Creating a Sensor System for Safe Motor Navigation using Deep Learning and YOLOV5: HOMESWEET

Lare Samuel Adeola
l.adeola@bolton.ac.uk

School of Creative Technologies
University of Bolton
December 21, 2023

Abstract

'HOMESWEET' is a deep learning model created with the YOLO object detection algorithm that has been trained to detect certain human physical states that could result in road accidents and deaths. The model achieved an accuracy score of 83% and a Precision rate of over 90% but had a fairly modest Recall rate of just over 70%. The model can be deployed in various other applications as it was able to detect not only facial cues but other micro-expressions and gesticulations that lead to the various states; in particular, in this research, it was created to detect fatigue, drowsiness and lack of total concentration while driving.

Introduction

Unsafe driving is one of the leading causes of death not only in the United Kingdom but also in the world. According to the data from ons.gov.uk (2019), an average of 2400 people die each year from year 2000 to 2020 as a result of road accidents. 34% of these road accidents are caused by motorists who have failed to look properly (statista.com 2019). Figures 1, 2 and 3, show various plots of the amount of lives lost, those who sustained serious injuries and the total injuries as a result of road accidents.

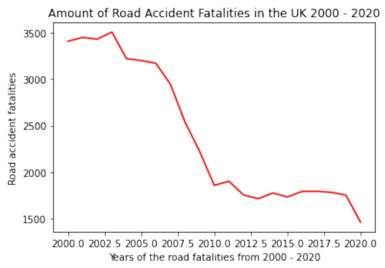


Figure 1: Amount of Road Accident Fatalities in the UK 2000 - 2020

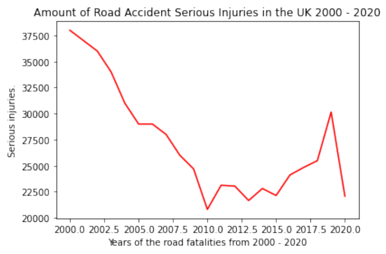


Figure 2: Amount of Road Accident Serious Injuries in the UK 2000 - 2020

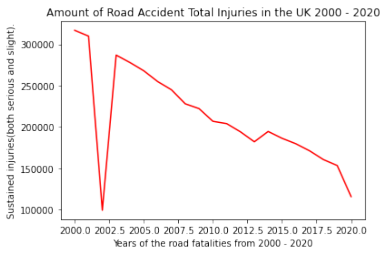


Figure 3: Amount of Road Accident Total Injuries in the UK 2000 - 2020

The charts show a markedly downward trend in the amount of road accidents fatalities from the year 2000 to the year 2020. While this is commendable, it is not enough. For example in the year 2020, there were a total of 1460 deaths caused by road accidents. This means that an average of 4 people died from road accidents per day in the year 2020. This coupled with the fact that road accidents can be avoided, it is imperative to always strive to reduce these figures further.

This makes the development of a sensor application that can help curtail unsafe driving habits paramount and important. As this application can not only save the lives of drivers and motorists but also save the lives of members of the public like pedestrians and cyclists. In this research, we present a sensor application system that has achieved an accuracy score of 83% in predicting selected unsafe driving habits(like fatigue, drunkenness, distractions) and other causes that may lead to both road accident fatalities and serious injuries.

Aims and Objectives

1. Build a Computer Vision Model that can predict the various labels used in creating this application. In this application, two labels were created:

   - Safe
     (a) Focused
     (b) Total Concentration
     (c) Awake
   - Unsafe
     (a) Fatigue
     (b) Drunk
     (c) Distracted

2. Create a User Interface, User Experience System and Feedback System, that shows the design of the system for mobile, tablets and personal computers and how users would navigate through such system.

Brief Overview of Relevant Literature

In the paper by (Singh et al. 2023), they created a system that focus on the eye-blinking rate then later advanced this blinking rate to create a statistic called - 'PERCLOS' which stands for the percentage of eyelid closure over the pupil over time (Singh et al. 2023). Their model achieved an accuracy of 80% but it was not stated in the research if this was on the training data or the test data; hence it's difficult to know how generalized their model is. Also their sample size was remarkably small, as they ran just 10 experiments of which their system couldn't detect 2. With such a small sample size and absent of data on the model's performance on test/validation data, the performance of the model they built can't be definitely measured. However, their PERCLOS statistic was a good feature extraction.

In the research done by (Dipu et al. 2021), the used an advanced Convolutional Neural Network(CNN) architecture for classification and the Single-Shot Detection Algorithm (SSD) for object detection. The maintained their reason for using the SSD over other popular detection algorithms like the YOLO V5 was because the SSD algorithm is much faster though less precise. Their approach involved extracting physiological features from various sets of images that show fatigue like yawning, drowsiness etc. They also created a data-set for this research and trained their model using a combination of both primary and secondary data. Their total data-size was 4500 out of which 700 was earmarked for validation. Their model achieved high accuracy in the 90s region - but it isn't clear if this was on the train set or the test set. They also found that their model performed poorly under low-light conditions because of the low amount of images in low light conditions in the training set. The researchers didn't mention anything about image augmentation, which is used to train the classifier to recognize augmented images, in this instance it could be low-light, image obstruction (like the steering wheel or a not so tall person driving a car), also the issue of demographics though mentioned was not elaborately defined/shown in any table or figures. Lastly, their test/validation data as a percentage of the total data-set size is too small - it stands at 15.56% which is too small as a good percentage for a validation/test data should be between 20-30% to avoid over-fitting.

Sensor

Collecting Sensor data for fatigue, stress, tiredness, drunkenness and other subsets of labels used in creating this application can be approached from various ways. One way is using the pressure sensor. This was done in the work by Yonekawa et al. (2009) whereby they created a system to detect fatigue in older people so as to prevent them from falling down. The pressure sensor could have been used also in detecting a motorists fatigue level. But how sustainable would this be? One must understand that this problem is greatly affected by distance, and as such is a problem that may fluctuate over time. Secondly, the pressure sensor being very sensitive may capture other data and though this could be clean and preprocessed in real time, there may be computational cost overheads whose solution may be too great for a problem so small.

In the work by Yonekawa et al. (2009), their problem was somewhat related to walking which is a type of motion but not the motion that is going to be captured in this project.

Another approach for Sensor Capturing would have been the use of measuring fatigue eye movement as was done in the research by Miao et al. (2022), in their approach, they created an algorithm that uses eye movement to collect fatigue information and based on three class labels - mild fatigue, moderate fatigue and severe fatigue, categorise the user's fatigue state(Miao et al. 2022). While this advanced Sensor collection has a high effective rate, it's use case seems to be too complex for the purpose it is trying to achieve. The eye organ is one of the most complex and evolved sensory organ of the human body. It cleanses itself, it auto adjusts based on the amount of light in an environment, in fact it can be argued that it is one of humanity's most advanced natural system. It is so advance that till today, modern cameras are still trying to catch up to it's complexity and it is indeed the source and inspiration for a separate branch of Computer Science called Computer Vision.

To be able to capture such sensory abilities in real time, would require a sort of intelligence that can evolve like the eye, that is aware not just of itself - locally like the research by Miao et al. (2022) but also of its surroundings to be able to provide further context. Hence, why Deep Learning was used in creating this application along with Computer Vision. The context of the environment is as important as the object of the environment.

Methodology

Deep Learning and YOLO

Using Convolutional Neural Networks to capture not only the eyes and eye movements but also other micro-interactions like the yawn or the tired lower lip proved to be effective. Deep Learning helped in training the model at times approaching Precision levels of 0.9998 and Recall levels of 1. What these means is that if more relevant data is fed to this model(a total of 167 images were used to train the deep leaning model), it's capabilities at detecting fatigue, drunkenness and other sub-classes would be immensely improved. And this is the desired outcome, a process that is not static, but rather continually improves itself and this is what makes Deep Learning an efficient technique for detecting fatigue and drunkenness.

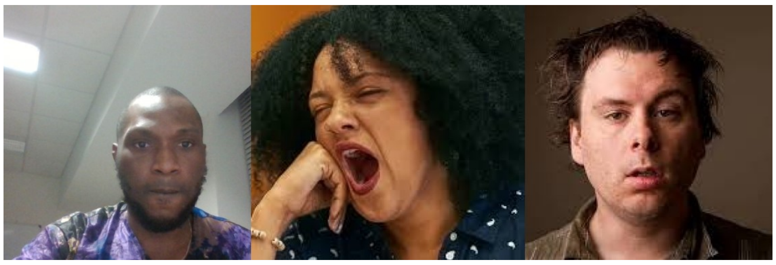


Figure 4: A few of the images used to train the model. The sub-classes from left to right: awake, tired and drunk. the sources from left to right: primary source, secondary(from google images), secondary(from pixabay)

YOLO is a python package and it means "You Only Look Once". It is a family of object detection architectures and models pre-trained on the COCO(it is a large object detection and segmentaion dataset) dataset. It basically divides an image into a grid system then classifies the different images in the grid system. The YOLO package was used in training our deep learning model.

Webcam, PhoneCam, Computer Vision

One of the advantages of using the camera as a sensor is its ubiquitous nature in modern day life. Almost everywhere one goes, there's a camera. From trying to cross the zebra crossing to photo-bombing a couple taking a selfie in the park not fault of ours, there's a camera a breath away. This makes this sensor particularly useful for the problem set. With the advent of navigation technology like GPS and Google Maps, the phone has become an important staple for car motorists. This makes the camera sensor a perfect choice for this problem set and the phone screen a perfect GUI(Graphical User Interface) for interaction, navigation and feedback.

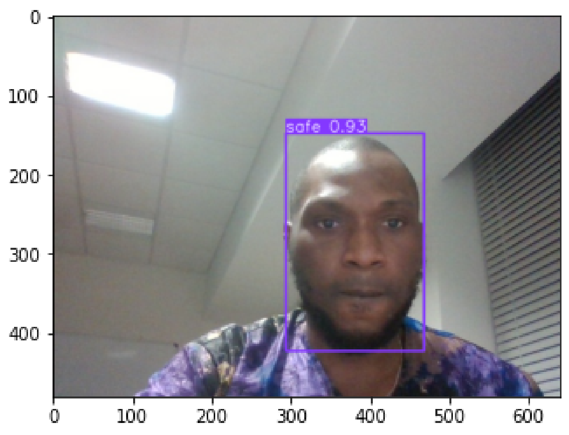


Figure 5: A primary sourced image captured by a laptop webcam. The rectangular box is the model capturing the state of the object. In this instance, the model captures the subclass - awake and correctly categorizes it as safe with a confidence score of 0.93.

Advantages of using this Sensor

The advantages of using the Camera Sensor include:

1. It is ubiquitous in nature.

2. It is part of the phone or laptop or personal computer so no extra sensor device to carry.

3. Camera pixels of phones have vastly increased over the years, figure 6 shows its exponential then steady then constant growth using the Iphone as a case study. Hence, it's accuracy in capturing the object is an advantage, as the higher the pixels the closer to reality the image becomes.

4. Using the Camera Sensor lends itself to using Computer Vision algorithms in performing Binary Classification(in this project - safe and unsafe), which would lead to increased accuracy and relevancy.

5. Zero Cost. Since most people have a mobile phone as shown in figure 7 whereby in 2022, 93% of the UK's total population are using a smartphone, using the Camera Sensor comes at no cost to the user.

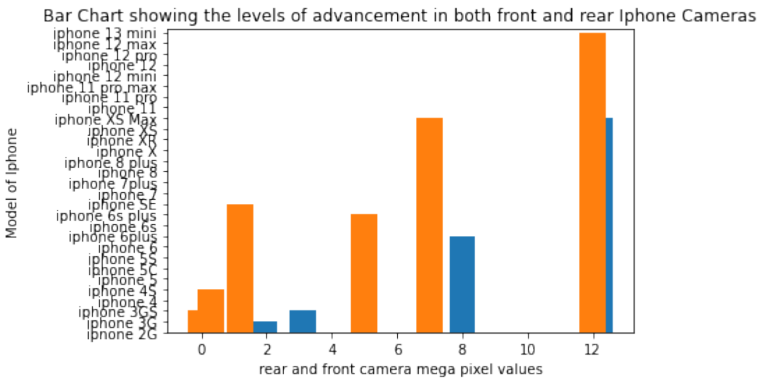


Figure 6: Bar Chart showing the levels of advancement in both front and rear Iphone Cameras. The blue bars represent the front camera mega pixels(mp) while the orange bars represent the rear camera mp.

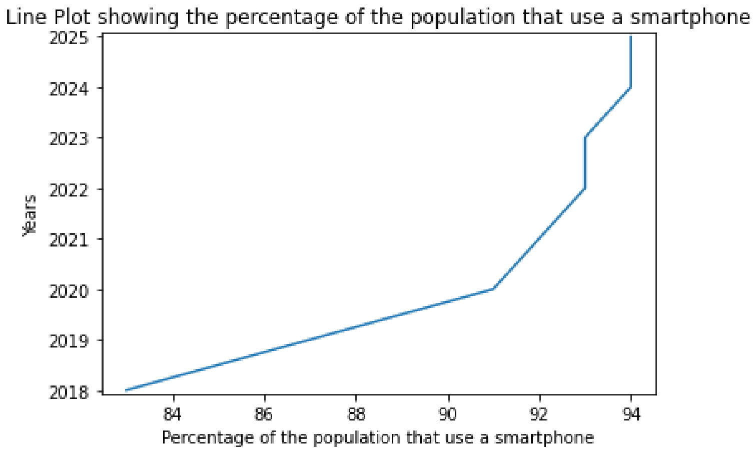


Figure 7: Line Plot showing the percentage of the UK population that use a smartphone. The percentages were gotten from the total amount of smartphone users in the UK(in millions) divided by the total population in the UK(in millions). The calculations are shown in the appendix.

Disadvantages of using this Sensor

1. Though Camera pixel advancements have occurred over the years, there are still slight problems like Occlusion, depth perception etc.

2. In order to get real-life accuracy, dedicated cameras may have to be employed like the Digital Single Lens Reflex Cameras(DSLR). Also, we have very little control on what type of phone camera the driver would be using, hence we cannot assume that they would be using a phone with good camera quality.

3. Deep Learning has made Computer Vision Classification more accurate and powerful in recent years (Sharifani & Amini 2023), but there are computational costs. For example while training the Deep Learning model for this project, it took me two days as I trained the model for 500 epochs(in hindsight this epoch amount was too much as after 180

epochs there were only infinitesimal improvements in the Precision level of the model on the training data).

Solutions to the Disadvantages

Not all of the problems were solved or catered to but to a large aspect, most of them were factored in and various techniques were employed to combat them. I would list a few of them below:

- The problem of occlusion and depth perception were catered to by feeding the model images with this particular problem. Especially that of depth. The images fed to the model were taken from different ranges. Figure 8 (images from different ranges and side angles) shows a few of the images taken from different depth ranges in order to combat this problem.

- For the second problem, the model was fed images that were a bit blurred/grainy in some instances, under-lit in other instances and even those where the colour space was monochrome or greyscale.

- The third depends on many variables ranging from if one was using the computers GPU or a dedicated GPU to the RAM of the CPU. Other factors like the number of epochs and amount of images also affect the total compute time.

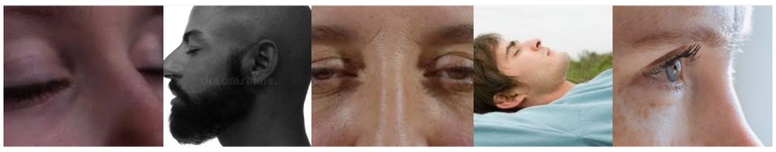


Figure 8: Variations of the different types of images used to train the model. Side-view, close-ups etc

Data Collection

Primary Data

The training of 'HOMESWEET' involved both primary and secondary sources of data. The primary source of data was of the author and it was taken through the webcam of the author's laptop. A series of different images and angles were taken to make the model consistent in accurately predicting the various sub-classes. These positions include:

- At different focal lengths. This was achieved albeit mechanically by moving further away.

- Blurry/grainy images. This was achieved by being in slight motion while the camera was capturing the author's image

- Side views.

- Tilts.

- Uncontrolled Environment. All the images from the primary data were taken in the Peter Stocker Laboratory(sci 2.27) in the Department of Computing Sciences, Univesity of East Anglia.

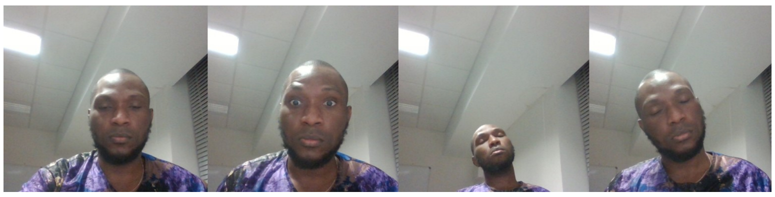


Figure 9: Caption

Secondary Data and Other Image Selection Criteria

The secondary data was sourced from the web. They are:

- Google Images
- Pixabay
- istockphoto

There were three cardinal attributes/features sought after when selecting an image -

1. Posture - Does the image narrate honestly the class to which it would belong?
2. Clarity and Crisp - How clear and crisp is the image? What is the pixel density?
3. Image Shot - Is it a full view or a side view? Is it a close-up shot or a mid-range shot?
4. Nationality - Algorithm Bias can hinder the general effectiveness and adaptability of a model in the wild. Hence, various images of different nationalities were selected. Three nationalities in total were represented:
   (a) African/Black/Black British/Caribbean
   (b) Asian/Asian British/Arab
   (c) White/Caucasian
5. Gender - In order to further reduce bias, images of both women and men were selected and sourced.
6. Lighting - Is the image well lit or under-lit or mid-lit? Efforts were made to select images that tick one of the previous options.

Figure 10 shows the spread of the different criteria used in selecting, sourcing and sorting the images for the secondary data while figure 11 shows some of the images used to train the model.

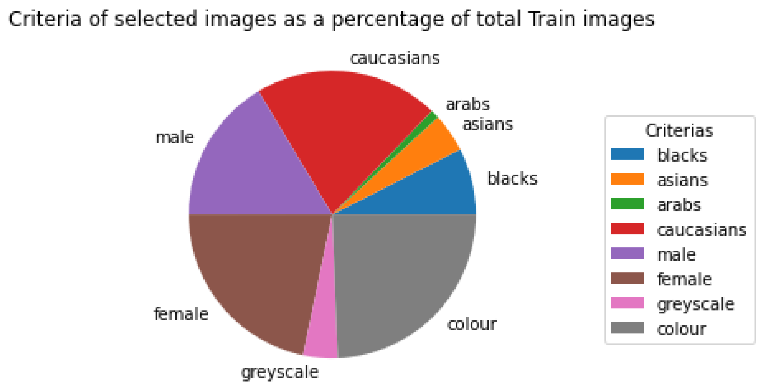


Figure 10: Criteria of selected images as a percentage of total Train images. The total amount of train images was 167.

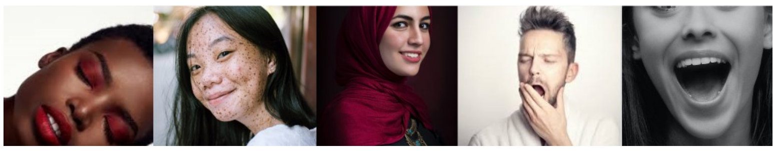


Figure 11: A selection of different images used in training the model. The criteria from left to right: nationality - black, asian, arabian, caucasian and colour - greyscale

Algorithm used for capturing Live Data

```
**Require:** $computer\_vision\_package = cvp$
**Ensure:** $variable\_name = cvp.VideoCaptureClass(id\_of\_video\_capture\_device)$
  **while** $variable\_name.method\_to\_check\_if\_webcam\_is\_open$ **do**
    $object\_1, object\_2 = variable\_name.method\_to\_read\_images$
    $image\_read = classification\_model(object\_2)$
    **if** $cvp.wait\_for\_pressed\_key\_method(delay\_in\_seconds)\ \&\ binary\_code\_of\_quit$ **then**
      $break\_out\_of\_loop$
    **end if**
    $variable\_name.method\_to\_release\_VideoCaptureClass$
    $cvp.method\_to\_destroy\_all\_image\_windows$
  **end while**
```

Micro-expressions - A Brief Detour

Micro-expressions are those semi-voluntary, often not lasting too long, expressions that our body makes frequently in different behavioral states (Zhao et al. 2023). For example whenever we meet someone we genuinely like, there's a web that pulls our eyes and our pupils expand (Greene 2018). There are also micro-expressions in other states like fatigue, drunkenness etc. A good example of a micro expression for fatigue can be the yawn. Whenever we yawn, more often than not it is a sign of either current fatigue or future fatigue; with the way we fold our hands(mostly one's dominant hand) to cover our mouths while yawning, this was a common micro-expression noticed in a lot of images sourced from our secondary data sources and the model was fed with these information. Figure 12 shows the various images which had this micro-expression and how the model performed in predicting their label. Other micro-expressions like the closing of the eyes(drowsy/sleepy) or the alertness level(wider opened eyes or a smile or laugh) or common gesticulations(like the stretch when one is tired) were also used to train the model.

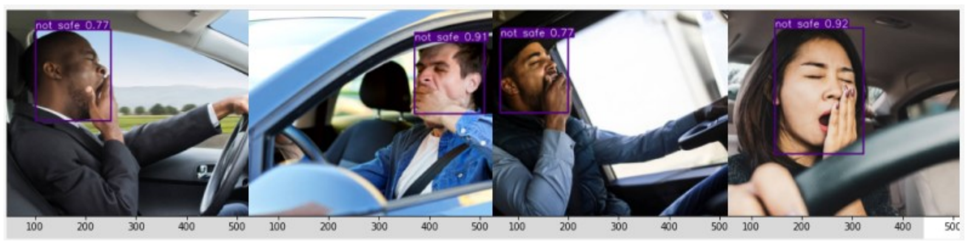


Figure 12: Showing the various fatigue/drowsiness micro-expressions. You can see in the image the prevalence of a yawn and the body movements that trail such a fatigue state.

Data Processing/Interpretation

Most of the Data Processing done for this project involved careful selection and sorting of images that would be able to train the model adequately to be able to classify images both captured and live into two categories - safe and unsafe.

In the process of selecting these images, various criteria were itemized and they have been elaborated in the previous chapter. However, there was one more step that involved processing both the primary and secondary data before feeding them to the model.

Image Annotation

Image Annotation can be roughly defined as the process of labelling images present in a dataset for the purpose of training a machine learning model. It is an important stage in the Computer Vision Machine Learning process and one important reason for this step is main ly o sim plify access to these selected images, using them as the ground-truth for further classification and object detection tasks (Hanbury 2008). Labelling or annotating a picture is paramount in Computer Vision tasks and it brings to the fore the perceived difficulties present in comprehending a visual scene (Barriuso & Torralba 2012). For this project, the open-source tool LabelImg was used to annotate the train images into the two classes - safe and unsafe.

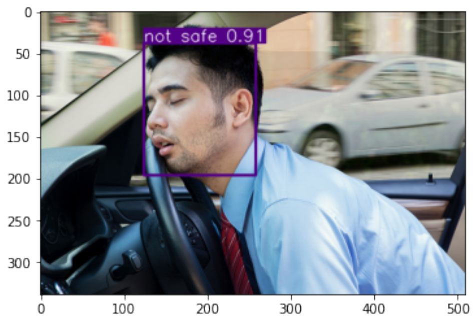


Figure 13: An example of an annotated image from the Test data which HOMESWEET has correctly classified as 'not safe' and the decimal number beside it is the model's confidence level of the predicted label.

Tools, Equipments and Libraries

Below is a list of the tools, equipments and libraries in the Data Processing, Analysis and Interpretation stage.

1. Python - Programming Language
2. Jupyter Notebooks - Integrated Development Environment
3. Open CV - This is an open source Computer Vision library.
4. Numpy - A popular library used for manipulating arrays.
5. YOLO - A Computer Vision object detection algorithm framework.
6. LabelImg - an open source tool used for Image Annotation.
7. Pytorch - A python deep learning framework.
8. Matplotlib - A python visualization library. This was used to view images, capture live data and view the model predictions.
9. Webcam - The primary sensor used to capture live and primary data.

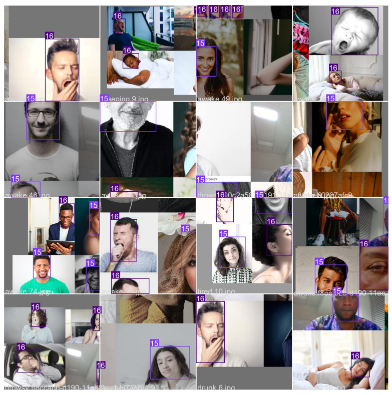


Figure 14: Train Batch of the 'HOMESWEET' model

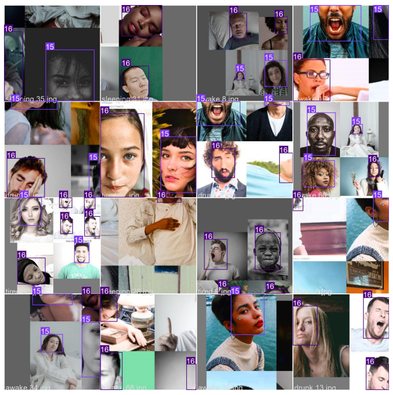


Figure 15: Train Batch of the 'HOMESWEET' model

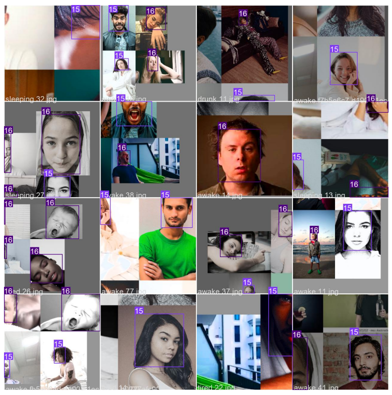


Figure 16: Train Batch of the 'HOMESWEET' model

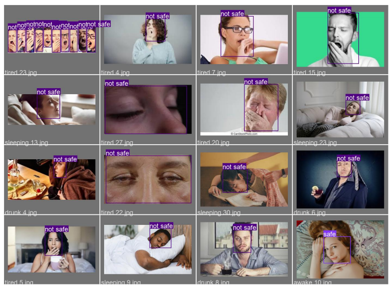


Figure 17: Validation Batch of the 'HOMESWEET' model

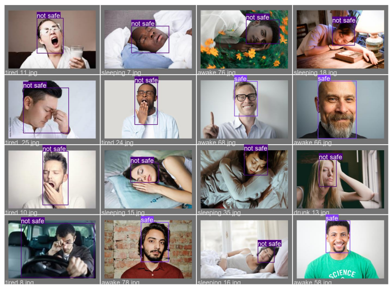


Figure 18: Validation Batch of the 'HOMESWEET' model

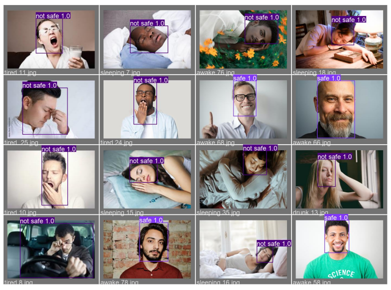


Figure 19: Validation Batch of the 'HOMESWEET' model

Model Performance on Selected Train Images

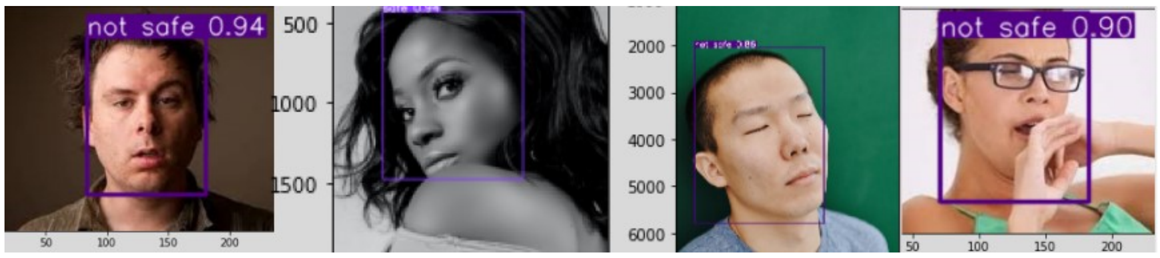


Figure 20: Selected images from the train data showing HOMESWEET'S predicted label annotations.

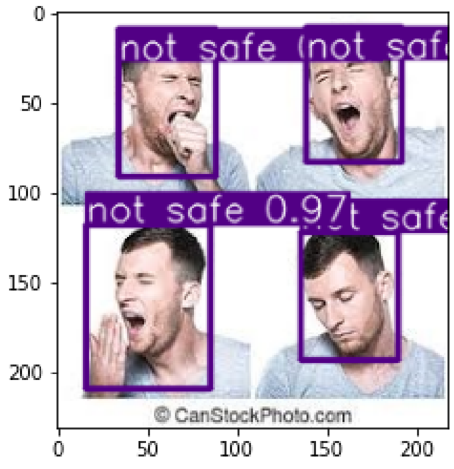


Figure 21: Selected images from the train data showing HOMESWEET'S predicted label annotations.

0.1 Model Performance on Selected Test Images

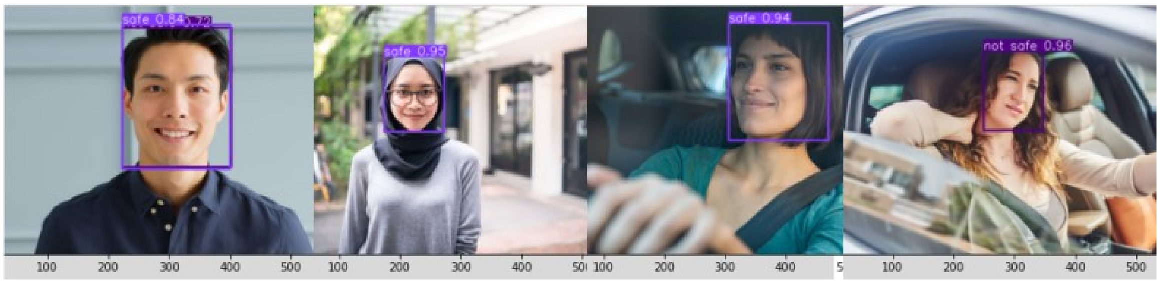


Figure 22: Selected images from the test data showing HOMESWEET'S predicted label annotations.

System Output & Feedback

Various Fidelity System Design and Output for Smartphones

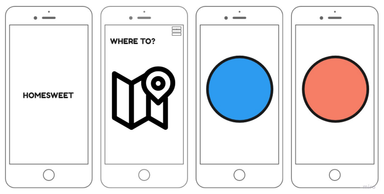


Figure 23: Snapshot of the low fidelity interface and feedback system for tablets

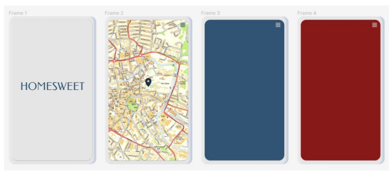


Figure 24: Snapshot of the high fidelity interface and feedback system for smartphones

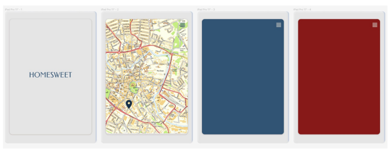


Figure 25: Snapshot of the high fidelity interface and feedback system for tablets

Feedback System

Two feedback systems were created in the application as a signal to the user based on the user's label category. There are:

1. Visual Feedback
2. Sonic Feedback

Visual Feedback

Figure 26: the 'sweet blue'(rgb: 49, 84, 116) used for the safe class to signal safe driving condition to the user

Figure 27: the 'sour red'(rgb: 135,25,25) used for the 'not safe' label to signal not safe driving condition to the user - prior or during

Sonic Feedback

One of the drawbacks of the Visual Feedback is that it can be easy to ignore. While this can be tolerated in the safe mode, in the not safe mode, when the driver/motorist is drowsy or sleepy or quite drunk, this can lead to unintended consequences. Hence why it is important to have another layer of signal feedback to the system.

The sonic feedback is a simple and short feedback whose repetition would depend on the amount of confidence level the system has predicted from the driver in the not safe label. A sonic feedback design system is important when creating interactive and augmented everyday contextual environments (Monache et al. 2010). The higher the confidence level, the more repetitive the sonic alert would be and vice versa. The purpose is to be a bit annoying or inconvenient to the driver so they can tap the screen - upon which sonic alert would stop and the mode would revert to the safe label since the driver is now awake. Care would be taken to design and implement this sonic feedback so that the User Experience of 'HOMESWEET' would not be affected.

Design Assets used in creating the User Interface and User Experience

- Typography

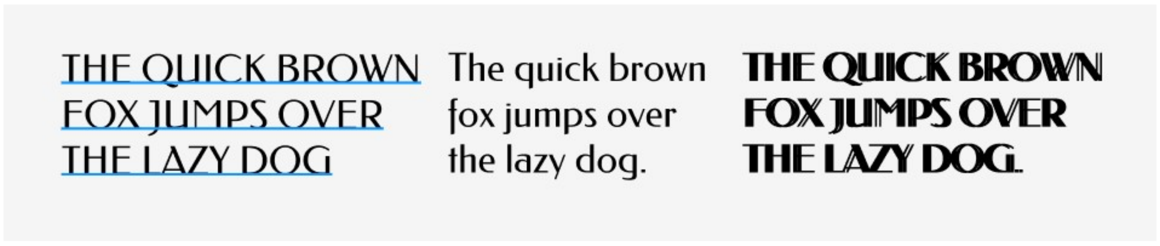

THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG
The quick brown fox jumps over the lazy dog.
THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG.

Figure 28: Typography used in the interface system. It is called the Federo typeface. The third image by the extreme right is a remixed version of the same type by the author.

- Colour Palette

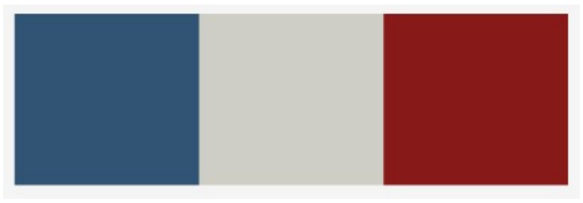


Figure 29: Color Palette used to design the interface. From left to right - primary, secondary & neutral, and tertiary colours

Results

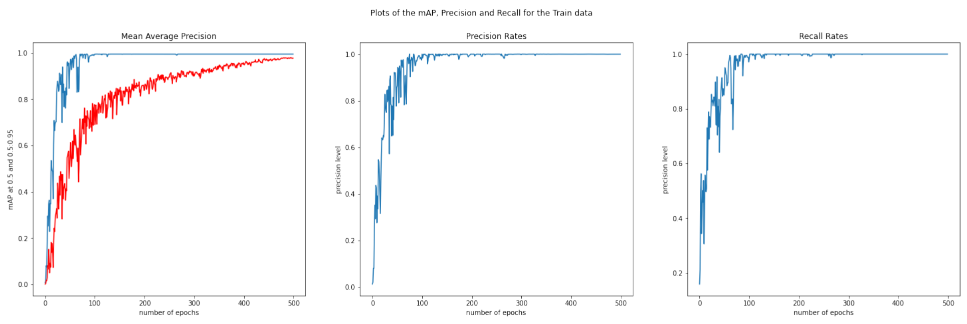


Figure 30: Plot showing the mAP, Precision and Recall rates for the train data model

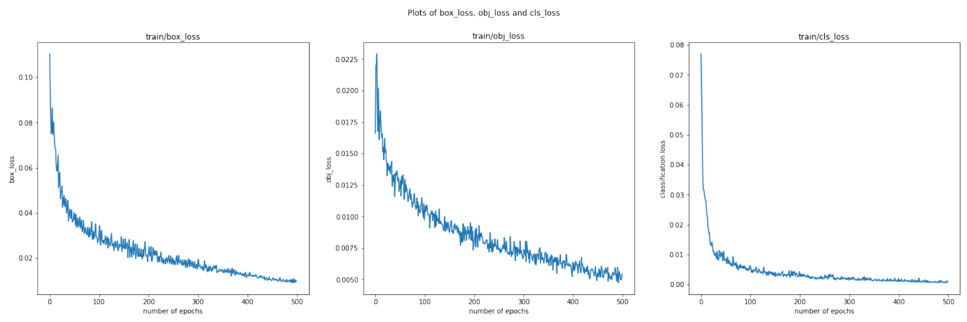


Figure 31: Plots showing the downward trend of the train box__loss, obj__loss and classification loss as the number of epochs increases

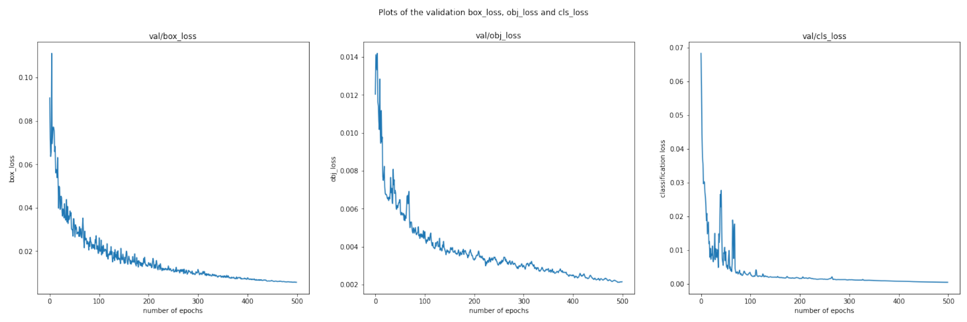


Figure 32: Plots showing the downward trend of the validation box__loss, obj__loss and classification loss as the number of epochs increases

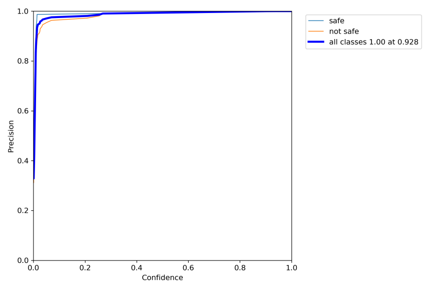


Figure 33: Plot showing Precision Curve with the Confidence Level in the x-axis

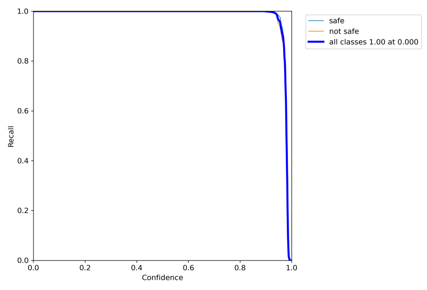


Figure 34: Plot showing the Recall curve with the confidence level in the x-axis

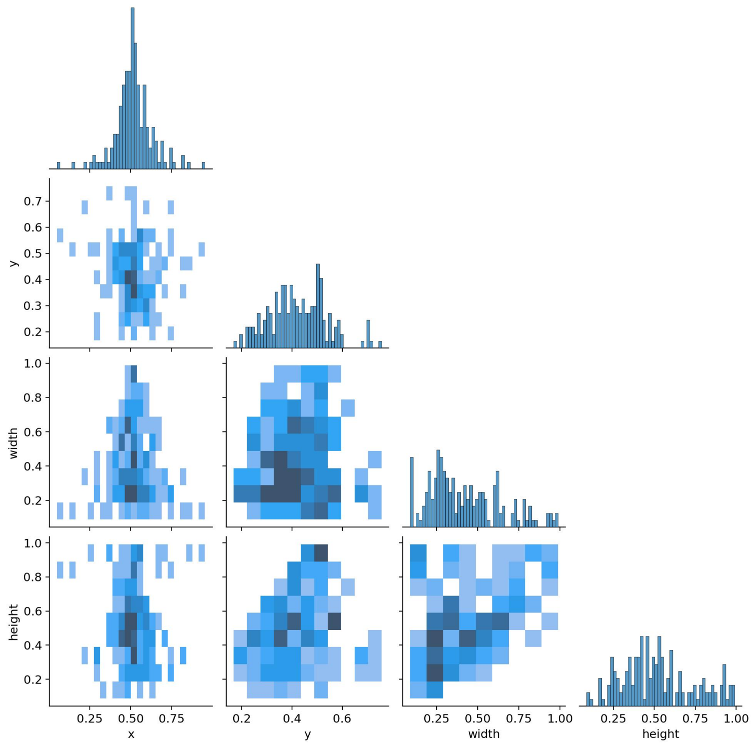


Figure 35: Different plots of the labels correlogram, the part of the image segmented during the image annotation phase; this plot shows the model capturing those image segments

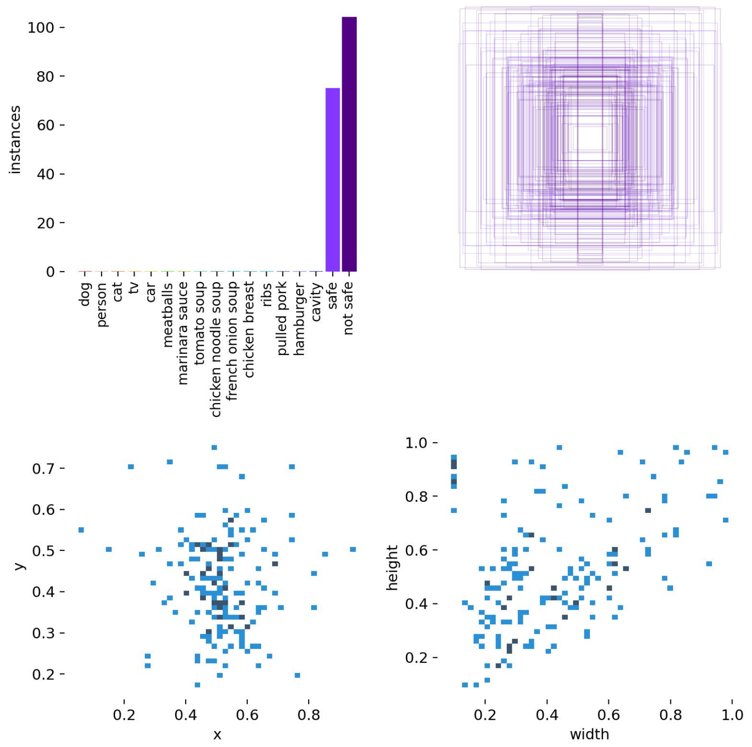


Figure 36: Different plots showing the models activity in the different labels. The top left plots the different labels against instances and the other labels not relevant to the research are from the YOLO algorithm. The other plots provide various information like the width and height of the different label images.

Interpretation of the Training Data Results

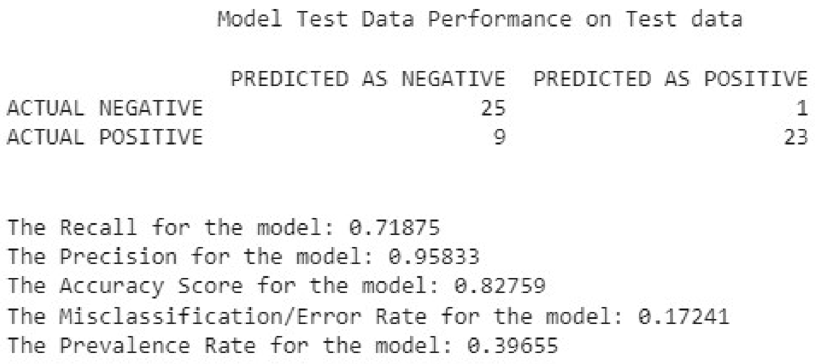
Figures 23-29 show the different classification metrics and performance of the model on the train data. There are a few things to note in the results:

1. The model achieved a high precision and recall rate as can be seen in figure 23 and also in figures 26 and 27.
2. The image annotation preprocess stage was effective as can be seen in figure 28.
3. The model was able to differentiate the two labels adequately as can be seen in top left image of figure 29.
4. The f1 score of the model on the train data is fairly good, varying over different confidence levels before settling at 0.5.

While the model seems to have performed well in the train data, in order to create a Generalized Model, we must test its mettle in the wild.

HOMESWEET IN THE WILD: Test Data Analysis and Results

A total of 66 images (39% of the total data) were used to test the generalized ability and predictability of the model. Of this number, the model found no detections in 5. Figure 30 below shows a table of the various Classification metrics of the model.



```
                 Model Test Data Performance on Test data

                  PREDICTED AS NEGATIVE  PREDICTED AS POSITIVE
ACTUAL NEGATIVE                      25                      1
ACTUAL POSITIVE                       9                     23


The Recall for the model: 0.71875
The Precision for the model: 0.95833
The Accuracy Score for the model: 0.82759
The Misclassification/Error Rate for the model: 0.17241
The Prevalence Rate for the model: 0.39655
```

Figure 37: The table shows both the Confusion Matrix and the Classification Performance of the Model on the Test Data

Interpretation of the Test Data Results

Figure 30 shows the classification performance of the model on the selected test data. We can see that the model has a Recall rate of 0.71875, a Precision of 0.95833 and an Accuracy score of 0.82759. This means that the model has a high specificity rate and a reasonably good sensitivity rate. The model's accuracy score is also good at 0.82759 and it must be noted that this accuracy can be improved over time as the model is trained with more relevant pictures.

The size of the train images would also have an impact on the strength and generalized nature of the model. This model was trained with only 167 images, a very small number in the Computer Vision domain. A good place to start with training a model would be north of

100,000 - 1,000,000 images and a test data that would be between 15 - 20% of the train data. For this project however, the concern was on taking little appreciable steps and further down the line, the model would be fed with a larger amount of varied and relevant data.

Despite this shortcoming of train data size, the model still performed superbly in the wild which means it is ready for deployment. Section 4.9, shows the predictive ability of the model on randomly selected test data from the two classes - safe and not safe.

Model's Test Data Predictions

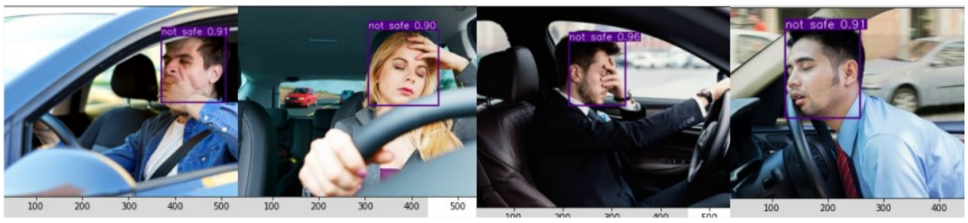


Figure 38: Some selected test data showing the model's performance at classifying the images into the not safe label

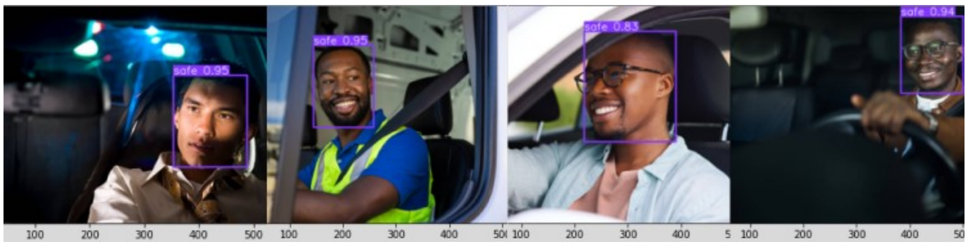


Figure 39: Some selected test data showing the model's performance at classifying the images into the safe label

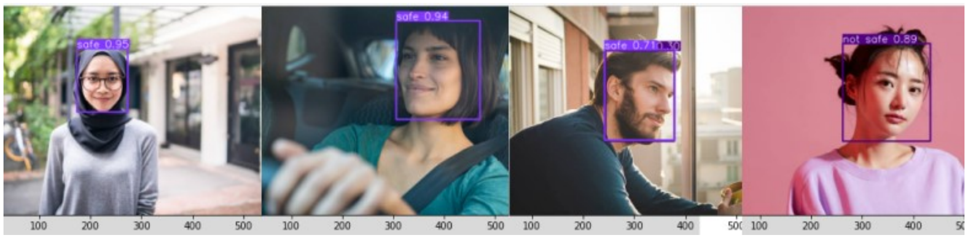


Figure 40: More selected test data showing the model's performance at classifying the images into the safe label

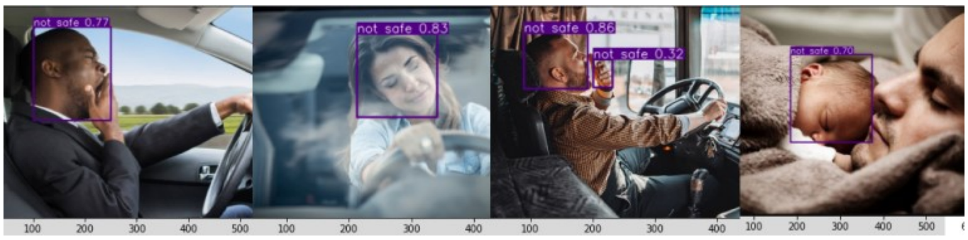


Figure 41: More selected test data showing the model's performance at classifying the images into the not safe label

Conclusion

With an accuracy score of 0.82759, the 'HOMESWEET' model has proven to be an effective deep learning model able to predict two class labels - safe and not safe - driving habits. The model has a high precision rate but its recall rate is not as high, this means that there would be times in the real world scenarios where it may incorrectly predict some class labels. While in other innocuous settings and applications, a recall of 0.71875 is good enough, in this application that aims to reduce road accidents and consequentially deaths, this may not be good enough.

This can be solved by feeding the model more data and continually providing it with information under various criteria like lighting conditions, accessories on the face, more micro expressions, more nationalities, backgrounds with artificial lighting etc to further improve 'HOMESWEET' accuracy score and recall rate and make it a more genralized deep learning model.

Links

UI/UX and Feedback signals - https://codepen.io/laresamdeola/pen/BaYwmoz Deep Learning Model Code - https://github.com/laresamdeola/homesweet/blob/main/HOMESWEET.ipynb

References

Barriuso, A. & Torralba, A. (2012), 'Notes on image annotation', *arXiv preprint arXiv:1210.3448* .

Dipu, M. T. A., Hossain, S. S., Arafat, Y. & Rafiq, F. B. (2021), 'Real-time driver drowsiness detection using deep learning', *International Journal of Advanced Computer Science and Applications* **12**(7).

Greene, R. (2018), *The laws of human nature*, Penguin.

Hanbury, A. (2008), 'A survey of methods for image annotation', *Journal of Visual Languages & Computing* **19**(5), 617–627.

Miao, X., Xue, C., Li, X. & Yang, L. (2022), 'A real-time fatigue sensing and enhanced feedback system', *Information* **13**(5), 230.

Monache, S. D., Polotti, P. & Rocchesso, D. (2010), A toolkit for explorations in sonic interaction design, *in* 'Proceedings of the 5th audio mostly conference: a conference on interaction with sound', pp. 1–7.

ons.gov.uk (2019), 'Road casualties'.
URL: *https://www.ons.gov.uk/aboutus/transparencyandgovernance/freedomofinformationfoi/roadcasualties*

Sharifani, K. & Amini, M. (2023), 'Machine learning and deep learning: A review of methods and applications', *World Information Technology and Engineering Journal* **10**(07), 3897–3904.

Singh, J., Kanojia, R., Singh, R., Bansal, R. & Bansal, S. (2023), 'Driver drowsiness detection system: An approach by machine learning application', *arXiv preprint arXiv:2303.06310* .

statista.com (2019), ‘Distribution of contributing factors leading to road accidents in great britain in 2019’.
URL: *https://www.statista.com/statistics/323079/contributing-factors-leading-to-road-accidents-in-great-britain-uk/*

Yonekawa, K., Yonezawa, T., Nakazawa, J. & Tokuda, H. (2009), Fash: Detecting tiredness of walking people using pressure sensors, *in* ‘2009 6th Annual International Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous’, IEEE, pp. 1–6.

Zhao, G., Li, X., Li, Y. & Pietikäinen, M. (2023), ‘Facial micro-expressions: An overview’, *Proceedings of the IEEE* .