Not Deep Enough: Autoencoders for Automatic Feature Extraction in Wireless Cognitive Load Inference

Anže Kristan[1], Daniel Pellarini[1] and Veljko Pejović[1]

[1]University of Ljubljana, Faculty of Computer and Information Science, Večna pot 113, 1000 Ljubljana, Slovenia

Abstract

Automatic and unobtrusive inference of a user's cognitive load could revolutionise interaction with computing systems. Wireless ranging represents an attractive alternative to the current state-of-the-art solutions relying on wearable physiological sensors. Yet, which wireless signal features will be the most informative for cognitive load inference remains an open research question. Due to their way of deconstructing input features into intermediate hidden layers, and reconstructing them into an output vector as similar as possible to the input data, autoencoders are on paper perfectly suited to help highlight the most important features in input data. In our study, we discover that using autoencoder for cognitive load inference from wireless signals can indeed sometimes beat classical machine learning classifiers in terms of accuracy. Nevertheless, despite the theoretical advantage, we find that autoencoders often struggle to reach the same accuracy levels as advanced traditional machine learning solutions. Consequently, we warn against over-reliance on deep learning methods in human-computer interaction research.

Keywords

autoencoders, machine learning, deep learning, cognitive load inference

1. Introduction

Thirty years after Mark Weiser's vision of human-computer interaction (HCI) being as seamless as a walk in the woods is still not a reality [1]. Computers do not understand us. Empowering computers with the ability to infer their users' cognitive load would open a range of possibilities for improved HCI. For instance, if a device senses that a student has stopped paying attention to a lecture or a learning exercise, it could remind the student to focus, present relevant content to pique the student's curiosity or relay a helpful hint or a tip. Or, in another example, if a device could know when a person is distracted or losing focus while performing a critical task, such as driving or operating machinery, it could alert the user to start paying attention or take a break in order to avoid potential accidents.

Efforts to enable automatic cognitive load inference often rely on the physiological response a person experiences when put under high cognitive load. In such cases, the person's heart activity becomes faster and more uniform, skin temperature varies, onsets of sweating appear,

Human-Computer Interaction Slovenia 2021, November 11, 2021, Koper, Slovenia
ak4352@student.uni-lj.si (A. Kristan); daniel.pellarini@fri.uni-lj.si (D. Pellarini); veljko.pejovic@fri.uni-lj.si (V. Pejović)
https://www.fri.uni-lj.si/en/about-faculty/employees/veljko-pejovic (V. Pejović)

and pupils dilate, to name a few relevant physiological reactions. Initial attempts to infer a user's cognitive load relied on lab-based equipment that required a user to be strapped to specialised sensors, thus were of little practical utility in the field of HCI [2]. Recently, wearable sensing relying on smartwatches, smart wristbands, and chest straps has enabled cognitive load inference where the participants can be put in real life situations [3]. Nevertheless, such approaches still require that each user is equipped with sensors.

A few recent attempts have been made to wirelessly sense cognitive load [4, 5]. One such method is Wi-Mind [5], which uses radio waves to detect changes in the movement of the user's chest. From the signal of these radio waves, breathing and heart rate features can be extracted. With these features, engineered upon the related work on physiological signal sensing, inferring whether the user is resting or busy (and the difficulty of the task they are solving) can be attempted. However, the wireless signal may contain much more information than just the features relating to breathing and heart rate. Whether extraction that goes beyond the well-known features would yield higher cognitive load inference accuracy remains an open question.

In this work, we use automatic feature extraction to try and find new features in order to improve cognitive load inference from wireless signals. For this purpose we use the autoencoder [6], a special type of a neural network that is composed of an encoder and a decoder part. The encoding layer transforms the input into an encoded representation (hidden layer), which is of a smaller size than the input. Then the decoder transforms the encoded representation into a reconstruction aiming to resemble the input as closely as possible. Since the hidden layer has fewer dimensions than the input data, the autoencoder tries to learn and extract important features of the data in order to be able to reconstruct the input from those features.

Using data obtained with wireless ranging of 23 participants solving six different task types at various difficulty levels, we assess the autoencoder's ability to derive features enabling high-accuracy cognitive load inference. Specifically, the contributions of our work include:

- We craft four different types of autoencoders with the goal of automatic feature extraction from wireless phase signals;
- We systematically evaluate the above autoencoders on various flavors of the cognitive load inference problem;
- We compare the autoencoder-based solutions with a range of conventional machine learning and deep learning solutions to assess the potential of autoencoders for real-world wireless cognitive load inference.

Contrary to the expectations, our results show that the traditional breathing and heart rate features usually trump autoencoders and produce the highest prediction accuracy. There are a few cases, however, where autoencoders can match or improve the prediction accuracy. We suspect our findings demonstrate that, despite their modelling power, deep learning-based methods need not be the most appropriate machine learning solution in cases where the data is noisy, or when the amount of data is not enough, and where a substantial body of expert knowledge on feature extraction already exists.

2. Background

2.1. Related work

Cognitive load is defined as "*a multidimensional construct representing the load that performing a particular task imposes on the learner's cognitive system*" [7]. Consequently, it is rather difficult to assess one's cognitive load – the load is modulated by the difficulty of a task one is solving, one's cognitive capacities, but also interests and motivations for solving a particular problem. Thus, measuring cognitive load is traditionally done through subjective self-evaluation after completing a task by using a survey, such as the NASA-TLX [8]. However, this does not allow for real time unobtrusive measuring of cognitive load, which is necessary should we want to use cognitive load information for human-computer interaction adaptation.

Harnessing the physiological reaction catalyzed by an increase in the cognitive load, researchers have devised methods for inferring cognitive load from various physiological signals. A study from 2010 used a contactless eye tracker, an ECG-enabled armband, a wireless EEG headset and a wireless HR monitor to measure a participant's physiological signals while solving elementary cognitive tasks. Using some of those signals the authors were able to achieve over 80 % classification accuracy of the participant's cognitive load defined as a two-category construct (high vs low load) [2].

Moving towards in-situ inference, off-the-shelf wearable devices have been recently used for cognitive load inference. In [3] researchers have demonstrated that cheap wristbands with photoplethysmogram (PPG) and electrodermal activity (EDA) sensors can be used for inferring cognitive load when users are exposed to specifically tailored tasks. The same has been shown in the context of playing a mobile video game [9]. The drawback of such solutions that users still need to be fitted with a particular device, which may not be possible in all situations.

2.2. Wireless cognitive load inference

Wireless cognitive load inference has recently been investigated in the Wi-Mind study [5]. This solution uses a software-defined radio-based radar to measure sub-millimeter movements of a person, which relate to their breathing and heart rates. For this, Wi-Mind records raw wireless signal phase of waves sent from a transmitter, reflected at a person's chest, and received at the receiver.

In the accompanying paper, the authors of Wi-Mind conducted a study with 23 volunteers who, while seated in front of a computer, individually solved tasks of six different task types, which are specifically designed to elicit a cognitive load response. Each task type has three difficulty levels – low, medium and high – for a total of 18 tasks. The tasks fall into the group of elementary cognitive tasks (described in [2]) and include:

- Finding A's (FA),
- Gestalt Completion (GC),
- Finding hidden pattern (HP),
- Number comparison (NC),
- Pursuit test (PT),
- Scattered X's (SX).

To infer cognitive load from wireless signals the authors of Wi-Mind used carefully engineered features representing different aspects of a person's breathing and heartbeat activity. The resulting features were then used in various classifiers, yielding a 75% accuracy rate in differentiating between busy (working on a task) and relax times, and a 65% accuracy rate in detecting transitions between two task difficulties, while achieving 52% accuracy of discerning between low and high cognitive engagement (corresponding to low and high task difficulty). A more fine-grained distinction among easy, medium, and high difficulty tasks was not possible.

2.3. Autoencoders

Autoencoders (AE) are a type of neural networks [6]. In essence, AEs are trained to attempt to copy their input to their output through a hidden layer, thus learning the characteristics of the data and how to represent (encode) it. An AE is made up of two parts: an encoder and a decoder, where the encoder acts as a function that maps the input to an encoded representation the size of the hidden layer, while the decoder acts as the inverse of the encoder function and maps the encoded representation back into the input. These properties render AEs a suitable tool for extracting features from sensor data, e.g. [10].

Similarly to standard neural network models, AEs are trained through backpropagation, with the difference being that the input provided to the AE is also the target output, whereas standard neural networks have a target output different to their input. Different regularization mechanisms, such as the limitation on the sparsity of network layers, or the Frobenius norm of the Jacobian matrix of the encoder activations with respect to the input, lead to different flavors of AEs, such as sparse, contractive AEs [11] and others.

3. Machine learning evaluation pipeline

In this paper we examine whether automatic feature extraction using autoencoders can help with cognitive load inference from wireless data. We use the wireless signal phase data collected from 23 participants performing the cognitive load elicitation experiments described in Section 2.2. Each participant was assigned a randomly generated identification string (ident), which is used instead of personally identifiable information.

We segment the wireless signal phase data into 30-second time frames (the length of the relaxation period during the Wi-Mind study). We examine several aspects of detecting cognitive load for which we try to infer the user's state during a time frame:

- Busy vs Relaxed (BvR): whether a user is busy or relaxed (solving a task vs being instructed to relax);
- Low vs High (LvH): whether a busy user is solving a low or high difficulty task;
- Low vs Medium vs High (LvMvH): whether a busy user is solving a low, medium or high difficulty task.

For the BvR problem we use one relax time frame and one busy time frame for each task (before preprocessing), because each period of relaxation only last as long as one time frame. While for the LvH and LvMvH problems we use 10 consecutive time frames from each task (before preprocessing). For each of these problems we first use all task types (FA, GC, HP, NC, PT, SX), then we explore how our models perform when using only one task type at a time. For classification purposes we use a range of machine learning models:

- Simple Dense (SD): a simple neural network with 1 dense hidden layer made using Keras;
- Long Short-Term Memory (LSTM): an LSTM based classifier made using Keras, based on a classifier made in the original Wi-Mind paper (an LSTM is a type of neural network especially useful for time series data);
- k-Nearest Neighbors (kNN): a k-nearest neighbors algorithm from the scikit-learn library;
- Support Vector Machine (SVM): a support vector machine based classifier from the scikit-learn library;
- Naive Bayes (NB): a naive Bayesian classifier from the scikit-learn library;
- Random Forest (RF): a random forest based classifier from the scikit-learn library;
- XGBoost (XGB): the XGBoost classifier from the XGBoost python package.

Using these classifiers we experiment with features extracted by experts (taken from the Wi-Mind paper) and features extracted by AEs. There are 8 feature groups we used as input for classification, 4 of which are not based on AEs and 4 of which are based on AEs:

- Phase (PHS): unwrapped phase data;
- Breathing (BR): 12 features related to a person's breathing extracted from the phase data;
- Heartbeat (HB): 10 features related to a person's heartbeats, extracted from the phase data;
- Combined (CMB): concatenated BR and HB features (22 features total),
- Undercomplete (under.): encoded representation of data as extracted by the undercomplete AE. This AE has no explicit regularization term, but contains a restricted number of nodes in the hidden layer;
- Sparse (sparse): encoded representation of data as extracted by the sparse AE. Here, the loss function penalizes activations within a layer;
- Deep (deep): encoded representation of data as extracted by the deep AE;
- Contractive (contr.): encoded representation of data as extracted by the contractive AE. Here, the loss function penalizes large derivatives of the hidden layer activations with respect to the input training examples (i.e. the AE should be more robust to variations in the input data).

The PHS, BR and HB features are based on the Wi-Mind paper. Each AE was set to extract 30 features (chosen empirically to be slightly larger than the number of CMB features and due to 30 seconds being the length of each time frame). Altogether there were 7 task types, 3 classification problems, 7 machine learning models and 8 feature groups, which brings the total number of combinations to 1176. For each combination of test ident, classifier and feature group used, we saved the classification accuracy (CA) to a dictionary. Once we have calculated the CA of each combination, we save the dictionary containing the CAs to a file. This was done using data from each task type separately, and then using data from all task types at once (7 times in total).

3.1. Noise filtering

For each ident there is raw phase data with timestamps describing when a volunteer was solving each task (or relaxing), how much time was spent on that task and how they performed on that task. Each instance of a class is based on a 30-second time frame, which means that any time an instance of a class was shorter than 30 seconds, it is instead skipped. Additionally, there is a meta-feature which denotes whether there is significant noise present in a certain time frame. Taking the noise meta-feature into account, more instances may be skipped.

Table 1 and 2, compare the number of instances before and after noise filtering. The prefix "before" means that there were no instances removed due to the noise meta-feature. The prefix "after" means that the noise meta-feature was used to remove noisy instances. The suffixes determine the class of that row.

Table 1
Comparison of the number of class instances for each task type for the busy vs relax problem.

task types	FA	GC	HP	NC	PT	SX	total
before_relax	66	55	68	68	62	52	371
before_busy	66	55	68	68	62	52	371
after_relax	38	22	40	37	19	22	178
after_busy	38	22	40	37	19	22	178

Table 2
Comparison of the number of class instances for each task type for the low vs medium vs high problem.

task types	FA	GC	HP	NC	PT	SX	total
before_low	230	230	230	230	230	230	1380
before_medium	230	230	230	230	230	230	1380
before_high	230	230	230	230	230	230	1380
after_low	177	102	185	177	135	150	926
after_medium	171	126	193	174	159	187	1010
after_high	189	135	191	166	129	178	988

4. Experimental results

For each problem we evaluated all combinations on each task type separately and then all task types together. Reported are the mean accuracies of the *leave-one-out* approach averaged over all participants.

4.1. Busy vs Relaxed

For the BvR problem we aim to classify whether the user is resting or solving a task. Before solving each task, participants in the study were instructed to relax for a period of time. The data from that period of relaxation was used to define the relaxed class. In contrast, the data obtained while the user was solving any task (regardless of the task type and difficulty) was used to define the busy class.

The results are summarized in Table 3. Some notable highlights for the BvR problem include:

- The highest overall accuracy 0.87 was achieved with the XGB classifier using the BR feature group using the NC task type;
- The highest overall accuracy using all task types is 0.76 with the RF classifier and BR feature group;
- The highest accuracy using only AE based feature groups is 0.74 with the NB classifier and *Deep* feature group using the NC task type;
- The highest accuracy using only AE based feature groups and all task types is 0.66 with the RF classifier and *Deep* feature group.

Table 3
Top accuracies for the BvR problem with the corresponding classifier and feature group used to achieve the given accuracy. The rows with the "_AE" suffix only take into account combinations with AE-based feature groups (written in lowercase), while the rows without the suffix take into account all classifier/feature combinations.

Task types	FA	GC	HP	NC	PT	SX	all
Accuracy	0.70	0.85	0.79	**0.87**	0.77	0.67	0.76
Classifier	SVC	NB	RF	XGB	NB	LSTM	RF
Feature	BR	HB	CMB	BR	BR	under.	BR
Accuracy_AE	0.61	0.61	0.64	**0.74**	0.62	0.67	0.66
Classifier_AE	SD	SD	RF	NB	NB	LSTM	RF
Feature_AE	sparse	under.	deep	deep	contr.	under.	deep

4.2. Low vs High

For the LvH problem we aim to classify whether the user is solving a task with low or high difficulty, as each task type was performed at three different difficulty levels (low, medium, high) by each of the participants. For this problem we only use data gathered while the participants were solving a task with low difficulty or high difficulty. Table 4 summarizes the results. Some notable highlights for the LvH problem include:

- The highest overall accuracy 0.67 was achieved with the NB classifier and *Contractive* feature group on the FA task type;
- The highest overall accuracy for all task types is 0.59 with the RF classifier and BR feature group;

- The highest accuracy using only AE-based feature groups is the same as the highest overall accuracy with the same classifier, feature group, and task type;
- The highest accuracy using only AE-based feature groups and all task types is 0.57 with the SVM classifier and *Undercomplete* feature group.

Table 4
Top accuracies for the LvH problem with the corresponding classifier and feature group used to achieve the given accuracy. The rows with the "_AE" suffix only take into account combinations with AE-based feature groups (written in lowercase), while the rows without the suffix take into account all classifier/feature combinations.

Task types	FA	GC	HP	NC	PT	SX	all
Accuracy	**0.67**	0.63	0.63	0.58	0.59	0.65	0.59
Classifier	NB	XGB	RF	NB	SVC	RF	RF
Feature	contr.	HB	BR	CMB	HB	CMB	BR
Accuracy_AE	**0.67**	0.50	0.63	0.47	0.50	0.61	0.57
Classifier_AE	NB	NB	SD	XGB	SVC	LSTM	SVC
Feature_AE	contr.	deep	contr.	under.	sparse	contr.	under.

4.3. Low vs Medium vs High

For the LvMvH problem we aim to infer whether the user is solving a task with low, medium or high difficulty. Table 5 summarizes the results. Some notable highlights for the LvMvH problem include:

- The highest overall accuracy 0.49 was achieved with the SD classifier and PHS feature group on the PT task type;
- The highest overall accuracy using all task types is 0.40 with the SVM classifier and CMB feature group;
- The highest accuracy using only AE-based feature groups is 0.47 with the NB classifier and *Sparse* feature group on the PT task type;
- The highest accuracy using only AE-based feature groups and all task types is 0.37 with the LSTM classifier and *Contractive* feature group.

We finally graphically summarize the results for task-invariant classifiers in Figure 1 and across different task types in Figure 2. We see that AE-based solution in general tend to perform worse than solutions based on features extracted from the existing domain expertise published in the literature. Nevertheless, we do see that in certain situations (e.g. certain task type, certain problem definition), AE-based solutions tend to be on a par with the expert feature-based solutions.

Table 5
Top accuracies for the LvMvH problem with the corresponding classifier and feature group used to achieve the given accuracy. The rows with the "_AE" suffix only take into account combinations with AE-based feature groups (written in lowercase), while the rows without the suffix take into account all classifier/feature combinations.

Task types	FA	GC	HP	NC	PT	SX	all
Accuracy	0.46	0.43	0.42	0.43	**0.49**	0.46	0.40
Classifier	NB	SD	SVC	XGB	SD	SD	SVC
Feature	contr.	HB	PHS	CMB	PHS	CMB	CMB
Accuracy_AE	0.46	0.39	0.39	0.39	**0.47**	0.43	0.37
Classifier_AE	NB	SVC	SVC	LSTM	NB	XGB	LSTM
Feature_AE	contr.	contr.	contr.	under.	sparse	under.	contr.

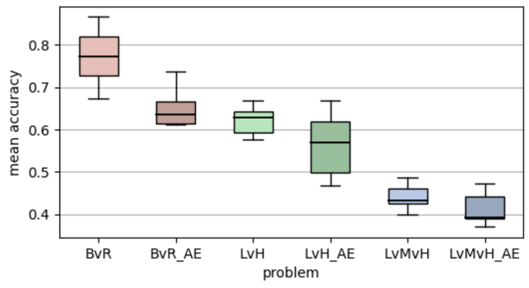

Figure 1: Boxplot for each problem.

A boxplot diagram using highest mean accuracies (as listed in the tables in section 4) for each problem. No suffix means that all feature groups were taken into account, while the "_AE" suffix means only AE based features were taken into account.

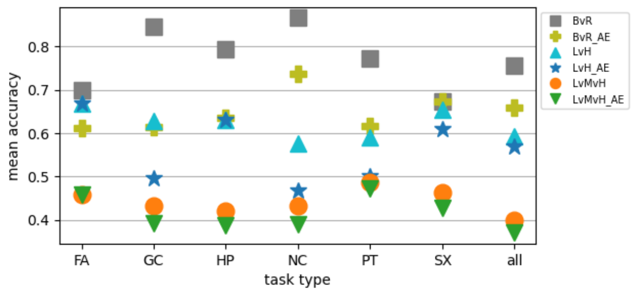

Figure 2: Accuracies for each task type.

Comparison of highest mean accuracies for each task type. No suffix means that all feature groups were taken into account, while the "_AE" suffix means only AE-based features were taken into account.

5. Discussion

In the previous section we constructed a large number of machine learning pipelines for cognitive load inference. The dataset we used was previously analysed in [5], thus we juxtapose our results with those presented by the dataset collectors. First, for the busy vs relaxed problem, the Wi-Mind authors achieve 75% classification accuracy (CA) using 1-D CNN + LSTM classifier. We, on the other hand, achieve 76% CA for the general (task-oblivious) model using RF classifier. Furthermore, we reach up to 87% for a particular task type using XGBoost – a classifier not experimented with in [5] – implying the importance of experimentation with diverse classifiers. For the general two-class low vs high cognitive load inference, the authors of [5] achieve 52% CA with a random forest classifier, while we achieve 59% CA with a random forest classifier. We assume that the improvement comes from a more detailed hyperparameter tuning we performed. Finally, for the three-class task-oblivious low vs medium vs high cognitive load inference problem, they achieve 37% CA with a random forest classifier, while we reach 40% CA with a fine-tuned SVM classifier.

Confident that our work delivers highly-optimised classification pipelines, we assess the autoencoders' ability to extract features beyond those already known and reported in the related literature. The results from Section 4 show that in most cases handcrafted features (breathing- and heart rate-related features) are sufficient, and features extracted from autoencoders do not offer improvements. However, for certain problem-task type combinations autoencoder-based features can offer the highest CA. For the BvR problem, for the SX task type, AE based features achieve the highest CA of 67%. For the LvH problem, in case of the FA task type, AE based features achieve the highest CA of 67%. Finally, for the LvMvH problem, for the FA task type, AE based features achieve the highest CA of 46%. Only for the LvH and FA task type, the highest CA using AE based features is also the overall highest CA for any task type in that problem.

The findings of our research are surprising. In theory, any function could be modeled with a deep learning model, therefore an AE should be able to find features that are at least as good as the handcrafted ones. Yet, in most cases, despite a rather detailed hyperparameter tuning, we failed to match the performance of handcrafted features. We postulate that the following could be the causes:

- **Insufficient training data.** On average, AEs contain significantly more parameters than classical machine learning algorithms. Consequently, a larger training set is needed. In Section 3.1 we see that for the BvR problem there are less than 400 data instances, while for the LvH and LvMvH problems, each class has more than 900 instances of data. Looking at the tables with results in Section 4, we see that the discrepancies between the highest mean accuracies using only AE feature groups and the highest mean accuracies using all feature groups are the highest for the BvR problem.
- **Noise in the data.** Wireless signals are notoriously noisy. While the handcrafted features may be robust to certain noise artefacts, AEs could mistakenly "hook" on intermittent noise, which could then lead to over-fitted classifiers. Indeed, the contractive AE, which should be the most immune to noise, appears to be performing slightly better than other AE flavors.
- **Suboptimal neural architectures.** In this work we used fixed simple AE architectures. Whether architectures with a different number of layers, different layers widths, and other parameters would perform better is an open question.

6. Conclusion

Riding on the wave of impressive results in computer vision, speech recognition, and other domains, deep learning was quickly adopted by human-computer interaction (HCI) researchers. Yet, while there are no doubts about the dominance of deep learning for problems such as visual object recognition[1], its suitability for various HCI problems is not that clear.

[1]In 2012 a CNN called AlexNet outperformed all other approaches submitted to the ImageNet competition by a large margin, and neural networks have dominated the said competition ever since.

In this paper we pitted deep learning-based automatic feature extraction against features extracted using conventional expert knowledge on the problem of human cognitive load inference from wireless ranging signals. Our analysis encompassed combinations of four different autoencoders for automatic feature extraction, seven different classifiers, as well as four different expert-feature groups, all tested on three different cognitive load inference problem flavours and six different cognitive tasks performed by 23 volunteers.

The thorough examination hints that deep learning is by no means a panacea. While autoencoder-based feature extraction achieves comparable results as the best expert-based feature extraction for certain tasks, such as SX, it is often outperformed by expert-based extraction and learning. We believe that the particularities of the problem and the dataset, including a relatively modest number of training datapoints, a high level of noise in the data, as well as the lack of a wider neural network architecture search lead to such results. Based on our experience, we caution against an overly optimistic use of deep learning for less-examined problems in the area of human-computer interaction.

7. Acknowledgments

This paper is based on the primary author's bachelor's thesis [12], with the other two authors acting as the mentor and co-mentor for that thesis. We would also like to thank the authors of Wi-Mind [5] for making their data and code publicly available.

References

[1] M. Weiser, The computer for the 21st century, Scientific American 265 (1991) 94–104. URL: http://dx.doi.org/10.1038/scientificamerican0991-94. doi:10.1038/scientificamerican0991-94.

[2] E. Haapalainen, S. Kim, J. F. Forlizzi, A. K. Dey, Psycho-physiological measures for assessing cognitive load, in: Proceedings of the 12th ACM international conference on Ubiquitous computing, 2010, pp. 301–310.

[3] M. Gjoreski, M. Luštrek, V. Pejović, My watch says i'm busy: Inferring cognitive load with low-cost wearables, in: UbiTtention 2018: 3rd International Workshop on Smart and Ambient Notification and Attention Management (with UbiComp'18), 2018.

[4] Y. Abdelrahman, E. Velloso, T. Dingler, A. Schmidt, F. Vetere, Cognitive heat: exploring the usage of thermal imaging to unobtrusively estimate cognitive load, Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies 1 (2017) 33.

[5] V. Pejović, T. Matkovič, M. Ciglarič, Wireless ranging for contactless cognitive load inference in ubiquitous computing, International Journal of Human–Computer Interaction (2021) 1–25.

[6] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[7] F. G. Paas, J. J. Van Merriënboer, Instructional control of cognitive load in the training of complex cognitive tasks, Educational psychology review 6 (1994) 351–371.

[8] S. G. Hart, L. E. Staveland, Development of nasa-tlx (task load index): Results of empirical and theoretical research, in: Advances in psychology, volume 52, Elsevier, 1988, pp. 139–183.

[9] M. Gjoreski, T. Kolenik, T. Knez, M. Luštrek, M. Gams, H. Gjoreski, V. Pejović, Datasets for cognitive load inference using wearable sensors and psychological traits, Applied Sciences 10 (2020) 3843.

[10] T. Plötz, N. Y. Hammerla, P. L. Olivier, Feature learning for activity recognition in ubiquitous computing, in: Twenty-second international joint conference on artificial intelligence, 2011.

[11] S. Rifai, P. Vincent, X. Muller, X. Glorot, Y. Bengio, Contractive auto-encoders: Explicit invariance during feature extraction, in: ICML, 2011, pp. 833–840. URL: https://icml.cc/2011/papers/455_icmlpaper.pdf.

[12] A. Kristan, Examining the potential of autoencoders for automatic feature extraction in wireless cognitive load inference, Bachelor's thesis, University of Ljubljana, Faculty of computer and information science, 2021. URL: https://repozitorij.uni-lj.si/IzpisGradiva.php?lang=eng&id=129656.