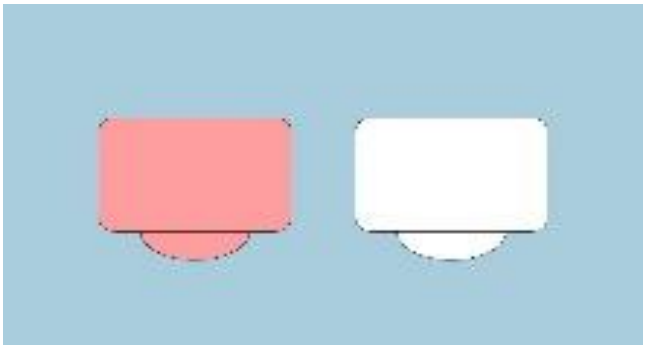

Figure 4. Color changes when student looks away.

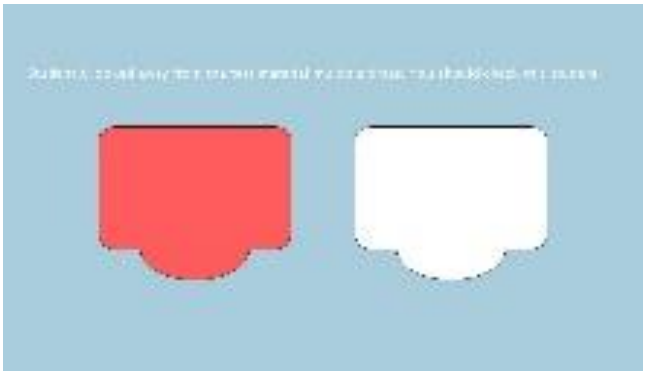

Figure 5. Textual explanation after hoovering over a table (XAI).

The system is designed to warn the invigilator when a student acts suspiciously. This warning should encourage the invigilator to keep an eye on certain students and to address these students when necessary. For ethical reasons, it was decided not to warn the students directly and, in this way, make them the agents that interact with the system. Communicating directly to students can distract them while taking the exam. Besides that, it can increase the feeling of nervousness. The system will make the work of invigilators more efficient, and it will ensure students will cheat less frequently during exams. For this reason, the system will bring value to society.

The communication between the AI proctoring system and the UI is done with OOCSI. OOCSI is a prototyping middleware that enables communication between multiple systems.

3.2 Intelligent behavior and embodiment.

As was just described the embodied intelligence is placed at the center of the invigilating routine. Physically this agent is present in the form of a collection of cameras and the UI the invigilator user to interact with it. Virtually it is present as a decision-making model that is fed live video and transforms this input into an array of classifications of looking behavior. These classifications are made by determining the direction a person looks in a video feed. The classification of direction is then used to classify behavior of a person as either suspicious, not looking at their own desk, or unsuspicious, looking at their desk, based on the amount of time a person shows either of these behaviors. Bringing this last classification from the system to the invigilator can be seen as the main function the agent fulfills in the interaction. However, the agent is acting throughout more of the interaction then the fulfillment of its main function as the explanation through the UI provides transparency by given the invigilator insight in the elements the system bases its decision on, time and direction of not looking at desks. This extra layer is paramount for the functioning of the system as a whole. Invigilators can see with the tool what areas of the exam hall deserve extra attention but can dismiss advice on review of their environment.

The learning algorithms previously mentioned in this paper are used to train our model to be able to show this intelligent behavior with live input data. The designed model was thought to recognize and map landmarks of faces using an existing model and then thought to classify the direction of a person's head. This training is offline as storing the data of students during examinations is deemed as undesirable making a designed model that learns during the interaction unlikely. This could however be made desirable if ethical concerns regarding privacy can be catered to.

The presentation of the classifications made by the deployed model is designed with the main goal of the design in mind, pointing the invigilator to where suspicious behavior may be observed. The final translation of the model's output, going from classification of head direction to suspicious behavior based on observed directions over time, makes for a usable system that enables this. The translation of the classification gives other desirable qualities as well such as the robustness of the system to error resulting from a wrong classification and the differentiation of areas where a lot of suspicious activity has happened, making the system more useful.

3.3 Testing and analysis

To evaluate the system, we use a test dataset to calculate the overall accuracy of the system as well as a confusion matrix (figure 6). Using the train_test_split method from the SKlearn library we split our original training dataset into a training dataset (66% of the original dataset) and a test dataset (33% of the original dataset). Each entry in the original training dataset is randomly assigned to either the test or the training dataset. The Random Forest Classifier is then trained using the new training dataset. Subsequently it is tasked with classifying the entries in the test dataset. The algorithm's classification of each entry in the test dataset is then compared to that entry's original label to

calculate the accuracy of the algorithm. Using this method an overall accuracy of 94.4% was calculated. Furthermore, the confusion matrix tells us only one entry was incorrectly classified, namely one image where the person was looking to the right was classified as the person looking to the center. These results are more than acceptable for our purposes, especially considering the results are cumulative since it only matters whether students have been looking in one direction for an extended period. Thus, a single incorrect classification is less impactful as it will be overruled by a majority of correct classifications.

```
RF Accuracy: 0.9444444444444444
[[7 1 0]
 [0 6 0]
 [0 0 4]]
```

Figure 6. Accuracy and Confusion Matrix

4. Discussion

The goal of the system, as described earlier, is to support invigilators during physical exams by means of using a learning algorithm and XAI. We do not know whether the system would actually support invigilators. However, we have designed a system that successfully detects whether a person is looking to the *right*, *center* or *left*. Moreover, this has also been linked to a visual interface that can support invigilators in doing their job. Therefore, we can state that, within the given timeframe, we successfully created a system that uses ML and XAI which in the future can be used by invigilators to support them with supervising during physical exams.

Our work does cover the biggest limitation of the work of Bancud and Palconit (2021). Their solution was not able to recognize whether students are cheating while not changing their body posture. Since our ML algorithm uses facial landmarks, this is no limitation anymore. Moreover, our algorithm does not need images and does not need to be updated from time to time.

Furthermore, our work covers two major limitations of the work of Justin Thomas and Adam Jeffers (2021). For their solution, they had to watch the recorded footage of the students making the exam, to determine whether a student had cheated or not. Our proposed solution automatically evaluates the live feed of the student. So, it is not needed to watch footage after the exam, which thus saves time. Besides, in our solution, no video is recorded. Moreover, our proposed solution does not require students to wear a wearable.

There are however several limitations to the current design. Firstly, because of the time constraint, the learning algorithm was trained to classify students only in three classes. For a real exam setting, this would not be enough to determine whether a student is cheating. However, it shows that a machine learning algorithm can be made to perform this task. Secondly, no physical prototype was made. It has yet to be determined how such an exam table set-up including a camera would look like. Thirdly, no tests were performed in a real exam room setting. Therefore, we cannot state whether a camera placed on an exam table has influence on the students and thus their exam results. Future work should therefore, amongst other things, focus on the influence of the presence of a camera on an exam table.

Apart from investigating the influence of the presence of a camera, there is more future work yet to be done before such a system could be implemented. First of all, a future system should categorize more classes than the current three. This would be needed to categorize students more specifically whether they are cheating or not. A student could also look up and to the left, and thus not cheating, while in the current system the student would be classified as prone to cheating. To build on that, an invigilator must be able to easily indicate on specific tables which classes are said to be prone to cheating. With this we mean that an invigilator must be able to tell the system that for instance for a specific row of tables, looking to the right is not cheating, because a window is located there. In this case the system would not color these tables if these students are harmlessly looking outside the window. To create this feature for invigilators, future work should also consider elaborating on the (design of the) UI. It should be made easy and intuitive for invigilators to interact with the UI.

5. Conclusions

This paper proposes a solution that supports invigilators in their work by using ML do detect whether a student is prone to cheating, and by making use of XAI to make it understandable for the invigilator. A machine learning algorithm was made that detects whether a student is looking to the left, center or right, and classifies this into the corresponding classes. In addition, a UI was made that supports the invigilator by using Explainable AI in the form of color coding and visual text, to make the classification of a student understandable. Within the given timeframe, we can state that we successfully created a system with respect to embodying intelligent behavior in social context. Specifically, our work shows to have potential in the educational context. Future work should elaborate on the working of the system in order to be determined whether it actually supports invigilators in their work.

6. REFERENCES

Baijnath, N., & Singh, D. (2019). Examination cheating: Risks to the quality and integrity of higher education. South African Journal of Science, 115. https://doi.org/10.17159/sajs.2019/6281

Baldwin, D. C. J., Daugherty, S. R., Rowley, B. D., & Schwarz, M. D. (1996). Cheating in medical school: A survey of second-year students at 31 schools. Academic Medicine, 71(3), 267–273.

Bancud, G. E., & Palconit, E. (2021). Human pose estimation using machine learning for cheating detection. https://doi.org/10.13140/RG.2.2.12686.28481

Canu, S. (2019). face-landmarks-detection-opencv-with-python.[Python]. https://pysource.com/2019/03/12/face-landmarks-detection-opencv-with-python/.

Cavalcanti, E. R., Pires, C. E., Cavalcanti, E. P., & Pires, V. F. (2012). Detection and Evaluation of Cheating on College Exams using Supervised Classification. Informatics in Education, 11(2), 169–190. https://doi.org/10.15388/infedu.2012.09

Diedenhofen, B., & Musch, J. (2017). PageFocus: Using paradata to detect and prevent cheating on online achievement tests. Behavior Research Methods, 49(4), 1444–1459. Scopus. https://doi.org/10.3758/s13428-016-0800-7

Felzenszwalb, P. F. Girshick, R. B. McAllester, D. and Ramanan D., (2010, Sep) "Object Detection with Discriminatively Trained Part-Based Models," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627-1645, doi: 10.1109/TPAMI.2009.167.

Galloway, M. K. (2012). Cheating in Advantaged High Schools: Prevalence, Justifications, and Possibilities for Change. Ethics & Behavior, 22(5), 378–399. https://doi.org/10.1080/10508422.2012.679143

Intel. (2021). OpenCV library (Version 4.5.4). [library] https://opencv.org/releases/

Kamalov, F., Sulieman, H., & Santandreu Calonge, D. (2021). Machine learning based approach to exam cheating detection. PLoS ONE, 16(8), e0254340. https://doi.org/10.1371/journal.pone.0254340

King, D.E. (2021). Dlib c++ library (Version 19.22). [library]. http://dlib.net/

Tsiakas, K. (2021). supervised-learning-1b. [Python]. https://canvas.tue.nl/files/3409965/download?download_frd=1.

Morgan, M., & Whitley, H. (2008, April 7). Academic dishonesty among pharmacy students: Does technology have a role?

Thomas, J., & Jeffers, A. (2020). Mobile eye tracking and academic integrity: A proof-of-concept study in the United Arab Emirates. Accountability in Research, 27(5), 247–255. https://doi.org/10.1080/08989621.2019.1646645

Tiong, L. C. O., & Lee, H. J. (2021). E-cheating Prevention Measures: Detection of Cheating at Online Examinations Using Deep Learning Approach -- A Case Study. ArXiv:2101.09841 [Cs]. http://arxiv.org/abs/2101.09841

7. Appendix

A.1 Python code for test data generation

'''

Code to generate a test dataset for

Written by Niek Rutten for the Embodying Intelligen Behavior in Social Contexts course

2020 TU/e

Based on:

Canu, S. (2019). face-landmarks-detection-opencv-with-python.[Python]. https://pysource.com/2019/03/12/face-landmarks-detection-opencv-with-python/.

and

Tsiakas, K. (2021). supervised-learning-1b. [Python]. https://canvas.tue.nl/files/3409965/download?download_frd=1.

'''

#code is largely the same as realtime predictions, differences are commented

```
import cv2
import numpy as np
import dlib
import pandas as pd

cap = cv2.VideoCapture(0)
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")

dataCount=0
df = pd.DataFrame(columns=('noseX', 'noseY', 'direction'))
direction = 2



while True:
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = detector(gray)
    for face in faces:

        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()
        #cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 3)

        landmarks = predictor(gray, face)
        x=[0] * 68
        y=[0] * 68
        for n in range(0, 68):
            multiplier=170/(landmarks.part(16).x-landmarks.part(0).x)

            x[n] = (int((landmarks.part(n).x - landmarks.part(0).x)*multiplier))
            y[n] = (int((landmarks.part(n).y - landmarks.part(19).y)*multiplier))
            cv2.circle(frame, (x[n], y[n]), 4, (255, 0, 0), -1)

    cv2.imshow("Frame", frame)

    key = cv2.waitKey(1)
    if key == 27:
        break

    if key == ord('s'): #save the current datapoint in a dataframe
        new_entry = {'noseX':x[33], 'noseY':y[33], 'direction':direction}
        df.loc[len(df)] = new_entry
        print(df)
        #export dataframe to CSV
    if key == ord('e'):
```

```
df.to_csv(r'C:\Users\20174718\Documents\ID\Year5\Q1\Embodying intelligent behavior in social context\test data.csv', index = False)
        print("exported!")
        #set which gaze direction is recorded
    if key == ord('r'):
        direction=1
        print("direction set to right")
    if key == ord('l'):
        direction=3
        print("direction set to left")
    if key == ord('c'):
        direction=2
        print("direction set to center" )


cap.release()
cv2.destroyAllWindows()
```

A.2 Python code for real time predictions

```
'''
Code to predict gaze direction using machine learning.
Written by Niek Rutten for the Embodying Intelligen Behavior in Social Contexts course
2020 TU/e

Based on:
Canu, S. (2019). face-landmarks-detection-opencv-with-python.[Python]. https://pysource.com/2019/03/12/face-landmarks-detection-opencv-with-python/.
and
Tsiakas, K. (2021). supervised-learning-1b. [Python]. https://canvas.tue.nl/files/3409965/download?download_frd=1.

'''

import cv2
import numpy as np
import dlib
import pandas as pd
from oocsi import OOCSI

from sklearn.ensemble import RandomForestClassifier



cap = cv2.VideoCapture(0) #video feed from webcam

detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")


sessions = pd.read_csv('test data.csv'); #load in the training data


# create training dataset - input and output -- create classes
X = sessions[["noseX","noseY"]].to_numpy(); # position of the nose
Y = sessions[["direction"]].to_numpy(); #looking direction


#training the algorithm
rf = RandomForestClassifier(n_estimators=150, criterion = 'entropy')
rf.fit(X, Y.ravel())


# variables for the text on screen
font = cv2.FONT_HERSHEY_DUPLEX
org = (200, 450)
fontScale = 2.5
color = (255, 0, 0)
thickness = 5


# connect to OOCSI running on the local machine ('localhost'), for connecting to processing
oocsi = OOCSI()




while True: #keep looping
    _, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)#turn the vidoe feed into grayscale


    faces = detector(gray) #detect faces in the image
    for face in faces:#loop trough all faces


        landmarks = predictor(gray, face) #detect landmarks for the face
        x=[0] * 68 #create variables for x and y to record landmark data
        y=[0] * 68
        for n in range(0, 68):
            multiplier=170/(landmarks.part(16).x-landmarks.part(0).x)#calculate the distance between left jaw and right jaw and calculate how much the face needs to be resized
```

```
        x[n] = (int((landmarks.part(n).x - landmarks.part(0).x)*multiplier))#record x and y data of landmarks
        y[n] = (int((landmarks.part(n).y - landmarks.part(19).y)*multiplier))
        cv2.circle(frame, (x[n], y[n]), 4, (255, 0, 0), -1)


    X_test=np.array([x[33], y[33]])#create array with data that is used to predict gaze direction
    rf_prediction = rf.predict(X_test.reshape(1,-1))#run prediction
    if rf_prediction[0]==1: #put text on screen with the predicted direction and send data to processing
            text='right';
            oocsi.send('colorChannel', {'direction': 1})
    if rf_prediction[0]==2:
            text='center';
            oocsi.send('colorChannel', {'direction': 2})
    if rf_prediction[0]==3:
            text='left';
            oocsi.send('colorChannel', {'direction': 3})
    cv2.putText(frame, text, org, font,
              fontScale, color, thickness, cv2.LINE_AA)
    #print(rf_prediction)


  cv2.imshow("Frame", frame)


  key = cv2.waitKey(1)#quit program when esc is pressed
  if key == 27:
    break



cap.release()
cv2.destroyAllWindows()
```

A.3 Python code for evaluation algorithm

```
'''
Code to evaluate the accuracy of a random forest classifier.
Written by Niek Rutten for the Embodying Intelligen Behavior in Social Contexts course
2020 TU/e

Based on:
Tsiakas, K. (2021). supervised-learning-1b. [Python]. https://canvas.tue.nl/files/3409965/download?download_frd=1.

'''

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn import tree, metrics
from sklearn.svm import SVC
import pandas as pd;

sessions = pd.read_csv('test data.csv'); #read the traning data

# create training dataset - input and output
X = sessions[["noseX","noseY"]].to_numpy();
Y = sessions[["direction"]].to_numpy();

# create training/test dataset
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.33, random_state=2);

#train algorithm
rf = RandomForestClassifier(n_estimators=150, criterion = 'entropy')
rf.fit(X_train, Y_train)
#run algorithm for test dataset
rf_prediction = rf.predict(X_test)
#print the accuracy
print("RF Accuracy:",metrics.accuracy_score(Y_test, rf_prediction))
print(confusion_matrix(Y_test, rf_prediction));
```

A.4 Processing code

```
//https://github.com/iddi/oocsi-processing

import nl.tue.id.oocsi.*;

int color1 = 255;
int color2 = 255;

int increase = 2;

String facingDirection = "right";

boolean suspicious = false;


void setup() {
  size(1000, 600);
```

```
  background(#AACDDE);
  textSize(20);

  // connect to OOCSI server running on the same machine (localhost)
  // with "receiverName" to be my channel others can send data to
  // (for more information how to run an OOCSI server refer to: https://iddi.github.io/oocsi/)
  OOCSI oocsi = new OOCSI(this, "EIBISC_GROUP2", "localhost");
  // connect to OOCSI server running at the adress oocsi.example.net:
  //OOCSI oocsi = new OOCSI(this, "unique_name", "oocsi.example.net");

  // subscribe to channel "testchannel"
  // either the channel name is used for looking for a handler method...
  oocsi.subscribe("colorChannel");
  // ... or the handler method name can be given explicitly
  // oocsi.subscribe("testchannel", "testchannel");
}

void draw() {
  background(#AACDDE);
  //tafel links
  fill(color1, color2, color2);
  rect(150, 200, 300, 200, 28);
  arc(300, 400, 170, 100, 0, PI, CHORD);

  //tafel rechts
  fill(color1);
  rect(550, 200, 300, 200, 28);
  arc(700, 400, 170, 100, 0, PI, CHORD);

   if (mouseX > 150 && mouseX<450 && mouseY > 200 && mouseY <400 && suspicious==true) {
   text("Student X is looking to their " + facingDirection + " a lot. You should check this student.", 162, 120);
  } else {
   text(" ", 40, 120);
  }
}

void colorChannel(OOCSIEvent event) {


  int direction = event.getInt("direction", 0);
  if(direction==1){
   color2-=increase;
   if(color2<0){color2=0;}
   facingDirection="right";
   suspicious=true;
  }
  if(direction==3){
   color2-=increase;
   if(color2<0){color2=0;}
   suspicious=true;
   facingDirection="left";
  }
  if(direction==2){
   color2+=increase;
   if(color2>255){color2=255;}
   suspicious=false;
  }
  //println(direction);
  //println(color2);
}
```

A.5 Link to video

https://youtu.be/LoICO9QNcvY