Mental Workload Prediction from EEG Data using Machine and Deep Learning Methods

1st Donald Pfaffmann
Ubiquitous Media Technology Lab
Deutsches Forschungszentrum für Künstliche Intelligenz
Saarbrücken, Germany
dopf00001@stud.uni-saarland.de

2nd Ganavi Basavaraju
Ubiquitous Media Technology Lab
Deutsches Forschungszentrum für Künstliche Intelligenz
Saarbrücken, Germany
gaba00002@stud.uni-saarland.de

3rd Nicolas Tabellion
Ubiquitous Media Technology Lab
Deutsches Forschungszentrum für Künstliche Intelligenz
Saarbrücken, Germany
s8nitabe@stud.uni-saarland.de

***Abstract*—This report explores the possibility of alarming drivers during times that they are cognitively distracted in order to reduce car accidents. To this end, brain waves of drivers have been recorded using electroencephalography during times of non-distracted and distracted driving. A state of distracted driving is artificially induced by having drivers perform a secondary task that invokes cognitive load, thus reducing performance on the primary task, i.e. driving. Methods of machine and deep learning are utilized in an attempt to build a system that is capable of distinguishing between non-distracted and distracted driving based on the cognitive load of drivers during live, realistic driving scenarios. Once cognitive distractions have been detected, an interface alarms drivers with visual and auditory cues, allowing them to disengage from actively driving by transferring control over to the autonomous vehicle. A specifically constructed experimental setting is utilized for validating the conceptual feasibility of this approach. The best performing machine learning classification algorithm exhibits an accuracy of 89.74%.**

***Index Terms*—EEG, BCI, cognitive load, Machine Learning, Neural Network**

I. Introduction

In the United States alone, over 3,100 people were killed and about 424,000 were injured in crashes involving a distracted driver in 2019, according to the National Highway Traffic Safety Administration[1]. Furthermore, it has been known for years that the prime cause (72%) of car accidents is human error [1]. Distractions can be of numerous kinds: They are either physical (e.g. taking your hands off the steering wheel), visual (e.g. taking your eyes off the road) or cognitive (e.g. "taking your mind off driving"). In this seminar project, we are interested in detecting cognitive distractions by estimating cognitive load (CL) of drivers in real-time during realistic driving scenarios. CL can be estimated by using subjective or objective measurements. Subjective measurements are widely used in the scientific field of psychology and include, but are not limited to, questionnaires, such as the NASA-TLX. A major drawback of subjective measurements is that they do not lend themselves to real-time evaluation. Therefore, objective measurements of the psycho-physiological factors of CL must be utilized. For doing so, electroencephalography (EEG) is used to measure the driver's brain activity during multiple scenarios. Ultimately, the goal of this seminar project is to train machine learning algorithms on historic EEG data, such that they are capable of predicting the CL of drivers from live EEG data. The prediction distinguishes between low and high cognitive load. Based on the current cognitive load, drivers are then prompted to disengage from actively driving themselves and activate autonomous driving. This transfer of control during times of cognitive distraction of the driver might help in reducing traffic accidents.

[1]https://www.nhtsa.gov/risky-driving/distracted-driving

II. Experimental Setup

A major design choice is how a high cognitive workload is being induced. In the context of this seminar project, the widespread primary/secondary task paradigm is being utilized [2]. In this paradigm, the primary task is taken as a baseline measure of cognitive workload and compared to the data obtained when additionally performing a secondary task. Since we are concerned with the CL of drivers during realistic driving scenarios, the primary task is chosen to be that of driving. For the secondary task, an auditory 2-back task is used. The n-back task is a continuous performance task that is commonly used as an assessment in cognitive neuroscience and has effectively proven to be one of the best ways to measure part of working memory and its capacity [3], [4], [5].

In the process of achieving the aforementioned use-case, it has to hold true that the secondary task increases cognitive load sufficiently - where sufficiently is defined by the ability of a machine learning classifier to predict the scenario of a given data point at a rate that is at non-trivially better than chance. In essence, this amounts to testing the effectiveness

of the primary/secondary task paradigm in this particular experimental setting.

Other ways of inducing high CL have also been suggested in the scientific literature. [6] and [7] argue that using a combination of the variables traffic occurrence and road complexity sufficiently induces cognitive load in drivers. In this context, an easy road segment (straight road) paired with low traffic occurrence corresponds to low CL and a difficult road segment (intersection, roundabout, etc.) in combination with high traffic (rush hour) corresponds to high CL.

The primary and secondary tasks define two scenarios: *driving* and *driving with 2-back*. In addition to these, two more scenarios are recorded, namely *idle* and *2-back*. In the *idle* scenario the subject is sitting with eyes closed as still and relaxed as possible. During the *2-back* scenario the subject is exclusively performing the secondary task, while sitting still with eyes closes. The two scenarios are recorded in the hope of gaining further insight into the relationship between *driving* and *driving with 2-back*. Against this background, some interesting questions are:

- What is the cognitive load of the primary task (*driving*) in comparison to an overall baseline (*idle*)?
- Is the cognitive load of *driving with 2-back* the sum of the loads of *driving* and *2-back*?
- Are the cognitive loads associated with each scenario clearly different from each other?

We expect to see the designed conceptual differences between scenarios to be reflected in the subjects' brain activity, with *idle* evoking the lowest and *driving-with-2-back* evoking the highest CL. The CL induced by *driving* and *2-back* is expected to be in the interval spanned by the former two, but the exact relationship between *driving* and *2-back* is difficult to discern apriori.

In general, data will be recorded from three subjects for 5 minutes for each scenario. A total of three independent recording sessions have been conducted, resulting in a total recording duration of 3 hours. The 3 hours can thus be split into 3 x 1 hour individual recording sessions, which in turn are split into 4 x 15 minutes of recorded EEG data for each scenario. A scenario can be further divided into 3 x 5 minutes recording blocks, where each block is associated with a single subject.

A. Hardware

EEG data was recorded using the wearable EEG-Headset Unicorn Hybrid Black[2]. A laptop or PC connects with the headset via Bluetooth. The Unicorn Hybrid Black allows to record EEG data using 8 electrodes (Fz, C3, Cz, C4, Pz, P07, Oz, P08) with 24-Bit and 250Hz.

B. Software

In order to simulate realistic driving scenarios, OpenDS[3] was deployed. In an effort to streamline working together on code in a group of three, Google Colaboratory was utilized. All programming has been done in Python. For machine and deep learning algorithms the Scikit-learn and Keras libraries have been used. Furthermore, a proprietary GUI (Unicorn Hybrid Black Suite) and library (Unicorn Python API Hybrid Black[4]) have been utilized for establishing communication between laptop/PC and headset. Additionally, pre-processing EEG data was done with the MNE library[5]. For the interface Tkinter[6] has been used.

III. Data Evaluation

The simplest and most straightforward way to validate if the secondary task sufficiently increases cognitive load, would be to simply feed the raw data to a ML algorithm. While this simplicity does sound appealing, it comes with a major drawback. In the context of the raw data, a single data point is a 250^{th} of a second, or 4 milliseconds long, which is an extremely tiny time frame for a model to make an accurate prediction. Furthermore, the raw data might exhibit a low signal-to-noise ratio (SNR) because of various artifacts, such as muscle and eye movements as well as inter-subject and inter-session variations. Therefore, it is common practise to pre-process the raw EEG data, which includes, but is not necessarily limited to: filtering, trimming, standardization, merging, averaging and manually computing a set of features that is ultimately passed to a ML algorithm as input.

A. Data Pre-processing

Several pre-processing steps are required before the raw EEG data can be used to train ML algorithms.

First and foremost, the recorded data has to be cropped to roughly 5 minutes. Usually, the .fif recording files are several minutes longer, because of the practical implications of assuring the adequate usage of the EEG headset (proper positioning of the headset, proper contact of the electrodes with the skull, etc.). This recording overhead is just noise and must be discarded. Each data set pertaining to a specific session, scenario and subject is thus of the shape (number of channels, sampling frequency x length of the data set in seconds). Concretely speaking, this amounts to (8, 90.000).

The cropped EEG data exhibits low frequency drifts and power line noise, which can both be removed by a band-pass filter that removes very low (0-0.1Hz) and very high frequencies (49.9Hz and above). For doing so, a finite impulse response (FIR) filter with a hamming window has been used.

In order to prevent outliers from disproportionately affecting the standardization procedure, extreme values at the very low and high end of the value spectrum are replaced by the 1^{th} and 99^{th} percentile, respectively. Afterwards, the data is standardized to have zero mean and a standard deviation of one.

For all following processes, it is important to note that we do not differentiate data according to subjects. Instead,

[2]https://www.unicorn-bi.com/de/

[3]https://opends.dfki.de/

[4]https://www.unicorn-bi.com/de/python-api/

[5]https://mne.tools/stable/index.html

[6]https://docs.python.org/3/library/tk.html

the data of all subjects is merged, randomly shuffled and subsequently seen as one homogeneous data set of 15 minutes recording time for each scenario and session. This approach produces a generalized model, that is trained on data from all subjects. Thus, each data set pertaining to a specific session and scenario is of the shape (8, 225.000). This cross-subject paradigm is contrasted by the intra-subject paradigm, where a personalized model is trained and tested for each individual subject respectively. It is assumed that the intra-subject paradigm is actually easier than the cross-subject one, because it is expected that there exist inter-subject differences that make generalizing across subjects difficult.

It has been mentioned before that a single data point's expansion in time is to little to draw empirical conclusions from. Therefore, it stands to reason to average multiple data points in one way or the other. A well established way of doing this, is to calculate the power spectral density (PSD) of the EEG frequencies [9]. For this, Welch's method with Blackman's window function has been used for the frequencies from 0-50Hz [10]. The width of one window is one second and windows have an overlap of 0.5 seconds. Thus, each data set pertaining to a specific session and scenario is of the shape (number of channels, number of frequencies, length of data set in minutes x seconds in a minute x overlap), resulting in (8, 51, 1800).

Having computed the PSD of the data, features are manually computed. A commonly used set of features in the scientific literature corresponds to a number of frequency bands and their ratios [8]. More specifically, these are the delta (0-4Hz), theta (4-8Hz), alpha (8-13Hz), beta (12-31Hz) and gamma (31-50Hz) frequency bands as well as the following ratios: (alpha + theta)/beta, alpha/beta, (alpha+theta)/(alpha+beta), theta/beta. In order to obtain a single value for a given frequency band, the mean of the different frequencies, that make up that band, is computed. Using features that are common in the scientific literature allows for a straightforward comparison in performance and thus inference about the quality of the data and the adequacy of the experimental setup. Once comparable results have been achieved, less well established methods might be explored. In particular, these include deep neural network architectures. Since these architectures commonly lack model interpretability, it is all the more important to have a transparent point of reference for comparing performance and judging the quality of the data accordingly. The aforementioned set of features is computed for each channel separately. This will result in a total of 72 features, thus the shape of the data is of the following dimensions: (72, 1800).

Finally, pandas data frames are constructed from the fully processed data for easy use with ML algorithms later on.

Two vastly different paradigms have to be distinguished in what follows. Firstly, there is the inter-session paradigm, where the data sets of different sessions are kept separate. Here, one particular session is designated to be the train set, while the remaining ones are seen as one homogeneous test set. Secondly, there is the cross-session paradigm, that does not distinguish between different recording sessions. The entire data, containing all sessions, is split into train and test set, where the train set contains 30% of the total data. The inter-session paradigm allows for testing the generalizability of any method between sessions, i.e. it allows to answer the question whether a given algorithm trained on only one session, is also capable of accurately predicting data points from different sessions. In other words, the model's ability to generalize across time is tested

B. Data Visualization

An important step in any exploratory data analysis, is to visually inspect the data. The most straightforward way to get a broad overview of the collected EEG data is to view it in it's totality across all channels, scenarios and subjects for each session. This has been done in figures 1, 2 and 3. In order to make all plots more expressive, the EEG data has been partially pre-processed, i.e it has been filtered, trimmed and standardized as specified in section III-A. It is important to note that filtering is only done for plotting the data, all other methods use unfiltered data, as filtering removes information that is actually beneficial for analysis.

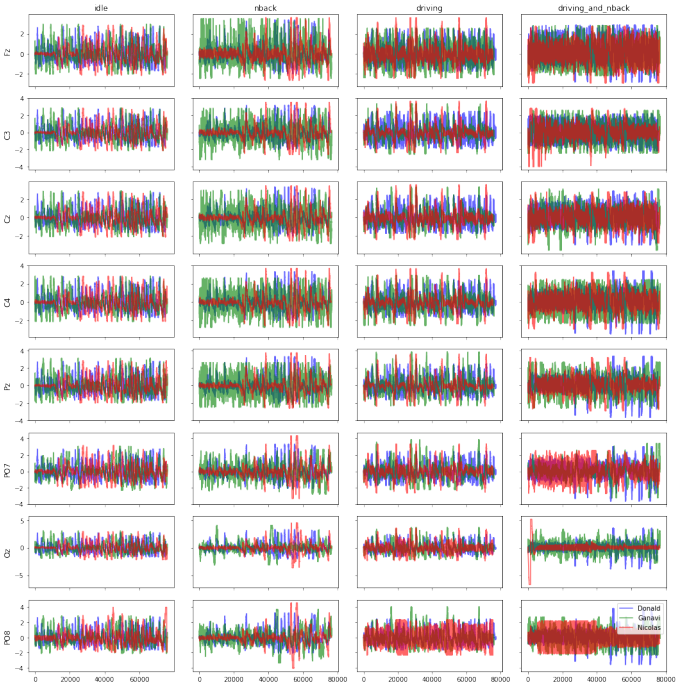

Fig. 1. Plot of the first recording session's filtered, trimmed and standardized data across all channels, scenarios and subjects.

Figures 1, 2 and 3 are arranged like a 3 dimensional matrix. The rows correspond to the eight different EEG channels, captured by the eight electrodes of the headset. The columns are associated with the four different recording scenarios. Each cell displays line-plots in three different colors, each representing a different subject's recording data over 5 minutes.

The majority of the first recording session's data appears to be of decent quality. For most of the individual cells, one can see a thin, straight line stretching along the x-axis. From

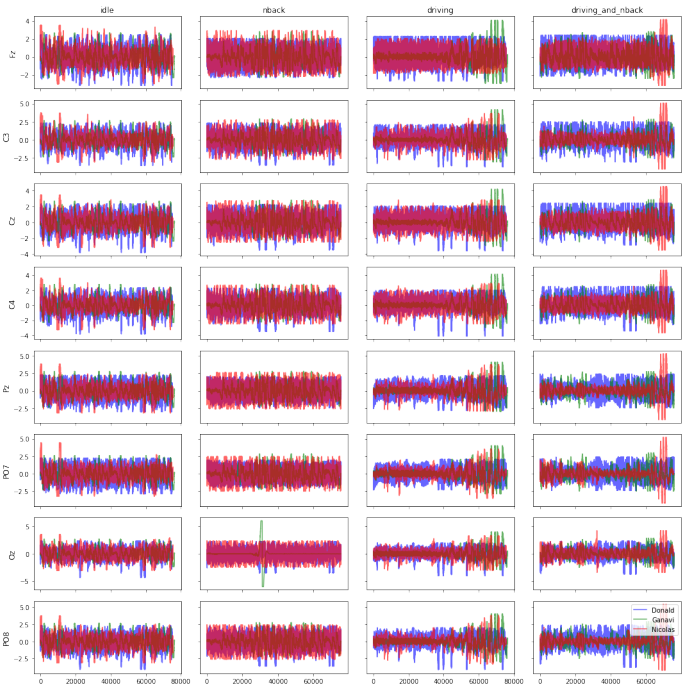

Fig. 2. Plot of the second recording session's filtered, trimmed and standardized data across all channels, scenarios and subjects.

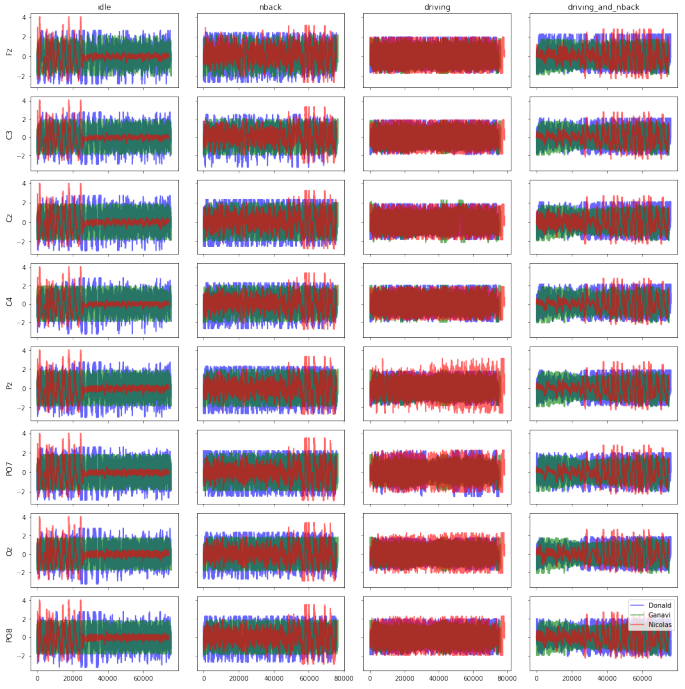

Fig. 3. Plot of the third recording session's filtered, trimmed and standardized data across all channels, scenarios and subjects.

this horizontal line extend thin, individual lines vertically. The presence of thin, individual lines that are clearly separated from one another speaks for the absence of noise. That being said, some irregularities can be spotted in the red data for the last column (*driving with 2back*). For some segments, one can see a thick horizontal line from which no lines extend vertically. This becomes especially evident in the cell of the last row (P08). Here the red data looks like a uniform rectangular block, which implies noisy data.

This block pattern is also present in the data of the second recording session, in particular for red line-plots in the second column (*driving*) as can be seen in figure 2. In the third recording session it is even more pronounced. There, the majority of the line-plots in all cells appear as if they were painted by a single stroke of a thick brush.

Having inspected the low level aspects of the data, we now turn to a more compact visual representation. In order to visually analyze the fully pre-processed data, i.e. the data sets with 72 features, in a number of dimensions that is comprehensible by humans, two dimensionality reduction techniques have been deployed, namely Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA).

Before moving forward, it is important to note that the PCA and LDA models, that follow the inter-session paradigm, have been fit solely on the first recording session data, i.e. the lower dimensional subspace was found by only referencing data from the first recording session, while the second and third recording sessions have been merely projected onto that subspace. This was done in order to prevent data leakage and test generalizability across sessions. Otherwise, information from train and test set would be used to find the lower dimensional subspaces, which would mean that subsequent ML algorithms that would be trained on the projected train set, also had implicit access to information from the test set.

1) PCA: PCA does not make use of labels, thus it is an unsupervised method. Furthermore, PCA assumes that interestingness is adequately captured by the variance of the data. Moreover, only linear relationships can be captured using PCA.

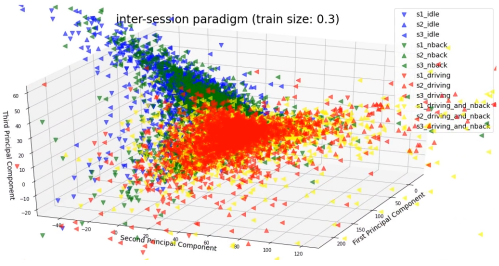

Fig. 4. Projection of inter-session data onto the first three principal components.

Data points colored blue have been recorded during *idle*, green during *2-back*, red during *driving* and yellow during *driving with 2-back*. The first recording session is represented by triangles pointed downwards, the second by triangles pointed upwards and the third by triangles pointed left. As

can be seen in figure 4, the first three principal components (PCs) of the data do not allow for a satisfactory separation of the different scenarios, as there is a lot of overlap. One might be able to say that the data point clouds associated with *driving* and *driving with 2-back* cluster together as do the data point clouds of *idle* and *2-back*. The former has a tendency to extend towards the right (high values of the second PC), while the latter stretches to the left (low values of second PC) and upwards (high values of the third PC). Thus, if it all, one might be willing to recognize a weak separation of scenarios into two clusters associated with driving vs non-driving activities, but intra-cluster separation is weak to non-existent. The first three PC capture 45, 19, and 10 percent of the variance, respectively.

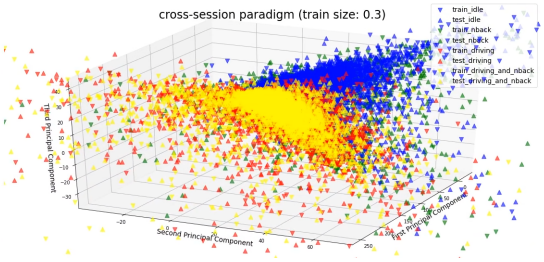

Fig. 5. Projection of cross-session data onto the first three principal components.

PCA fitted using data from the cross-session paradigm paints a similar picture, as can be seen in figure 5. Again, one might argue about the presence of two data point clouds, separated by their association with either driving or non-driving activities. These clouds appear to be primarily different in their values on the first and third PC. The cloud containing points from *idle* and *2-back* tends to extend to the right (high values of the second PC) and upwards (high values of the third PC), while the cloud containing points from *driving* and *driving with 2-back* tends to extend to the left (low values of the second PC) and remains flat (constant values of the third PC). Here, the first three PC capture 73, 10, and 5 percent of the variance, respectively.

Both subspaces found by PCA, fitted either on the inter- or cross-session paradigm data, are not suitable for classification, as the projected data does not lend itself to an obvious separation of the scenarios. In particular, the data points of *driving* and *driving with 2-back* cluster in one cloud with a high degree of overlap and as these are the classes that we ultimately care to predict, projecting the data using PCA doesn't appear to be beneficial for doing so.

2) SVD: Nevertheless, a more elaborate analysis of the data in terms of it's variance might produce further insight and can be seen in figure 6. Figure 6 depicts the Singular Value Decomposition of the entire, fully pre-processed data. It can be seen that in fact only 23 dimensions carry information that influence the variance of the data. Considering that the original data is described by 72 dimensions, this amounts to a reduction-factor of roughly 3. Furthermore, from these 23 variance-relevant dimensions only the first 16 describe the signal, while the remaining 7 actually capture noise [12]. Thus,

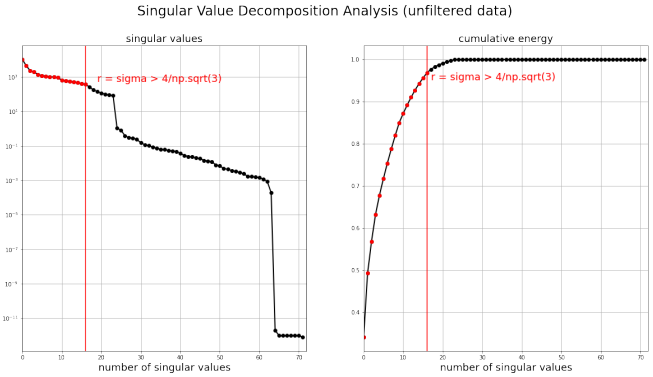

Fig. 6. Singular Values Energy (left) and Cumulative Energy (right).

it might be interesting to train a ML algorithm on these first 16 dimensions of the data, in the hope that the additional 13 dimensions on top of the ones already calculated by PCA help to separate the scenarios.

3) LDA: A supervised dimensionality reduction method makes use of the labels to project the data into a lower dimensional subspace. A well established method for doing so is Linear Discriminant Analysis. LDA assumes that the data of each class comes from an independent normal distribution with shared covariance matrices and unique mean vectors. Like PCA, LDA is also essentially concerned with the variance of the data: It tries to minimize intra-class spread while maximizing inter-class distance. Ideally, this objective results in cleanly separated data point clouds, each associated with a different class. In figure 7 we can see the projection of the inter-session data onto the three dimensional subspace computed by LDA. The color and marker scheme is the same as for the PCA plots. It is clear from the plot that the data not only separates according to the four different scenarios, but also further along the number of different sessions. This becomes especially apparent in the case of *driving with 2-back*, where the data point clouds for the first, second and third recording sessions are completely disjoint. While not as strongly pronounced, a similar pattern is evident for the other scenarios as well. This nicely portrays the fact that the LDA model has issues generalizing between different sessions. It is incapable of computing a three dimensional subspace that captures data points from the same scenario and across sessions in a homogeneous, continuous cloud.

This finding is strengthened by the results of the cross-session paradigm displayed in figure 8. Here, it is easy to see that data points from different sessions cluster in the same cloud, because the lower dimensional representation has been computed using data points from all sessions. Moreover, the data point clouds associated with each scenario exhibit a high degree of separation. Due to this separation, it becomes possible to train a linear classification algorithm on the train set of the cross-session paradigm, in order to quantify the degree of separation.

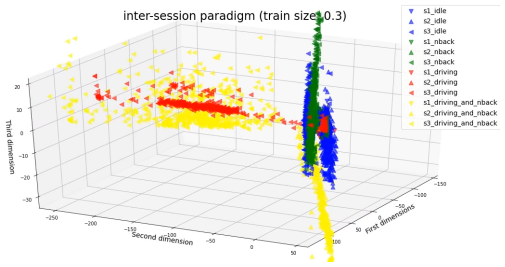

Fig. 7. Projection of inter-session data onto the three dimensional subspace computed by LDA.

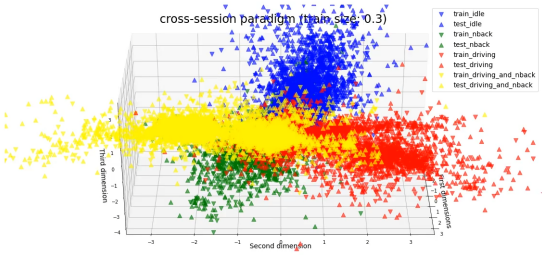

Fig. 8. Projection of cross-session data onto the three dimensional subspace computed by LDA.

4) Quantifying Class Separation: In order to quantify the degree of separation across scenarios of the cross-session paradigm data in the subspace computed by LDA, a Linear Support Vector Classifier (LSVC) has been trained as a One-Versus-Rest-Classifier. This means, that four classifiers are trained in total, each viewing a given scenario as one, and all of the other scenarios jointly as the opposing class. Thus, the multi-class setting is decomposed to four unique binary-class settings.

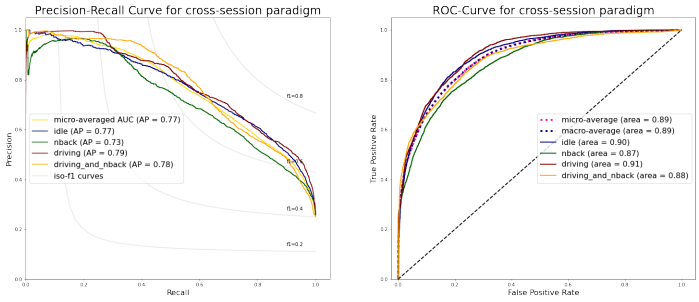

Fig. 9. Receiver Operator Characteristic (left) and Precision-Recall (right) curves of the LSVC for the cross-session data.

Figure 9 shows the Receiver Operator Characteristic as well as the Precision-Recall curves for the LSVC that has been trained on and tested against the LDA-projected cross-session data. The legend depicts the area under the curve (AUC) for the 4 different individual binary classifiers as well as the micro- and macro-averaged AUC. The micro-averaged PR-AUC is 0.77, while the PR-AUC for the individual scenarios ranges from 0.73 (*2-back*) to 0.79 (*driving*). Similarly, the micro-averaged ROC-AUC is .89, while the ROC-AUC for the individual scenarios ranges from .87 (*2-back*) to .91 (*driving*). The difference in the micro-averaged area under the curves can probably be explained by the model's low precision score, as both metrics share the same recall scores.

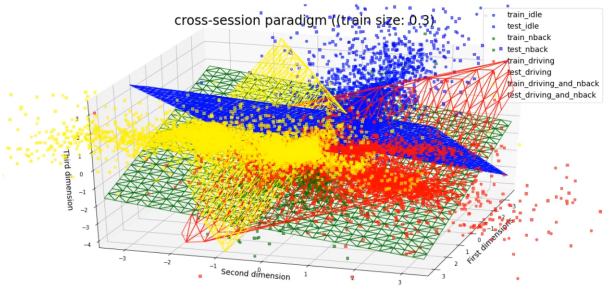

Fig. 10. Decision surfaces of the 4 binary LSVC of the LDA-projected cross-session data.

Figure 10 displays the learnt decision surfaces of the 4 binary LSVC in the LDA-subspace. Four decision surfaces are depicted in total, each corresponding to a different classifier. The color of the decision surface represents the single scenario that it tries to separate jointly from all the other ones. This visual sanity check confirms that the LSVCs are learning reasonable decision boundaries. Ultimately, this tells us a few things: Firstly, the signal-to-noise ratio of the data is sufficiently low to work with the data in a meaningful way. Secondly, the expected conceptual difference between the different scenarios is reflected in the data. Thirdly, it is possible to train a classifier that is capable of predicting which scenario a given data point belongs to at a rate that is non-trivially better than chance. Having established these findings, we move towards the use cased outlined in section I.

IV. Predicting Mental Workload

Once the data is processed and the feature extraction is complete, the next step is to train the models to make the best prediction. The data set (Cross-Session) is then trained on KNN and Random forest models for binary classification. Figure 11 shows the performance of the KNN and RF models for Binary classification.

The Figure 11 shows that KNN (Acc: 65%) performs slightly better than RF (Acc: 62.54%). The recall of the RF model is too poor and by comparing the results KNN performs better among the two.

The next approach was to use Neural networks to perform classification. The models are developed using Dense and Fully connected networks available in Keras library. The models have been trained with multiple epoch size (100, 150, 200) and batch sizes (32, 64, 128) and the best ones are documented.

The data from all the recording sessions were combined (Cross - Session Paradigm) and split into 6:2:2 for training, test, and validation sets. Starting the with most simple architecture with 1- Hidden layer perceptron, figure 12, 13 shows

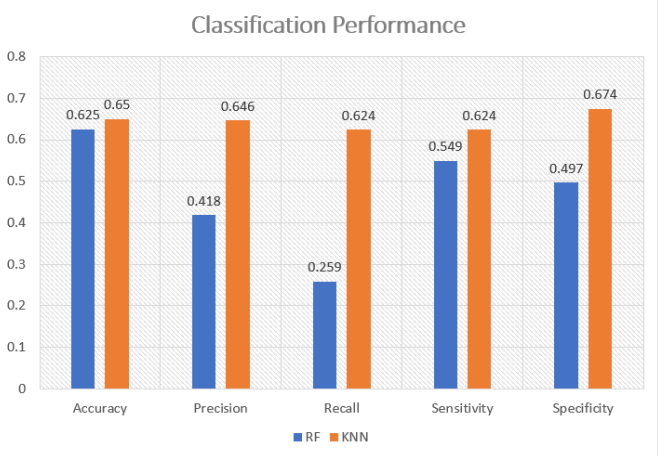

Fig. 11. Classification Performance RF vs KNN

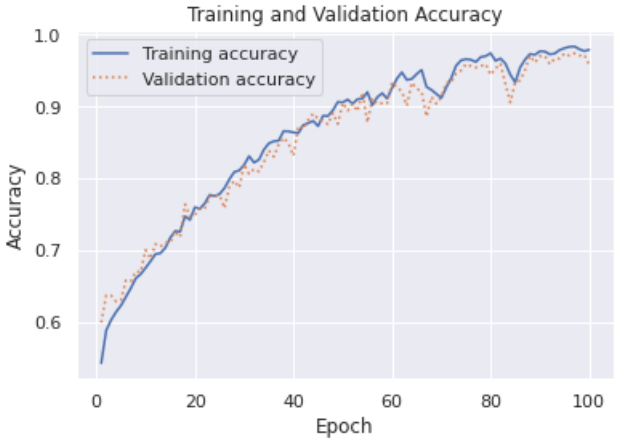

Fig. 12. Model 1: 1-Hidden Layer Perceptron - Accuracy

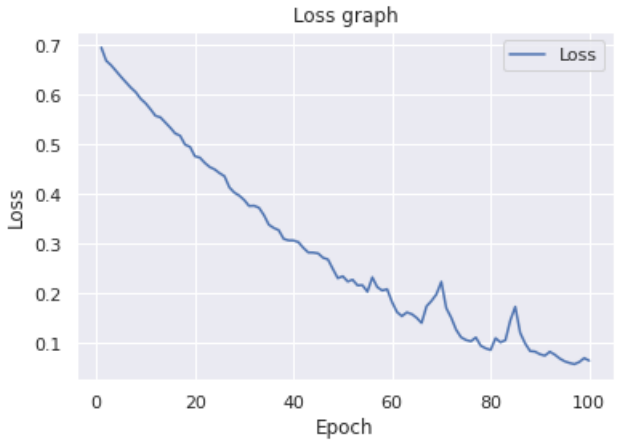

Fig. 13. Model 1: 1-Hidden Layer Perceptron - Loss

the accuracy and loss graphs. The achieved test accuracy was 76.34%.

Next, the architecture with more layers (5 layer - MLP) is used to improve the dataset. (Figure 14 and 15). The achieved test accuracy is 89.74%.

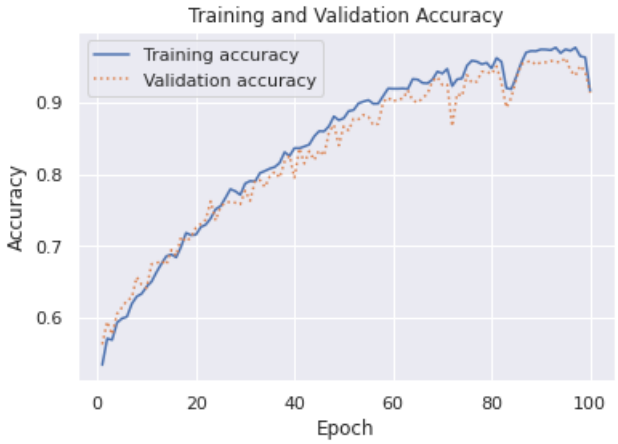

Fig. 14. Model 2: 5 Layer MLP - Accuracy

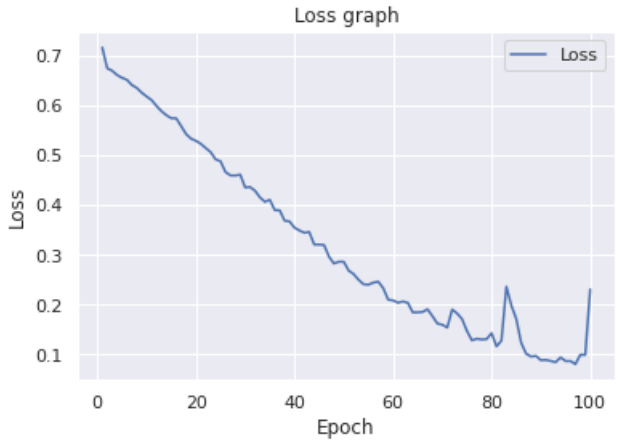

Fig. 15. Model 2: 5 Layer MLP - Loss

Then we also considered Inter-session approach, where the model was trained on one session's data and tested on the other. This resulted in test accuracy of 73.4%. The observed paradigm (Figure 16 and 17) shows that the gap between train and validation sets is more even though the data is from same session.

Another explored approach is to use intra-subject and cross-session data to train the model to generalize better on individual's data. Figure 18 and 19 represents the accuracy and loss of the model. The test accuracy was 70.82%.

The obtained results are satisfactory and the intersubject trained models performed slightly better.

The above models were trained with data collected on different sessions. Although the models performed better with the test data, it failed to generalize much of the unseen data.

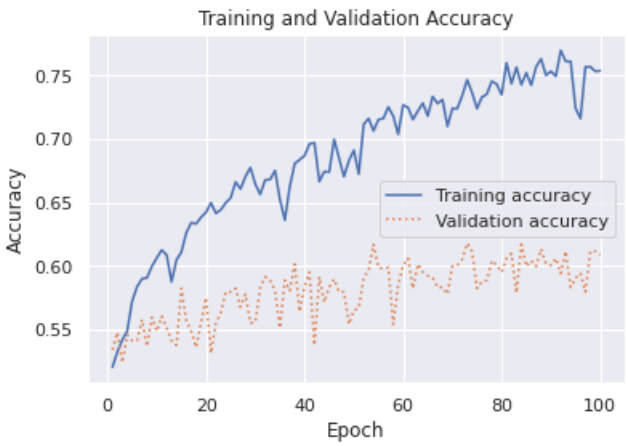

Fig. 16. Model 3: Inter-session Model- Accuracy

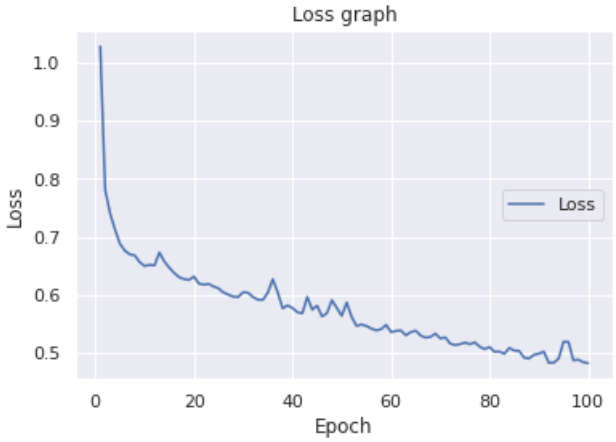

Fig. 17. Model 3: Inter-session Model- Loss

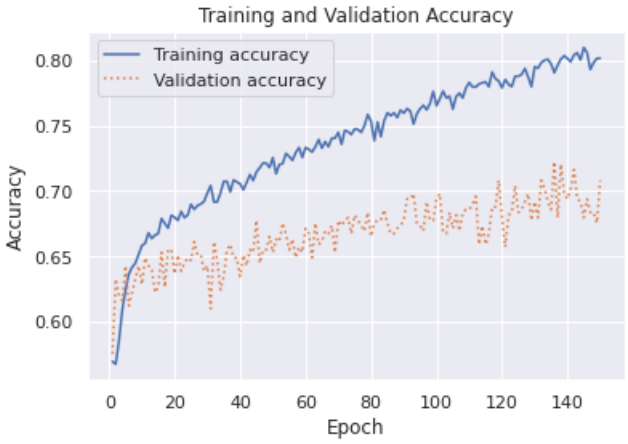

Fig. 18. Model 4: Trained with a subject's data alone - Accuracy

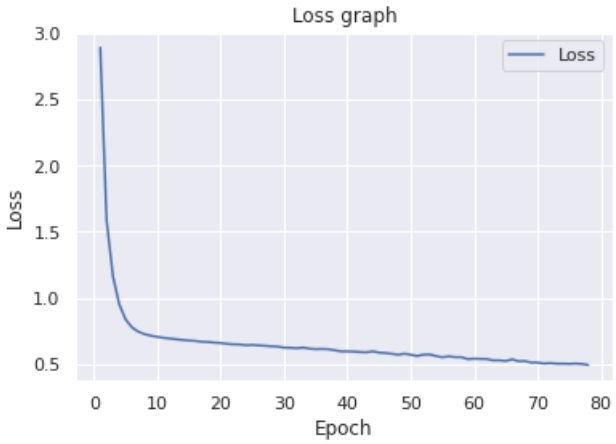

Fig. 19. Model 4: Trained with a subject's data alone - Loss

V. Interface

Once the model detects a high CL, the driver is alarmed via auditory and visual cues, urging them to disengage from actively driving themselves and transfer control over the the autonomous vehicle. An interface communicates this intent via beeping and displaying a transition from a green to a red driving status. The driver than has to manually transition control over to the car by pressing a button.

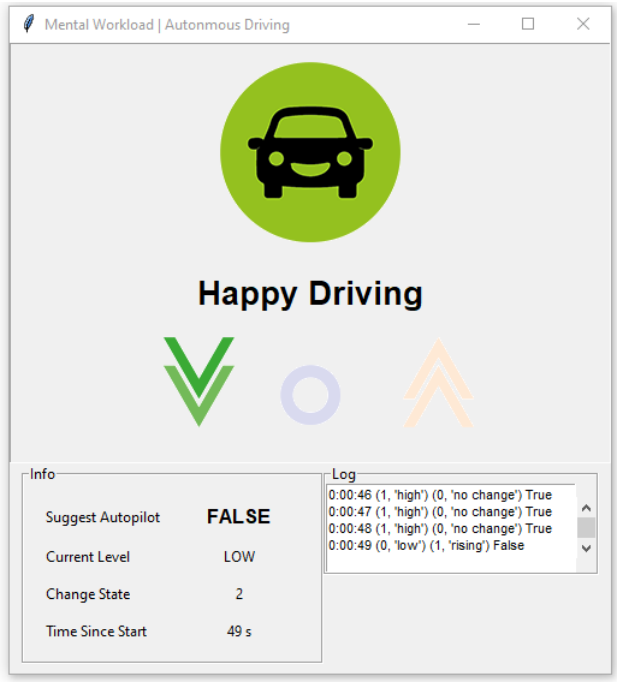

Fig. 20. The interface shows prediction results in real-time and suggest activation or deactivation of the autopilot.

Figure 20 depicts the interface, which sums up the essential information based on the results of the prediction model. Figure 21 shows the interface suggesting to activate the autopilot.

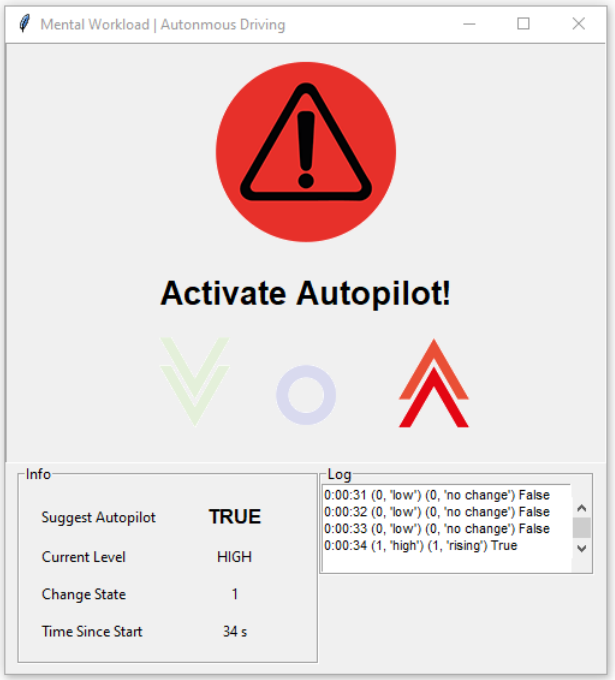

Fig. 21. Interface suggesting to activate autopilot

Fig. 22. Assigned key on steering wheel for autopilot.

```
<keyAssignments>
    <keyAssignment function="auto_steer" key="BUTTON_3" />
</keyAssignments>
```

Fig. 23. Map autopilot to button on steering wheel via OpenDS settings.xml.

It keeps track of the current time, CL of the driver, the change in CL and whether a suggestion to activate the autopilot is given - in real-time. The interface consists of a visual view of the current level of MWL, a info view with a brief summary of all relevant values, as well as a history log.

The way we feed the interface with the model's prediction is by writing the results to a text file and reading the results in set time intervals. From the binary prediction results showing either a high or low MWL, a potential state change is calculated and implemented in the interface accordingly.

Based on the current state a suggestion is given in the "Info" view. The proposed auditory cue, is realised by playing a short repeating beep sound, when MWL is at a high level. Once MWL is falling to a low level again a longer beep sound is played to signal to the driver to take back control form the auto-steering system.

As can be seen in figure 22, the driving simulator setup at the DFKI laboratory, that we had access to, is equipped with a steering wheel (Fanatec BMW GT2). To make the action of toggling the autopilot for the driver as comfortable as possible and reduce additional distraction, the respective button is mapped onto the steering wheel. The necessary code for doing so is depicted in figure 23. This enables the driver to change the state of the autopilot by pressing a button in comfortable reach.

Furthermore, we investigated the option to use a middle device to imitate the center console of a car, in order to provide information about the CL of the driver and the current state of the autopilot, to the driver at a glance. To achieve such a system it would be required to use a Message Queuing Telemetry Transport (MQTT) protocol to make communication between clients, e.g. EEG data processing client, a mobile device and OpenDS, possible.

Together with OpenDS' component LiveDataInterface/Settingscontroller, a simple network socket with XML-based information exchange, this could have been accomplished. By sending a XML message, e.g. requesting the state of the autopilot in the driving simulation, to OpenDS an answer will be sent back to the software. There is also the option to subscribe to data and get a message from OpenDS in regular time intervals.

This would have enabled us to implement our own widget in OpenDS and show selected information on screen in the driving simulation. As we only focused on the core functionalities in our interface this has been left for future work in the context of this project.

VI. Results & Outlook

As is evident from the results of machine learning algorithms trained on inter-session data, a major remaining challenge, is the ability to generalize across sessions. Judging from what we have gathered so far, the inter-session differences are the greatest challenge when it comes to predicting CL from live EEG data - even more so than inter-subject differences. It is surprising that the performance of the MLP trained on and tested against the intra-subject data, is worse than of the MLP using the cross-subject data. One would expect that the absence of the inter-subject differences makes generalization easier. But in order to make a fair comparison, both models

would have to be trained on the same number of data oints. Currently though, the cross-subject data is thrice the size of any intra-subject data set (by design).

Having established that data quality is sufficient in terms of the signal-to-noise ratio, and that the data reflects the conceptual differences between the scenarios, a next step would be to remove expert knowledge from the pipeline, and with it the time and effort involved to manually craft a feature set. For doing so, [11] suggest an autoencoder architecture that utilizes convolutional layers in order to find a lower dimensional representation of the data, that can be used instead of the manually crafted feature set, while achieving greater performance. That being said, deep architectures usually do not allow for insightful conclusion to be drawn from the data, as model interpretability is generally speaking quite low.

ACKNOWLEDGMENT

We would like to extend our heartfelt thanks towards Guillermo Reyes Fuentes and Maurice Rekrut, without whom this work would not have been possible.

REFERENCES

[1] Thomas, P., Morris, A., Talbot, R., and Fagerlind, H. (2013). Identifying the causes of road crashes in Europe. Annals of advances in automotive medicine, 57, 13.

[2] Engström, J.; Markkula, G.; Victor, T.; Merat, N. Effects of Cognitive Load on Driving Performance: The Cognitive Control Hypothesis. Hum. Factors J. Hum. Factors Ergon. Soc. 2017, 59, 734–764.

[3] Jaeggi, S.M.; Buschkuehl, M.; Perrig, W.J.; Meier, B. The concurrent validity of the N -back task as a working memory measure. Memory 2010, 18, 394–412

[4] Kane, M.J.; Conway, A.R.A.; Miura, T.K.; Colflesh, G.J.H. Working memory, attention control, and the n-back task: A question of construct validity. J. Exp. Psychol. Learn. Mem. Cogn. 2007, 33, 615–622.

[5] Mehler, B.; Reimer, B.; Wang, Y. A comparison of heart rate and heart rate variability indices in distinguishing single-task driving and driving under secondary cognitive workload. In Proceedings of the 6th International Driving Symposium on Human Factors in Driver Assessment, Training, and Vehicle Design, Olympic Valley-Lake Tahoe, CA, USA, 27–30 June 2011; pp. 590–597.

[6] Paxion, J., Galy, E., Berthelon, C.: cognitive load and driving. Front. Psychol.5, 1344 (2014) pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[7] Verwey, W.B.: On-line driver workload estimation. Effects of road situation and age on secondary task measures. Ergonomics 43(2), 187–209 (2000)

[8] Barua S, Ahmed MU, Begum S. Towards Intelligent Data Analytics: A Case Study in Driver Cognitive Load Classification. Brain Sci. 2020 Aug 6;10(8):526. doi: 10.3390/brainsci10080526. PMID: 32781777; PMCID: PMC7465999.

[9] Di Flumeri, G., et al.: EEG-based cognitive load neurometric to evaluate the impact of different traffic and road conditions in real driving settings. Front. Hum. Neurosci. 12, 509 (2018).

[10] Solomon Jr., O.: PSD computations using welch's method. NASA STI/Recon Technical Report N 92 (1991).

[11] Islam, Mir Riyanul, et al. "Deep learning for automatic EEG feature extraction: an application in drivers' cognitive load classification." International Symposium on Human cognitive load: Models and Applications. Springer, Cham, 2019.

[12] Gavish, Matan, and David L. Donoho. "The optimal hard threshold for singular values is $4/\sqrt{3}$." IEEE Transactions on Information Theory 60.8 (2014): 5040-5053.