Keeping Drivers Alert: A Solution for Monitoring Driver Attention in Assisted-Driving Vehicles

Robert Jutreša*, Peter Peer* Žiga Emeršič*, Jihun Kim†
* Faculty of Computer and Information Science/Computer Vision Laboratory, Ljubljana, Slovenia
† School of Electronic and Electrical Engineering, Kyungpook National University, Daegu, South Korea
rj7149@student.uni-lj.si, peter.peer@fri.uni-lj.si, ziga.emersic@fri.uni-lj.si, soji@knu.ac.kr

***Abstract*—The advent of the new industrial revolution has led to a surge in the number of self-driving and assisted-driving vehicles on the roads. Although this development enhances the overall quality of life, it also poses new challenges and risks. One such difficulty is the need for driver attention in the event of system errors. To overcome this issue, a driver monitoring system is necessary to ensure that the driver remains focused and can take control if necessary, while also sending alerts in case of driver distraction. To address this challenge, we propose a method that employs a convolutional neural network based on the ResNet architecture. This system can recognize selected types of driver distractions in captured infrared images. We experimented with different depths of the ResNet architecture to determine the most optimal results. The best model achieved an accuracy rate of 94% in detecting various forms of driver distraction.**

***Keywords*—drowsy driving, distraction detection, CNN, ResNet**

I. INTRODUCTION

The topic of Highly Automated Driving (HAD) has been widely discussed in recent times, with leading companies such as Tesla and Waymo spearheading this technological revolution. However, there have been reports of accidents resulting from driver distraction or failure to intervene promptly if necessary. This research seeks to address this issue through a feasibility study of deploying a Convolutional Neural Network (CNN) model in a Driver State Monitoring System (DSMS) to achieve accurate classification outcomes. To this end, a dataset of driver images engaged in distraction tasks was collected, and a model was chosen based on related work, the implementation of which was then optimized. Our objective is to develop a model that can accurately identify driver distraction in real-world driving scenarios.

II. RELATED WORK

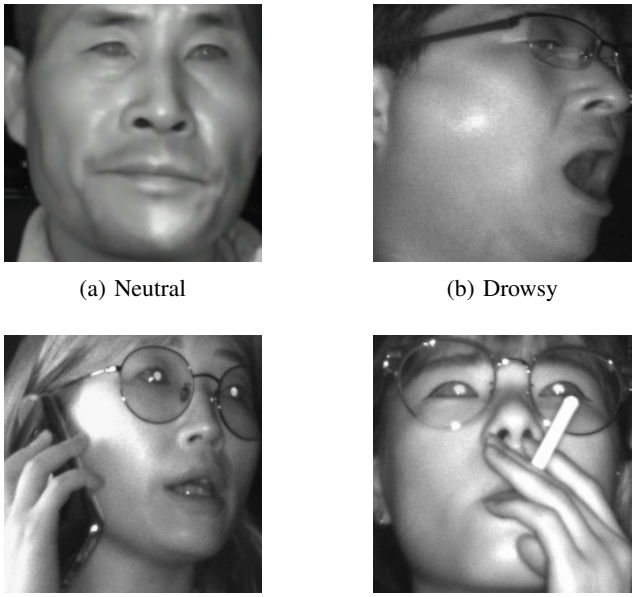
Gonçalves et al. [1] defined the important monitoring characteristics of DSMS for HAD. They said this introduces a new paradigm where driving performance metrics are no longer viable, and approaches like detecting engagement in nondriving tasks or fatigue-countering behaviors are becoming more important. They defined several distraction and fatigue indicators, based on eye and body behavior. Among these yawning was the one we focused on the most.

In 2021 Kashevnik et al. published a literature review paper [2], in which they outlined the process of developing a distraction detection system from sensor data acquisition to data processing and so forth. They also reviewed multiple papers that had neural network (NN) based solutions, which were using either self-created datasets or the AUC Distracted Driver Dataset [3], [4], the purpose of which is detecting driver distraction using driver posture from a side view. Most of the reviewed papers achieved an accuracy of 92% and above, which we set as our target benchmark.
Mase et al. [5] also performed their tests on the AUC Distracted Driver Dataset. They employed the InceptionV3 CNN architecture achieving accuracy in line with accuracies reported in papers reviewed by Kashevik et al.
Craye et al. [6] based their approach on using RGB-D cameras to capture driver state and classifying into one of five distinct classes including making a phone call, drinking, etc. They used the AdaBoost classifier and Hidden Markov Model achieving an accuracy of 90% for distraction detection. This approach is most in line with our own among the reviewed literature and thus represents the secondary benchmark for our metrics. The big difference from our proposed approach is the focus on the position of the driver's arms through a 3D point cloud and gaze estimation through iris localization. In contrast, we decided to focus primarily on the usage of convolution filters.
Liu et al. [7] approached the detection of driver detection problems, using their own developed system and dataset, where they used Support Vector Machines (SVM) and other methods of Semi-Supervised Machine Learning, achieving results as high as 97% accuracy. While not related to our work, as they used head and gaze positions and rotations as measures for their classification, it is important to acknowledge this different approach as it can also produce valuable results.
Fernandez et al. [8] made a study on the success of different approaches and techniques when approaching the problem of driver distraction detection (including the above mentioned [6], [7]), where they outlined some of the important areas to consider when developing such a system. Among these, they drew attention to the importance of different factors that modulate distractions, including but not limited to factors outside of the vehicle (traffic,

road accidents) and the driver's emotional state.

III. METHODOLOGY

A. Data Overview



(a) Neutral (b) Drowsy

(c) Phone (d) Cigarette

Fig. 1: Examples of images from each class in the train set

For our research, we used a dataset provided by the National Information Society Agency (NIA) of South Korea. The Driver Status Information Images for Preventing Drowsy Driving dataset [9] was developed for use with AI technology and the development of service applications that monitor the driver's condition through changes in facial expressions and feature points. The dataset is composed of over $250,000$ images of frontal views of $1,000$ subjects using an infrared camera with corresponding labels captured in three different settings, those being real-world driving images, images from a semi-controlled environment, and images from a controlled environment. The labeling provided for the images consists of bounding box locations and locations of facial key points as well as driver identification (gender, age, etc.) and object information. The object information includes data about the presence or absence of various occlusions (sunglasses, masks, etc.) and objects such as phones and cigarettes, in addition to data about the subjects' eyes and mouth (visibility, etc.).

B. Model Selection

As our problem was at its core a multi-label problem, we decided it would be a good choice to take it on using a CNN-based approach [10]. Kashevnik et al. [2] reported that most reviewed papers had achieved the best quality results using a CNN with a ResNet [11] architecture. According to the available data we decided on testing various depths of this architecture.

IV. EXPERIMENTS

A. Image Selection and Preprocessing

Due to the size of the original dataset, the number of images had to be limited due to time and hardware limitations. This subset was selected from both real-world driving images and images from a controlled environment. The set of images from a semi-controlled environment was omitted. From these two groups, a random sample of $26,000$ images was selected without replacement. Among these, $6,500$ belong to each of the 4 classes that were defined using the provided labeling. These 4 classes included 3 that classified a form of distraction and one neutral class. The classification schema was as follows:

- Image was classified as "subject is distracted by a phone" if there is a phone present in the image (e.g. Fig. 1c).
- Image was classified as "subject is distracted by a cigarette" if there is a cigarette present in the image (e.g. Fig. 1d).
- If the mouth of the subject isn't visible, the image was classified as "subject is drowsy" if they had their eyes closed (assuming eyes aren't obscured), otherwise, the image was classified as "subject is drowsy" if they were yawning (mouth is opened) (e.g. Fig. 1b).
- If none of the above were applicable to an image it was classified as "neutral" (e.g. Fig. 1a).

The images were then cropped based in such a way that they included the face of the subject, and potentially present phones or cigarettes. Lastly, these images were morphed to the shape of 224×224 which is the standard input size of the ResNet architecture.

TABLE I: Values of Applied Perturbations

Feature	**Range**
brightness	$[0.8, 1.2]$
rotation [°]	$[-5, 5]$
width shift	0.1 of image width
height shift	0.1 of image height
shear [°]	$[0, 0.1]$
zoom	$[0.9, 1.1]$

B. Model set up and Parametrization

All models were implemented using the open-source Python library TensorFlow [12] and following their implementation instructions. The exception to this were the ResNet-18 and ResNet-34 implementations, where the publicly available image-classifiers pip package was used. All tested models used the same set of selected images split into the train and test sets with the latter consisting of 20% of all images. The train set was then additionally split into the validation set, again with a 20% ratio, resulting in 16640 images in the train set, 4160 images in the validation set, and 5200 images in the test set. The ratio of images between classes was preserved. When loading the images into the programs additional prepossessing was applied in accordance with the documentation and

perturbation to images in the test set. The ranges and values of applied perturbations can be seen in Table I and were selected based on what changes could be present in a real-world setting. The horizontal flip attribute is excluded from this table as it doesn't have a value to list (parametrization is boolean).

TABLE II: Batch Size and Parameter Count of Models

ResNet	Batch Size	Number of parameters	
		Trainable	Nontrainable
18	60	11,708,359	7,942
34	60	21,816,519	15,366
50	40	25,636,868	53,120
101	20	44,655,108	105,344
152	20	60,321,796	151,424

All of the tested models shared this configuration when loading the images as well as the previously defined input shape. In addition to this they initialized with weights trained on ImageNet [13], to take advantage of transfer learning, frozen on top of the model since we were working with a closed set evaluation problem and a 2D average pooling layer at the end of the model. To this was attached a dense layer utilizing the rectified linear activation function (ReLu) with an output size of 1024 nodes, and lastly a dense layer utilizing the softmax function to give out the required predictions. All models used the Adam optimizer [14] and a universal early stopping system configured to reduce the learning rate by a factor of 0.5 when necessary (i.e. after 20 epochs without improvement), and stop the learning process when it reaches $1e{-}10$. Categorical cross-entropy and categorical accuracy were also universally used as measures during training. The information about the batch size used for each model and the trainable/nontrainable parameter count of the models can be seen in Table II.

V. RESULTS AND DISCUSSION

The first section below presents the numerical results of the experiments, while the second section provides commentary on those results. It is important to take into context how the experiments were run. The models were trained and tested on a personal computer equipped with an AMD Ryzen 9 5950X 16-Core processor, 16 GiB$\times$4 3200 MHz RAM, NVIDIA GeForce RTX 3070 graphics card with 8 GiB of video memory, and a 1 TB Samsung SSD 970 EVO NVMe drive. Used software included the Tensorflow 2.11 library for GPU and CUDA 11.7, the latest version of the image-classifiers package as well as necessary accompanying packages. All the times were averaged over multiple runs.

A. Results

Several statistical and empirical data points were collected during training and testing. The first set of data, seen in Table III, represents both the training and evaluation time of each model depth. The times listed are the execution times of the model training code blocks. The inconsistencies here are attributed to the varied batch size for different models (set as high as allowed by the hardware limitations for each model) and the configured early stopping preventing further training of some models at an earlier stage than others.

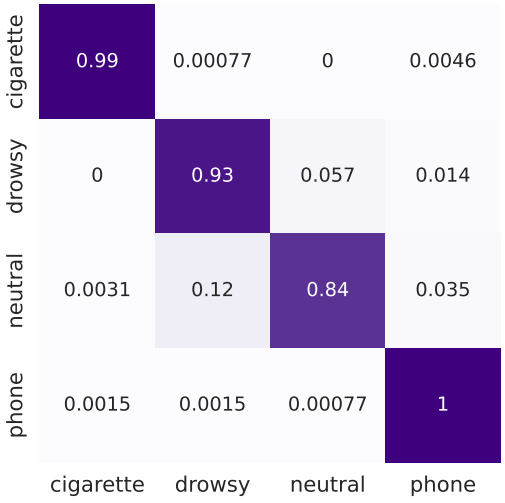


Fig. 2: ResNet-34 confusion matrix.

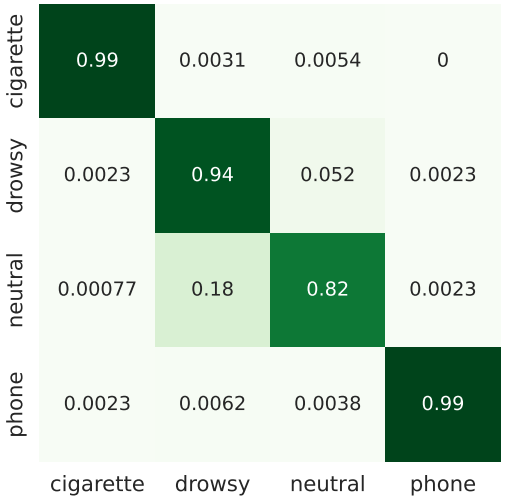


Fig. 3: ResNet-50 confusion matrix.

The results of statistical measurements done for model evaluation can be seen in the following figures. Fig. 2 and Fig. 3 show computed confusion matrices, showing the class-specific accuracy, of the two best-performing models according to statistical measurements done during model evaluation. Full results of this can be seen in Fig. 4. In the Figure the y-axis is limited, to clarify the differences between model implementations.

B. Discussion

When comparing the data it was important to consider the nature of the application of such a model. As the model is aimed toward HAD applications, it must be fast as well as accurate. With this in mind, it is quite easy to determine

TABLE III: Training and Evaluation time of various model depths

ResNet	Training time [min]	Evaluation time [s]	Batch Size	Epochs Training
18	60	7.1	60	26
34	120	7.2	60	51
50	77	8.7	40	34
101	163	16	20	55
152	109	22	20	28

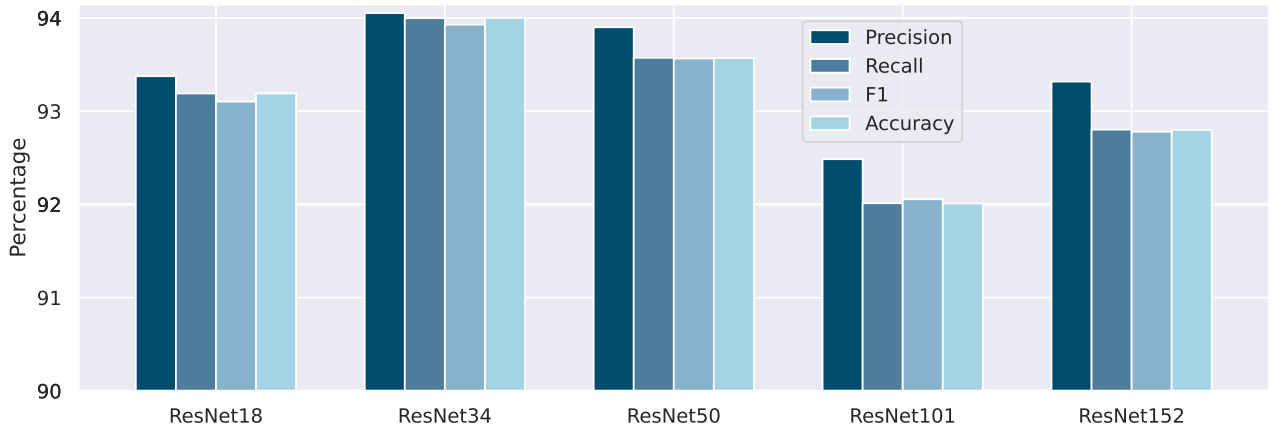


Fig. 4: Comparison of different model depths (y axis limited for clarity).

the optimal depth among the ones tested for the ResNet architecture.

Firstly looking at Table II, we can see that the ResNet-18 and ResNet-34 implementations would take up the least amount of resources based on their parameter count (size of the model), with the ResNet-50 implementation not far behind. The ResNet-101 and ResNet-152 implementations could be deemed too big for a lightweight computer installed in a car. When looking at Table III, we can see that all of the models would perform well in a real-world situation when regarding their evaluation speed of a provided image. According to Zhuk et. al. [15] the response time of a driver is between 0.773 and 2.43 seconds, meaning that even the slowest model (ResNet-152 which takes 4ms to evaluate a single image) would be sufficient to classify the current state of the driver in negligible time. The training speed is in this case, not an important factor as models would be installed in the HAD system with already trained weights.

Next, we should look at the metric data seen in Fig. 4 (note that the y axis is limited for clarity). We can see that again the lower-depth versions return better results when it comes to classification. In particular, the ResNet-34 implementation outperforms the others across all metrics, followed by the ResNet-50 implementation.

Lastly, to compare these two we look at the produced confusion matrices seen in Fig. 2 and Fig. 3. Even tho they are mostly similar, we can see that ResNet-34 has slightly higher rates of classifying the images correctly, as well as a slightly lower rate of miss-classifying the neutral as the drowsy class.

This was the most common problem when it came to the classification of every implemented depth of the ResNet architecture. As the images in the phone and cigarette classes had the items in question present in the image, and the models were loaded with pre-trained weights on the ImageNet dataset, the classification of these posed no issues. On the other hand, the drowsy and neutral classes had to be fine-tuned, and due to the nature of the dataset or our process of sampling, the images from the two classes might not have been distinct enough.

To better understand the classification described in Section IV-A, an example of the provided labels can be seen in Fig. 5. The labels presented locations and annotations for selected facial features and objects, and thus some generalization had to be done when classifying images based on those labels. While the "phone" and "cigarette" classes were straightforward, the labels didn't allow for much variability when choosing the features for the "drowsy" class. Thus we looked at the visibility and openness of the mouth and eyes in order to classify images as "drowsy". For this, we made assumptions that the subject was alone in the vehicle, and thus not talking to anyone, as well as that the potential miss-classification of a non-drowsy state as drowsy in a real-world scenario is better than vice versa. Because of these assumptions, no images, where the mouth was labeled as opened, were classified as "neutral" only "drowsy".

VI. CONCLUSION

The results of this research are promising. Based on available information a fitting CNN model was selected, and experiments were run to find the optimal implementation of the selected model. In the end, this proved to be the ResNet-34 implementation, proving to not only produce the most accurate classification

```
{
    "FileInfo": {
    ...
  },
  "UserInfo": {
    ...
  },
  "Accessory": {
    "Mask": false,
    "Glasses": false,
    "Cap": false
  },
  "Annotation": 1,
  "ObjectInfo": {
        "KeyPoints": {
            ...
        },
        "BoundingBox": {
      "Face": {
        "isVisible": true,
        "Position": [ ... ]
      },
      "Leye": {
        "isVisible": true,
        "Opened": true,
        "Position": [ ... ]
      },
      "Reye": {
        "isVisible": true,
        "Opened": true,
        "Position": [ ... ]
      },
      "Mouth": {
        "isVisible": true,
        "Opened": true,
        "Position": [ ... ]
      },
      "Cigar": {
        "isVisible": false,
        "Position": [ ... ]
      },
      "Phone": {
        "isVisible": false,
        "Position": [ ... ]
      }
    }
    }
}
```


Fig. 5: Data Labeling Schema

while remaining lightweight enough for it to be feasibly implemented in a HAD application. In addition to this we also matched and surpassed the set benchmarks outlined in Section II.

Possible improvements and future work could include further tests on already mentioned implementations, in an attempt to improve both the accuracy and portability of the model. This could include expanding the number of samples taken from the database, testing it on other possible databases, introducing new classes, freezing layers of the model, or in other ways altering the training to lower the number of parameters. Furthermore, an actual prototype setup involving both an infrared camera and a computer to test the appropriateness of the model in a real-world application could be made, together with long-term tracking of both – motion and emotion.

REFERENCES

[1] J. Goncalves and K. Bengler, "Driver state monitoring systems–transferable knowledge manual driving to had," *Procedia Manufacturing*, vol. 3, pp. 3011–3016, 2015.

[2] A. Kashevnik, R. Shchedrin, C. Kaiser, and A. Stocker, "Driver distraction detection methods: A literature review and framework," *IEEE Access*, vol. 9, pp. 60 063–60 076, 2021.

[3] Y. Abouelnaga, H. M. Eraqi, and M. N. Moustafa, "Real-time distracted driver posture classification," *CoRR*, vol. abs/1706.09498, 2017. [Online]. Available: http://arxiv.org/abs/1706.09498

[4] H. M. Eraqi, Y. Abouelnaga, M. H. Saad, and M. N. Moustafa, "Driver distraction identification with an ensemble of convolutional neural networks," *CoRR*, vol. abs/1901.09097, 2019. [Online]. Available: http://arxiv.org/abs/1901.09097

[5] J. M. Mase, P. Chapman, G. P. Figueredo, and M. T. Torres, "A hybrid deep learning approach for driver distraction detection," in *2020 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 2020, pp. 1–6.

[6] C. Craye and F. Karray, "Driver distraction detection and recognition using RGB-D sensor," *CoRR*, vol. abs/1502.00250, 2015. [Online]. Available: http://arxiv.org/abs/1502.00250

[7] T. Liu, Y. Yang, G.-B. Huang, Y. K. Yeo, and Z. Lin, "Driver distraction detection using semi-supervised machine learning," *IEEE transactions on intelligent transportation systems*, vol. 17, no. 4, pp. 1108–1120, 2015.

[8] A. Fernández, R. Usamentiaga, J. L. Carús, and R. Casado, "Driver distraction using visual-based sensors and algorithms," *Sensors*, vol. 16, no. 11, p. 1805, 2016.

[9] N. I. S. Agency, "Driver condition information video to prevent drowsy driving," 2021, accessed on 9. 12. 2022. [Online]. Available: https://aihub.or.kr/aihubdata/data/view.do?currMenu=115&topMenu=100&aihubDataSe=realm&dataSetSn=173

[10] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu, "Cnn-rnn: A unified framework for multi-label image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[12] T. Developers, "Tensorflow," 2022. [Online]. Available: https://doi.org/10.5281/zenodo.7604251

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] M. Zhuk, V. Kovalyshyn, Y. Royko, and K. Barvinska, "Research on drivers' reaction time in different conditions," *EasternEuropean Journal of Enterprise Technologies*, vol. 2, pp. 24–31, 04 2017.