Université de Montréal

Évaluation de la charge mentale des pilotes en manœuvre aérienne

Par

Maxime Antoine

Unité académique d'Informatique et de Recherche Opérationnelle, Faculté des Arts et Sciences

Mémoire présenté en vue de l'obtention du grade de Maîtrise

en Informatique, option Intelligence Artificielle

Août, 2022

Université de Montréal

Département d'Informatique et de Recherche Opérationnelle, Faculté des Arts et Sciences

---

Ce mémoire intitulé

Évaluation de la charge mentale des pilotes en manœuvre aérienne

Présenté par

Maxime Antoine

A été évalué par un jury composé des personnes suivantes

Eugene Syriani
Président-rapporteur

Claude Frasson
Directeur de recherche

Michalis Famelis
Membre du jury

Résumé

La charge de travail cognitive d'un pilote d'aviation, qui englobe sa capacité mentale à effectuer les manœuvres d'un avion, varie selon l'étape de pilotage et le nombre de tâches convergeant simultanément sur le pilote. Cette charge de travail peut entraîner des erreurs de pilotage aux conséquences graves. La plupart des erreurs se produisent pendant la procédure de décollage ou d'atterrissage. Cette étude vise à mesurer et prédire la charge de travail cognitive d'un pilote lors d'une procédure de décollage afin de mieux comprendre et potentiellement prévenir ces erreurs humaines. Pour y parvenir, nous avons créé une solution logicielle pour mesurer et surveiller en temps réel la fréquence cardiaque, la dilatation pupillaire et la charge de travail cognitive d'un pilote. Le logiciel est également capable de déclencher des événements de défaillance pour déclencher une modification de la charge de travail cognitive à la demande. À l'aide d'un casque électroencéphalogramme (EEG) permettant de mesurer l'activité électrique du cerveau, d'un moniteur de fréquence cardiaque, d'un eye tracker et d'un simulateur, nous avons créé une configuration d'environnement où les pilotes devaient faire décoller un avion A320 avec et sans pannes sans le savoir au préalable. Cette étude a rassemblé 136 décollages sur 13 pilotes pour plus de 9 heures de données de séries chronologiques, soit 2 millions de lignes combinées. De plus, nous avons étudié la relation entre la fréquence cardiaque, la dilatation de la pupille et la charge de travail cognitive lors d'une tâche critique. Cette étude a révélé, à l'aide d'une analyse statistique, qu'un moment critique, comme une panne de moteur, augmente la fréquence cardiaque, la dilatation de la pupille et la charge de travail cognitive d'un pilote. Ensuite, cette recherche a utilisé différents modèles d'apprentissage automatique et d'apprentissage en profondeur pour prédire la charge de travail cognitive d'un pilote pendant le décollage. Nous avons constaté qu'en utilisant un modèle d'apprentissage en profondeur long short-term memory empilé, nous étions en mesure de prédire la charge de travail cognitive 5 secondes dans le futur. Le modèle long short-term memory empilé a donné une erreur quadratique moyenne (MSE) de 44,09, une erreur racine quadratique moyenne (RMSE) de 6,64 et une erreur absolue moyenne de 5,28, prouvant qu'il est possible de prédire la charge de travail cognitive.

Mots-clés : charge cognitive, pilote, apprentissage profond, apprentissage machine, EEG

Abstract

The cognitive workload for an aviation pilot, which englobes a pilot's mental capacity to perform aircraft's maneuvers, varies according to the piloting stage and the number of tasks converging simultaneously on the pilot. This workload can lead to piloting errors with severe consequences, where most errors occur during the takeoff or landing procedure. This study aims to predict the cognitive workload of a pilot during a takeoff procedure in order to better understand and potentially prevent these human errors. To achieve this, we created a software solution to measure and monitor in real-time the heart rate, pupil dilation, and cognitive workload of a pilot. The software is also capable of triggering failure events to trigger a change in cognitive workload on demand. Using an electroencephalogram (EEG) headset which measures the electrical brain activity, a heart rate monitor, an eye tracker, and a simulator, we created an environment setup where pilots had to take off an A320 airplane with and without failures without priorly knowing it. This study gathered 136 takeoffs across 13 pilots for more than 9 hours of time-series data, or 2 million rows combined. Moreover, we investigated the relation between heart rate, pupil dilation, and cognitive workload during a critical task. This study found, using statistical F-test analysis, that a critical moment, such as an engine failure, augments the heart rate, pupil dilation, and cognitive workload of a pilot. Next, this research utilized different machine learning and deep learning models to predict the cognitive workload of a pilot during takeoff. We found that, when using a stacked long short-term memory deep learning model, we were able to predict future cognitive workload 5 seconds into the future. The stacked long short-term memory model resulted in a mean square error of 44.09, a root mean square error of 6.64, and an mean absolute error of 5.28, demonstrating that it is possible to predict future cognitive workload.

Keywords: cognitive load, aviation, heart rate, pupil dilation, machine learning, deep learning, EEG

Table of content

Résumé 5
Abstract 7
Table of content 9
List of tables 15
List of figures 17
List of acronyms and abbreviations 19
Acknowledgments 23
Chapter 1 Introduction 25
1.1 Cognitive workload in aviation 26
1.2 Research hypotheses and objectives 27
1.3 Research outline 28
Chapter 2 State of the art 29
2.1 Cognitive workload 29
2.1.1 Types of measurement 30
2.1.1.1 Subjective measures 30
2.1.1.2 Performance Measures 31
2.1.1.3 Physiological Measures 31
2.1.1.4 Behavioral Measures 32
2.1.2 Types of cognitive workload 32
2.1.3 Human body and cognitive workload 33
2.1.3.1 Heart rate and cognitive workload 33
2.1.3.2 Pupil Dilation and cognitive workload 33

2.2 Heart rate 34
2.2.1 ECG 35
2.2.2 BPM 37
2.3 Pupil dilation 38
2.3.1 Eye gaze 38
2.4 Brain 40
2.4.1 Brain areas 40
2.4.2 Brain frequencies 41
2.5 Machine learning models 43
2.5.1 Linear regression 43
2.5.2 Support vector regression (SVR) 43
2.5.3 Multi-layer perceptron regressor (MLP) 44
2.5.4 Convolutional neural networks (CNN) 46
2.5.5 Long short-term memory (LSTM) 47
2.6 Performance measures 48
2.6.1 Mean Square Error (MSE) 49
2.6.2 Root Mean Square Error (RMSE) 49
2.6.3 Mean Absolute Error (MAE) 49
2.7 Conclusion 50
Chapter 3 - Event Generator & Connector Software 51
3.1 Event Generator 53
3.1.1 Architecture 53
3.1.1.1 Functional interface 54
3.1.1.2 Synchronizer server 54

3.1.1.3 Expendability 55
3.1.2 Functionalities 56
3.1.2.1 Events page 57
3.1.2.2 Scenarios page 58
3.1.2.3 Control panel page 59
3.1.2.4 Output Panel page 60
3.1.2.5 Statistics page 60
3.2 Connector 60
3.2.1 Architecture 61
3.2.2 Modules 62
3.2.2.1 Heart rate module 62
3.2.2.2 Eye tracking module 63
3.2.2.3 EEG module 63
3.2.2.4 X-Plane executor module 64
3.2.2.5 Logging module 65
3.2.2.6 Screen recording module 65
3.2.3 Multithreading 65
3.2.4 Multiple Connectors 66
3.3 Conclusion 67
Chapter 4 – Experiment 69
4.1 Participants 71
4.2 Setup 71
4.2.1 Measurement devices 71
4.2.1.1 Polar H10 72

4.2.1.2 Gazepoint GP3 72

4.2.1.3 NCO EEG headset 73

4.2.2 Experiment environment 73

4.3 Procedure 75

4.3.1 Greetings 75

4.3.2 Familiarization flight 76

4.3.3 Experiment flight 76

4.4 Conclusion 77

Chapter 5 – Data 79

5.1 Time synchronization 79

5.2 Data synchronization 80

5.3 Data features 81

5.4 Data merge 82

5.4.1 Solution: Average lowest frequency 83

5.5 Data interpretation 85

5.5.1 Hypothesis 1 85

5.5.2 Hypothesis 2 91

5.6 Data analysis 93

5.6.1 Noise 93

5.6.2 Groups 94

5.7 Conclusion 96

Chapter 6 – Workload prediction models 97

6.1 Preprocessing 97

6.2 Feature selection 99

6.3 Results ... 101

6.3.1 Models results ... 101

6.3.2 Other results ... 103

6.4 Discussion ... 104

6.5 Conclusion ... 106

Chapter 7 – Conclusion ... 107

7.1 Limitations ... 108

7.2 Future Work ... 108

Bibliography ... 109

List of tables

Table 1 – An overview of the different scenarios. 70

Table 2 – An overview of the groups and their respective sessions. 70

Table 3 - An overview of the captured data with its corresponding description. 82

Table 4 - A representation of the merged table. 83

Table 5 - A representation of the merged table using the lowest frequency. 84

Table 6 - Models performances without using CW as input 102

List of figures

Figure 1. – A visual representation of a PQRST wave (Madona et al., 2021). 36
Figure 2. – An example of depolarization and repolarization (Wikipedia, 2022). 36
Figure 3. – An example of an electrocardiogram paper (My-EKG, n.d.). 37
Figure 4. – An overview of the PCCR logic (Ball, 2014) 39
Figure 5. – A visual representation of the cortex areas (Farnsworth, 2021) 41
Figure 6. – Example of SVR with Slack Variables (scikit-learn, n.d.) 44
Figure 7. – The structure of an artificial neuron (Goodfellow et al., 2016) 45
Figure 8. – The MLP model architecture 46
Figure 9. – The CNN model architecture 47
Figure 10. – The Stacked LSTM architecture 48
Figure 11. – An overview of the complete architecture 51
Figure 12. – An overview of the events page 57
Figure 13. – An overview of the Scenario page. 58
Figure 14. – An overview of the control panel page 59
Figure 15. – An overview of the Connector architecture. 62
Figure 16. – An overview of multiple Connectors running on different computers. 67
Figure 17. – The experiment with the participant (left) and the pilot monitor (middle) 74
Figure 18. – The pilot's environment with the measurement tools 74
Figure 19. – A complete flight output representation of an A320 pilot. 87
Figure 20. – A complete flight output representation of a non-licensed pilot 88
Figure 21. – Min-max value of CW within a 4-second timeframe 91
Figure 22. – Average HR before and after engine failure 92
Figure 23. – Average PD before and after an engine failure 93
Figure 24. – Average workload per scenario per group 94
Figure 25. – Average HR and PD per group per scenario 95
Figure 26. – Structure overview of the dataset 97
Figure 27. – Visual representation of the NaN values in the dataset. 98

Figure 28. – The Pearson correlation matrix of the dataset. 100
Figure 29. – Ten best features for ML models. 100
Figure 30. – SVR model predictions (red) vs true labels (blue) 103
Figure 31. – Stacked LSTM model predictions (red) vs true labels (blue) 103
Figure 32. – Stacked LSTM architecture predicting HR 5 seconds into the future 104

List of acronyms and abbreviations

API: Application programming interface

BLE: Bluetooth low energy

BPM: Beats per minute

CW: Cognitive workload

ECG: electrocardiogram

EEG: Electroencephalogram

EF: Engine failure

GATT: Generic attribute protocol

HR: Heart rate

PD: Pupil dilation

SGD: Stochastic Gradient Descent

UI: User interface

UUID: Universally unique identifier

XPC: XPlaneConnect

Acknowledgments

Completing this thesis would never have been possible without the expertise, knowledge, and kindness of my research supervisor, Prof. Claude Frasson. I want to express my sincere gratitude and sympathy for his guidance throughout this thesis.

Moreover, I would like to thank Dr. Hamdi Ben Abdesslem for his help during this thesis. Without his patience, knowledge, and kindness, this research would never have been as fun and enjoyable as it has been.

To continue, this project would never have been possible without the expertise, sympathy, and resources from BMU, Bombardier, CAE, and CRIAQ, whom I would like to thank. A special thank you to Jocelyn Archambault for his kindness and help during this research. Furthermore, I would personally like to thank NSERC for giving me the opportunity to pursue this research.

I would also like to thank my girlfriend, Emmanuelle, for her love, patience, and support these last few months. Thank you for keeping me motivated throughout this thesis and sticking with me through the difficult times.

Finally, I would like to thank my parents for giving me the opportunity to pursue a master's degree across the world. Thank you for believing in me since day one.

Chapter 1 Introduction

Hart and Staveland from NASA describe cognitive workload (CW) as the user's perceived level of mental effort influenced by task load and task design (Hart & Staveland, 1988). Because of this, CW has been a hot topic in the research community for the last decades. While many studies exist around measuring CW, few studies are done on how to measure CW in real-time. This method of measuring CW is relevant for different industries where measuring the CW of a person could help predict future behavior and avoid poor decisions. This method is particularly relevant in aviation, where passengers' security is the number one priority.

To further develop this idea, let us consider a dramatic event from a couple of years ago. In May 2018, A Boeing 737-200 flying from Havana to Holguin in Cuba crashed a couple of minutes after takeoff killing 112 people inside (Robinson, 2019). The Civil Aviation Institute of Cuba (IACC) reported that: "The most probable cause of the accident was the collapse of the aircraft as a result of its entry into abnormal positions immediately after liftoff during the takeoff, which led to the loss of control of the plane due to a chain of errors, with a predominance of the human factor."

The report stated that a lack of training, poor decision-making, and other factors contributed to this dramatic event. What black boxes were unable to measure was the pilot's mental state when taking off. Was he tired? Was he not paying enough attention when following the procedures? Was he distracted when taking off? Predicting these factors and responding to them accordingly could have helped the pilot and maybe prevent such a dramatic event. Predicting the CW of a person, together with other cognitive measurements, could help understand the pilot's mental state at any given time and respond to it accordingly to ensure the safety of passengers. The next section will give more details about CW within the aviation industry.

1.1 Cognitive workload in aviation

CW within the aviation industry, which englobes the mental capacity of a pilot to perform the maneuvers of an aircraft, varies according to the piloting stage and the number of tasks converging at the same time on the pilot. This load can lead to pilot errors with serious consequences. While the aviation industry has a variety of procedures to minimize human errors, they still happen.

The National Transportation Safety Board (NTSB), the organism that tracks aviation accidents in the United-States, reports that 80% of all aviation accidents are due to human errors, with sometimes dramatic consequences (Panish et al., n.d.). Most errors occur during the takeoff or landing procedure. Because of this, measuring their CW could help the aviation industry to better understand a pilot’s mental state when errors happen and help pilots during flights by recommendation. To understand and measure the CW of a pilot in real-time, different cognitive and physical measurements need to be combined and analyzed to understand a pilot's mental state at a given time and predict it accordingly. For example, physical measurements such as the airplane metrics combined with other measurements related to CW, such as the heart rate (HR) or the pupil dilation (PD) of a pilot, could be used to predict the pilot's future CW better. Moreover, this study is part of the Pilot AI project with BMU, Bombardier, CAE, CRIAQ, and NSERC to further develop the knowledge and expertise of artificial intelligence within the aviation industry.

This study aims to predict the CW of pilots during a takeoff procedure using different measurement tools. Thanks to this prediction, future research from the Pilot AI project will be built on top to determine and suggest the following best action to take in case of cognitive overload. Moreover, this study aims to gather data during a takeoff procedure that will be used in future research on the Pilot AI project. This research can be seen as an essential starting block that will fuel considerable research within the Pilot AI project. Finally, this study aims to create a plug-and-play machine learning model which will gather data from different software and hardware devices in real-time. The program will transfer data to the model and predict it in real-time so other research can be built on top of it. The following section will describe the hypotheses and objectives that are necessary in order to achieve such predictions.

1.2 Research hypotheses and objectives

Considering previous problem and investigations, we formulate the following hypotheses:

Hypothesis 1: Is it possible to measure the CW of pilots in real-time during a takeoff experience?

Hypothesis 2: Is it possible to establish a correlation between the measured CW and the measured PD and HR during a critical event?

Hypothesis 3: Is it possible to predict the CW of a pilot based on his previous behavior?

To answer the described hypotheses, we must achieve the following objectives:

Objective 1: To demonstrate that we can measure the CW, we must create an environment that generates a change in CW on demand. To achieve this, we will use a flight simulator and integrate software that can generate failures during a flight.

Objective 2: Being able to measure a pilot's CW, HR, and PD. We will use an EEG headset, a heart rate monitor, and an eye tracker with the previously mentioned software solution to achieve this. In addition to the software requirements in objective 1, this software must also be capable of monitoring, tracking, and saving flight events and cognitive measurements in real-time. With objective 1 and objective 2, we can prove hypothesis 1.

Objective 3: Compare the different data entries and curves extracted from experiments during a workload-intensive event. We will use statistical analysis and correlation methods to demonstrate a relationship and prove hypothesis 2.

Objective 4: Research, train, and compare different machine learning and deep learning models able to predict future CW. We will investigate, compare, optimize, and train state-of-the-art machine learning and deep learning models from the literature. The data from the experiment will be utilized and adequately prepared for training. Different performance measures will be used to compare each model. Finding a model with good performances on the test set will prove hypothesis 3.

1.3 Research outline

Chapter 2: Contains the state-of-the-art research around CW. The basic definition of CW and research linking CW, HR, and PD. Each part will be reviewed and argued with different research papers and data to understand better the link between each part and how we could utilize them. Moreover, the different machine learning models and performance measures will be presented.

Chapter 3: Contains an in-depth overview of the functionalities and architecture of the software that was implemented to generate and capture CW, HR, PD, and other data points. This software implementation is part of objective 1 and objective 2.

Chapter 4: Explains the experiment's details created to gather the pilot's data. This chapter describes the environment, and the procedure details, together with the different questionnaires and documents used during the experiment. This chapter also gives more information about the participants during the experiment.

Chapter 5: This chapter discusses and analyzes the data generated from the experiment. This chapter also provides information about the final data output architecture. This chapter will prove hypothesis 1 and use the methods described in objective 3 to prove hypothesis 2.

Chapter 6: Contains the approach used and the results of machine learning and deep learning models. The training and evaluation of the different models will be given and compared as explained in objective 4. It will also answer hypothesis 3 and argument the discovered results.

Chapter 7: Concludes this research regarding the different hypotheses and objectives introduced in the previous section, the overall limitations of this research, and the future work perspectives of this research.

Chapter 2 State of the art

2.1 Cognitive workload

Cognitive workload represents a particular mental effort put by an individual in response to one or more cognitive tasks (Sevcenko et al., 2021). These tasks can vary from a range of situations such as writing, reading a book, solving math equations, etc. which requires mental effort. Tsang and Vidulich describe the mental effort as the number of cognitive resources that are necessary such as the treatment of information, making decisions, short-term memory memorization, etc. (Tsang & Vidulich, 2006).

This subject, while complex, has been of major interest to the scientific community, mainly around the subject of learning. Miller explains that our working memory is limited to approximately seven chunks of information (Miller, 1956). Because of this known limitation, much research has been done on how to optimize this limited space. While using our working memory, two situations can happen, cognitive overload and cognitive underload. Mayer and Moreno describe the state of cognitive overload as the situation where the demands evoked by a learning task exceed the cognitive capacity of the learner's working memory (Mayer & Moreno, 2003). This situation can lead to a decrease in performance and an increase in error-making. Another negative cognitive state is cognitive underload. According to Paas et al., learners undergo a cognitive underload when the amount of mental effort is inferior compared to what the task is requiring (Paas et al., 2003). This inferior cognitive effort also affects the performance of the learners in addition to an increase in error-making. We will focus on cognitive underload and overload during this research as it will showcase pilot errors that can later be analyzed.

This section will focus on the different state-of-the-art methods being used to measure the cognitive workload of a person. The types of measurements (Section 2.1.1) together with the different areas of the body of a person and their link with cognitive workload (Section 2.1.2) will be explained.

2.1.1 Types of measurement

To measure the cognitive workload of a person, different techniques are used. One can distinguish between four categories of measurements techniques: subjective, performance, behavioral, and psychological measures (Sevcenko et al., 2021). Each measure has its unique way of measuring different aspects of the cognitive workload. This section will focus on each technique, what they measure and how it can be applied in the context of pilots.

2.1.1.1 Subjective measures

Gopher and Braune report that subjective measurements are based on the observation that people can interpret and describe their experienced cognitive load during a particular task (Gopher & Braune, 1984). To achieve this, different questionnaires are used in the scientific community. The two most popular questionnaires are the SWAT, created by Reid and Nygren (Reid & Nygren, 1988), and the NASA-Task Load Index (NASA-TLX) developed by Hart and Staveland at NASA (Hart & Staveland, 1988). According to Sperling and Dosher, subjective measures are the easiest measures to collect, are inexpensive, and provide consistent results compared to other methods (Sperling & Dosher, 1986). Therefore, these questionnaires are heavily used when testing the cognitive workload of an individual. The NASA-TLX questionnaire requires the learners to evaluate their experiences, using scales that are predefined, immediately after finishing a specific task. With this, the learner can evaluate the mental, physical, and temporal demands of the task in addition to the performance, the effort, and the frustration encountered during the cognitive task.

These questionnaires, while popular, have also some limitations. One main issue is that learners must answer the questionnaire immediately after completing a certain task. Hancock explains that depending on the success or failure of a task, the perceived cognitive load during the filling of the questionnaire could be biased and influence the answers to it (Hancock, 1989). Another problem is that a retrospective view of an experienced cognitive load may be distorted by fading memory. Sevcenko et al. explain that only a summary of the experienced cognitive workload can be clutched this way, and not fine variations of it (Sevcenko et al., 2021).

2.1.1.2 Performance Measures

Performance measures will measure the variation in the learner’s performance at different tasks (Sevcenko et al., 2021). In case of cognitive overload or underload, the performance of the learner should decrease (Babiloni, 2019). While being an obvious choice for measuring cognitive workload, this measurement also has some limitations. Brünken, Roland et al. show evidence that it cannot be determined that the variation in performance during a certain task is due to cognitive overload (Brünken, Roland et al., 2010). Arousal or motivation could also yield a variation in the overall performance of the learner. Another limitation of this measurement is that performance cannot always be measured during the task but only at its completion. This limitation makes it often a useless measurement tool for prediction or adaptation purposes (Sevcenko et al., 2021).

In the case of measuring a pilot’s cognitive workload, performance-based measures can become ambiguous. The performance of the pilot will only be based on if he followed the protocol or not. If a pilot does follow the procedures as expected every time, he will have a perfect score, and no change in performance will be observed. One interesting point of performance-based measures would be to calculate the number of errors the pilot did and create a performance score based on that.

2.1.1.3 Physiological Measures

Physiological measures concentrate on the physiological changes associated with cognitive states (Sevcenko et al., 2021). Different physiological signals are then recorded while the learner performs a particular task. Depending on the goal of the study, heart rate variation (HRV), pupil dilatation (PD), or electrodermal activity (EDA) sensors are used. One of the most popular physiological measurement tools is the electroencephalogram (EEG), which measures the brain's electrical activity. Jerčić et al. have demonstrated that pupil dilatation and heart rate measurements could be used and combined with EEG measurements to indicate a learner's cognitive workload (Jerčić et al., 2020). Section 2.1.3.1 and Section 2.1.3.2 will give more details regarding pupil dilation and heart rate and how they are linked with the cognitive workload.

The main advantage of physiological measures, and particularly EEG measures, is the ability to measure the learner's physiological changes in real-time. For this reason, we decided to use this measurement method to measure a pilot's CW during a takeoff.

2.1.1.4 Behavioral Measures

Sevcenko et al. describe behavioral measures as the analysis of differences in interaction behavior during task processing (Sevcenko et al., 2021). Depending on the studies and the goal of it, different behavioral measurements can be taken. Pouw et al. used the reaction time of the learners in addition to subjective measurements (Pouw et al., 2016). Ruiz et al. used speech and voice patterns of the learners to measure their cognitive workload (Ruiz et al., 2010). Another study of Pouw et al. used the eye movements of the learners as an indicator of cognitive activity (Pouw et al., 2016).

This measurement technique also gives the advantage of measuring a learner's cognitive load in real-time while doing a specific task. Given this, behavioral measures will be used together with physiological measurement techniques to measure the cognitive workload of a pilot in real-time during a takeoff.

2.1.2 Types of cognitive workload

According to the cognitive load theory, Sweller explains that three types of cognitive load exist (Sweller, 2011):

- **Intrinsic load**: This type of cognitive load refers to the fundamental difficulty of a specific problem. The brain must use its long-term memory, short-term memory, and strategic cognition to solve the problem. This difficulty and load do not change regardless of external factors.
- **Extraneous load**: This load refers to how new information is presented. The information could be presented visually or just spoken by another person. Each person will have more ease processing visual information and thus have less extraneous load than others and vice versa. It is the external factor that creates the cognitive load in this case.
- **Germane load**: This cognitive load refers to how a person uses their personal intelligence and memory capacity to create mental schemas. These schemas are used to

solve various problems presented by previous cognitive loads. This type of cognitive load is present when the brain creates a learning process to assimilate new information and use it to solve problems.

2.1.3 Human body and cognitive workload

Studies have been made regarding the link between some regions of the human body and cognitive workload. While lots of research have found links with different body parts, this research focused on two specifically: pupil dilation (PD) and the learner's heart rate (HR). These categories were chosen because their measurement tools can easily be integrated into a pilot's cockpit. Moreover, these categories have been well studied in the research community and offer a solid base to expand further during this research.

While each area will be explained separately, it is common to use a combination of these body parts to have a more precise measure of the learner's CW. For example, Jerčić et al. used the HR in addition to the PD of each learner to measure its CW and attention (Jerčić et al., 2020).

2.1.3.1 Heart rate and cognitive workload

One area of the body that is linked with the CW of a person, is a person’s HR. Jerčić et al. state that HR is linked to both the sympathetic and parasympathetic nervous system (Jerčić et al., 2020), where sympathetic activity tends to increase HR, while parasympathetic activity decreases the HR of a learner. Studies have shown that recall of pleasant and unpleasant memories prompts acceleration of the HR, showing that arousal determines the HR of a person (Lang et al., 1993). Sosnowski et al. demonstrated in their study that an increase in the task’s difficulty increased the HR of the learners (Sosnowski et al., 2004). To measure the HR, electrocardiograms (ECG) have been used in many studies (Jerčić et al., 2020) (Rani et al., 2007). ECG provides a cheap and reliable way to measure a learner's HR and can be used in combination with an EEG headset to compare and correlate the HR and CW of a person.

2.1.3.2 Pupil Dilation and cognitive workload

Kahneman et al. used PD as part of his empirical foundation for his attention theory. Since then, several studies have been done using pupillary dilation as a proxy for evaluating CW. The pupil dilates when cognitive load increases, until task demands exceed the available cognitive resources (Zekveld et al., 2011). Zekveld et al. further explain that three aspects of the pupil response are analyzed: the peak amplitude, the latency of the peak dilation (peak latency), and the mean dilation in a relevant time window.

Different studies used PD as a proxy for cognitive load. Siegle et al. explain that PD and blinking provide complementary, mutually exclusive indices of information processing that are both linked with CW (Siegle et al., 2008). According to Siegle et al., there is a correlation between blinking and PD with regards to CW. The occurrence of a blink precludes the measurement of pupil diameter giving a better indication of when to measure the pupil dilation of a learner. Zekveld et al. used PD to observe the correlation between hearing loss, age, and cognitive ability. PD was used to measure the CW from different learners (Zekveld et al., 2011). His study showed that the pupil response systematically increased with decreasing speech intelligibility. Lastly, Palinko et al. used PD to estimate the CW of drivers that were speaking and driving simultaneously (Palinko et al., 2010). While these studies only represent a small portion of the research done around the link and use between PD and CW, other research studies utilize PD in a similar manner than the one described.

Sections 2.2 and 2.3 will give an in-depth description of the functioning of measuring HR and PD and how it can be extracted from physical impulses and movements of the body.

2.2 Heart rate

One of the most reported physiological measures that correlate with CW is heart rate (HR) (Xi et al., 2019). As mentioned in Section 2.1.3.1, Different studies use HR in accordance with CW. This section gives further information about ECG signals and how they can be transformed into heart rate in beats per minute (BPM). Fortunately, modern tools and trackers do this calculation in real-time and transfer the information into a readable BPM.

2.2.1 ECG

ECG sensors directly use electrical signals produced by heart activity as compared to PPG which uses electrical signals derived from light reflected due to changes in blood flow during heart activity (Neurosky, n.d.). When producing these electrical signals, ECG sensors will capture them in real-time and output a PQRST wave, like Figure 1. To understand Figure 1, the terms depolarization and repolarization, as shown in Figure 2, must be introduced. Depolarization is the electrical change in the resting membrane which causes a positive electrical increase. Repolarization is the direct opposite of depolarization (Lakna, 2018). This difference can be detected by the ECG sensors in real-time. The P-wave shown in Figure 1 represents the atrial depolarization or atrial contraction of the heart (AL-Ziarjawey, 2015). Then follows a small downward Q peak followed by the dominant R peak and a small S inferior downwards peak together known as the QRS complex. The QRS complex represents the depolarization of the right and left ventricles of the heart combined with the contraction of the large ventricular muscles (Madona et al., 2021). Finally, at the end of this wave, Figure 1 shows the T-wave, representing the ventricle's repolarization. The change in polarization is shown in Figure 2 by the different upward or downward peaks. This cycle can be represented as a graph where the y-axis is the amount of mV and the x-axis the time, in seconds, at which this value was measured. While obvious, it is worth mentioning that this cycle can be measured for every heartbeat.

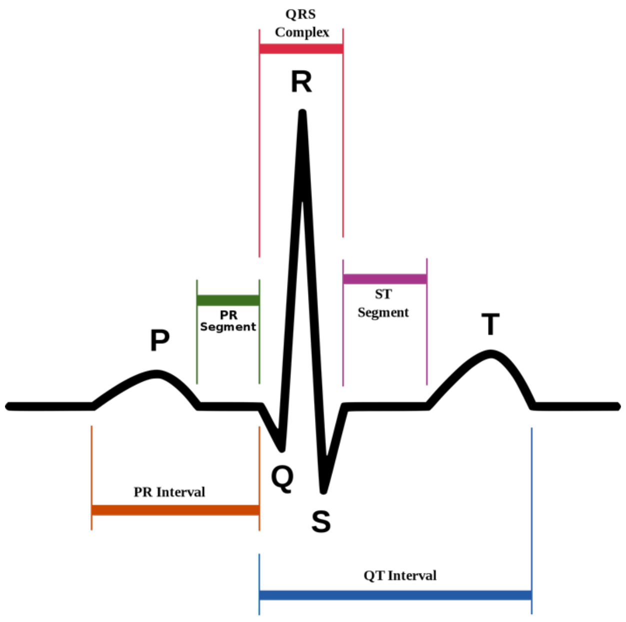


Figure 1. – A visual representation of a PQRST wave (Madona et al., 2021).

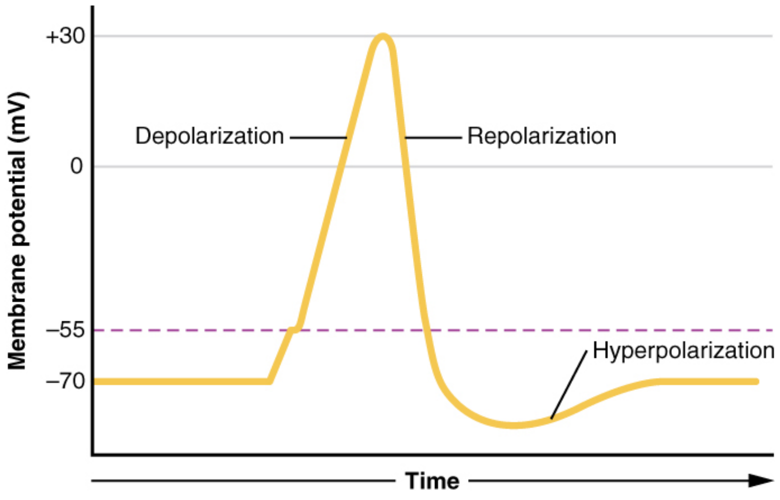


Figure 2. – An example of depolarization and repolarization (Wikipedia, 2022).

2.2.2 BPM

Now that the heart depolarization and repolarization can be measured via ECG, it is also possible to extract the beats per minute (BPM) out of it. To be able to measure the BPM, the rate of measurement must be known. This rate acts as a reference for measuring the distance between two heart beats. The following example explains how to extract the BPM out of an ECG graph. When calculating ECG, the volts measured by the polarization and depolarization of the heart are written on a squared paper as shown on Figure 3. The representation of one square in seconds or in mV depends on the machine that is being used. Obviously, modern heart rate devices do not use the same setup to calculate the BPM of person. Nevertheless, the same logic is used to measure it digitally.

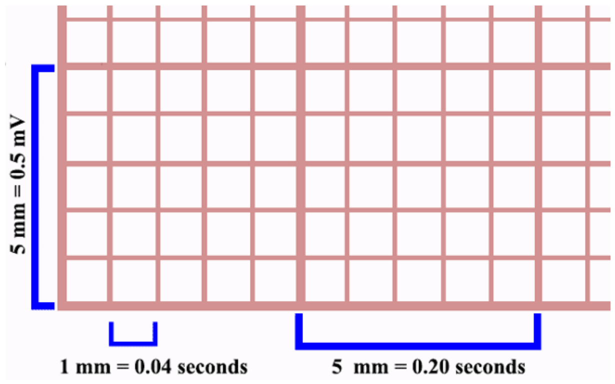


Figure 3. – An example of an electrocardiogram paper (My-EKG, n.d.).

Let's give an example of an ECG paper moving at a speed of 25mm/second. This means that the paper moves at:

25mm x 60 sec = 1500mm/minute.

The BPM, commonly referred as HR, can be represented as the number of cardiac cycles per minute or the number of times the heart beats every single minute (Sattar & Chhabra, 2022). This can be achieved by measuring the interval between two R peaks for an entire minute. This measurement is commonly referred as the RR interval. If, for example, one RR interval measures *20mm* on paper, then the HR of this person is simply:

1500 / 20 = 75 BPM

It is important to mention that this BPM value represents the instantaneous HR of a person, meaning it is the number of times the heart would beat in one minute if the duration of successive cardiac cycles were constant (Sattar & Chhabra, 2022). Therefore, it is possible to have different BPM values within the same minute and why it is not necessary to wait one complete minute to measure the BPM of a person.

2.3 Pupil dilation

The pupil is an opening at the center of the iris surrounded by the iris muscles that will respond to outside stimuli which in turn will control the amount of light that reaches the retina (Jerčić et al., 2020). There exist two types of iris muscles: the sphincter muscle, forming a band around the inner margin of the pupil, and the radial muscle. Also, two synergistic pathways are present, the parasympathetic and the sympathetic pathway. The sympathetic pathway innervates the radial dilator muscle which causes the dilation of the pupil (Jerčić et al., 2020). On the other hand, the parasympathetic pathway innervates the sphincter, which causes the constriction of the pupil.

As explained in Section 2.1.3.2, this dilation and constriction of the pupil has a link with the CW of a person. To measure PD, different methods are being used. One popular method, used by different Tobii eye tracker products, consist of using basic trigonometry equations. The optical sensor of the eye tracker registers an image of the eyes (Tobii, 2015). The eye tracker must also acquire information about the distance between the camera and the actual pupil (Tobii, 2015). The firmware can then calculate the pupil size by measuring the diameter of the pupil on the image and multiply it with some scaling factor using simple trigonometry (Tobii, 2015). There exist different ways of representing the size of a pupil. A pupil can be represented in millimeters or in pixels. However, the measurement unit does not matter as this paper, and most scientific research, focuses on the pupil size variations over time. This variation gives us key information about the CW of a pilot while flying.

2.3.1 Eye gaze

Another data point that can be useful in understanding a person's CW is his eye gaze. The eye gaze gives information about where a person is currently looking at. For most eye trackers, the area where the eye gaze can be tracked is limited to a computer screen. To know a person's eye gaze, at a given time, most eye trackers use a technique called the pupil center corneal reflection (PCCR) method (Tobii, 2015). Figure 4 shows a visual explanation of this technique. For this method to function properly, a light source is used to illuminate the eye of the subject. This light will cause a highly visible reflection on the projected eye. This reflection, visible on the cornea, is then captured by a camera, together with the pupil (Tobii, 2015). With this information, a vector is then created between those points, giving the current eye gaze of the user on the screen. Because every eye is different, a calibration is necessary beforehand. The calibration will make the subject look at the center of the screen, followed by every corner of the screen. Using this technique, it is possible to capture the XY-position of the person looking at the screen. It is important to mention that this method only adds more context to why a person's CW was high or low at a certain time *t.* By knowing where a person was looking at, it makes it easier to analyze and interpret the PD of that same person.

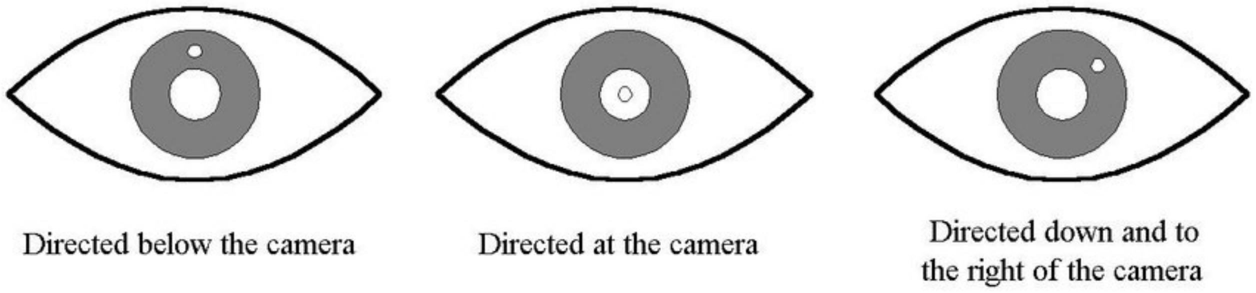

Directed below the camera
Directed at the camera
Directed down and to the right of the camera

Figure 4. – An overview of the PCCR logic (Ball, 2014)

2.4 Brain

This section gives an in-depth overview of the different vital components of the brain to understand CW and how to calculate it. The following references are from Dr. Bryn Farnsworth's article "What is EEG (Electroencephalography) and How Does it Work?" (Farnsworth, 2021). The brain is composed of billions of cells, where half are neurons, and the other half help and facilitates the activity of neurons. These neurons interconnect via synapses, which act as gateways of synaptic activity. This synaptic activity generates an electrical impulse commonly referred to as a postsynaptic potential. Fortunately, more than one neuron fires when doing any activity. These thousands of neurons, all firing in sync, generate an electrical field that can be measured outside the skull, on the head surface. Electroencephalography is the physiological method of recording the brain's electrical activity via electrodes placed on the scalp surface. This headset with electrodes is referred to as an electroencephalogram or EEG. The output of this brain activity is measured in Volts (V). Because the actual measured electrical activity is of such a small magnitude, the recorded data is sent to an amplifier to showcase the resulted Voltage better.

2.4.1 Brain areas

The brain is divided into different areas, all responsible for processing and reacting to certain types of information at a given time. Thanks to EEG, which records the electrical activity generated by the brain, it is possible to interpret which area of the cortex is triggered given a particular piece of information. Figure 5 gives a visual representation of the following brain areas:

- **Occipital cortex:** All visual information is processed in this cortex area. An EEG will gather electrical impulses from this area when given a particular image or video.
- **Parietal cortex:** The parietal cortex gathers information from external sources and internal sensory feedback. This cortex area will take all this information and transform it into a logical representation of how our bodies should relate to the environment and vice versa. For example, this cortex area processes, stores, and retrieves the shape, size, and orientation of objects to be grasped.

- **Temporal cortex:** The temporal cortex processes sensory input to something logical using visual memories, language, and emotional associations. The left temporal cortex involves comprehending written and spoken language. The middle regions are used for spatial navigation.
- **Frontal cortex:** This cortex area manages executive functions: it helps us maintain control, plan, and monitor our behavior.

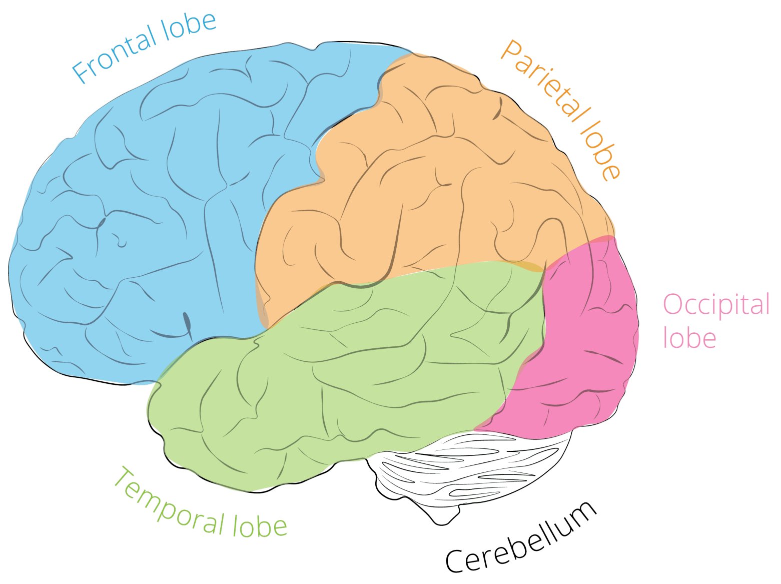


Figure 5. – A visual representation of the cortex areas (Farnsworth, 2021)

2.4.2 Brain frequencies

When measuring brain activity via EEG, not only the regions of electrical activities are essential but also the different frequencies at which they are captured. Apart from the regional characteristics of specific electrical activity, it is also possible to analyze which frequencies primarily drive the ongoing activity. These frequencies give more information about a person's cognitive, affective, or attentional state. Here is a summary of the different brain frequencies and what they represent:

- **Delta:** Delta waves are related to sleep and relaxation. These waves are defined between 1 - 4 Hz.
- **Theta:** Theta waves are linked to creativity, intuition, and concentration. These waves are between 4 and 8 Hz frequencies.
- **Alpha:** Alpha waves are prominent when the brain achieves to be in a calm, relaxed state. These waves are linked to relaxation and rest. Therefore, alpha waves are often used to calm people when doing therapy. These waves are defined between 8 - 13 Hz frequencies.
- **Beta:** Beta waves are prominently measured during information processing and sustained attention. These are often measured between 13 - 22 Hz. Moreover, beta waves are also related to anxiety and hyperactivity. These are measured between 28 and 32 Hz frequencies and are known as high beta waves.
- **Gamma:** The last type of wave is the Gamma wave. Gamma waves represent 38 – 42 Hz frequencies but are most likely around 40Hz in practice. These waves are associated with processing complex tasks. Moreover, gamma waves are associated with rapid eye movement, known as micro-saccades. These waves could be helpful to and correlated with the pupil dilation and eye gaze data extracted from the different available eye trackers.

This brief overview of the different common brain waves helps us give a better understanding of the brain's functioning and what is being targeted when measuring someone's cortex. Given previous information, we can use the brain waves that are being measured to calculate the cognitive workload of a person. It is essential to focus on beta waves as those waves give the most information about cognitive workload. Moreover, we will focus on theta waves to measure the level of concentration of a pilot in real-time. An EEG headset will extract those measurements, analyze the frequencies, and transform these into coherent data types.

2.5 Machine learning models

To predict the future CW of a pilot, different machine learning techniques and models will be used to generate realistic predictions. This section introduces the different machine learning and deep learning models used to predict the CW throughout this research. In order to have a worst-case reference, a naive linear regression model is used as the baseline, meaning that every other model should at least have better performances than this naive, non-optimized linear model. Each model is explained, and arguments are given about why this research chose this model.

2.5.1 Linear regression

This model fits a linear model with coefficients w = (w1, …, wp) and aims to minimize the residual sum of squares between the actual targets and the predicted targets by linear approximation. The objective function for one feature looks as follows, where y_i is the target, w_i is the coefficient, and x_i is the feature:

$$MIN \sum_{i=1}^{n} \left(y_i - w_i x_i \right)^2$$

This model will be used as the baseline model. If another model outputs worse performances than this model, the chosen model was not a good fit and will be thrown out. Moreover, it gives a good starting point that is easy to understand and compare. This research used the ridge regression model as the basic linear regression model. This model is a linear least-squares model with l2 regularization (scikit-learn, n.d.). To find the best hyperparameters for this model, gridSearchCV from the Scikit-learn library was used. For this model, the hyperparameters are: {"alpha": 90, solver: "svd"}.

2.5.2 Support vector regression (SVR)

This model gives the ability to reduce the error to a particular range. For example, when predicting a pilot's CW, accept having a predicted CW that is +-5 from the authentic CW. SVR allows defining how much error is acceptable and will find a line or plane to fit the data. The purpose of SVR is to minimize the l2-norm of the coefficient vector. The objective function of SVR looks as follows:

$$MIN\ \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}|\xi_i|$$

To better illustrate this formula, consider Figure 6. This figure shows that epsilon (ϵ) corresponds to the error range. On the other hand, hyperparameter C gives the ability to adjust the tolerance for points outside of ϵ. The deviation from the margin $w_i x_i + \varepsilon$ or $w_i x_i - \varepsilon$ is denoted as ξ.

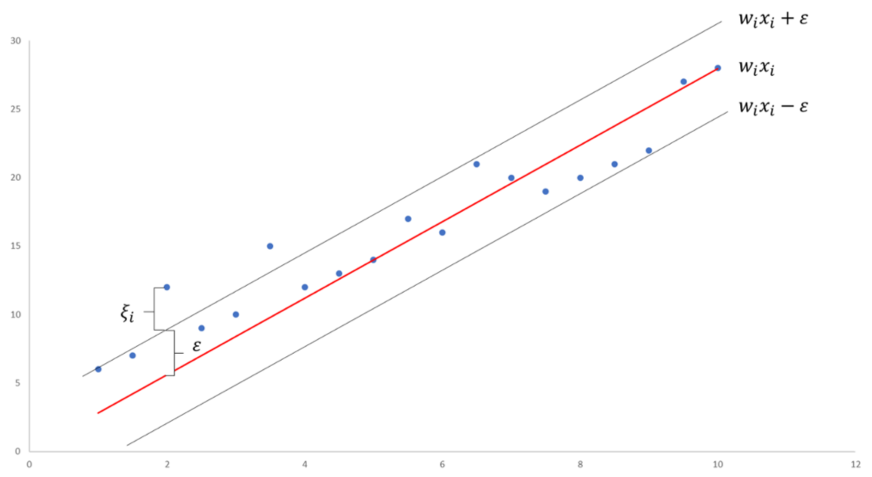


Figure 6. – Example of SVR with Slack Variables (scikit-learn, n.d.)

Although this research tries to predict the exact CW of a pilot at a particular time *t*, giving the model some error range could make the model more robust and better predict the CW without being too penalized when outputting a wrong prediction (scikit-learn, n.d.).

This research used a SVR regression model with hyperparameters: kernel = linear, gamma= auto, tol = 1e-1, C = 5, epsilon = 1. These values were found using GridSearchCv from the sklearn library.

2.5.3 Multi-layer perceptron regressor (MLP)

This supervised learning model learns the function $f(\cdot): R^d \rightarrow R^o$ where *d* represents the number of dimensions for the input and o represents the number of dimensions for the output. Given a set of features *X=x1, x2, ..., xd* and a target y, it can learn to approximate a non-linear regression function. This is possible thanks to the multiple artificial neurons that are stacked on top of one another. Figure 7 shows the structure of an artificial neuron.

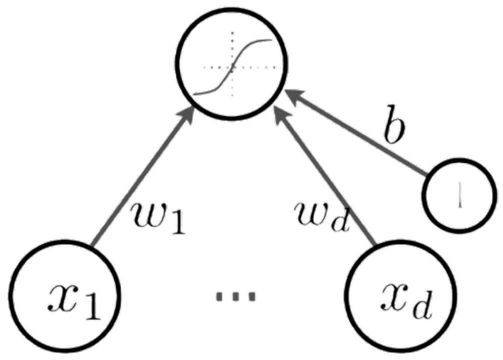


Figure 7. – The structure of an artificial neuron (Goodfellow et al., 2016)

The bottom layer represents a set of neurons *{xi|x1, x2, ..., xd}* representing the input layer. These inputs are combined with a set of weights (w) and a bias (b) giving the formula (Goodfellow et al., 2016):

$$a(\boldsymbol{x}) = b + \sum_i w_i x_i$$

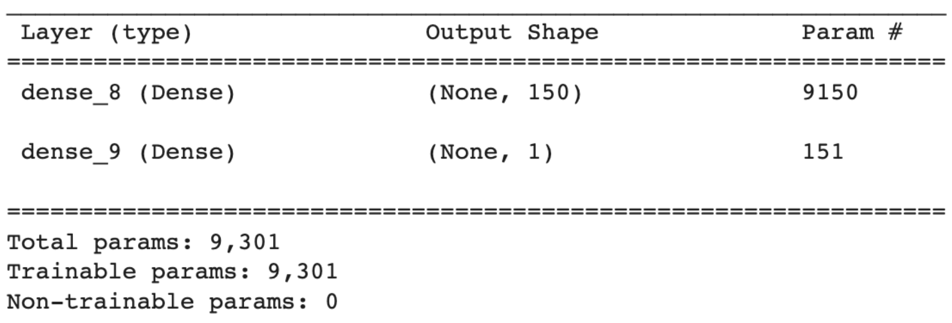
Finally, the result of the previous formula is passed into an activation function which outputs the result (Goodfellow et al., 2016):

$$h(\boldsymbol{x}) = g(a(\boldsymbol{x}))$$

This mechanism can be stacked using hidden layers, resulting in more complex non-linear representations. How the weights are updated falls outside this research's scope. Nevertheless, stochastic gradient descent using the Adam optimizer and ReLU activation function will be used throughout the different deep learning models this research will cover. As of writing this research, these parameters have been chosen as they are the state-of-the-art optimizer and activation function in the research community.

This MLP can also be considered the standard baseline for the deep learning models used throughout this chapter. Nevertheless, it can be used as a viable solution for predicting pilots'

CW, giving better results than a simple linear regression. This research used an MLP regression model with a ReLU activation function, using the ADAM optimizer and MSE loss function. This model ran over 150 epochs, with a batch size of 64.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 150)               9150

 dense_9 (Dense)             (None, 1)                 151

=================================================================
Total params: 9,301
Trainable params: 9,301
Non-trainable params: 0
```

Figure 8. – The MLP model architecture

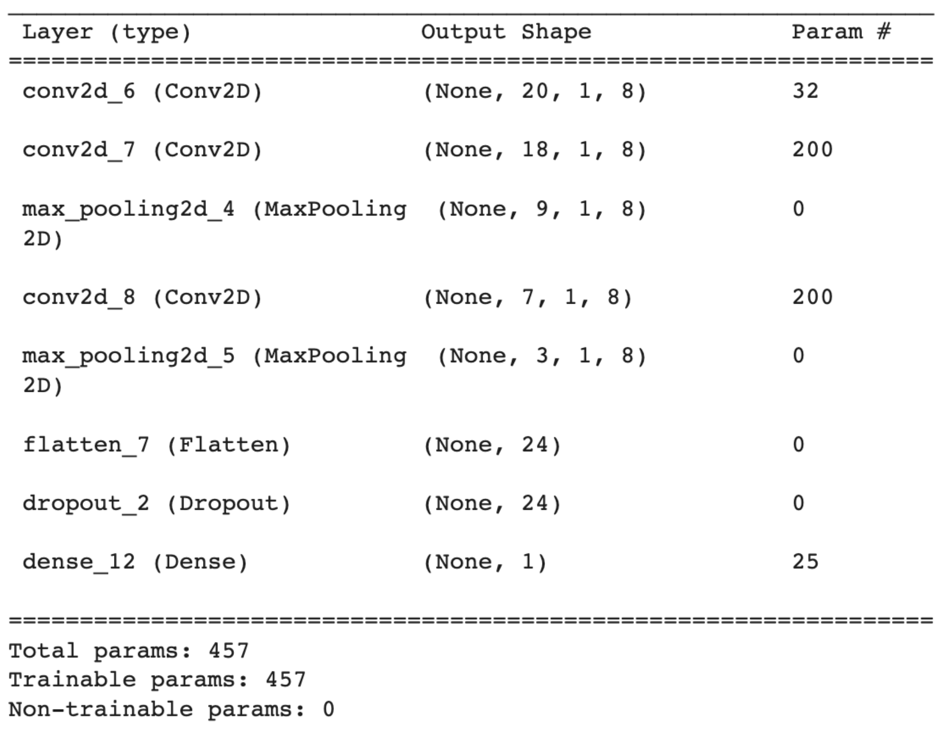
2.5.4 Convolutional neural networks (CNN)

While convolutional neural networks are often used for image, speech, and audio inputs, they can also be used for regression purposes. A CNN can be divided into three layers:

- Convolutional layer
- Pooling layer
- Fully connected (FC) layer

Multiple convolutional layers and pooling layers are often stacked together to form more complex CNN models (Goodfellow et al., 2016). The FC layer is the last layer of the model. CNN can be seen as a regularized MLP. Regularization methods like dropout or weight decay, together with the fact that a CNN does not need to be fully connected, make this model more robust to overfitting.

Figure 9 shows the CNN model used to predict the CW of a pilot. Different model architectures were tested, but this model implementation gave overall better results on average. The results are discussed in Section 6.5. This research used a CNN regression model with a ReLU activation function, the ADAM optimizer, and the MSE loss function. This model ran over 150 epochs, with a batch size of 64.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 20, 1, 8)          32

 conv2d_7 (Conv2D)           (None, 18, 1, 8)          200

 max_pooling2d_4 (MaxPooling  (None, 9, 1, 8)          0
 2D)

 conv2d_8 (Conv2D)           (None, 7, 1, 8)           200

 max_pooling2d_5 (MaxPooling  (None, 3, 1, 8)          0
 2D)

 flatten_7 (Flatten)         (None, 24)                0

 dropout_2 (Dropout)         (None, 24)                0

 dense_12 (Dense)            (None, 1)                 25

=================================================================
Total params: 457
Trainable params: 457
Non-trainable params: 0
```

Figure 9. – The CNN model architecture

2.5.5 Long short-term memory (LSTM)

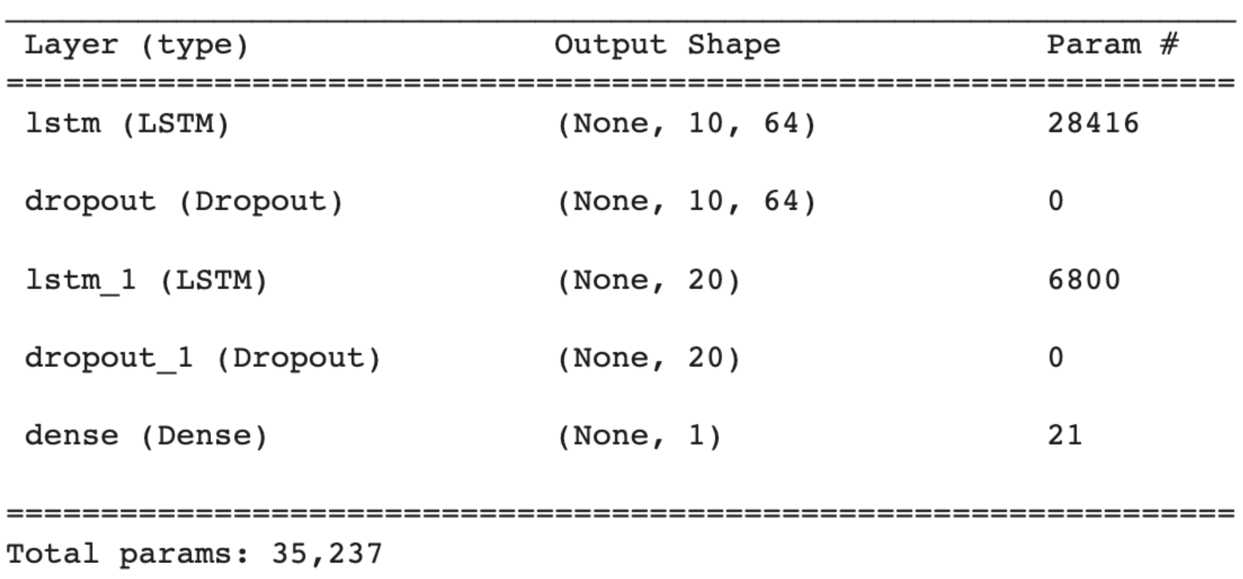
Long short-term memory model, or LSTM, is a popular supervised learning algorithm from the family of recurrent neural networks (RNN) (Goodfellow et al., 2016). To understand how LSTMs work and what problems they solve, let us start with RNNs. An RNN is a supervised learning algorithm that predicts the next target based on previous inputs. Since the pilot's HR, PD, and CW are measured over time, it makes this algorithm a perfect candidate for predicting multivariate time series data. One drawback of RNNs is that problems can arise when the time series data is too long. These problems are known as the vanishing or exploding gradients problem (Goodfellow et al., 2016). To solve this issue, LSTM models are used. The inner workings of LSTM fall outside the scope of this research, but it can be seen as a viable solution for the vanishing and exploding gradient problem.

Different LSTM architectures were tested to predict a person's CW. This research implemented three different architectures with the following hyperparameters:

- **Vanilla LSTM**: one LSTM layer, ADAM optimizer, MSE loss function, dropout of 0.2

- **Bidirectional-LSTM**: one bidirectional LSTM Layer, ADAM optimizer, and MSE loss function with no dropout.
- **Stacked LSTM**: 2 LSTM layers, ADAM optimizer, MSE loss function, and two dropout layers at 0.5.

Moreover, each model was trained during 150 epochs, with a batch size of 32. Figure 10 shows the stacked LSTM architecture. This architecture resulted in the best performances described in Section 6.5. In total, 35 237 parameters were trained for this model.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 10, 64)            28416

 dropout (Dropout)           (None, 10, 64)            0

 lstm_1 (LSTM)               (None, 20)                6800

 dropout_1 (Dropout)         (None, 20)                0

 dense (Dense)               (None, 1)                 21

=================================================================
Total params: 35,237
```

Figure 10. – The Stacked LSTM architecture

2.6 Performance measures

Lastly, performance measures are necessary to compare the predicted results of the different models explained in the previous section. Since this research deals with multivariate time series data and regression models, different regression performance measures are necessary. The following sections describe the different performance measures used to evaluate and compare the models. The actual results of each model are explained in Section 6.5.

2.6.1 Mean Square Error (MSE)

As its name says, the mean square error measures the average of the squares of the errors. The MSE is measured using the following formula, where y is the true value and y-hat is the predicted value:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y_i}\right)^2$$

This metric takes the average squared difference between the estimated values and the actual value. The advantage of MSE is that it shows how much the predicted results deviate from the actual result.

2.6.2 Root Mean Square Error (RMSE)

The Root Mean Square Error, or RMSE, is the square root of MSE. This metric is more often used than MSE because the output values of the MSE of a model can sometimes become quite large. The square root tunes down the squared part of the MSE formula, avoiding large values. One advantage of both the MSE and RMSE is that they give values that are in direct relationship to the actual target value. The formula for RMSE is:

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(y_i - \widehat{y_i}\right)^2} = \sqrt{MSE}$$

2.6.3 Mean Absolute Error (MAE)

Like previous metrics, the Mean Absolute Error calculates the magnitude of difference between the predictions and the accurate labels. This time, the average of all absolute errors is taken. MAE is also commonly referred to as the L1 loss function. The formula for MAE is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y_i}|$$

This measurement gives yet another quantifiable value for each model to later compare. It is essential to notice that although MAE and MSE are necessary performance measures, RMSE is more commonly used in combination with R-squared to evaluate the performance of a regression model.

2.7 Conclusion

In this chapter, we went over the different studies related to CW. We defined what cognitive workload is and why it is such a popular research topic. Furthermore, we explained the different types of measurement currently available to measure CW and how certain body parts such as PD and HR are linked to CW. Ultimately, we chose to use physiological and behavioral measurement techniques as they can be measured in real-time. Moreover, we chose to use PD and HR measurements together with CW from EEG as all three can be measured in a simulator environment or in a cockpit. It is essential to note that EEG measures the actual CW of a pilot, whereas PD and HR are indicators related to CW. Finally, this research will concentrate on the beta and theta waves as they offer a better understanding of concentration and information processing. Furthermore, we introduced the different machine learning and deep learning models used during this research together with their corresponding performance measures. The next chapter will go over the software solution that was created in order to measure the CW, HR, and PD of a pilot during a takeoff exercise. The knowledge of this chapter will be used throughout the implementation of the software in the next chapter.

Chapter 3 - Event Generator & Connector Software

Flight simulators are a crucial aspect of a pilot's training. They offer a safe environment to sharpen their skills and prepare them for different flight scenarios. While most simulators are very good at such tasks, they often lack a way of triggering or monitoring specific events during a simulation. The first objective of this research is to create an environment that can start, capture, and log different aircraft steps and allow the generation of failures to analyze the pilot's resulting CW and measure the variation of a pilot's CW. Moreover, the software must also capture these measurements in real-time.

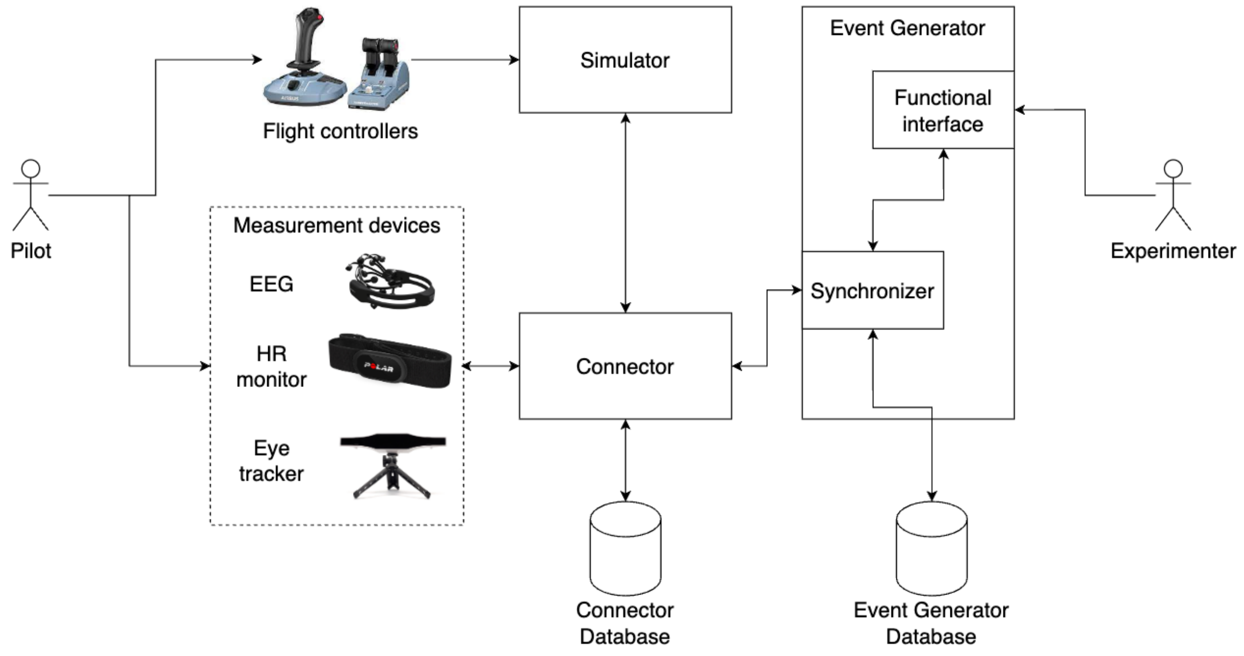


Figure 11. – An overview of the complete architecture

To solve this first objective, we created the architecture shown in Figure 11, which consists of:

- **The Event Generator:** A web application with an interface used by the experimenter to interact, trigger, and monitor events in real-time.
- **The Connector:** A multithreaded Python server whose objective is to capture all the data from the measurement devices in real-time. Moreover, the Connector interacts directly

with the simulator, sending requests from the Event Generator to the simulator and vice versa.

- **Simulator:** The flight simulator. It is connected with a throttle and a joystick for the pilot to interact.
- **Measurement devices:** The devices used to measure the pilot's CW, HR, and PD in real-time. The devices are either on the pilot directly (EEG, HR monitor) or pointing directly at the pilot (eye tracker).

While many commercial flight simulators exist, this research focused on one simulator, in particular, X-Plane 11. X-Plane 11 is a realistic flight simulator created by Laminar Research[1]. The simulator has various commercial and military airplanes available for pilot training. Compared to others available, the simulator's main advantage is its ability to add external plugins to change many of its functionalities. Specific commands, called Datarefs, can be triggered to interact with the simulator. The term Datarefs will be used throughout this chapter. The inner workings of these Datarefs fall outside the scope of this research. Nevertheless, it can be compared to an HTTP request being sent to a server. Moreover, it is also possible to acquire realistic replicas of different aircraft. Using the knowledge and expertise of Bombardier and CAE, this research chose the Airbus A320 as the airplane that will be used during the experiments. The reason is that the A320 is a famous airplane with many pilots already trained for it, which augments the chances of having participants. Within X-Plane 11, the Flight Factor A320 ultimate pack[2] was chosen as the A320 aircraft plugin. The following sections will give an in-depth description of the two software solutions. Both functionalities and architectures will be explained.

One might wonder why this particular architecture has been chosen. Using this web architecture with a python server offers the possibility for expansion in the future. The web interface can be used on any device with an internet browser, is easy to update, and can be used and shown remotely from the computer where the participant is doing the experiment. Moreover, the Python server has been chosen as it allows multithreading within a server. The complete architecture has been created with expandability in mind from start to finish. Each

[1] https://www.x-plane.com
[2] https://store.x-plane.org/A320-Ultimate_p_688.html

module can be easily expanded using popular technologies known by many developers. Moreover, the hardware pieces of equipment have an intuitive SDK or API.

This chapter will first focus on the Event Generator, the reason for such a web application, its functionalities, and more. Next, this chapter will give details about the Connector, its different modules, how it is linked to the Event Generator and simulator, why it is multithreaded, and some solutions to avoid CPU overload.

3.1 Event Generator

To trigger and monitor different aircraft failures, an application called the Event Generator was created. This web application aims to facilitate the experiment's procedure and know that the data is being captured correctly. As will be explained further throughout this chapter and thesis, this architecture interacts with different hardware modules, each having its data transfer mechanism and logic. Because of the potential hardware bugs during the experiment, we created this web interface to monitor each hardware module in real-time. Moreover, this web interface also facilitates the triggering of certain events during a simulation in real-time. A web application was chosen because it allows connecting external measurement tools via API calls or WebSockets to the application with ease. This section will explain in detail all the Event Generator core functionalities, its architecture, and the expandability of the program.

3.1.1 Architecture

In this section, we will focus on the architecture of the Event Generator. As explained in the previous section, we chose to develop a web application. The reason is that a web app can connect to almost any other devices, simulators, and many more in real-time on any computer. This cross-operability ensures that, when delivering the project to the partners, it will be easy for them to start and run the Event Generator. Figure 11 shows that the Event Generator comprises a functional interface (Section 3.1.1.1) connected to a Synchronizer server (Section 3.1.1.2) which communicates with the Connector and the database.

3.1.1.1 Functional interface

The functional interface is a user interface made to facilitate the experimenter's work during the experiment. For example, failure events need to be triggered at a specific time during the experiment. This interface allows to do it using a button instead of typing commands in the terminal. Similar to a website page, the functional interface displays specific inputs and buttons so the experimenter can quickly trigger events and monitor the experiment from one central place. The JavaScript framework React has been chosen as the frontend library. React[3] is a popular JavaScript framework for building user interfaces. Ant design[4] has been chosen as the UI library. This UI library is primarily used on many applications, making the users feel immediately familiar with the overall UI when seeing it for the first time. Aside from the framework and its UI, WebSockets have been used to make the functional interface send and display data in real-time. The WebSocket API makes it possible to open a two-way interactive communication session between a client and a server (MDN, n.d.).

3.1.1.2 Synchronizer server

The Synchronizer server communicates with the Connector and the Event Generator database. Like the functional interface, it communicates with the Connector via WebSockets to ensure real-time communication between the two servers. Curious readers might wonder why this server is called the Synchronizer. To understand the name, some context is required. As will be explained in Section 3.2, the Connector server is made of different modules where specific modules require up to 80% of the CPU processing power. The Connector, combined with the flight simulator and the Event Generator, can cause CPU overload on the computer. To avoid this, we made it possible to run multiple Connector servers on different computers. Section 3.2.4 gives more information about this. To come back to the name, the Synchronizer's purpose is to synchronize the time of the different Connector servers, so the data has matching timestamps across computers. This synchronization plays a significant role as all the captured data are time series data.

[3] https://reactjs.org/

[4] https://ant.design/

Another purpose of the Synchronizer is to transfer the experimenter's input from the functional interface to the Connector, which will then send it to the simulator (see Section 3.2). To achieve this, it uses WebSockets to communicate with the Functional interface and other WebSockets to communicate with the Connector server(s). Lastly, the Event Generator is capable of creating scenarios made of different failure events. These scenarios are saved in a MongoDB database which can later be retrieved and triggered when necessary.

The server is a Node.js with TypeScript, a typed version of JavaScript, application. The server uses express, a popular web framework for Node.js that handles HTTP requests and WebSockets communication. Here is an example to understand the role of each module within the Event Generator. The experimenter interacts with the Functional interface. Clicking certain buttons triggers different WebSockets calls to the Synchronizer server. Depending on the call, the server might send the request to the Connector module or store data in the database. These requests to the Connector server happen when the experimenter triggers a failure event (or scenario) or asks for data related to the simulator or the measurement devices. When done, the Connector will send the data back to the Synchronizer server, showing it to the experimenter using the Functional interface.

3.1.1.3 Expendability

Using this architecture makes the Event Generator expandable for future improvement. The Synchronizer and the Functional interface were made to be expanded should it be necessary. Another critical aspect of the project was the ability to interact with other programs from the Pilot AI project. The architecture of the program was created around this principle. The Functional interface and Synchronizer server can easily replace or connect to other programs using standard communication protocols like HTTP or WebSockets. Thanks to this architecture, other students can quickly call GET and POST requests from the Event Generator and expect some results. A standalone API module could be developed and documented in the future to facilitate this process.

3.1.2 Functionalities

The Event Generator is divided into five pages. The following sections will describe each page, together with its utility. As explained in the previous section, the Event Generator was created focusing on expandability. This expandability offers the possibility to add, modify, or remove pages should it be necessary. The following pages are the minimum of distinct pages required to achieve this research's first objective.

3.1.2.1 Events page

As described in the first objective of this research, different flight events must be triggered to measure the variation in CW correctly. To achieve this quickly, the events page was created. On this page, individual events, each representing a white box, can be triggered with a button. Figure 12 demonstrates the page and the different types of events that are available. The events can be divided into:

- **Aircraft events:** Events that have a direct impact on the aircraft. For example, an engine fire event can be enabled or disabled. The experimenter must select the adequate input and click the trigger button.
- **Environment events:** Events that will trigger a change in the environment around the aircraft. For example, during a flight simulation, the experimenter can activate thunder, rain, and turbulence in real-time. Here too, the experimenter must put the correct percentage and trigger the event by clicking on the trigger button.

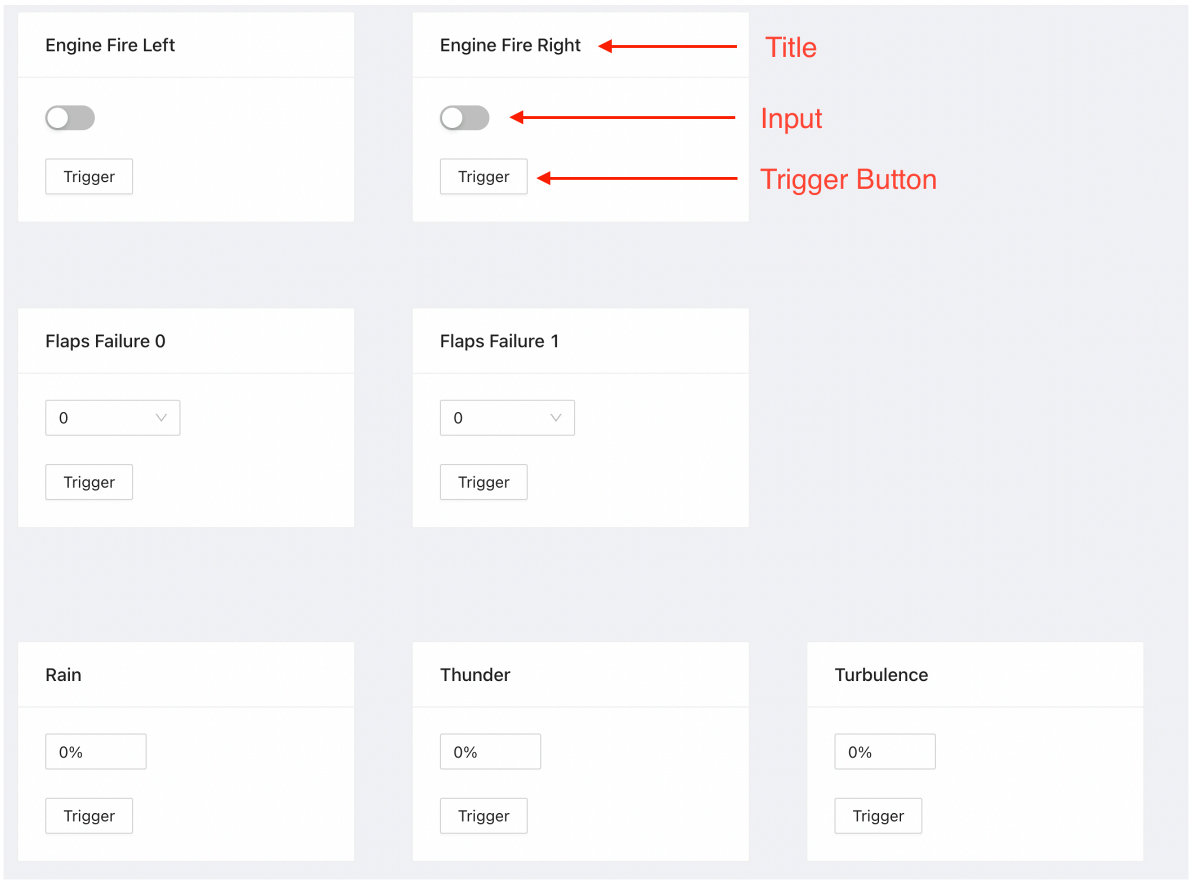
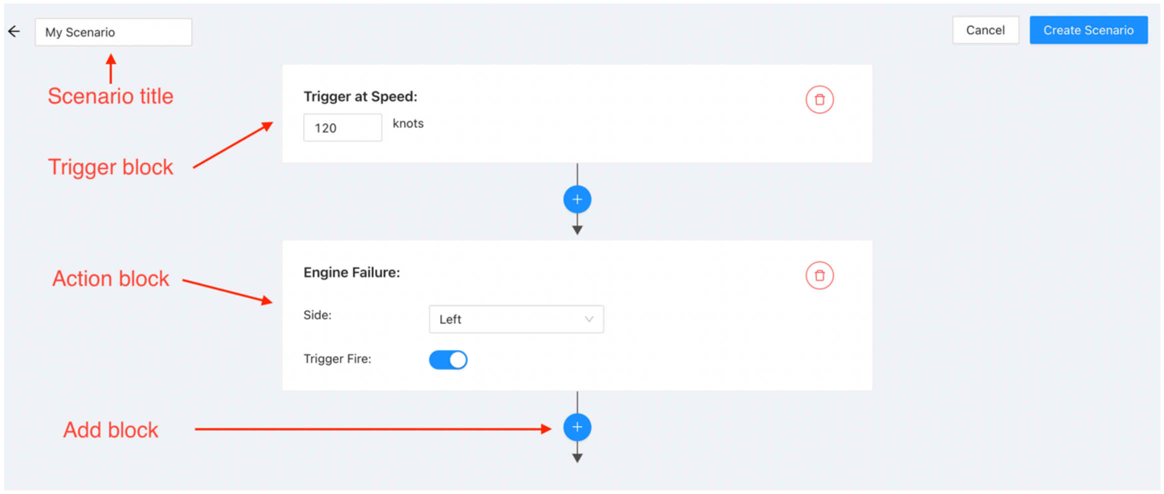


Figure 12. – An overview of the events page

Moreover, Figure 12 shows, in red, the different parts of an event box. Each event has a title, some inputs, and a trigger button. Each event's input has been created to match the Datarefs values of the X-Plane simulator. The event will be sent to the simulator and trigger a change when clicking the button. When run locally on the same network, the response time between clicking on the event and the actual event in the simulator is not noticeable.

3.1.2.2 Scenarios page

To further facilitate the experimenter's job, the scenarios page was created. This page aims to quickly create pre-defined scenarios with different triggers and actions to reproduce during the experiments. All the events on the events page are available on the scenarios page. Figure 13 gives an overview of the scenario creation page. Each scenario has a title, blocks representing a trigger or an event, and buttons to add blocks. When creating a scenario, one must start with a trigger block. During a simulation, the program will watch for the trigger value and execute the other blocks in sequence afterward. For example, one scenario used during the experiment is a left engine failure at 140 knots. With this scenario builder, one would only need to add a trigger speed block at 140 knots and an action block triggering a left engine failure. This way, the scenario will be triggered punctually for every participant and easily modified should it be necessary. All failure scenarios were created using this scenario builder. Moreover, since the triggering of the scenario is automatic, it will always occur at the exact moment for every pilot, hence reducing errors during the experiment


My Scenario

Trigger at Speed:
120 knots

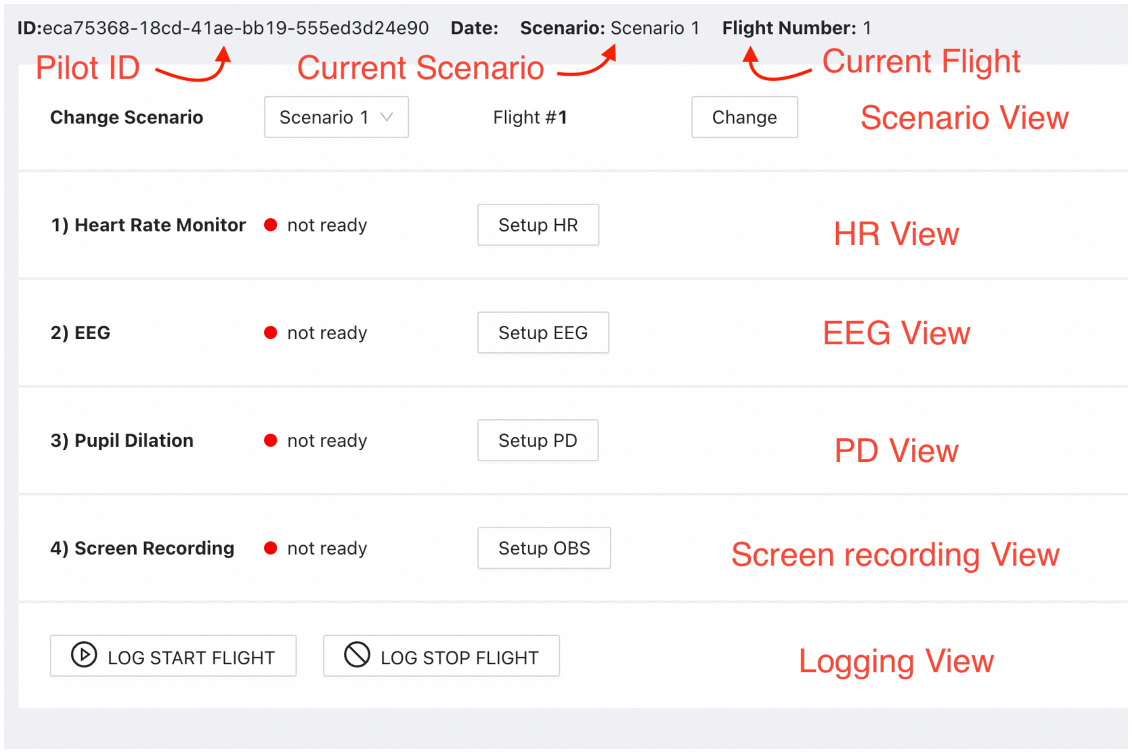
Engine Failure:
Side: Left
Trigger Fire:

Figure 13. – An overview of the Scenario page.

3.1.2.3 Control panel page

The control panel page is used during the experiment to overview all the essential information and enable the experimenter to manage the different actions required. Figure 14 gives an overview of this page. First, the page gives an overview of the different hardware modules that are connected in real-time. Each module has its view and an indicator indicating if the module is running correctly (green) or not (red). Each module can be started and stopped when necessary. Within this page, it is possible to control and command different actions to the Connector server that will, in turn, communicate with the simulator or the hardware components. Here is an overview of the different functionalities on this page:

- It shows the pilot ID, the current scenario being used in the simulator, and the current flight number
- Activate or deactivate a specific hardware module
- Monitor if a hardware module is still functioning or not in real-time.
- Switch and select scenarios during the experiment
- Log the start and stop of the takeoff simulation.


ID:eca75368-18cd-41ae-bb19-555ed3d24e90 **Date:** **Scenario:** Scenario 1 **Flight Number:** 1

Change Scenario Scenario 1 Flight #1

1) Heart Rate Monitor ● not ready

2) EEG ● not ready

3) Pupil Dilation ● not ready

4) Screen Recording ● not ready

Figure 14. – An overview of the control panel page

3.1.2.4 Output Panel page

On this page, the experimenter can extract a pilot's experiment data using its pilot ID. It will output a ZIP file containing the raw and merged data of the experiment. The file structure and the data structure are explained in Chapter 5. This page, while not necessary, was created to offer a complete solution where other researchers could further use this software in upcoming research studies around the same subject.

3.1.2.5 Statistics page

This page will show real-time statistics when measuring a pilot's CW, HR, or PD. Data captured during an experiment is sent directly to the Event Generator and shown on the statistics page. This page is used to confirm that the data is being captured correctly. Indirectly, this page also demonstrates that it is possible to send real-time data being captured by different hardware solutions. It is essential to note that this page could be expanded in the future to show a more in-depth, real-time analysis of the data that is being measured.

3.2 Connector

A different multithreaded program was necessary to communicate with different hardware and software modules in real-time. Because of this, we created a python server called the Connector. It communicates with every data gathering device and safely saves the current extracted data in a specific file on the computer running the Connector. The Connector server was also created with scalability in mind. Adding or removing different measuring devices and modules to the Connector is easy. In the future, if new methods or products are necessary to extract the CW of a person, it would be easy to integrate them into the server. Figure 15 shows a more detailed view of the Connector server. Notice that, compared to Figure 11, some parts that are not linked to the Connector server were removed for simplicity's sake. As mentioned in Section 3.1.2.3, the control panel page groups all hardware components together, giving a modular and easy setup to monitor the functioning of the Connector server.

The coming sections explain the Connector architecture (Section 3.2.1), the different modules within that architecture (Section 3.2.2), and why multithreading was chosen (Section 3.2.3). Finally, because multiple modules are running simultaneously, communicating in real-time on different threads, it was essential to give the ability to separate the Connector into multiple servers to avoid overloading the CPU of one computer. Section 3.2.4 will give more information about the solution found regarding this CPU overload problem and why it was necessary.

3.2.1 Architecture

The Connector is a multithreaded Python Flask[5] server that communicates exclusively via WebSockets. A Flask server was chosen because it creates an accessible API to communicate with and can run as a standalone server. When running, the Connector server can receive different Socket messages from the Event Generator, the simulator, or some measuring device and answer appropriately. The Event Generator communicates mainly via WebSockets with the Connector. This communication protocol ensures that both apps can send messages back and forth in real-time to trigger or display certain events accordingly. When initiated, the Connector will create different CSV and TSV files for each connected module. These files all share a unique UUID, the current date, and flight number to be able to merge related flights a posteriori. Chapter 5 gives more details about the merging logic of the files.

Figure 15 gives a detailed overview of the Connector's architecture. The Connector server has six distinct modules connected to different third-party software (green) and hardware (measurement devices). Each module is connected with the Synchronizer server from the Event Generator. Only the X-Plane Executor module is connected to the flight simulator. Each module runs on a separate thread to allow parallel processing. It is essential to note that the communication mechanism between a module and third-party software or hardware differs depending on the software or hardware (Bluetooth, UDP, WebSockets). The following section will give more details about each module within the Connector server.

[5] https://palletsprojects.com/p/flask/

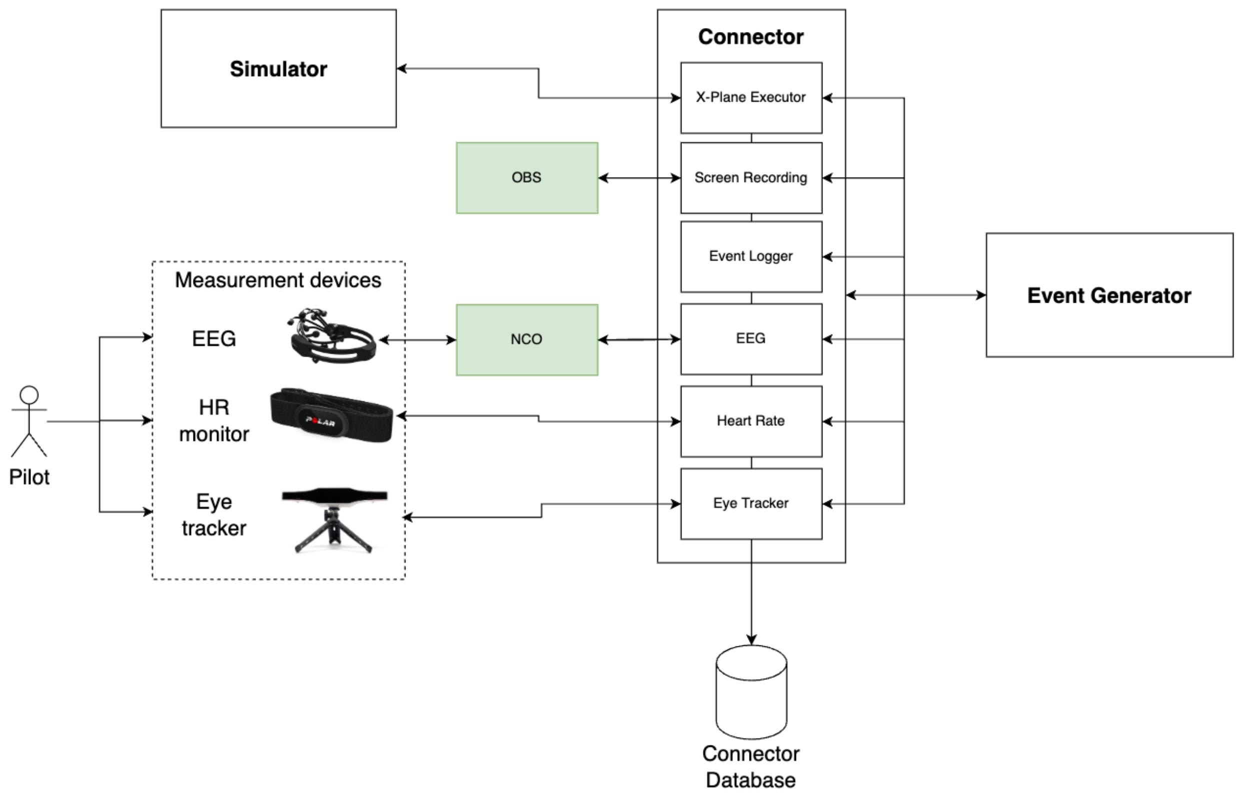


Figure 15. – An overview of the Connector architecture.

3.2.2 Modules

To better understand the different layers of the Connector, the following Section gives an overview of all the modules presented within the program. Each module has unique functions and characteristics, which are addressed in the following sections. Each module can be seen as a Python class, running independently.

3.2.2.1 Heart rate module

This section will focus on the internal architecture of the heart rate module. To be able to receive continuous data from the heart rate monitor, a specific architecture was created. This architecture is based on the Bluetooth GATT logic. The architecture can be seen as an infinite while loop. When instantiated, the module will enter an infinite loop that will indefinitely wait for data to be sent by the HR monitor. This way, the HR module will never miss incoming data, and the amount of data that can be received is also manageable. After multiple trials and errors, we used 1 second between two data entries. This delay means that a new BPM will be captured from the HR monitor every second. When a BPM is received, the HR module will immediately save the BPM, together with the current time that the BPM was received, in a CSV file. The module can also send this captured data in real-time to another server if necessary.

3.2.2.2 Eye tracking module

The eye tracker module works differently than the HR module. Unlike the HR module with its infinite loop, the eye tracker module does not need to gather the data. The data is gathered by the eye tracker software within the eye tracker immediately, at 60Hz. Adding an extra layer above the software, called the eye tracker module, ensures that additional data is saved and that the data generated by the software is stored where required. The module adds 60 times a second, the current timestamp at which the data gathering occurred because the eye-tracking software does not store that information by default. When a flight simulation is done, the eye tracker module asks the eye-tracking software for the data file and stores it where requested.

3.2.2.3 EEG module

This module measures the pilot's CW in real-time by connecting the module to the EEG headset. To achieve this, the module must use a third-party software called Mentor (Chaouachi et al., 2015). This software is a module from the NCO software, a proprietary software of the BMU lab that represents the convergence of multiple years of research into one extensive program called NCO (Benlamine et al., 2016). This software enables to read and interpret the raw data from an EEG headset in real-time and outputs different data. It is from the NCO software that the workload is extracted. NCO reads the EEG data, transforms the data, and outputs a file with a comprehensible workload score. Unfortunately, this software has no API available yet, so it was essential to find an alternative solution to automate the initialization and recording process of the data. Fortunately, NCO has an interface that could be exploited. This EEG module has two purposes:

- Simulate mouse clicks on the NCO interface to automate the whole process.
- Extract the data from the NCO program after each scenario and store it in a known place.

To simulate mouse clicks, a python script was created. This script includes different functions that achieve specific tasks within the NCO interface: initialize, record, and stop recording. Based on the frontend's input, specific functions will be triggered. Of course, this research does not see this as a viable long-term solution. This solution was found as an alternative because NCO does not have an API available for their program. Should BMU provide one in the future, only a couple of functions within the EEG module should be modified to make everything work.

Another critical point was the extraction of the data logs between each scenario. Unfortunately, NCO was not designed to enable offline and multiple recordings of data one after the other without re-initializing the whole program. To save a scenario's data, the file location of the recorded data within the NCO program is used to extract the data and copy it to a safe folder before it is destroyed when starting a new recording. This solution was needed because the program would overwrite the files between two recordings. This process was made possible thanks to BMU, who kindly updated the program to allow offline data storage and multiple recordings one after the other without the need to restart the program. Moreover, BMU helped with the analysis of the raw data together with the analysis of the workload data to ensure its validity of it.

3.2.2.4 X-Plane executor module

This module aims to connect the Connector and Event Generator application to the X-Plane 11 simulator. One essential plugin was necessary to achieve this connection, XPlaneConnect[6] (XPC). The XPC plugin is an open-source research tool created by NASA and used to interact with the X-Plane simulator. The particularity of this plugin is that it has been converted to C, C++, Java, MATLAB, and Python and gives the ability to communicate in real-time over the network via UDP. Being written in Python, the Connector uses the Python 3 version of the XPC plugin to communicate with the simulator. The module triggers all the environmental changes within the simulator using XPC and logs those changes with the logging module.

It can be seen as a wrapper around the XPC plugin. One requirement from the partners of this research was to create a program that could connect to any simulator. If the simulator should

[6] https://github.com/nasa/XPlaneConnect

change, only this module should be modified to create the connections between the program and the new simulator.

3.2.2.5 Logging module

This module was designed to give the ability to any other module to log data within a CSV file. More specifically, this module logs all X-Plane-generated events that X-Plane itself did not trigger and hence not saved in an X-Plane log file. The X-Plane Executor module heavily uses this module. A reference to this module is passed to other modules needing it to solve this. Whenever needed, it can save specific data in any structure wanted. An example of this would be when executing a scenario event triggered by the Event Generator; whenever a specific part of the scenario is executed and sent to X-Plane, a log of this event is saved within a CSV file with specific information regarding the event.

3.2.2.6 Screen recording module

The name of this last module spoils its core functioning. The screen recording module will record a specific scenario using the recording software OBS. When starting the recording, a timestamp is saved to indicate the beginning of the recording. The same is done when the recording stops. This timestamp logic gives colleagues more freedom to use information or not or to create programs on top of this generated data further down the line. It is also interesting to compare the data against the video to find helpful information and visually understand what the data is telling with these timestamps. When initializing the module, a WebSocket is created to enable communication with OBS. When started via the frontend, the WebSocket will tell OBS to start or stop the video recording and store the MP4 video in a specific folder that will be later extracted.

3.2.3 Multithreading

While reading the previous sections, attentive readers might have encountered one problem: the HR monitor module is in an infinite loop state. If only using one thread to run the Connector program, the Connector would stay in this infinite loop, only acquiring HR but nothing else. To avoid this problem, it is vital to create a multithreading architecture for the Connector.

Multithreading is a programming model that allows multiple threads to be created within a process, executing independently but concurrently and all sharing process resources (Burns, n.d.). This model allows applications to process requests without waiting for one request to finish simultaneously. This particularity is the sole reason why the Connector needs a multithreading architecture. When initiated, the Connector will create seven separate threads, one for each module and the python server itself. Each module can be used and extract data simultaneously while the Connector module remains operational for further requests. This architecture allows the Connector to be scalable for further improvements. If ever needed, it would be possible to add other modules to the Connector and spawn a new thread for it without causing any issues. After initiation, the different modules are allocated to each thread and asked to start immediately. The Connector will ask the different threads to stop working when the application stops. This method also allows the Connector module to trigger specific tasks within a thread. Finally, it is essential to mention that this procedure is also possible using a multiprocessing architecture. The critical difference is that multiprocessing does not share resources with other processes. This difference is problematic since the Connector application needs to share data between different modules on different occasions. For this reason, this research chose to use multithreading over multiprocessing.

3.2.4 Multiple Connectors

As explained at the beginning of this chapter, CPU overload could happen because all these modules are running concurrently. This potential overload is because two programs, X-Plane and NCO, require up to 80% of the processing power. To overcome this issue, we made it possible to run multiple Connectors on different machines, each executing specific modules. Figure 16 gives a better overview of this implementation. Multiple computers can start with a Connector server that runs only a particular set of modules. The Event Generator server must know which computer is running and what module each computer executes. To achieve this, the computer's local IP addresses are saved within a .env file within the Synchronizer. One problem that arises when implementing this architecture is the internal clock time. While running in the same time zone, two computers can have different internal clock times. This is where the Synchronizer comes in. The main goal of the Synchronizer server from the Event Generator is to synchronize the time between the different Connectors. To achieve this, the Synchronizer sends a request to all connected Connector servers asking for the computer's internal clock time. It stores these times in a CSV file and is later used to synchronize the time between all files from the different Connector servers. Chapter 5 goes more into debt on how this is achieved. Since all captured data are time-series data, it makes this synchronization a critical step and avoids the CPU overload problem. Moreover, it is essential to notice that the Event Generator does not need to rub on a computer running a Connector server. The Event Generator could run standalone on a third computer if desired. During the experiments, we decided to run two Connector servers with the Event Generator on one of the two computers.

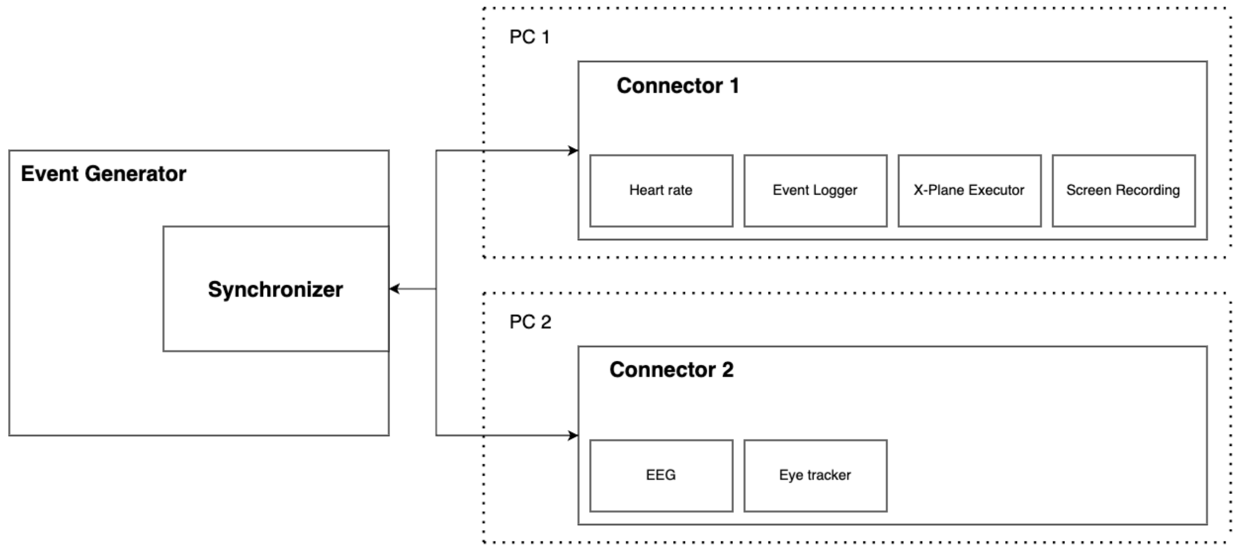


Figure 16. – An overview of multiple Connectors running on different computers.

3.3 Conclusion

In this chapter, we gave an in-depth overview of the software solution created to measure the CW, HR, PD, and other simulator events in real-time during a simulated takeoff. The inner workings of the Event Generator, which englobes the Functional interface and the Synchronizer server, were explained. The Connector servers were explained as well as the link between the Event Generator and the Connector server. Moreover, this software solution is connected and integrated with other research projects from the Pilot AI project. Finally, as requested by the partners, this solution is simulator agnostic and can be exported to other simulators should it ever be necessary. The next chapter goes over the experiment where we used the software solution in this chapter. Each component of this software was created with the experimental procedure in mind to avoid problems during the experiment.

Chapter 4 – Experiment

An experiment was done to gather the CW of pilots in real-time. This chapter gives a detailed overview of this experiment. To put into perspective, the previous chapter gave information about the architecture behind the experiment setup and how it gathered each critical data point in an automated way. This chapter, in comparison, is about what the experiment is, how it took place, the procedure, the participants, the environment, and the measurement tools used during the experiment. Moreover, because this experiment requires humans to participate, this research created an ethical certificate (CERSES-3849) that got approved by the ethics committee of University of Montreal. This certificate ensures that the experiment was done ethically and obliges the experimenters to follow it thoroughly to stay compliant.

To better understand the following chapter, some context is required. The experiment aims to measure pilots' CW, HR, and PD during a takeoff procedure in an Airbus A320. The plane is ready for takeoff on lane 06L of the Montréal airport (YUL). The participant must:

- Release the parking brake
- Do the takeoff procedure
- Climb until 3000 ft without using autopilot

If those objectives are achieved, we consider the takeoff successful. We would consider a takeoff not successful if:

- The airplane went too far outside the lane
- The participant crashes the airplane

We created different scenarios with different procedures to generate as much CW as possible. Some scenarios include failures and a different procedure to follow in case of failure. In total, six different scenarios were introduced. Table 1 gives an overview of each scenario. Each scenario changed in terms of time, weather conditions, and whether a failure would occur or not. Scenarios one to three were used for the regular takeoff sessions, while scenarios four to six were used for the failure sessions.

Table 1 – An overview of the different scenarios.

Scenario	Time	Weather	Failure
1	1:45 pm	No wind, no clouds	No
2	6:00 am	Clouds at 2700ft, rain	No
3	9:00 pm	No wind, no clouds	No
4	5:30 am	No wind, no clouds	Yes, EF at 80 knots
5	6:00 am	15 knots crosswind	Yes, EF at 140 knots
6	6:00 am	Low visibility, rain	Yes, EF at 80 knots

Participants were divided into two groups. Group one would start with a regular takeoff session of 20 minutes and later switch to a 20-minute failure session and vice versa for group two. Table 2 gives a better overview of the sessions each group had. There were no limits on the number of takeoffs that one participant could take during a session. This research observed that participants took between four to six takeoffs per session. This number of takeoffs means that certain scenarios were done twice during a specific session.

Table 2 – An overview of the groups and their respective sessions.

Group	Session 1	Session 2
1	Normal	Failure
2	Failure	Normal

Lastly, it is essential to notice that an Airbus A320 requires a pilot and a pilot monitor to operate. The experimenter would act as the pilot monitor during the experiment. It is a scripted role in which the experimenter performed specific actions or made certain calls to help the participant take off the airplane. For example, the pilot monitor would announce when the airplane reached 100 knots or raise the landing gear when the participant would say "gear up". These calls and actions follow the standard A320 takeoff procedure.

4.1 Participants

Participants were recruited from Bombardier and CAE. Participants were required to work from Bombardier or CAE and work in a field closely related to the aviation industry. For example, participants with no piloting license were mainly engineers working on a specific aircraft at Bombardier and CAE, knowing most aircraft procedures even if they were not pilots. We did not require them to have a piloting license, nor to be a pilot of a specific aircraft.

In total, 13 participants completed the experiment for a total of 136 takeoffs. The average age was 36 years old (+- 8.8 years). All participants were males. Out of the 13 participants, 7 were pilots. Out of those 7, 5 are or were A320 pilots. Pilots had an average of 604 hours of flight hours and 8.5 years of piloting experience. While 6 participants did not have a pilot's license, they still had an average of 98 hours of training in simulation. While these hours cannot be compared to actual flight hours, they indicate the aviation knowledge these non-pilots had.

While the number of participants is lower than for a typical experiment, it is essential to notice that each participant did a minimum of 10 individual takeoffs. In comparison, other experiments would have more participants but make them do fewer exercises during the experiment. In concertation with the different partners of the project, we considered the number of participants together with their number of takeoffs sufficient to construct a considerable dataset with enough variation and robustness within it.

4.2 Setup

This section gives more information about the setup that was created for the experiment. From the different measurement devices to the actual environment composition, this section explains why specific measurement tools were chosen compared to others and why the environment was set up this way. The experiment was set up at UDEM lab: Room 3332 (3rd floor) - Pavillon André-Aisenstadt.

4.2.1 Measurement devices

The CW of a pilot is measured using different hardware tools. This section is an overview of these tools. Before the experiment, an in-depth cost-value analysis was made with the different partners to determine which tools would be used. A focus was made on the precision of the measurements and the compatibility with other programs running. Ideally, pilots should not feel obstructed while achieving specific tasks. However, the precision of the evaluation of CW was the most important criteria. While many solutions exist for measuring the CW of a learner, this research focuses on tools for measuring PD and HR.

4.2.1.1 Polar H10

Today, different technologies exist to measure the HR of a person. From actual medical ECG devices to smartwatches, different choices were considered for the experiment. Ultimately, the Polar H10[7] strap was chosen as a measurement device as it offers the best precision compared to other devices (Neurosky, n.d.). The Polar H10 is a heart rate strap that attaches to the person's chest. Polar offers the possibility to connect via BLE to capture the data measured by the strap without using their SDK

The Polar H10 measures and calculates the BPM directly on the device. The device provides the instantaneous heart rate of a person extracted using the GATT protocol. For the experiment, we only used the BPM data of the Polar H10 as research show a relation with CW. Nevertheless, it is possible to add more precise data, like the ECG graph, the RR interval, the heart rate variability, and more in addition to the BPM should it be necessary.

4.2.1.2 Gazepoint GP3

Measuring the PD of a person turns out to be a complicated task. To measure PD, the measurement tool must always track the eyes of the user and calculate the variance in pupil diameter in real-time. Fortunately, eye tracking devices have become more common and precise over the years. Different eye tracking options were studied and compared. Finally, this research chose the Gazepoint GP3[8] because it offers pupil dilation measurement and the real-time eye gaze of a person. This device is also popular among the research community and has ready-to-use tools to calibrate and measure the pupils instantly. It is vital to notice that this tool can only work in a similar computer screen simulator environment. Should these experiments ever occur

[7] https://www.polar.com/en/sensors/h10-heart-rate-sensor
[8] https://www.gazept.com/product/gazepoint-gp3-eye-tracker/

in a real cockpit, a different eye-tracking device will have to be used. This is because, in a real cockpit, pilots move their heads all over the place to execute a specific task. The Gazepoint GP3 vision window in which it can track the eyes of a pilot is too small compared to the size of a cockpit.

4.2.1.3 NCO EEG headset

Last, this experiment uses the EEG headset from BMU[9] to measure the CW of the participant. This EEG headset offers the facility to capture real-time brain activity and emotions without calculating the combined brain waves at a particular time t as explained in Chapter 2 (Chaouachi et al., 2015). The headset itself comes from the company OpenBCI[10] where we used the NCO software on top of it to extract the real-time CW.

4.2.2 Experiment environment

Figure 17 shows the complete environment for the experiment. This section will break down the setup to better understand how the experiments were done. Figure 17 shows the entire configuration during an experiment. The person on the left is the participant, while the others are the experimenters. The person in the middle plays the role of the pilot monitor or copilot. The participant has the EEG headset, the HR monitor strapped to his body, and the eye tracker pointed at him. The participant is constantly looking at the screen in front of him. To control the A320 in the X-Plane simulator, the participant has a joystick, throttles, and pedals like those inside an Airbus A320. At the same time, the experimenters use the three other screens to monitor the different measurement tools, video recording, and simulator data that are incoming in real-time. The pilot monitor is constantly juggling between the takeoff procedure and the data monitoring.

[9] https://bmu.co/
[10] https://openbci.com/

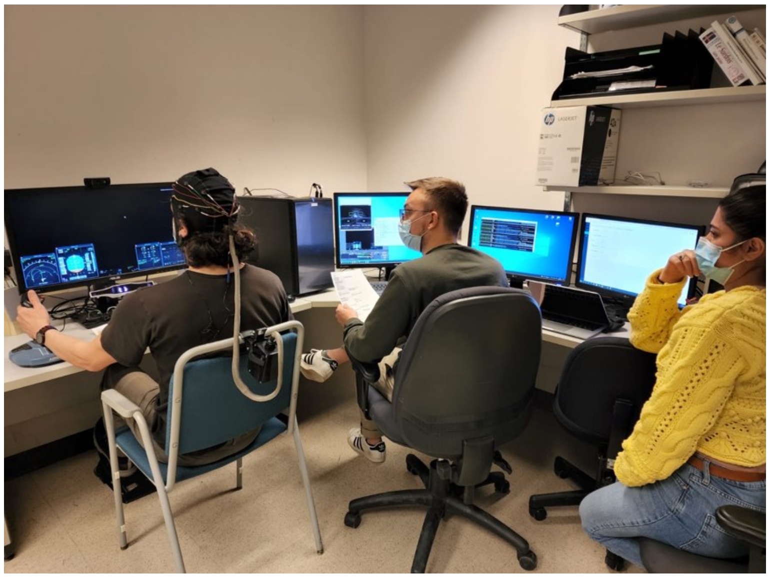


Figure 17. – The experiment with the participant (left) and the pilot monitor (middle)

Figure 18 shows the pilot's view of the experiment. The different measurement tools, together with the joystick, throttles, and pedals (under the table), are visible in the picture. It is essential to notice that the participant did not see the experimenters during the experiment. This positioning was done to ensure that distractions were kept to a minimum. The computer's volume was loud enough so the participant could immerse himself and focus on the different takeoffs he had to do.

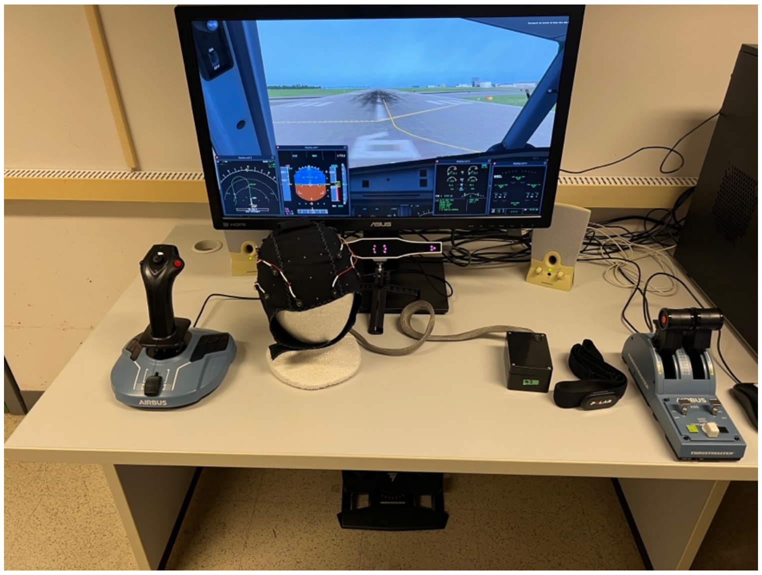


Figure 18. – The pilot’s environment with the measurement tools

4.3 Procedure

This section gives a detailed overview of the procedure respected throughout the experiments. This procedure was mentioned in the accepted ethics certificate. In the week preceding the experiment, participants received a detailed document of the A320 takeoff procedure to familiarize themselves with the simulator's handling characteristics. Without going into the details, this document includes:

- All the essential information of the primary flight display, the navigation display, engine/warning display, system display.
- A simplified takeoff procedure. It was chosen to simplify the procedure to allow participants who are not familiar with the A320 cockpit to understand and participate in the experiments.
- A simplified rejected takeoff procedure.
- A simplified engine failure after V1 procedure.
- The dialogs between the pilot and the copilot during each takeoff procedure.

Notice that flight experts at CAE and Bombardier have reviewed this document. The document follows the Airbus A320 takeoff procedure, but some less important parts have been removed for simplicity's sake.

4.3.1 Greetings

When arriving, the participants were greeted and introduced to the environment. The experiment's objective, together with the measurement tools that were going to be used, was introduced. Particular attention was paid to make sure all questions regarding the experiment, the project, and data privacy were answered. It was mentioned on multiple occasions during the experiment that the captured data would be anonymized as soon as it ended. Lastly, each measurement tool was introduced together with an explanation of why we needed to use each tool. Lastly, to reassure participants, it was made clear that the purpose of the experiment was not to test the participant's piloting skills but rather to gather data in different piloting situations, regardless of their skills.

After an initial tour of the environment, an explanation of the experiment's objective, and the tools that were going to be used, participants received a consent form that explained the experiment objectives and details of the data handling. This consent form ensures that both the participant and the experimenter have proof and consent to the experiment. After that, the participants were asked to answer a questionnaire mainly asking about their piloting skills. The answers to this questionnaire are analyzed in Chapter 5.

4.3.2 Familiarization flight

Before the data capturing, participants had time to familiarize themselves with the simulator’s handling characteristics and ensure the procedures between the participant and pilot monitor (experimenter) were clear and understood. For 30 minutes, the participant had time to learn how to do a normal takeoff, stop the plane during a rejected takeoff, and follow the procedure during a takeoff with engine failure after V1. The dialogs for each procedure were printed so the participant could look at them during each takeoff procedure. The experimenter would give feedback to participants who had little knowledge about the A320 to sharpen their skills. Each flight began at the runway 6L of the Montréal airport, with a standard atmosphere and perfect weather conditions.

4.3.3 Experiment flight

The Polar H10 heart rate monitor, GP3 eye tracker, and NCO EEG headset were calibrated before the data-gathering flights. The screen recording was started, and every hardware module was checked to see if data was incoming. When all tools were calibrated and sent data as expected, the experiment would start. As explained at the beginning of this chapter, participants would be assigned to one group and do two 20-minute takeoff sessions, one with failures and one without failures. During the session with failures, the participant knew there could be failures during each takeoff but did not know which type of failure or when a failure could happen. Moreover, the participant did not know the weather conditions in advance of each scenario. Between takeoffs, the participant would stay on his chair, give feedback on the takeoff, and wait for a new takeoff to start. When done, the participant would remove all the measurement tools and be thanked for his participation.

4.4 Conclusion

In this chapter, we went over every step of the experimental procedure in which pilots had to perform a takeoff with or without failures. This chapter included several details about the participants, the different scenarios introduced, the group division logic, the step-by-step procedure, and more. Moreover, this chapter argues why we chose the Polar H10, the Gazepoint GP3, and the NCO EEG headset for this experiment. The setup, together with pictures of the experiment, were presented as well. Thanks to this setup, the data of 13 pilots were gathered for a total of 2 million rows combined or 9 hours of continuous data. The next chapter combines and analyzes the gathered data and tries to answer the first and second hypotheses of this research.

Chapter 5 – Data

Previous chapters gave more information about how certain data types were extracted, interpreted, and saved into logs using a specific architecture following a strict procedure. This chapter focuses on how all these data logs are merged and the different data points that are present. The first goal of this chapter is to defend choices that were made regarding the data, together with the limitations and expandability of the extracted data. The second goal of this chapter is to give an in-depth overview of the data through different statistical analyses. It is crucial to notice that this chapter only talks about the raw data. Chapter 6 uses the data described in this chapter to create different machine learning models.

5.1 Time synchronization

In previous chapters, it was mentioned that multiple Connector modules ran on different computers during the experiment. One computer had the dashboard and the EEG module running, while the other had all the other modules together with the flight simulator running to avoid CPU overload. This architecture creates a time synchronization problem. Since this research deals with time-series data, it is crucial to ensure that the time registered on both computers is the same. The problem is that Windows does not always synchronize the time using the same time servers. This synchronization issue creates a problem where time would be slightly off between computers, saving data at the wrong time.

Since we could never be sure of the time difference between the two computers, a solution was created. To better illustrate this solution, let us consider this example. Suppose computer one has an internal clock time of 1010 seconds since the epoch time while computer two has an internal clock time of 1000 seconds since the epoch time. When starting a new experiment, the Event Generator server sends a request to both computers to save their current internal time into a CSV file. Both files are later combined, and the difference in time is taken between both computers:

$$1010\ s - 1000\ s = 10\ s$$

we considered computer one, the one with the flight simulator running, being the computer with the accurate time. The time difference is applied to all the saved data logs on computer two, resulting in synchronized data logs across computers. It is crucial to notice that while this example seems rather simplistic, the actual time difference between computers is calculated in micro-seconds to ensure the best synchronization possible.

5.2 Data synchronization

After completing an experiment, approximately 100 different files are created and stored on different computers. In order to combine the correct files, a synchronization logic was necessary. Three essential components within the file name are necessary to ensure synchronization:

- **UUID:** A unique identifier that is given to a specific pilot.
- **Date**: The specific date when the experiment occurred.
- **Scenario number**: Which scenario was used during a takeoff.
- **Flight number**: The current takeoff number.

While the date is to make the search easier afterward, the generated UUID, the scenario number, and the flight number are the critical components in the file name that make the following logic work. When the Connector module is initiated, a unique identifier is created simultaneously. This UUID is used for every file generated throughout the lifetime of the Connector module, which is the time that the experiment will last. More than combining files from the same experiment, it was essential to combine files from the same takeoff. A single takeoff generates ten different CSV files that later need to be merged. Together with the flight number, the scenario number is being used to ensure a unique identifier for all files coming from one experiment. For example, here are the file names of the heart rate and pupil dilation files generated from one takeoff during one experiment:

- **heart rate file**: *hr_2022-05-10_b2fd5c85-7aea-4d1f-b487-87b66355d1e8_scenario-4_6.csv*
- **pupil dilation file**: *pd_2022-05-10_b2fd5c85-7aea-4d1f-b487-87b66355d1e8_scenario-4_6.csv*

Thanks to this method, a script is run post experimentation to merge all necessary files. This Python script will then merge every file with the corresponding UUID, scenario number, and flight number.

While this method indeed merges files done during the same experimentation, this method cannot combine data points done simultaneously under one row. The following section explains how data points are merged and the choices that were made regarding the merge of time series data.

5.3 Data features

This section provides an overview of the different data points that were gathered during the experiments. Not all the data points will be referenced here, as measurement tools gathered some data points without this research needing it. Table 3 gives an overview of the gathered data, its abbreviation, and a short description of the data.

Table 3 - An overview of the captured data with its corresponding description.

#	Attribute	Description
1	TIME	The time at which the data was recorded in milliseconds since the epoch.
2	HR	The instantaneous heart rate at a certain time t in BPM
3	LPD	Left pupil diameter in pixels
4	RPD	Right pupil diameter in pixels
5	BPOGX, BPOGY	The X- and Y-coordinates of the best eye point of gaze, as a fraction of the screen size.
6	REYEX, REYEY, REYEZ	The X-, Y- and Z-coordinates of the right eye with 3D space with respect to the camera focal point, in units of meters.
7	LEYEX, LEYEY, LEYEZ	The X-, Y- and Z-coordinates of the left eye with 3D space with respect to the camera focal point, in units of meters.
8	WORKLOAD	The percentage of workload
9	ENGAGEMENT	The percentage of engagement
10	STATUS	Video recording status events (start, stop)
11	EVENT_TYPE	Events during the experiment (start_scenario, stop_scenario, speed, engine_failure, …)
12	PARAMETERS	Optional parameters related to EVENT_TYPE parameters
13	FLIGHT	The current flight number
14	PILOT	The pilot number which uniquely identifies a pilot
15	SCENARIO	The scenario that is currently being played

5.4 Data merge

Timestamps are an essential part of this research since this research deals with time-series data. All timestamps in the different files reference the number of seconds since the time epoch. This choice was made to facilitate the merging of data points together. When dealing with time-series data, a problem arises: *How to merge non-matching timestamps?* Since the registered timestamps are in milliseconds, many data points are non-matching. Different merging options were analyzed and tested to provide the best results possible. One option was to fill in empty data points with the closest timestamp value of that data point. For example, if no data were provided for column A at the row with timestamp 1, this logic would have looked at timestamps 0 and 2 and taken the value from column A. This logic was used throughout the research as a viable solution to solve the timestamps merging issues but was removed as it did not provide coherent results. Nevertheless, we believe it could be a potential solution for future research using this data if done differently. It is crucial to note that while different merged data structures are compared and analyzed, all the individual raw files are available for use should they be necessary. This solution ensures that future research could still be done on the generated dataset without following this research logic. Nevertheless, the merged dataset structure constitutes an essential part of this research as it lays the foundation for the machine learning models discussed in Chapter 6. The following section provides the merge logic this research used to gather results in the coming chapters.

5.4.1 Solution: Average lowest frequency

The retained solution relies on the frequency at which each data type is being saved. The data type with the lowest frequency is used as a reference for merging to avoid rows where data types are missing. The HR is the data type with the lowest frequency, only saving one HR every second, or 1 Hz. This research abbreviated all timestamps to a specific second instead of a millisecond to better merge the data.

Table 4 - A representation of the merged table.

Timestamp	Sensor A	Sensor B
1	50	10
2	NaN	12
3	NaN	16
4	NaN	12
5	NaN	10
6	55	14

Let us take an example scenario to understand this decision better. Imagine a sensor providing data only once every 5 seconds while another would provide some data once every second. Table 4 shows a representation of the table after 6 seconds. In order to avoid having missing data for some rows, taking the sensor providing the least amount of data over a specific time frame makes sense. One might wonder how other, faster data types will be squeezed to match the slower provider. To answer this, here is an overview of the different data types that will be provided. The following example will assume that, compared to the slower sensor, *n* consecutive more data points were generated while the slow sensor only generated one data point:

- **Average**: *(p1 + p2 + ... + pn) / n*. The average of the n-consecutive points.
- **Median**: The median of all the n data values. This approach is more statistically robust than using the average because of potential outliers in the data.
- **Lowest value**: The lowest value out of the n values. While it might not tell as much as the median or the average, it can be helpful when training a machine learning model.
- **Highest value**: The same logic applies to the highest value out of the n values.

Table 5 - A representation of the merged table using the lowest frequency.

Timestamp	Sensor A	Average_B	Median_B	Min_B	Max_B
1	50	12	12	10	16
6	55	...	...	...	...

Table 5 gives a representation of the table using this logic. It is crucial to notice that the timestamp difference between both rows seems rather drastic in this example. In reality, since the HR saves a BPM every second, the time difference between two rows is precisely one second. This method augments the number of variables within the dataset. Using the described logic, the dataset goes from 15 variables to approximately 50 variables.

5.5 Data interpretation

5.5.1 Hypothesis 1

The first hypothesis that this research tries to answer is: "*Is it possible to measure the CW of pilots during a takeoff flight*?". Let us consider two different takeoff flights from the experiment to answer this hypothesis. Figure 19 shows the data output of pilot six during a rejected takeoff. Pilot six is an official A320 pilot with eight years of experience and more than 250 official flight hours. Figure 19 includes:

- The HR, PD, and CW of the pilot at a time t
- Important events which resulted in a CW increase.

The three plots are aligned vertically based on time. One can consider the start of this scenario when the parking brake is released (first vertical line) and the end when the pilot confirms the engine must be turned off (last vertical line). It is important to note that CW, PD, and HR changes are more important than their actual value. This change can be better analyzed because each participant has a different HR, PD, and CW based on his antecedent.

As explained in Chapter 2, PD relates to someone's attention. The figure shows that pilot six stayed focused during the entire takeoff, as seen on the PD graph (blue). Once the experiment stopped, his attention started decreasing. The workload graph shows that certain events increased the pilot's CW. The engine failure (red vertical line) increased the pilot's workload. Again, the pilot did not know about the failure in advance. The spontaneity of the failure increased the pilot's workload, making him think about the subsequent actions to take in this scenario. It can be noted that each CW increase (green plot) was the result of a specific action the pilot needed to perform, confirming the first hypothesis.

Let us look at a different output of a non-licensed pilot in order to confirm our first analysis. Figure 20 shows the output of a non-licensed pilot for the same scenario. Pilot five is an A320 engineer who knows the ins and outs of the A320 and has flown on the simulator with it for testing purposes but is not a licensed pilot. The workload chart (green) of Figure 20 also shows that the CW of the participant increased when the engine failure was triggered. Like pilot six, the CW of pilot five increased when he made specific actions. When comparing both participants, two significant differences are observed. Pilot five was less attentive than pilot six during the takeoff. His attention increased when the engine failure happened and after the experiment terminated. A second noticeable observation is that different actions triggered different changes in CW for both pilots. Both observations could be because of the difference in piloting experience and knowledge between participants. Lastly, it can also be observed that the engine failure has a future effect on the HR for both pilots, indicating a relationship between the pilot's CW, his attention (PD), and his heart rate (HR).

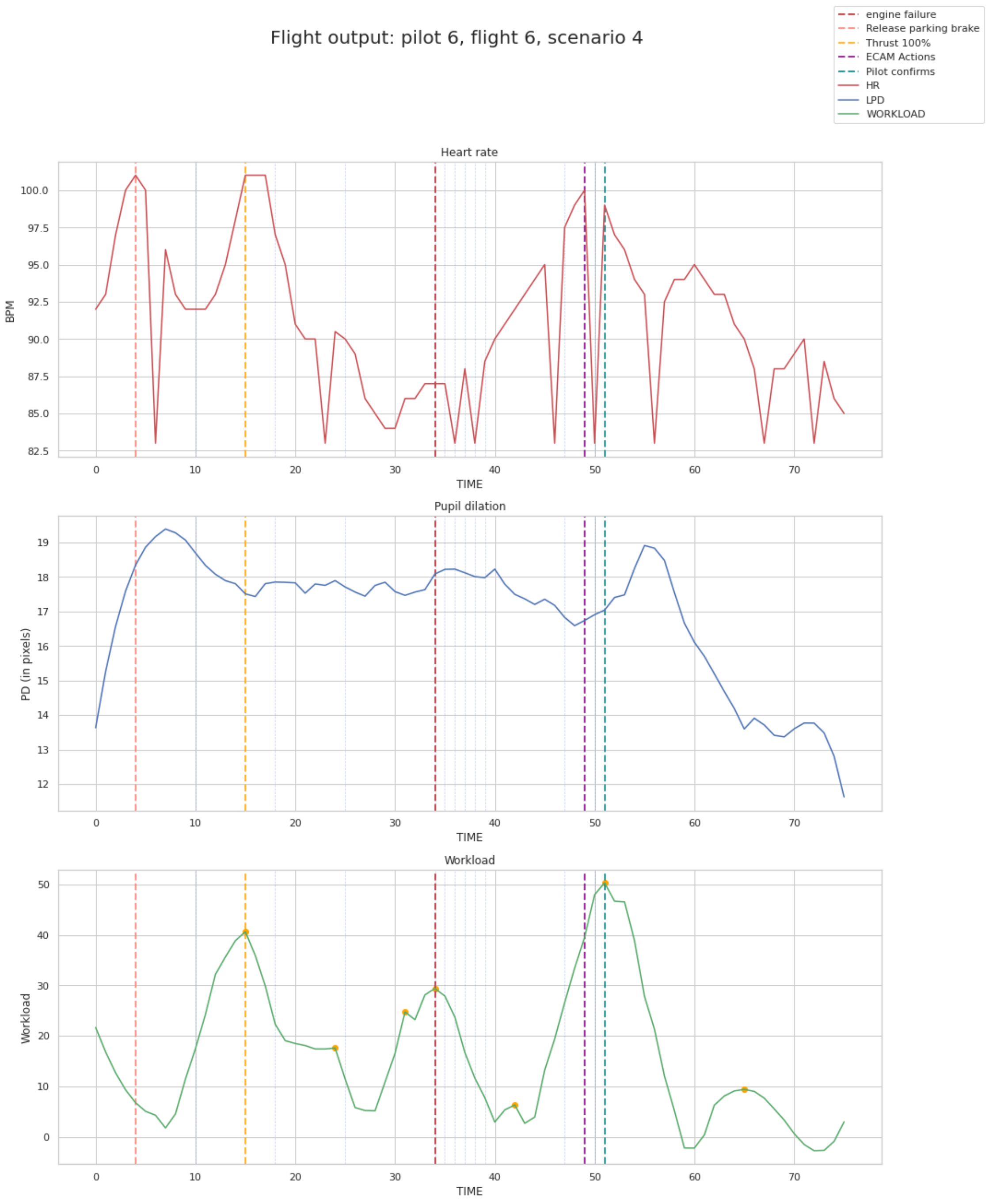


Figure 19. – A complete flight output representation of an A320 pilot.

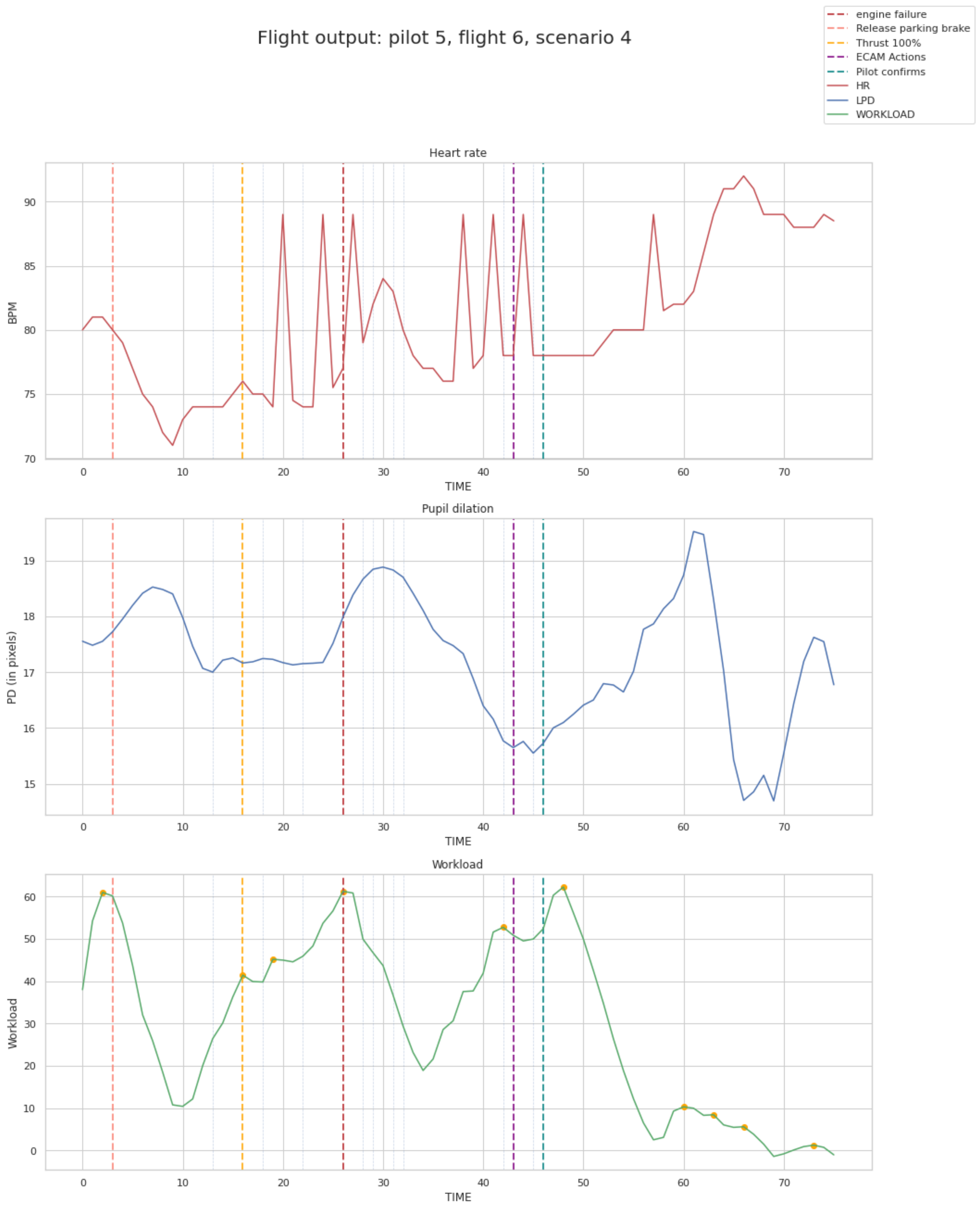


Figure 20. – A complete flight output representation of a non-licensed pilot

With this, let us establish a correlation between the measured CW and the measured PD and HR. To achieve this, this research will base its hypothesis on a specific moment when CW increases. After analysis, this research found that CW increases around the moment of an engine failure. This increase can be seen in Figures 13 and 14 at the red vertical line. To find a correlation, this research will use F-tests and t-tests.

First, this research tries to confirm the claim that CW increases when an engine failure happens. The null hypothesis would be that the CW does not increase during an engine failure. One problem arises when trying to reject the null hypothesis. The problem is that although the log indicates the time the failure happened, it does not mean that this is when the pilot saw it or when the pilot monitor said the failure was happening. To give more context: When doing the experiments, the pilot monitor was in charge of saying to the pilot when an engine failure happened. Because the pilot monitor knew when the failures would happen in a scenario, he tried to say it at the same time the trigger happened. The reality was that the pilot monitor would say the engine failed around the time the crash happened, but not at the exact logged time. Because of this, a different approach should be taken. When looking at a time window of 4 seconds (2 seconds before, 2 seconds after the engine failure), a significant difference in CW between the max and the min value is visible as shown on Figure 21. When doing a one-way F-test on the max versus the min value, this research obtains an F-value of 5.424, resulting in a p-value = 0.029, which rejects the null hypothesis. This confirms that within a 4-second time window, there is a significant change in CW when an engine failure happens, which rejects the null hypothesis. If the null hypothesis were true, the change in CW would not be significant enough to pass the F-test. These results answer and validate the first hypothesis of this research. The following calculations give more details about the obtained results. Notice that all measurements were done with a 95% confidence interval (CI):

The F-value is equal to the mean square for treatment (MST) over the mean square error (MSE):

$$F = \frac{MST}{MSE}$$

The MST and MSE can be calculated as follow:

$$MST = \frac{n_1(\bar{x}_1 - \bar{x})^2 + n_2(\bar{x}_2 - \bar{x})^2 + \cdots + n_K(\bar{x}_K - \bar{x})^2}{K-1}$$

$$MSE = \frac{(n_1-1)\,s_1^2 + (n_2-1)\,s_2^2 + \cdots + (n_K-1)\,s_K^2}{n - K}$$

With K representing the number of populations to compare (two in our case) and:

$$n = n_1 + n_2 + \cdots + n_K$$

$$\bar{x} = \frac{\Sigma x}{n} = \frac{n_1\bar{x}_1 + n_2\bar{x}_2 + \cdots + n_K\bar{x}_K}{n}$$

With all this, we can now compare the two hypotheses with a 95% CI or α = 0.05, which are:

$$H_0 : \mu_1 = \mu_2 = \cdots = \mu_K$$
$$\text{vs. } H_a : \text{ not all } K \text{ population means are equal}$$

Moreover, our df1 (degrees of freedom) = 2-1 = 1 and df2 = 13-1 = 12

When using the above formulas and comparing the min and max population using the 13 pilots, we obtain a F = 5.424. When looking inside the F-table, we see that for df1=1 and df2 = 12 with a 95% confidence interval, we find a value of 4.7472. This means that any values above 4.7472 rejects the null hypothesis, which is our case. Transforming these results to p-values, F = 5.424 results in a p-value = 0.029. Since our p-value is lower than our α, we reject the null hypothesis. The same logic is applied for other one-way F-tests throughout this chapter.

We chose a time window of 4 seconds when looking back at the experiment recordings. On average, the pilot monitor told the pilot an engine failure was happening within a 4-second time window of the logged engine failure in the dataset.

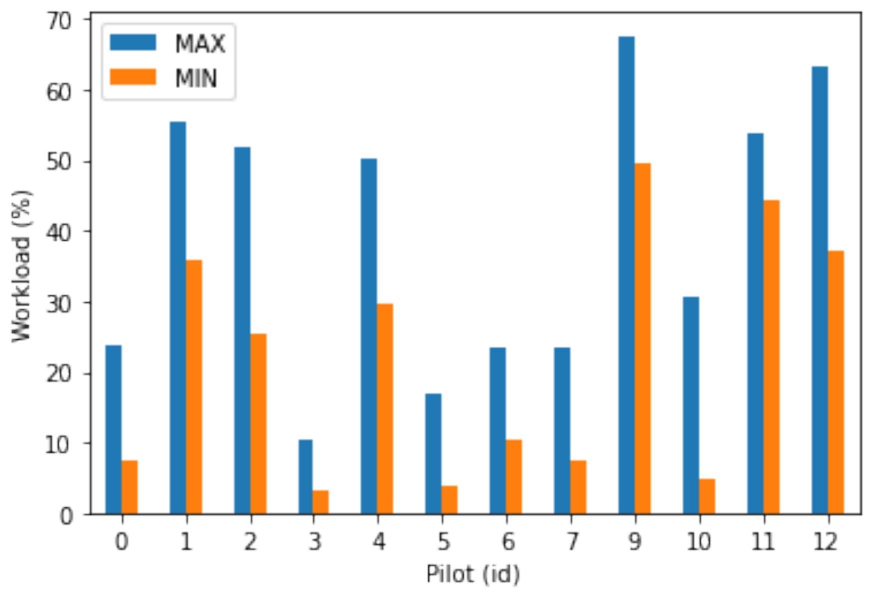


Figure 21. – Min-max value of CW within a 4-second timeframe

5.5.2 Hypothesis 2

The second hypothesis this research tries to prove is that a person's HR increases after a spike in CW to demonstrate the correlation between HR and CW. The null hypothesis would be that the HR stays the same before and after the spike. Figure 22 shows the average HR before and after the event over a 20-second timeframe (10 seconds before and after an engine failure). When doing a one-way F-test, this research found an F-value of 10.75, resulting in a p-value = 0.003, which rejects the null hypothesis. With this, this research shows a difference in HR before and after a CW spike caused by an engine failure.

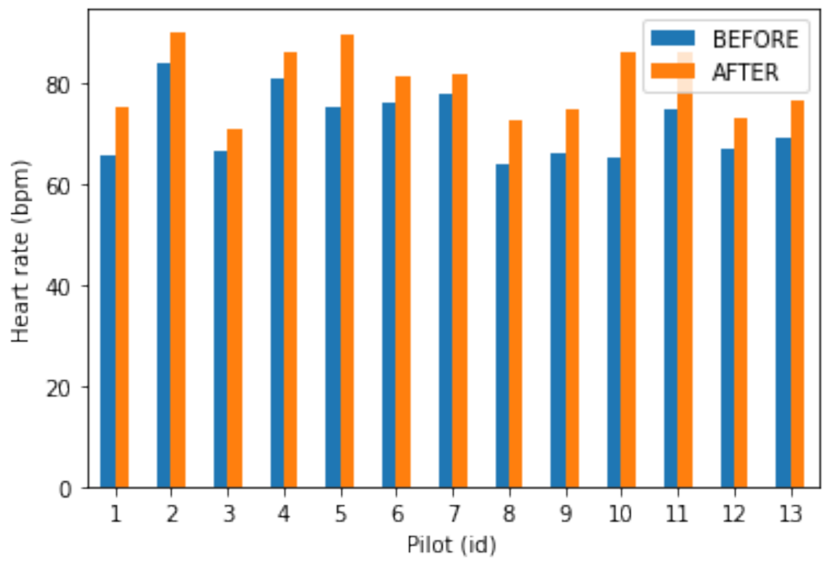


Figure 22. – Average HR before and after engine failure

Part of the second hypothesis too, we tried to prove that a person's PD increases after a spike in CW to demonstrate the correlation between PD and CW. The null hypothesis is that the PD value would not change before or after the spike. Figure 23 shows the average PD before and after the event over a 6-second timeframe (3 seconds before and after engine failure). When doing a one-way F-test, this research found an F-value of 6.36, resulting in a p-value = 0.0187, which rejects the null hypothesis. This result validates the hypothesis that there is a correlation between CW and PD. This result shows that a PD increase increases the pilot's attention, which in turn will increase the CW of the pilot.

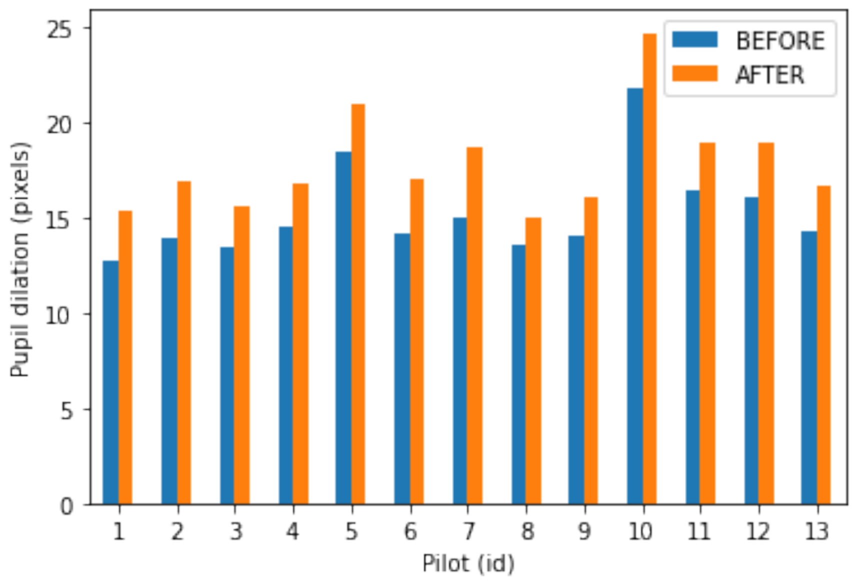


Figure 23. – Average PD before and after an engine failure

5.6 Data analysis

To better understand the data, let us analyze its output. More than 2 million rows were saved during the experiment. Since a new row was saved 60 times a second, or 60Hz, more than 9 hours of data were saved. This number corresponds to an average of 41 minutes of data per pilot. This average makes sense as each experiment consisted of two sessions of 20 minutes each.

5.6.1 Noise

When analyzing the data, this research saw noise within it. This noise makes sense as we used hardware equipment to extract and measure different physical body parts of the participant. The noise within the data can be due to the sensors of the hardware equipment failing during specific measurements, the participant moving during the experiment causing interference when measuring, or the transfer method used to transfer the data to the computer (Bluetooth or WIFI). All these events were unavoidable and mitigated as much as possible during the experiment. To solve this problem, the data has been smoothed using different smoothing strategies. Each strategy was used and compared with the goal of obtaining the best model's predictions. One of these smoothing filters, the Savitzky-Golay filter, smooths the data using convolutions to increase its precision (Gallagher, 2020). The main benefit of this filter is that it increases the precision of the data without distorting the signal. Another filter used during this research is the Butterworth filter (Mishra et al., 2020). The inner working of this filter falls outside the scope of this research, but it can be noted that this filter is used in the machine learning community to reduce noise in data. Lastly, the locally estimated scatterplot smoothing, or LOESS, was used to filter noise in the data (Pintus et al., 2014). Further comparison between smoothed data versus non-smoothed data is done in Chapter 6 when doing data pre-processing. The next chapter will compare the smoothed and raw data as input for the machine learning models. When smoothing the data, the margin of error was negligible, making it a viable solution.

5.6.2 Groups

As mentioned in the previous chapter, participants were divided into two groups. Group 1 consisted of 7 participants, with 4 pilots. Group 2 consisted of 6 participants with 3 pilots. Group 1 totaled 72 takeoffs, while Group 2 totaled 64 takeoffs. At the end of the experiment, participants were asked how difficult the experiment was on a scale from 1 to 10, with 10 being the most difficult takeoffs they ever did. The average difficulty was 3.9/10 (+- 1.79). Participants with a piloting license had a lower average difficulty, 2.78/10 (+-0.69), than participants without a piloting license, 5.25/10 (+- 1.78). This result makes sense as participants with a piloting license have more simulator experience than participants without one.

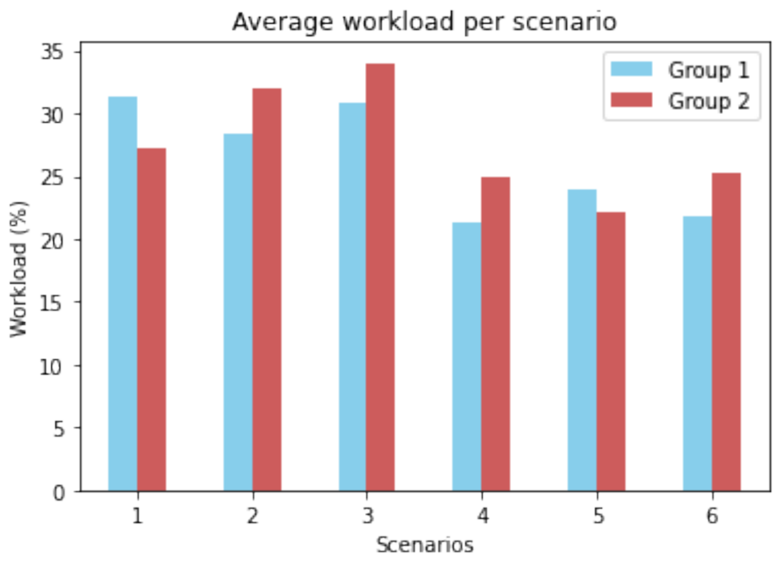


Figure 24. – Average workload per scenario per group

Moreover, this research compared the average workload per scenario for each group, as shown in Figure 24. The workload for each scenario between each group stays approximately the same. On the other hand, scenarios without engine failures generated more CW than scenarios with engine failures. This phenomenon could be explained because, on average, the duration of a normal takeoff was longer than a takeoff with engine failure during the experiments. Only looking at the average CW does not tell the change in intensity of CW during an engine failure versus during a normal takeoff. For example, let us consider two examples where the CW is calculated every second, one normal takeoff and one with engine failure. The normal takeoff lasted 40 seconds; the CW increased to 65% 10 seconds out of the 40, and the rest was 50% resulting in an average CW of 53.75%. Let us now consider a scenario with engine failure that lasted 20 seconds; the CW increased to 80% 2 seconds out of the 20, and the rest was 50% resulting in an average CW of 53%. As we can see, while the engine failure did increase the CW by 15% more, the average CW of the engine failure scenario was still lower than a normal takeoff.

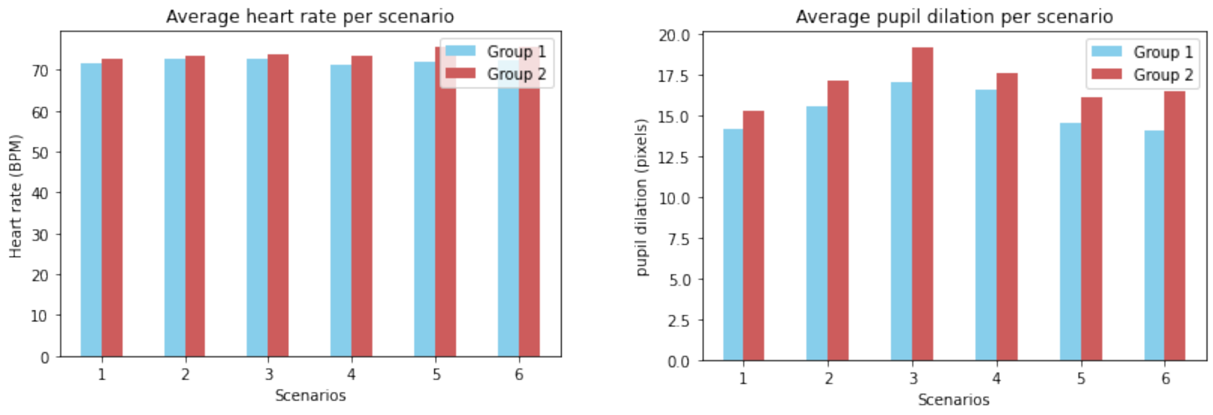


Figure 25. – Average HR and PD per group per scenario

Furthermore, the average HR and PD per scenario were analyzed as shown in Figure 25. While no significant differences were visible between each group or between each scenario, it can be noted that the PD for scenario 3 was, on average, higher than for other scenarios. This difference makes sense as scenario 3 made the participants do a night takeoff. The black screen illuminated less light causing the pupils to dilate more to allow more light in (Szabadi, 2018).

This section demonstrated that regardless of the pilot's group during the experiment, no difference in the resulting data could be seen, canceling any learning bias that could have occurred based on the scenario order. Similar results for HR, PD, and CW were found regardless of the group.

5.7 Conclusion

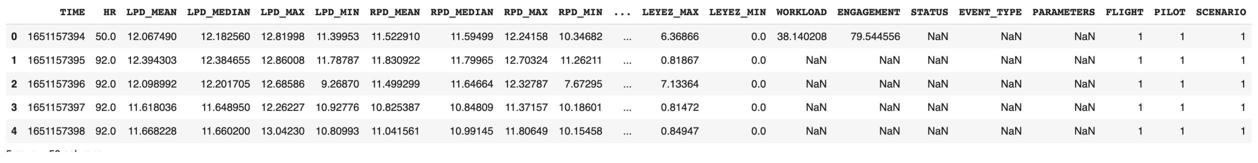
In this chapter, we introduced the different techniques that were used in order to synchronize and combine the different files that were gathered during the experiment. We found a way to synchronize data gathered on different computers and how to merge timestamps that were within the same second. When merging the data, the average, minimum, maximum, and median value within that second was added as value. This way, each row represents a single second and is easier to interpret. Moreover, this chapter introduced the different data features that were extracted during the experiment.

More importantly, this chapter answered the first hypothesis by analyzing the data gathered during the experiment for a specific pilot during a specific scenario. We found that each action the pilot took during a scenario led to an increase in CW, confirming the first hypothesis. Moreover, we found that during a workload-intensive event, the CW of a pilot increased together with the HR and PD of the pilot. These findings confirm the second hypothesis of this research.

Finally, this chapter analyzed the differences between the groups. Overall, the group repartition, thus scenario repartition, had no influence on the gathered data. The next chapter will now use this data and try to predict future CW and thus prove the third and last hypothesis of this research.

Chapter 6 – Workload prediction models

The purpose of this chapter is to answer the third hypothesis. This can be done by analyzing and implementing different machine learning or deep learning models and techniques to produce coherent results. During this chapter, we will walk through the different preprocessing, feature engineering, feature selection, machine learning, and deep learning models implemented during this thesis. The dataset extracted from the previous chapter and shown in Figure 26 will be used to train and evaluate the models. Section 6.1 describes the preprocessing done on the dataset to remove unnecessary or missing data and how to deal with the noise present in the data. Next, Section 6.2 explains the feature selection process using different statistical methods. Finally, Section 6.3 provides the results of all the steps done in the previous sections.



	TIME	HR	LPD_MEAN	LPD_MEDIAN	LPD_MAX	LPD_MIN	RPD_MEAN	RPD_MEDIAN	RPD_MAX	RPD_MIN	...	LEYEZ_MAX	LEYEZ_MIN	WORKLOAD	ENGAGEMENT	STATUS	EVENT_TYPE	PARAMETERS	FLIGHT	PILOT	SCENARIO
0	1651157394	50.0	12.067490	12.182560	12.81998	11.39953	11.522910	11.59499	12.24158	10.34682	...	6.36866	0.0	38.140208	79.544556	NaN	NaN	NaN	1	1	1
1	1651157395	92.0	12.394303	12.384655	12.86008	11.78787	11.830922	11.79965	12.70324	11.26211	...	0.81867	0.0	NaN	NaN	NaN	NaN	NaN	1	1	1
2	1651157396	92.0	12.098992	12.201705	12.68586	9.26870	11.499299	11.64664	12.32787	7.67295	...	7.13364	0.0	NaN	NaN	NaN	NaN	NaN	1	1	1
3	1651157397	92.0	11.618036	11.648950	12.26227	10.92776	10.825387	10.84809	11.37157	10.18601	...	0.81472	0.0	NaN	NaN	NaN	NaN	NaN	1	1	1
4	1651157398	92.0	11.668228	11.660200	13.04230	10.80993	11.041561	10.99145	11.80649	10.15458	...	0.84947	0.0	NaN	NaN	NaN	NaN	NaN	1	1	1

Figure 26. – Structure overview of the dataset

6.1 Preprocessing

To train better models, it is vital to preprocess the data to transform a raw value into qualitative data that makes sense for a model. To achieve this goal, we used different preprocessing techniques.

First, it was essential to deal with NaN values in the dataset. Figure 27 shows an overview of all the rows in the dataset, where the black color means that data is present and the beige color where data is missing. Out of all the columns, three columns are almost empty: STATUS, EVENT_TYPE, and PARAMETERS have 99,49 %, 98,80%, and 98,80% NaN values respectively. These results are because events were logged at the specific time they happened during the experiments, resulting in only a few events in the database compared to other columns. Moreover, those three columns represent the same event with other parameters and information. For this reason, we chose to remove the STATUS and PARAMETERS columns. During the feature engineering part of the dataset, the EVENT_TYPE column will be one-hot encoded. Other rows with NaN values were removed, resulting in an 8% drop from the original dataset.

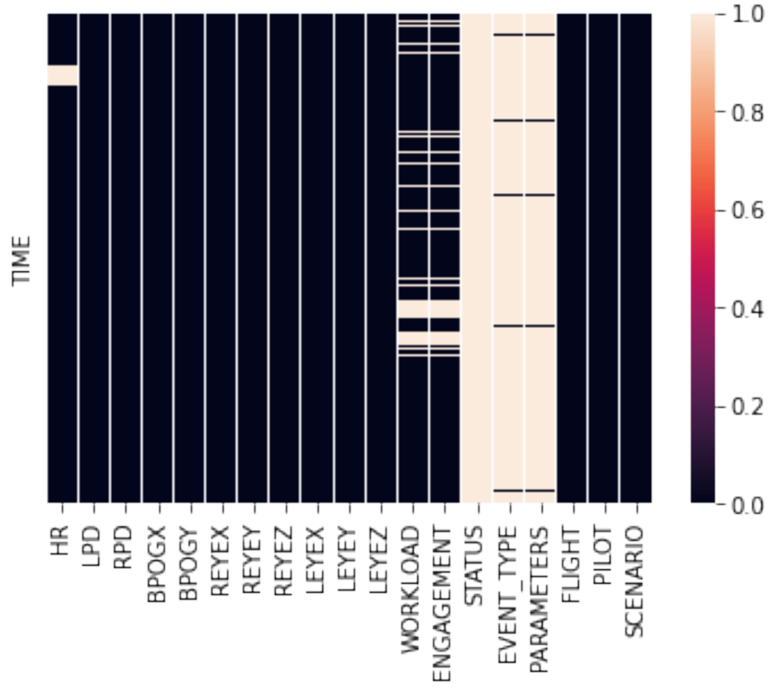


Figure 27. – Visual representation of the NaN values in the dataset.

Furthermore, another essential aspect of the preprocessing was how to deal with the noise in the data. Since the data was gathered using hardware sensors in real-time, the values contain some noise in some shape or form. To deal with this problem, it was chosen to smooth the data. The following was done to see the impact of smoothing on the data:

- Smoothing all the columns with noise.
- Only smoothing the target.
- Only smoothing the columns and not the target.
- No smoothing.
- Smoothing using the Savitzky-Golay filter, LOESS, and the Butterworth filter.

We found that smoothing all the columns using the Butterworth filter gave the best predictions for every model compared to other methods. For this reason, the LPD, RPD, HR, and WORKLOAD columns were smoothed.

Next, we scaled the data using different scaling methods. Scaling methods help models converge better because the values are scaled down. For example, HR values went from 50-120 BPM to a range of 0-1 with the same distribution using MinMaxScaler. MinMaxScaler and StandardScaler were compared the same way this research compared the smoothing methods. MinMaxScaler on all the columns appears to give better results than other methods for deep learning models, while StandardScaler gave better results for machine learning models. It is crucial to notice that while scaled values are used for training the model, they are then descaled at the end to represent the original values.

Lastly, all the values from the EVENT_TYPE column were one-hot encoded. NaN values were transformed to 0, and all the different events received their column. This created four new columns, EVENT_engine-failure, EVENT_speed, EVENT_start_scenario, and EVENT_stop_scenario, each describing a specific event. This way of encoding multiple events provides an easier way for machines to interpret the result of an event. Values in these columns can be interpreted as 1, meaning the event was triggered at timestamp t and 0 when nothing happened.

6.2 Feature selection

After preprocessing the data, it is necessary to distinguish and select critical features that generate the best predictions. A Pearson correlation matrix was used to compare features with one another. As seen previously, the dataset has over 50 different features. To avoid showing the correlation between unnecessary features, we focused on the correlation between the CW, the model's target, and other features. Figure 28 shows the correlation of the essential features. A comparison between raw and processed data is also performed in Figure 28. From it, it is possible to distinguish that preprocessed data have a better correlation with one another but do not have a high correlation.

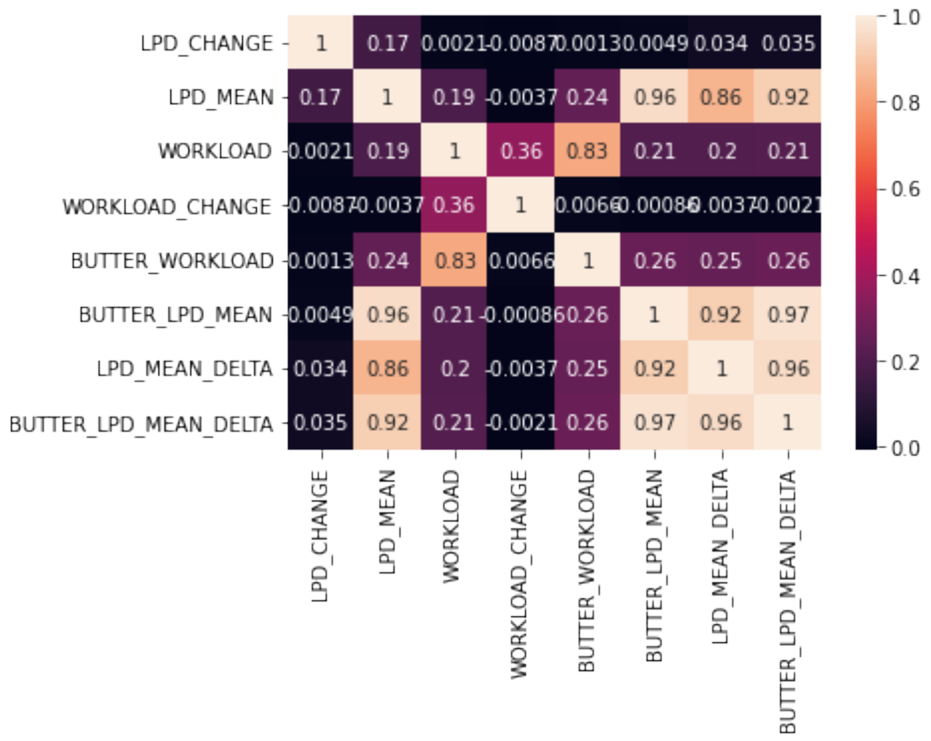


Figure 28. – The Pearson correlation matrix of the dataset.

After multiple trials and errors, this research found that taking the best ten features from the Pearson correlation matrix led to better performances for machine learning models and deep learning models. Figure 29 shows the ten best features that were used.

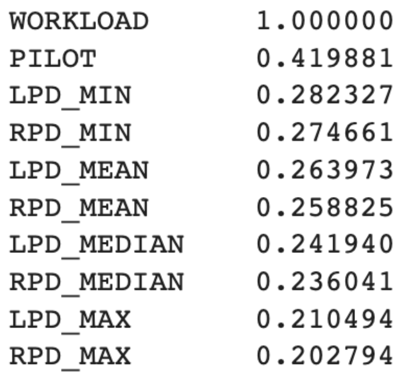

```
WORKLOAD       1.000000
PILOT          0.419881
LPD_MIN        0.282327
RPD_MIN        0.274661
LPD_MEAN       0.263973
RPD_MEAN       0.258825
LPD_MEDIAN     0.241940
RPD_MEDIAN     0.236041
LPD_MAX        0.210494
RPD_MAX        0.202794
```

Figure 29. – Ten best features for ML models.

6.3 Results

This section will demonstrate and compare the results from the previously mentioned techniques and machine learning and deep learning models used throughout this research. Moreover, these results will try to answer the third hypothesis. Furthermore, we will demonstrate other exciting results while trying to answer the hypothesis. The results of each model will be given together with a comparison between each model using the described performance metrics. Finally, the last subsection will give ideas and results outside the research objective but could help future research topics around the same subject.

6.3.1 Models results

Table 6 gives the average performance measures of all the models used to predict a pilot's CW on the test set. The values can be interpreted as the actual CW percentage. For example, the ridge regression model has an MAE of 19.67, meaning a mean absolute error of 19.67% of CW. Overall, a stacked-LSTM implementation produces the best results compared to other models. It is crucial to notice that the LSTM model used the previous workload to predict future CW. Other models did not use CW as input because of the models' inner working. Figures 29 and 30 show the predicted results for two different models. In Figure 30, the SVR model can detect a small trend but does not accurately predict the value of the CW. This phenomenon arises for all the models except for the LSTM approach. Section 6.4 explains why this happened and how future work could investigate this problem.

Figure 31 shows the prediction of a stacked-LSTM model using CW as an input for predicting the next CW, 5 seconds into the future. As shown in the picture, the model can predict the future CW 5 seconds into the future with good accuracy. This model has the following performance measurements: MSE of 44.09, an RMSE of 6.64, and an MAE of 5.28. It predicts better than previous models when previous CW is used as input. Different types of LSTM were trained and tested. A bi-directional LSTM also gave great performances compared to other models. Each model was trained for 150 epochs with early stopping regularization to avoid overfitting. We found the parameters of each model using GridSearchCV. Table 7 shows the stacked-LSTM model’s hyperparameters.

Table 6 - Models performances

Model	**MSE**	**RMSE**	**MAE**
Ridge Regression	474.70	21.79	19.67
SVR	626.62	25.03	22.31
MLP	537.11	23.18	23.18
CNN	497.73	22.31	18.52
BI-LSTM	99.6	9.98	7.81
Stacked-LSTM	**44.09**	**6.64**	**5.28**

Table 7 - Stacked-LSTM hyperparameters

Parameter	**Value**
Batch size	32
Epochs	150
Dropout	0.5
Loss function	MSE
Stochastic optimization method	ADAM optimizer

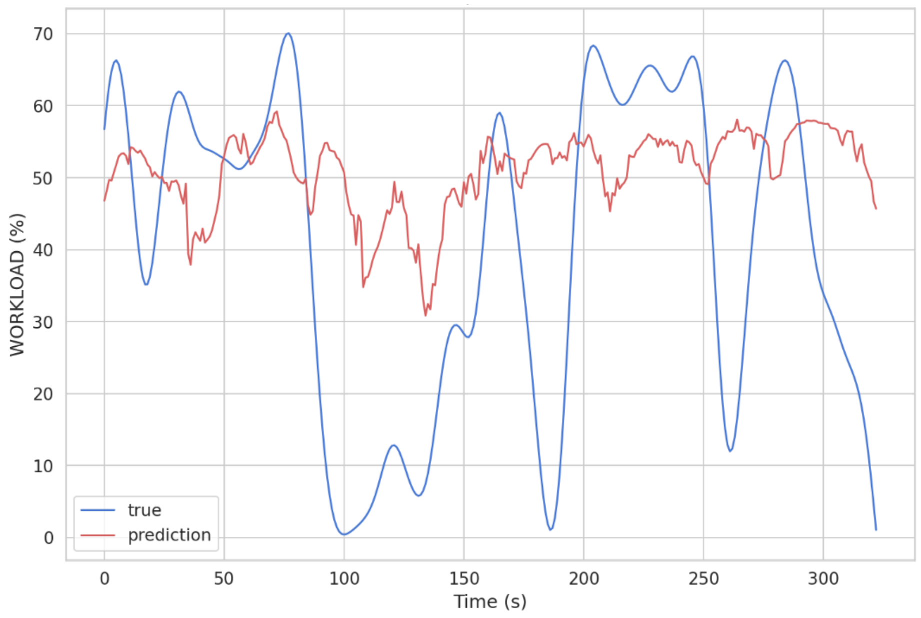


Figure 30. – SVR model predictions (red) vs true labels (blue)

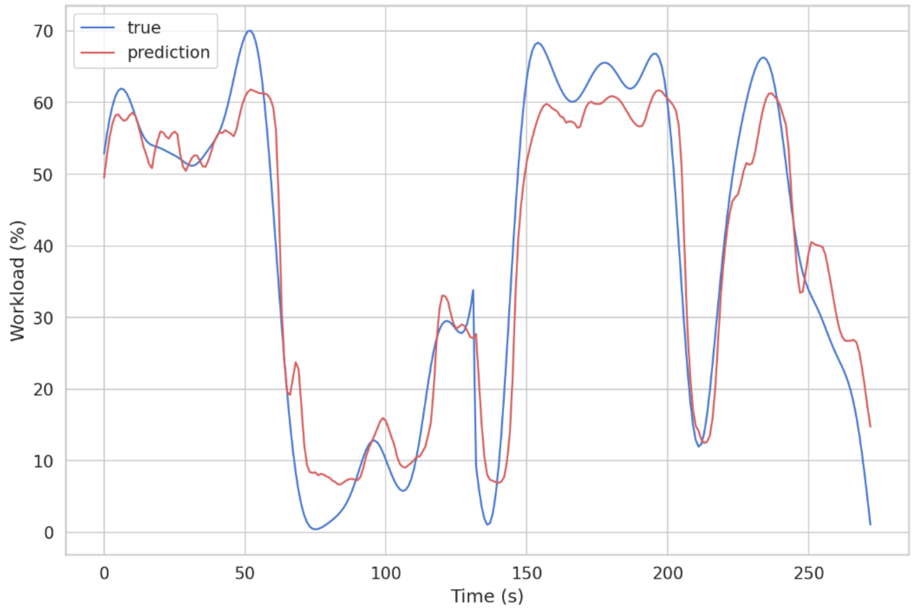


Figure 31. – Stacked LSTM model predictions (red) vs true labels (blue)

6.3.2 Other results

Moreover, this research could also predict the future HR of a participant 5 seconds into the future, as shown in Figure 32. The metrics for this model are an MSE of 4.84, an RMSE of 2.20, and an MAE of 1.73 using a stacked LSTM model.

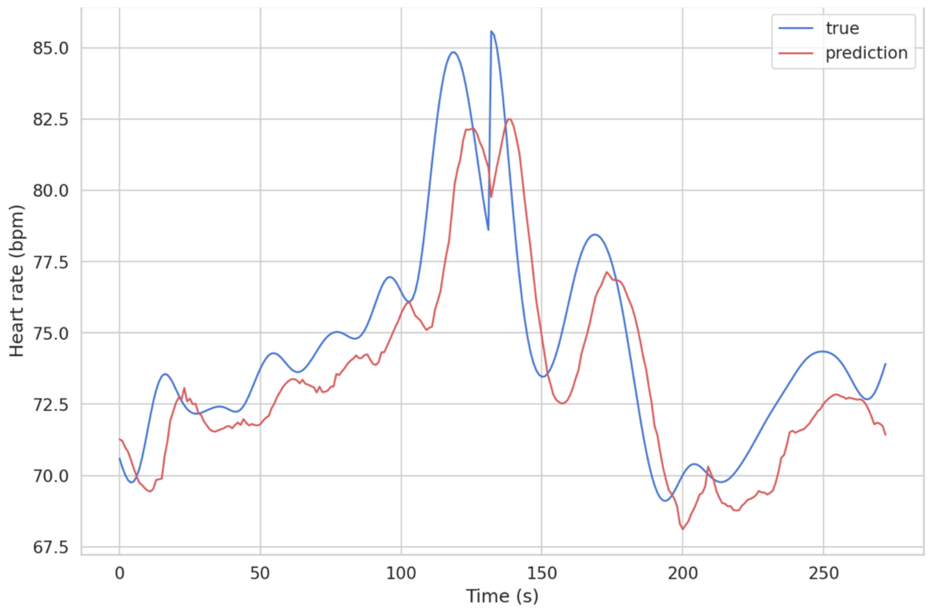


Figure 32. – Stacked LSTM architecture predicting HR 5 seconds into the future

6.4 Discussion

As mentioned in Section 6.5, this chapter tries to answer the hypothesis: Is it possible to predict the CW of a pilot based on his previous behavior? Based on the results in the previous section, we conclude it is possible to predict future workload based on a pilot's past behavior using data from the HR monitor, eye tracker, EEG, events, and more combined with a stacked-LSTM model. Different approaches were used to try to predict the CW of a pilot. One idea was not to use any previous CW to predict future CW, similar to the implementation of the SVR, Ridge regression, MLP, and CNN models. As seen in the previous section, this method did not result in reliable predictions. This could be because predicting CW only by looking at the current data does not give enough information for the models to predict anything confidently. LSTM models were trained without using CW as input and did not yield reliable results. We also tried to add lag in the CW data. This method shifted all CW x seconds into the future (shifted x rows), so the model could analyze the change in PD or HR to predict the current CW. Nevertheless, even when using lag, we could not predict the next CW.

Ultimately, we found that it is possible to predict the CW of a pilot based on his past behavior. Section 6.5.1 demonstrated that when using a stacked LSTM with the ADAM optimizer, the model could predict future CW 5 seconds into the future. Throughout past sections, attentive readers might have noticed that the stacked-LSTM model could predict future workload only for a specific timestamp, five seconds. We tried shorter and longer timestamps to predict the CW of a pilot using different LSTM models. We ultimately chose the five-second mark as it yields the best results while simultaneously being able to predict far enough into the future. When predicting over five seconds, the accuracy of the predictions starts to decline. Predictions less than five seconds into the future are slightly better but are not helpful for the aviation industry and the Pilot AI project. How useful is it to predict the future CW of a pilot one second into the future? We determined it was not, as it would not give enough time for the system to do something about it realistically. For this reason, we chose to predict five seconds into the future as it provides the best of both worlds.

To elaborate on this, let's take an example. During a takeoff, pilots must take off the plane from the runway after they've reached their V1 velocity. Above this velocity, the aircraft does not have enough runway to stop. This velocity is calculated based on the runway length, obstacles, temperature, runway slope, and airplane weight. Airplane manufacturers give these values for each plane. As seen during the experiments, procedures vary during an engine failure based on this V1 speed. If an engine failure happened before V1, pilots had to immediately stop the plane before reaching V1 to ensure a safe procedure. When the engine failure happens, pilots only have a couple of seconds to make the right decision: taking off or stopping the plane. In this context, predicting 5 seconds into the future could help make the right decision in a reasonable amount of time, whereas predicting 10 seconds in the future might be too late.

It is essential to note that while Figure 31 shows a reasonable prediction, the figure can be misleading initially. The predictions appear closer to the actual labels than they are. If we were to zoom in on a portion of the predictions, they would not appear as close as they are. Nevertheless, the model could predict the CW five seconds into the future. Moreover, we found it possible to predict the HR of a person 5 seconds into the future. This finding goes hand in hand with other research using similar methods. Luo and Wu used a similar LSTM architecture to predict the HR of a person. They compared different architectures using ADAM and SGD optimizers. Their proposed model gave an RMSE of 0.208 using the ADAM optimizer (Luo & Wu, 2020), giving similar predictions as this research's proposed model. Moreover, Tsai et al. used a

CNN-LSTM approach to predict the HR of a person using a non-contact radar. Using this approach, their study achieved a 98%-99% accuracy (Tsai et al., 2020).

With all these results, we conclude it is possible to predict the future CW of a pilot based on his past behavior, hence answering the third and final hypothesis of this thesis.

6.5 Conclusion

In this chapter, we introduced the different machine learning and deep learning techniques used during this research. To train these models, we had to preprocess, and feature engineer the data to yield better results. We also explained the different measurement techniques used to analyze and compare the performances of each model using the preprocessed and feature-engineered data. Overall, LSTM models performed the better, with the stacked-LSTM performing the best compared to other models. These LSTM models were trained using the ADAM optimizer and dropout regularization techniques to output the best possible results. The LSTM models can predict the CW of a pilot five seconds into the future. Moreover, we were able to predict the heart rate of pilots using the same LSTM model five seconds into the future too. Doing all this, this chapter proves the third and last hypothesis of this research. The next chapter will wrap up the previous chapter and give a description of this research's future work and conclusion regarding this research.

Chapter 7 – Conclusion

This research investigated the real-time cognitive load of airline pilots during a takeoff procedure. With this, we tried to investigate the possibility of measuring CW during takeoff using an EEG headset, a heart rate monitor, and an eye tracker. To achieve this, we created a software solution that is able to measure the CW, HR, PD, and other simulator events in real-time during a simulated takeoff. This software was made to run on multiple computers, be simulator agnostic, and be integrated and connected with other research projects from the Pilot AI project. This software was used during experiments that we created, which gathered 13 participants, including seven pilots. Out of those seven pilots, five are or were A320 pilots. In total, the experiment gathered 136 takeoffs, or 2 million rows of data combined. This is equivalent to more than 9 hours of continuous time-series data. The experiment consisted of taking off an A320 airplane to 3000ft without autopilot. Six different scenarios were made to trigger variation in CW using the previously mentioned software. Furthermore, the data of each pilot was cleaned, saved, and merged together so other researchers within the Pilot AI project could use it as well. With this data, we were able to prove that it is possible to measure the CW of a pilot in real-time during a simulated takeoff. Next, the data was analyzed using statistical analysis, and we determined that during a workload-intensive task, the CW, HR, and PD of a pilot increased across scenarios and across pilots, proving the second hypothesis of this research. Lastly, to prove the third hypothesis, we compared and analyzed the effectiveness of using machine learning and deep learning models to predict the cognitive load of a pilot in real-time using his past behavior. After comparison and analysis, we found that a stacked-LSTM resulted in the best predictions compared to other models with an MSE of 44.09, an RMSE of 6.64, and an MAE of 5.28. This model implementation used the ADAM optimizer together with dropout regularization to achieve these results. This model was able to predict the CW of a pilot 5 seconds into the future, as shown in Figure 31.

7.1 Limitations

It is essential to mention the limitations of this research and the results gathered during it. First, the model is limited to making CW predictions of a pilot in a takeoff procedure. Moreover, it can, at the moment, only make predictions during engine failures before or after V1 and standard takeoffs. In the future, more failures could be added such that the model could train and predict the CW during these events.

Another limitation is regarding the gathered data. As mentioned in the experiment chapter (Chapter 4), the participants had time to train for 30 minutes before the start of the experiment. This familiarization could have created a particular bias in the data, where participants could anticipate what would happen. One last limitation is that the relationship found between HR, PD, and CW is only during an engine-failure event. More analysis of other types of failures or maneuvers should be done to find if there is a relationship.

In addition, the model can only make predictions using the gathered data from the specific hardware equipment used during this research. For example, the eye tracker outputs different types of data in pixels using a particular algorithm. This makes the model depends on the hardware material used during the experiment. Future research should be done using different hardware equipment to make the model more robust. The same limitation goes for the gathered data. More data should be gathered using other hardware equipment to ensure the robustness of the data.

7.2 Future Work

This research is part of the Pilot AI project and is a starting point for future research. The gathered data and the experimental setup will be used for future research and models regarding the Pilot AI project. Originally intended as a proof of concept, the created setup demonstrated to be a building block for a more advanced setup. Future research could use this setup in a real simulator. Moreover, the generated model could be finetuned and used as a starting point for more advanced machine learning models. The findings regarding the prediction of HR 5 seconds into the future or CW 5 seconds into the future are part of the Pilot AI project objectives.

Other measurement tools in correlation with CW could be used during new experiments to see if it is possible to predict the CW of a pilot without using any EEG headset. Moreover, new measurements such as heart rate variability or the average amount of blinking could be used to better predict a person's CW. Regarding the data, new failures could be added and measured in a real A320 cockpit to give a broader range of data in a more realistic environment. A pipeline could be created to upload the measured data directly into a cloud server so all participants of the pilot AI project could access it immediately.

Bibliography

AL-Ziarjawey, H. (2015). Heart Rate Monitoring and PQRST Detection Based on Graphical User Interface with Matlab. *International Journal of Information and Electronics Engineering*. https://doi.org/10.7763/IJIEE.2015.V5.550

Babiloni, F. (2019). *Mental Workload Monitoring: New Perspectives from Neuroscience* (pp. 3–19). https://doi.org/10.1007/978-3-030-32423-0_1

Ball, L. J. (2014). Eye-tracking and reasoning: What your eyes tell about your inferences. In *New approaches in reasoning research* (pp. 51–69). Psychology Press.

Benlamine, M. S., Chaouachi, M., Frasson, C., & Dufresne, A. (2016). Physiology-based Recognition of Facial Micro-expressions using EEG and Identification of the Relevant Sensors by Emotion: *Proceedings of the 3rd International Conference on Physiological Computing Systems*, 130–137. https://doi.org/10.5220/0006002701300137

Brünken, Roland, Seufert, Tina, Paas, Fred, & Department of Psychology, Education and Child Studies. (2010). Measuring cognitive load. In *Cognitive Load Theory* (pp. 181–202). Cambridge University Press. https://doi.org/10.1017/cbo9780511844744.011

Burns, B. (n.d.). *What Is Multithreading: A Guide to Multithreaded Applications*. TotalView by Perforce. Retrieved July 30, 2022, from https://totalview.io/blog/multithreading-multithreaded-applications

Chaouachi, M., Jraidi, I., & Frasson, C. (2015). MENTOR: A Physiologically Controlled Tutoring System. In F. Ricci, K. Bontcheva, O. Conlan, & S. Lawless (Eds.), *User Modeling, Adaptation and Personalization* (pp. 56–67). Springer International Publishing. https://doi.org/10.1007/978-3-319-20267-9_5

Doumbouya, R., Benlamine, M. S., Dufresne, A., & Frasson, C. (2018). Game Scenes Evaluation and Player's Dominant Emotion Prediction. In R. Nkambou, R. Azevedo, & J. Vassileva (Eds.), *Intelligent Tutoring Systems* (pp. 54–65). Springer International Publishing. https://doi.org/10.1007/978-3-319-91464-0_6

Farnsworth, B. (2021, July 13). *What is EEG (Electroencephalography) and How Does it Work?* Imotions. https://imotions.com/blog/what-is-eeg/

Gallagher, N. (2020). *Savitzky-Golay Smoothing and Differentiation Filter*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

Gopher, D., & Braune, R. (1984). *On the Psychophysics of Workload: Why Bother with Subjective Measures?* https://doi.org/10.1177/001872088402600504

Hancock, P. A. (1989). The effect of performance failure and task demand on the perception of mental workload. *Applied Ergonomics*, *20*(3), 197–205. https://doi.org/10.1016/0003-6870(89)90077-X

Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In P. A. Hancock & N. Meshkati (Eds.), *Advances in Psychology* (Vol. 52, pp. 139–183). North-Holland. https://doi.org/10.1016/S0166-4115(08)62386-9

Jerčić, P., Sennersten, C., & Lindley, C. (2020). Modeling cognitive load and physiological arousal through pupil diameter and heart rate. *Multimedia Tools and Applications*, *79*(5–6), 3145–3159.

Lakna. (2018, August 19). *Difference Between Depolarization and Repolarization*. Pediaa.Com. https://pediaa.com/difference-between-depolarization-and-repolarization/

Lang, P. J., Greenwald, M. K., Bradley, M. M., & Hamm, A. O. (1993). Looking at pictures: Affective, facial, visceral, and behavioral reactions. *Psychophysiology*, *30*(3), 261–273. https://doi.org/10.1111/j.1469-8986.1993.tb03352.x

Luo, M., & Wu, K. (2020). Heart rate prediction model based on neural network. *IOP Conference Series: Materials Science and Engineering*, *715*(1), 012060. https://doi.org/10.1088/1757-899X/715/1/012060

Madona, P., Basti, R. I., & Zain, M. M. (2021). PQRST wave detection on ECG signals. *Gaceta Sanitaria*, *35*, S364–S369. https://doi.org/10.1016/j.gaceta.2021.10.052

Mayer, R. E., & Moreno, R. (2003). Nine Ways to Reduce Cognitive Load in Multimedia Learning. *Educational Psychologist*, *38*(1), 43–52. https://doi.org/10.1207/S15326985EP3801_6

MDN. (n.d.). *The WebSocket API (WebSockets)—Web APIs | MDN*. Retrieved July 30, 2022, from https://developer.mozilla.org/fr/docs/Web/API/WebSockets_API

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, *63*(2), 81–97. https://doi.org/10.1037/h0043158

Mishra, S. K., Gupta, M., & Upadhyay, D. K. (2020). Active realization of fractional order Butterworth lowpass filter using DVCC. *Journal of King Saud University - Engineering Sciences*, *32*(2), 158–165. https://doi.org/10.1016/j.jksues.2018.11.005

My-EKG. (n.d.). *Electrocardiogram Paper*. Retrieved July 30, 2022, from https://en.my-ekg.com/basic-principles/paper-ekg.html

Neurosky. (n.d.). *ECG vs PPG for Heart Rate Monitoring: Which is Best?* Retrieved July 30, 2022, from http://neurosky.com/2015/01/ecg-vs-ppg-for-heart-rate-monitoring-which-is-best/

Paas, F., Tuovinen, J. E., Tabbers, H., & Van Gerven, P. W. M. (2003). Cognitive Load Measurement as a Means to Advance Cognitive Load Theory. *Educational Psychologist*, *38*(1), 63–71. https://doi.org/10.1207/S15326985EP3801_8

Palinko, O., Kun, A. L., Shyrokov, A., & Heeman, P. (2010). Estimating cognitive load using remote eye tracking in a driving simulator. *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, 141–144. https://doi.org/10.1145/1743666.1743701

Panish, Boyle, Ravipudi, & Shea. (n.d.). *Aviation and Plane Crash Statistics (Updated 2022)*. Retrieved July 30, 2022, from https://www.psbr.law/aviation_accident_statistics.html

Paul, A. (n.d.). *Chapter 3 Measurement of Operator Workload 1 Summary*. Retrieved July 30, 2022, from https://www.academia.edu/6650878/Chapter_3_Measurement_of_Operator_Workload_1_Summary

Pintus, E., Sorbolini, S., Albera, A., Gaspa, G., Dimauro, C., Steri, R., Marras, G., & Macciotta, N. P. P. (2014). Use of locally weighted scatterplot smoothing (LOWESS) regression to study selection signatures in Piedmontese and Italian Brown cattle breeds. *Animal Genetics*, *45*(1), 1–11. https://doi.org/10.1111/age.12076

Pouw, W. T. J. L., Eielts, C., van Gog, T., Zwaan, R. A., & Paas, F. (2016). Does (Non-)Meaningful Sensori-Motor Engagement Promote Learning With Animated Physical Systems? *Mind, Brain, and Education*, *10*(2), 91–104. https://doi.org/10.1111/mbe.12105

Pouw, W. T. J. L., Mavilidi, M.-F., van Gog, T., & Paas, F. (2016). Gesturing during mental problem solving reduces eye movements, especially for individuals with lower visual working memory capacity. *Cognitive Processing*, *17*(3), 269–277. https://doi.org/10.1007/s10339-016-0757-6

Rani, P., Sarkar, N., & Adams, J. (2007). Anxiety-based affective communication for implicit human–machine interaction. *Advanced Engineering Informatics*, *21*, 323–334. https://doi.org/10.1016/j.aei.2006.11.009

Reid, G. B., & Nygren, T. E. (1988). The Subjective Workload Assessment Technique: A Scaling Procedure for Measuring Mental Workload. In P. A. Hancock & N. Meshkati (Eds.), *Advances in Psychology* (Vol. 52, pp. 185–218). North-Holland. https://doi.org/10.1016/S0166-4115(08)62387-0

Robinson, C. (2019, September 30). Final Report on Last Year’s Horrific Airplane Crash in Havana. *Havana Times*. https://havanatimes.org/features/final-report-on-last-years-horrific-airplane-crash-in-havana/

Ruiz, N., Liu, G., Yin, B., Farrow, D., & Chen, F. (2010). Teaching athletes cognitive skills: Detecting cognitive load in speech input. *Proceedings of the 24th BCS Interaction Specialist Group Conference*, 484–488.

Sattar, Y., & Chhabra, L. (2022). Electrocardiogram. In *StatPearls*. StatPearls Publishing. http://www.ncbi.nlm.nih.gov/books/NBK549803/

scikit-learn. (n.d.-a). *Sklearn.linear_model.Ridge*. Scikit-Learn. Retrieved July 30, 2022, from https://scikit-learn/stable/modules/generated/sklearn.linear_model.Ridge.html

scikit-learn. (n.d.-b). *Sklearn.svm.SVR*. Scikit-Learn. Retrieved July 30, 2022, from https://scikit-learn/stable/modules/generated/sklearn.svm.SVR.html

Sevcenko, N., Ninaus, M., Wortha, F., Moeller, K., & Gerjets, P. (2021). Measuring Cognitive Load Using In-Game Metrics of a Serious Simulation Game. *Frontiers in Psychology*, *12*. https://www.frontiersin.org/articles/10.3389/fpsyg.2021.572437

Siegle, G. J., Ichikawa, N., & Steinhauer, S. (2008). Blink before and after you think: Blinks occur prior to and following cognitive load indexed by pupillary responses. *Psychophysiology*, *45*(5), 679–687. https://doi.org/10.1111/j.1469-8986.2008.00681.x

Sosnowski, T., Krzywosz-Rynkiewicz, B., & Roguska, J. (2004). Program running versus problem solving: Mental task effect on tonic heart rate. *Psychophysiology*, *41*(3), 467–475. https://doi.org/10.1111/j.1469-8986.2004.00171.x

Sperling, G., & Dosher, B. (1986). Handbook of perception and performance. *Handbook of Human Perception and Performance*, *1*, 1–65.

Sweller, J. (2011). Cognitive load theory. In *The psychology of learning and motivation: Cognition in education, Vol. 55* (pp. 37–76). Elsevier Academic Press. https://doi.org/10.1016/B978-0-12-387691-1.00002-8

Szabadi, E. (2018). Functional Organization of the Sympathetic Pathways Controlling the Pupil: Light-Inhibited and Light-Stimulated Pathways. *Frontiers in Neurology*, *9*. https://www.frontiersin.org/articles/10.3389/fneur.2018.01069

Tobii. (2015a, August 10). *How do Tobii Eye Trackers work? | Learn more with Tobii Pro* [Information]. https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/how-do-tobii-eye-trackers-work/

Tobii. (2015b, August 14). *Are pupil size calculations possible with Tobii Eye Trackers?* [Information]. https://www.tobiipro.com/learn-and-support/learn/eye-tracking-essentials/is-pupil-size-calculations-possible-with-tobii-eye-trackers/

Tsai, Y.-C., Lai, S.-H., Ho, C.-J., Wu, F.-M., Henrickson, L., Wei, C.-C., Chen, I., Wu, V., & Chen, J. (2020). High Accuracy Respiration and Heart Rate Detection Based on Artificial Neural Network Regression. *2020 42nd Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 232–235. https://doi.org/10.1109/EMBC44109.2020.9175161

Tsang, P. S., & Vidulich, M. A. (2006). Mental workload and situation awareness. In *Handbook of human factors and ergonomics, 3rd ed* (pp. 243–268). John Wiley & Sons, Inc. https://doi.org/10.1002/0470048204.ch9

Wikipedia. (2022). Depolarization. In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Depolarization&oldid=1075672607

Xi, P., Law, A., Goubran, R., & Shu, C. (2019). Pilot Workload Prediction from ECG Using Deep Convolutional Neural Networks. *2019 IEEE International Symposium on Medical Measurements and Applications (MeMeA)*, 1–6. https://doi.org/10.1109/MeMeA.2019.8802158

Zekveld, A. A., Kramer, S. E., & Festen, J. M. (2011). Cognitive load during speech perception in noise: The influence of age, hearing loss, and cognition on the pupil response. *Ear and Hearing*, *32*(4), 498–510. https://doi.org/10.1097/AUD.0b013e31820512bb