Alerta - An application to detect fatigue and drowsiness signs among truck drivers.

Saturday 18th November, 2023 - 14:00

Bacem Etteib
University of Luxembourg
Email: bacem.etteib.001@student.uni.lu

This report has been produced under the supervision of:
Nicolas Guelfi
University of Luxembourg
Email: nicolas.guelfi@uni.lu

1. Plagiarism statement

This 350 words section without this first paragraph must be included in the submitted report and placed after the conclusion. This section is not counting in the total words quantity.

I declare that I am aware of the following facts:

- As a student at the University of Luxembourg I must respect the rules of intellectual honesty, in particular not to resort to plagiarism, fraud or any other method that is illegal or contrary to scientific integrity.
- My report will be checked for plagiarism and if the plagiarism check is positive, an internal procedure will be started by my tutor. I am advised to request a precheck by my tutor to avoid any issue.
- As declared in the assessment procedure of the University of Luxembourg, plagiarism is committed whenever the source of information used in an assignment, research report, paper or otherwise published/circulated piece of work is not properly acknowledged. In other words, plagiarism is the passing off as ones own the words, ideas or work of another person, without attribution to the author. The omission of such proper acknowledgement amounts to claiming authorship for the work of another person. Plagiarism is committed regardless of the language of the original work used. Plagiarism can be deliberate or accidental. Instances of plagiarism include, but are not limited to:

    1) Not putting quotation marks around a quote from another persons work
    2) Pretending to paraphrase while in fact quoting
    3) Citing incorrectly or incompletely
    4) Failing to cite the source of a quoted or paraphrased work
    5) Copying/reproducing sections of another persons work without acknowledging the source
    6) Paraphrasing another persons work without acknowledging the source
    7) Having another person write/author a work for oneself and submitting/publishing it (with permission, with or without compensation) in ones own name(ghost-writing)
    8) Using another persons unpublished work without attribution and permission (stealing)
    9) Presenting a piece of work as ones own that contains a high proportion of quoted/copied or paraphrased text (images, graphs, etc.), even if adequately referenced.

Auto- or self-plagiarism, that is the reproduction of (portions of a) text previously written by the author without citing that text, i.e. passing previously authored text as new, may be regarded as fraud if deemed sufficiently severe.

2. Introduction

How could we use deep learning to recognize fatigue signs among drivers in order to prevent vehicle crashes?

In 2020, almost 4,014 died in large fatal truck crashes[1]. The main goal of this project is to replace traditional ways of detecting fatigue, a primary cause of vehicle crashes, by a modern solution based on AI. While methods such as measuring distance of eyelid and eye tracking have proven their inefficiency, Computer Vision(C.V) seems to hold a great promise in achieving better results[2]. Artificial Intelligence could be defined as a simulation of how the human learning process works. It refers to the ability of a machine to executes certain tasks without being explicitly programmed to do them. Machine Learning, a subset of AI, is the technique of learning patterns from a given dataset. The latter corresponds to a

collection of relevant data points that could be treated by a computer. In deep learning, a subset of machine learning, we often make use of Convolutional Neural Networks(CNN). A CNN is a type of neural networks designed specifically to process visual inputs. The focus on detecting fatigue signs and not other factors in vehicle crashes is mainly because of the high correlation between the possibility of a road accident and the driver's consciousness state. In the majority of cases, driving while feeling tired or dizzy is associated with a fatal crash risk[7].

3. Project Description

3.1. Domains

3.1.1. Scientific

. In the scientific part of this project, we focus mainly on working with Artificial Intelligence and more precisely deep learning, a subset of Machine Learning. Therefore, we provide a scientific description of what is deep learning, how it is used in general and how is adapted to our problem. We explore various techniques used in deep learning and the possibility to integrate them with our solution.

3.1.2. Technical

. The technical domains include the design and development of a Convolutional Neural Netowrk by going through the different phases of preprocessing, training and evaluation. They also cover the usage of Python, a dynamic programming language to successfully build the CNN. Therefore, we expect certain pre-requisites from the reader that would be indicated in the following sections.

3.2. Targeted Deliverables

3.2.1. Scientific deliverables

. *How could we use deep learning to recognize fatigue signs among drivers in order to prevent vehicle crashes?*
scientific deliverables of this project are; to define what is deep learning, understand the types of neural networks and mainly the convolutional neural network and to discuss the possible implementations of pre-trained models and custom data sets to produce an automated system for fatigue detection. Thus, we cover in details, using the available references, the different phases of building an accurate deep learning model. We also approach alternatives to designing deep learning models from scratch such as the usage of pre-trained models.

3.2.2. Technical deliverables

. After learning about different deep learning models, we focus in the technical deliverable on the implementation of a Covolutional Neural Network with a classification model (Random Forest), to recognize fatigue signs in both a visual input and a set of information related to the driver's demographics. We discuss, in details, the different stages of the alogirthms development and the final product to be delivered to the customer.

4. Pre-requisites

4.1. Scientific pre-requisites

In order to fully grasp the content of this research, we expect a strong background in Computer Science and an overview about Artificial Intelligence. Some subdomains of AI as well as technical terms are explained throughly along the paper. In addition, the ability to read an analyze the references would be an advantage.

4.2. Technical pre-requisites

The main technical skills that should be met, before diving into technical part, are competencies in algorithms and Python or a similar high-level programming language. It is also assumed that the reader has profiency in Unix operating systems, terminal commands and running web servers locally. As a later step, the program package would be dockerized for easy implementation and increased usability.

5. How could we use deep learning to recognize fatigue signs?

5.1. Requirements

In order to correctly understand the scientific question, there are some key concepts we should talk about:

- Functional requirements:
After researching and studying the field of Artificial Intelligence through the mentioned references below, we conclude that deep learning is well suited for our problem. In fact, there are a bunch of techniques and models that we could use to spot the fatigue signs in a given image, from which we mention a specific type of neural networks called Convolutional Neural Netowrks. It is therefore primitive to define what is deep learning, how is a CNN able to recognize tiredness and how could we deploy a CNN to solve our problem.

- Non-functional requirements:
In this report, we make sure to use well-defined scientific terms and keep an ordered structured to understand and try to solve the scinetific question using a top bottom approach. Besides, we include high quality references in the last section of this report so that the reader could easily verify the credibility of the report and even perform further researchers on this topic.

5.2. Design

In this section, we describe the methodology behind producing the scientific deliverable for this project. In fact,

the first step was to define the main objective of our work: reducing the number of vehicle crashes. However, this objective was very general and open to many interpretations therefore we decided to narrow the scientific question and focus mainly on one factor highly related to road accidents which is fatigue.

Thanks to the work done by DAVID F. DINGES in his research paper about sleepiness and accidents[7], we have come to a proof that fatigue is associated with most vehicle crashes nowadays. Therefore, it would be very interesting to develop a system that could automatically detect fatigue signs and alert the driver on time to appropriately behave and avoid a possible accident. Unlike a normal person, a computer could not easily detect and interpret any given facial expressions. It is thanks to our eyes and cognitive system that we could break an image into parts and visually spot emotions, expressions and states such as tiredness. For a computer to do so, we need to teach it or more precisely to train it to recognize facial expressions. As it has been approached in this paper[6], we define the ability of a machine to simulate the human vision process as Computer Vision. Thus, we focus on explaining what is Computer Vision and how does it contribute to fatigue detection. Furthermore, after digging into the field of Computer Vision, we found out that the main technique used to detect facial expressions, nowadays, is through the deployment of what's called a Convolutional Neural Network.

A CNN, which is explained later in more details, is a specific type of neural networks that could learn patterns from a given set of images related to the main task and generalize on any new and unseen image. A concrete example would be to train a CNN on classifying images as dogs and cats and at a later stage, introduce a completely new image and let the CNN detect the type of object present on it. The current methods used to detect fatigue, in addition to CNN, were richly described in this paper[3].

Nevertheless, training and deploying a CNN from scratch seemed to be a very resources-consuming task. Besides, we could also face problems related to collecting relevant images for both categories (awake and drowsy). Therefore, we decided, after looking up for a bunch of pre-trained models to use the YOLO model thanks to its simple architecture and also significant speed in comparison to other models. The reasons behind choosing YOLO were very well depicted in this paper[5].

5.3. Production

The latest terrifying vehicle crash stats prove the urgent need of taking strict measures to flatten the curve and reduce the severity of accidents. Hence, the development of an intelligent system to tackle this problem is of utmost importance. One of the 'modern approaches' that has received a lot of attention over the last years, mainly because of its high efficiency, is Artificial Intelligence (AI). So how is this relevant to our work?

In fact, we would be mainly interested in a subset of AI called deep learning. In deep learning, a set of deep neural networks that consists of a higher number of hidden layers is used to simulate the way the human brain works. A deep neural network architecture corresponds to an ensemble of interconnected nodes, each built based on the previous layer, that map an input X to an input Y. This transformation could correspond to a classification, translation, speech recognition etc.. It is designed to solve any pattern recognition task without human intervention.

A neural network is based on 3 types of layers:

- **The input layer**: first layer where the input data instances are introduced.
- **The hidden layers**: where the data is processed by applying complex nonlinear functions.
- **The output layer**: the final layer where the predictions are obtained.

The movement inside those layers could happen from left to right at first (forward propagation) where random weights and biases are assigned to the input values, and from left to right (backward propagation) where the predictions are refined and optimized by reducing the error using mathematical approaches such as the Gradient Descent algorithm. Over time, we aim to minimize the error function and improve the accuracy, and that's what constructs the learning process for the machine.

Since our main input data for this project are images, we would be more interested in a special type of neural networks called Convolutional Neural Networks. CNNs are proven to have better performance with unstructured data such as images, speech or audio signals. Our main focus will be the images data. The architecture of a CNN is as follows:

***Convolutional Layer:** The convolutional layer is the main block of CNN, where most of the operations are performed. By taking an input image, for example an RGB image, the input will be a three-dimensional matrix consisting of pixels. The CNN will apply a filter (kernel) that will move across the matrix and detect the presence of the features using the convolution operation. The filter is typically a square matrix of dimensions 3*3 or 5*5. After all the operations of the filter are performed, we obtain a new output matrix called the feature map. The volume size of this feature map is affected by the number of filters applied, the number of pixels operated by the kernel and the padding which is fitting the input features to the kernel size by dropping the unfitted columns or adding zeros to match the same size. To introduce non-linearity, the CNN applies a ReLU function to the feature map. The Rectified Linear Unit function has been introduced recently as a replacement for traditional activation functions proven to be inefficient in the long term and causing problems such as the vanishing gradients problem. In fact, the ReLU function runs faster as the sigmoid function for example because it is

easier to computer thanks to the absence of exponential terms.

***Pooling Layer:** This layer introduces dimensionality reduction or down sampling. This will result in the reduction of the number of parameters to be learned and similarly the computational complexity. Depending on the type of the pooling layer, the matrix reduction could be demonstrated in different ways. For instance, the max pooling operation will choose the maximum pixel value from the region of the feature map.

***Fully Connected Layer:** As the name suggests, the fully connected layer(FC) consists of interconnected nodes, each dependent on the activation of the previous one, that perform the task of classification based on the previously obtained features. In order to classify the output correctly, the FC layers tend to use a softmax function that maps an input value to 0 or 1.

In order to correctly produce satisfying results, the development and production of the CNN should undergo several phases.

5.3.0.1. Preprocessing

. The visual inputs could be of different shape, color and position. Hence, we need to define a pipeline to process the inputs in a way that we obtain the same input charecterstics for all the visuals. Some of the basic operations may include pixel normalization for the sake of reducing complexity, color transformation(to grayscale) or resizing. In case of insufficient data samples, data augmentation techniques such as flipping images could be applied.

5.3.0.2. Training

. The training phase consists of feeding the preprocessed visual inputs to the Convolutional Neural Network through the input layers to detect the presence of features, apply filters, reduce the size and learn the relevant patters through the rest of the hidden layers. This has been explained in a deeper way in section 2.4.

5.3.0.3. Evaluation

. This phase consists of evaluating the model's performance based on the produced graphs and metrics from the training phase. Afterwards, a different set of approaches to improve the model's performance such as data augmentation, hyper-parameters tuning could be applied. It is very important that the the final version of the deep learning model generalizes well on unseen samples and meets the minimal requirements for the deployment stage.

However, training CNNs from scratch to recognize whether the driver is drowsy or not will be an extremely time-consuming task. Besides the high numbers of computational resources needed for this task, a reasonable number of relevant data points collected for both labels should be present. Despite the increasing number of available datasets online, this task is not always obvious. Therefore, we introduce YOLO[5], a pre-trained algorithm based on a single CNN that simultaneously predicts multiple bounding boxes and their probability classes. The YOLO architecture is incredibly fast. The base model could treat images in real-time at 45 frames per second. Hence we deploy this pre-trained model for our image classification function.

On the other hand, we also make use of data science techniques to achieve another output belonging to the accident risk assessment. In fact, data science is a field of computer science that focuses on rigorously analyzing data to extract patterns, drive useful insights and build meaningful conclusions. Different statistical techniques are applied on the data in order to conclude possible relations between introduced features. In our case, we make use of the driver's data such as the age, sex and driving proficiency to create a machine learning model that could classify the possible accident risk based on the entered information. This task lies under the field of supervised learning and is called data classification.

5.3.1. Answer to the scientific question

. Since fatigue is a major factor contributing to vehicle crashes, we need to develop a solution that is able to automate the task of detecting fatigue signs among vehicle drivers. As we are recognizing signs on given images, we need to use deep learning and precisely train a CNN to perform the facial expression detection. However, the CNN should be accurate enough to deliver satisfying results and this could be assessed through different evaluation metrics.

5.4. Assessment

The Computer Vision field offers a variety of techniques that we can implement to produce solutions to various issues we are facing nowadays. That being said, deep learning models have achieved astonishing results over the years in analyzing visual inputs. Therefore, the mentioned methodology about CNN structure answers our scientific question: How could AI contribute to the prevention of truck accidents. Nevertheless, the designed approach is always prone to errors since the accuracy of the deep learning model is dependent on multiple factors on which we mention data size and relevance. We could not completely conclude that the usage of a deep learning model would prevent potential vehicle crashes, but at least it would contribute significantly in reducing them.

6. Technical description

For this first version of the project, the technical deliverable is a semi-automatic system consisting of a web server running in the local machine. This web server is powered by a third-party called streamlit. It works by taking two different input data types. The first input is captured automatically by a connected camera and the second one is a set of information concerning the driver demographics entered manually. Hence, we define two distinct methods of input processing: One for the tabular data and the other for the images. The web server, once run locally, takes in a continuous way real-time

video frames within short intervals. Afterwards, the frames are preprocessed and fed to a pre-trained deep learning model to perform the main task's idea which is drowsiness detection. At this stage, the first output, based on the binary classification, is going to correspond to the driver's current state:. The second output, based on the manually entered background information of the driver, is going to be an assessment of the accident's risk. By combining both outputs, an alert sound would be emitted in the vehicle and a break suggestion would be displayed. At a later version of the project, the technical deliverable would correspond to a small app installed in the front of the driver's face and related to the vehicle's main screen and motor. Based on the program's output, different actions such as stopping the vehicle for a while would be taken.

6.1. Requirements

The software main goal is to accurately detect fatigue signs presented by facial landmarks and interpreted from text features of a truck driver in order to prevent potential accidents. The software is considered as "well-executed" when a set of three main functions are called and executed properly. The functions are split into two types: input and processing functions. The input function has to take an input image or a real time video and assure that it matches the processing functions requirements: image size and color. For instance, if the processing functions require a grayscale image and the input image is colored, we process the visual input accordingly. Moreover, the processing functions act as the main part of the program and map the visual and tabular data into new interpreted output. The output depends on the function design and could be a newly generated image with a specific label or an informative text about potential accident risk. For better understanding of the functions structure, check the appendix section where we explained in a more structured way the main functionalities provided. Once executed properly, the functions should be reusable. In fact, the function callback should be done independently of how the user's look like or the execution environment. They should also provide accurate results especially in case of possible occurrence of a vehicle crash since such an information is considered critical to both the driver and the responsible company.

6.2. Design

The technical deliverable for this project is a web server based on streamline and using in the backend the trained deep learning model to recognize whether a person is drowsy or non-drowsy. The derived information, in addition to a risk assessment function to evaluate the accident risk degree, is supposed to be able to accurately detect potential vehicle crash on a given time t. In order to produce the web application, we make use of a pertained algorithm YOLO, visual and tabular inputs and a web server (streamlit). Hereby, we will explore the different libraries used for this work and the design phases.

6.2.1. Libraries:.

6.2.1.1. PyTorch

. Initially developed by Facebook AI's team, pytorch is an open source framework used for deep learning applications. It is adapted both on GPUs and CPUs and allows faster tensor computations. We use it in our code to load the pertained YOLO base model and to make new predictions using the newly created weights file adapted to our specific visual inputs. Pickle: Developed mainly for the purpose of serializing and de-serializing object structures using binary protocols. In our case, it is used to save and load a machine learning model.

6.2.1.2. Numpy

. Lying under the domain of linear algebra, dumpy performs array operations in a faster way using vectorization.

6.2.1.3. Streamlit

. Another open source framework integrated with python to easily create web applications for deep learning models. It works smoothly with almost all python libraries such as PyTorch and Keras.

6.2.1.4. OpenCV

. This python library is specialized in image and video analysis. It offers a variety of functionalities to process and input visual data.

6.2.1.5. Pandas

. One of the most used packages for data science related tasks such as reading and processing data frames. In addition, since it is based on bumpy, it allows a wide range of multi-dimensional array processing such as reshaping data sets, grouping, merging and data filtration.

6.2.2. Data Collection

. It is well-known, at least among the data science community, that data collection and preprocessing represent the most time and effort demanding tasks. Despite the increase in the number of open-source data portals, the problem of data availability and accessibility is always persistent. During the production of the web application, we faced several data issues mainly for collecting visual inputs to train our YOLO model. Nevertheless, we partially solved this problem by generating a few personal data instances to partially train the model. As for tabular data, we managed to have access to some relevant data frames that could be of a use to evaluate the potential accident risk.

6.2.3. Data Preparation

. In this project, the data preparation phase could be summed up in the process of data annotation. We used RectLabel to manually draw boxes over the most relevant features in the visual inputs(face elements). By doing so, we generate metadata files containing the bounding box annotations.

6.2.4. Algorithm Development

. After successfully installing the required libraries, collecting and preparing the data, we feed our data set into the YOLO base model and start the training phase. At the end of this phase, we should obtain a new generated file containing the trainable parameters.

6.2.5. Deployment

. After successfully training and testing the deep learning model, we initialize the streamlit web server with a basic GUI design and we use the generated model as a backend model to classify new data instances.

6.3. Production

In this section, we will dive deeply into the program execution phases and the main functions used to achieve the described results.

6.3.1. Functions.

6.3.1.1. main function

. In the first 20 lines of the main function, we focus on creating page elements. We specify a title and generate a form with input boxes. We distinguish two different types of input values: visual input that accepts an uploaded image (real time video to be implemented later) and numeric input such as age, gender and current weather status. Once the form is submitted, we

```
1  model = torch.hub.load('ultralytics/yolov5', 'custom', path='last.pt', force_reload=True
2  def main():
3
4      title = '<p style="font-size: 42px;">Drowsiness Detection tool </p>'
5      title = st.markdown(title, unsafe_allow_html=True)
6      st.text("Detect drowsiness on truck drivers.\nCreated by Bacem Etteib")
7
8      img2 = input_function()
9      placeholder = st.empty()
10
11     with placeholder.form(key="submit-form"):
12         input_gender = st.selectbox(
13         "Gender:", ["Male","Female"])
14         input_weather = st.selectbox(
15         "Weather:", ["Clear","Cloudy",
16         "Foggy","Rainy","Severe Crosswindss",
17         "Drizzling","Blowing Sand"])
18         input_age = st.number_input("Select your age)",
19             min_value=18, max_value=90, value=40, step=1)
20         generate = st.form_submit_button("Submit")
21
22     if generate:
23         st.write(risk_assessment(input_gender,input_age,input_weather))
24         image_classifying(model,img2)
```

Fig. 1. main function

display the main output of the program using two callable functions: risk assessment and image classifying.

6.3.1.2. risk assessment function

. As the name suggests, we use this function to estimate the accident's risk degree. To do so, we need to gather information about the driver such as the gender and age. Those information are proven to be correlated with our target variable and could bring, when combined with current weather status, relevant information about the classification problem. For instance, we train a machine learning model that we would talk about later to predict the risk level based on those information. From line 46 to 51, we use the pickle library to load the model placed in current folder and to predict an array X containing the input values. We assign the prediction values (0 or 1) to a new variable and we decode the binary values into text data (Low Risk or High Risk) for better readability for the targeted user. From line 27 to 45, since we need to input an array of value combinations into the model, we encode the input values from the user accordingly. This approach is similar to the One-hot-Encoding approach in machine learning.

```
26 def risk_assessment(gender,age,weather):
27     X = np.array([int(age),0,0,0,0,0,0,0, 0,0])
28     if str(weather) == 'Clear':
29         X[2] = 1
30     elif str(weather) == 'Cloudy':
31         X[3] = 1
32     elif str(weather) == 'Foggy':
33         X[4] = 1
34     elif str(weather) == 'Rainy':
35         X[5] = 1
36     elif str(weather) == 'Severe Crosswinds':
37         X[6] = 1
38     elif str(weather) == 'Drizzling':
39         X[7] = 1
40     elif str(weather) == 'Blowing Sand':
41         X[1] = 1
42     if str(gender) == 'Male':
43         X[9] = 1
44     else:
45         X[8] = 1
46     loaded_model = pickle.load(open('finalized_model_2.sav', 'rb'))
47     result = loaded_model.predict([X])
48     if result == 0:
49         return 'Status: Low Risk'
50     else:
51         return 'Status: Medium to High Risk'
```

Fig. 2. risk assessment

6.3.1.3. image classifying function

. Although it looks very simple, in terms of length, this function acts as one of the core functions for our program. It takes as parameters the trained deep learning model along with a user image and uses the model to make predictions about the drowsiness status. It will, then, output a new image 'img3' with a labeled bounded box around the user' face as awake or drowsy.

```
53 def image_classifying(model,img2):
54     results = model(img2)
55     #results.print()
56     img3 = np.squeeze(results.render())
57     st.image(img3, caption='Proccesed Image.')
58
```

Fig. 3. image classifying function

6.3.1.4. input function

. In order for the previous function to execute properly, we need to pass as a parameter, an image 'img2'. In the input function, our interest will lie heavily in taking an input image from the user through a simple upload interface, read the image using the opencv library and return it for later usage. If we go back to line 8, we can see that the input function is assigned to a variable 'img2' so that we can pass it later to the deep learning model.

```
59 def input_function():
60     st.subheader("""
61     Upload a current picture of you.
62     """)
63     #upload image using only accepted formats
64     file = st.file_uploader('Upload Image', type = ['jpg','png','jpeg'])
65     if file!= None:
66     #put image in the same directory as the script
67         img = cv2.imread(file.name)
68         return img
```

Fig. 4. input function

Back to the first line, we can see that we used PyTorch library to load a YOLO base model with a custom weights file generated from the training phase. In fact, we have seen, until now, how the program acts in terms of core functions. For better understanding, we need also to explain how we trained the YOLO and RandomForest models. In the following section, we will explore, in more details, the data preparation, processing and training phases.

6.3.2. Models.

6.3.2.1. Data preparation and training for YOLO

. In order for the YOLO model to predict whether a given person is drowsy or awake, we need to train it on a custom dataset containing a balanced number of samples of the two labels. Since data collection is always a nightmare in the field of AI, we will simplify this process and generate 10 samples for each category using my camera and the opencv library. We place the gathered images in two folders such as the folder structure matches the categories names, and we use an open source software (RectLabel) to manually draw bounding boxes around the face with the correct label for each box (drowsy or awake). We will then use the generated yml file to train our base model. As we can see, the training command takes as parameters the image size, batch size (number of samples to be processed), number of epochs (iterations), the yml file and the initial weights. At the end of the training phase, we obtain our final deep learning model ready for deployment. The final version of the model is the one with higher accuracy. The accuracy should be meaningful, that means that we can have a model with higher accuracy but also with an overfitting problem. Thus, after experimenting with the model for a while, we obtain the final and the best version of it.

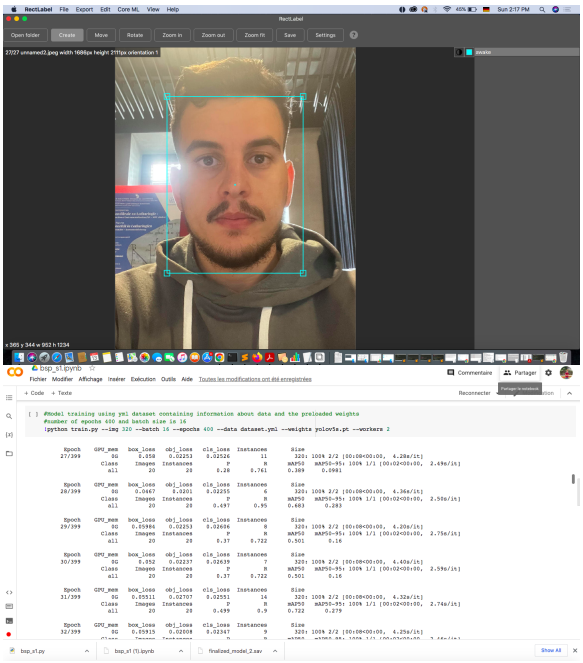
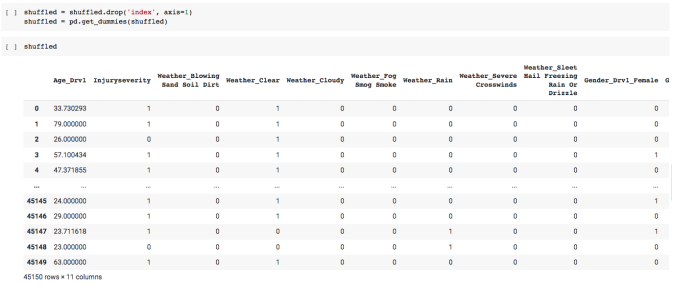


Fig. 5. image annotation and model training

6.3.2.2. RandomForest classifier

. We use an open source csv data frame of shape 45150x11 containing historical information about previous crash data in the US. The columns contain data about driver's age, gender and weather status. We remove irrelevant columns and we perform preprocessing techniques such as variable encoding, filtering and data frame splitting. Afterwards, we train an ensemble learning method called RandomForest to predict the target column (crash severity) based on the training features. Finally, we can see that we implemented pickle to save the model for future deployment.



```
[ ] shuffled = shuffled.drop('index', axis=1)
    shuffled = pd.get_dummies(shuffled)

[ ] shuffled
```

	Age_Drv1	Injuryseverity	Weather_Blowing Sand Soil Dirt	Weather_Clear	Weather_Cloudy	Weather_Fog Smog Smoke	Weather_Rain	Weather_Severe Crosswinds	Weather_Sleet Hail Freezing Rain Or Drizzle	Gender_Drv1_Female	G
0	33.730293	1	0	1	0	0	0	0	0	0	
1	79.000000	1	0	1	0	0	0	0	0	0	
2	26.000000	0	0	1	0	0	0	0	0	0	
3	57.100434	1	0	1	0	0	0	0	0	1	
4	47.371855	1	0	1	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	
45145	24.000000	1	0	1	0	0	0	0	0	1	
45146	29.000000	1	0	1	0	0	0	0	0	0	
45147	23.711618	1	0	0	0	0	1	0	0	1	
45148	23.000000	0	0	0	0	0	1	0	0	0	
45149	63.000000	1	0	1	0	0	0	0	0	0	

45150 rows × 11 columns

Fig. 6. data frame

6.4. Assessment

The program was first designed to take two distinct input types: visual and tabular data and output a risk assessment informative message and a newly generated image with the correct label. Globally, we were able to exactly produce this behavior and achieve satisfying results. Nevertheless, whenever we demonstrate an AI based solution, it is of utmost importance to mention the results accuracy. It is well-known that, in most cases, it is impossible to produce a completely accurate solution. Therefore, the software is always under development and could not be considered as achieved. In the next versions, we could introduce more data, more features and play with the hyperparamters of our deep learning model to increase the model's accuracy and the program precision.

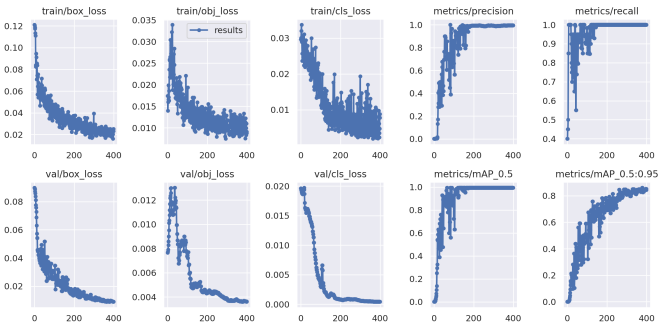


Fig. 7. training and validation losses

7. Appendix

- Operation: Input of real-time images
  - Users: A truck driver
  - Description: The function takes continuous real-time video frames of the truck driver through an attached camera.
  - Parameters: No required parameters, the function call is performed automatically and does not need any type of parameters.
  - Pre-condition: The attached camera should be running smoothly all along the process. It should also be positioned in a way that the user's face is close and his facial landmarks are clear enough.
  - Post-condition: The video box must be maintained so that it can be used to display the results from the next functions.
  - Trigger: The function starts automatically once the web server starts. It does not require any user intervention.
- Operation: Image classifying
  - Users: The system
  - Description: Once the images from the input function are generated, the function executes immediately and the pre-trained model starts to predict the facial recognition status according to the defined labels: drowsy or normal.
  - Parameters: An image/video from the input function
  - Pre-condition: The input function should be run efficiently and correctly: Facial landmarks are well visible on the taken video.
  - Post-condition: A box around the face with a confidence score should appear on the screen.
  - Trigger: The function starts immediately after the successful execution of the input function.
- Operation: Risk Assessment
  - Users: The system
  - Description: The function takes a set of information related to the driver's background: age, sex, proficiency etc.. and makes a prediction of the crash severity. The results of this function should be combined with those from the image classification one.
  - Parameters: Manual data input from the driver.
  - Pre-condition: The data should be entered in the required format.
  - Post-condition: Text displayed on the main screen showing accident risk prediction.
  - Trigger: Starts once at the beginning of the main program and the entered data is stored.

```
import streamlit as st
import pickle
from sklearn.ensemble import
    RandomForestClassifier
import cv2
import torch
import numpy as np
import os
import time ,sys
from PIL import Image, ImageEnhance
from streamlit_embedcode import github_gist
import sys

model = torch.hub.load('ultralytics/yolov5',
    'custom', path='last.pt',
    force_reload=True)
def main():
  title = '<p style="font-size:
      42px;">Drowsiness Detection tool </p>'
  title = st.markdown(title,
      unsafe_allow_html=True)
  st.text("Detect drowsiness on truck
      drivers.\nCreated by Bacem Etteib")
  img2 = input_function()
  placeholder = st.empty()
  with placeholder.form(key="submit-form"):
  input_gender = st.selectbox(
  "Gender:", ["Male",'Female'])
  input_weather = st.selectbox(
  "Weather:", ["Clear",'Cloudy',
  "Foggy","Rainy","Severe Crosswindss",
  "Drizzling","Blowing Sand"])
  input_age = st.number_input("Select your
      age)", min_value=18, max_value=90,
      value=40, step=1)
  generate = st.form_submit_button("Submit")
  if generate:
  st.write(risk_assessment(input_gender,input_age,input_
  image_classifying(model,img2)
  def image_classifying(model,img2):
  results = model(img2)
  #results.print()
  img3 = np.squeeze(results.render())
  st.image(img3, caption='Proccesed Image.')

def input_function():
  st.subheader("""
  Upload a current picture of you.
  """)
  file = st.file_uploader('Upload Image',
      type = ['jpg','png','jpeg'])
  if file!= None:
  img = cv2.imread(file.name)
  return img

def risk_assessment(gender,age,weather):
  X = np.array([int(age),0,0,0,0,0,0,0, 0,0])
  if str(weather) == 'Clear':
    X[2] = 1
  elif str(weather) == 'Cloudy':
    X[3] = 1
  elif str(weather) == 'Foggy':
    X[4] = 1
  elif str(weather) == 'Rainy':
    X[5] = 1
  elif str(weather) == 'Severe Crosswinds':
```

```
    X[6] = 1
  elif str(weather) == 'Drizzling':
    X[7] = 1
  elif str(weather) == 'Blowing Sand':
    X[1] = 1
  if str(gender) == 'Male':
    X[9] = 1
  else:
    X[8] = 1
  loaded_model =
      pickle.load(open('finalized_model_2.sav',
      'rb'))
  result = loaded_model.predict([X])
  if result == 0:
    return 'Status: Low Risk'
  else:
    return 'Status: Medium to High Risk'

if __name__ == '__main__':
main()
```

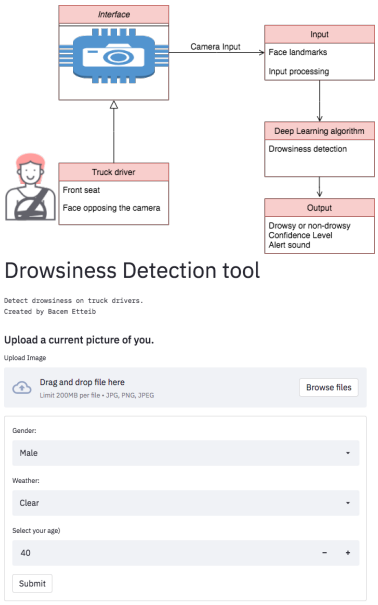


Fig. 8. Application Overview

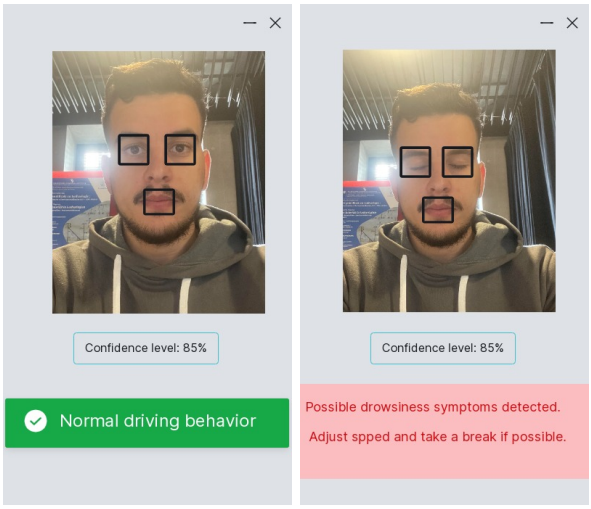


Fig. 9. Normal and Drowsy cases

References

[1] Fatality Facts- Large trucks." Insurance Institute for Highway Safety (IIHS) , May 2022.

Explanation: Statistics from 2020 about truck crashes to justify the need for our solution and why it is important.

[2] ZHAO, Zuopeng, ZHOU, Nana, ZHANG, Lan, YAN, Hualin, XU, Yi and ZHANG, Zhongxin. Driver fatigue detection based on convolutional neural networks using EM-CNN. At: Computational Intelligence and Neuroscience. Hindawi, 18 November 2020. available at: https://www.hindawi.com/journals/cin/2020/7251280/ .

Explanation: this paper, a multitask cascaded convolutional network was used for face detection and feature extraction. Interestingly, the proposed network architecture outperformed existing CNN methods such as AlexNet or ResNet50 models in terms of accuracy. We could possibly make use of this architecture to increase our detection accuracy and reduce false positive cases which might be annoying for both the driver and the company he is working for. I am currently playing around with the proposed architecture and testing the results on my own.

[3] KOVAČIĆ, Kristian, IVANJKO, Edouard and GOLD, Hrvoje. Computer Vision Systems in road vehicles: A Review. 1 October 2013. available at: https://arxiv.org/abs/1310.0315 .

Explanation: I used the paper to explore different methods and techniques to prevent accidents among truck drivers. In addition to techniques concerning lane detection, vehicle state, the author made reference to the usage of facial expressions to spot possible drowsiness. He mentioned more advanced techniques such as the combination of both driver's state, vehicle state and general conditions to develop a more generalized model. This could be applied in a newer version of the application. I also made use of the last section of this paper "Open problems" to get to know the limitations of the existing solutions as well as my solution and possibly think of ways to overcome them in the newer version.

[4] MAGÁN, Elena, SESMERO, M. Paz, ALONSO-WEBER, Juan Manuel and SANCHIS, Araceli. Driver drowsiness detection by applying deep learning techniques to sequences of images. Multidisciplinary Digital Publishing Institute, 22 January 2022. available at: https://www.mdpi.com/2076-3417/12/3/1145 .

Explanation: As the authors of this paper proposed a similar system to

my solution, I focused on the technical part concerning their project and how they applied CNN and RNN for feature extraction. For example, they tested an SVM model combined with regression trees to detect driver's facial landmarks instead of YOLO which seems to provide interesting results. The comparison of two methods illustrated with real world results allowed me to further estimate the performance of my to-be solution.

[5] REDMON, Joseph, DIVVALA, Santosh, GIRSHICK, Ross and FARHADI, Ali. You only look once: Unified, real-time object detection. 9 May 2016.. Available at: https://arxiv.org/abs/1506.02640 .

Explanation: This is the original paper for the YOLO pre-trained model. The paper explains the model architecture and how it works. It also points out how the model overcomes the computational limitations and provdes accurate results. Thus, for the sake of reducing our project complexity, the usage of YOLO could be a good starting point.

[6] VOULODIMOS, Athanasios, DOULAMIS, Nikolaos, DOULAMIS, Anastasios and PROTOPAPADAKIS, Eftychios. Deep Learning for Computer Vision: A brief review. Computational Intelligence and Neuroscience [online]. 1 February 2018. Available at: https://www.hindawi.com/journals/cin/2018/7068349/ .

Explanation: This paper provides a very accurate explanation of computer vision and the different deep learning techniques being applied to this field.

[7] Drew Dawsona, Amy C.Reynolds, Hans P.A., Van Dongen, Matthew J.W. and Thomas. Determining the likelihood that fatigue was present in a road accident: A theoretical review and suggested accident taxonomy. 6 September 2018. Available at: https://www.sciencedirect.com/science/article/pii/S1087079217302095bib1 .

Explanation: This paper explains the high correlation between fatigue signs and road accidents. It sheds the lights on the association of sleep cycles, tiredness factors and the awarness of the driver in front of the steering wheel.