Universidad del País Vasco

Euskal Herriko Unibertsitatea

Design and Electronic Implementation of Machine Learning-based Advanced Driving Assistance Systems

DISSERTATION

to obtain the degree of doctor at the University of the Basque Country, by

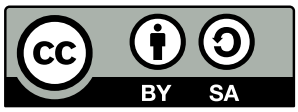
Óscar Mata Carballeira

Thesis supervisors:

Inés Juliana del Campo Hagelström

María Victoria Martínez González

February 2022



"Hutsera saltoten ez badogu
leku bardinien jarraituko dogu"

Gatibu

Contents

List of Figures

List of Tables

List of Abbreviations

ABS	Anti-lock braking system
ACAP	Adaptive compute acceleration platform
ACC	Adaptive cruise control
ADAS	Advanced driving-assistance system
ADC	Analog-to-digital converter
AEB	Autonomous emergency braking
AI	Artificial intelligence
AMBA	Advanced microcontroller bus architecture
ANFIS	Adaptive neuro-fuzzy inference system
ANN	Artificial neural network
APU	Application-processing unit
ARM	Advanced RISC machines
ASA	Automatic speed assistant
ASIC	Application-specific integrated circuit
AXI4	Advanced eXtensible Interface-4
BMU	Best-matching unit.
BP	Backpropagation
BP	Brake-pedal pressure
BRAM	Block RAM
BSP	Board support package
CAN	Controller area network
CC	Cruise control
CLB	Configurable logic block
CPU	Central processing unit
DAS	Driving assistance system
DBSCAN	Density-based space clustering of applications with noise
DMA	Direct memory access
DS	Driving style
DSP	Digital signal processing
ECU	Electronic control unit
EEG	Electroencephalogram
ELM	Extreme learning machine
EM	Expectation-maximization

(E)RPM	(Engine) revolutions per minute
ESC	Electronic stability control
EU	European Union
FCW	Front collision warning
FF	Flip-flop
FIS	Fuzzy inference system
FPGA	Field-programmable gate array
FSM	Finite-states machine
GHG	Greenhouse gasses.
GIC	Global interrupt controller
GMM	Gaussian mixture models
GNSS	Global navigation satellite systems
GP	Gas-pedal pressure
GPU	Graphics processing unit
HAC/HCA	Hierarchical agglomerative clustering
HMM	Hidden Markov model
HW	Hardware
I/O	Input and output
IMU	Inertial measurements unit
IR	Infrared
IRB	Institutional Review Board
ISA	Intelligent speed assistant
ISO	International Organization for Standardization
JTAG	Joint Test Action Group
KDE	Kernel density estimation
k-NN	k-nearest neighbors
LDW	Lane-departure warning
LED	Light-emitting diode
LEZ	Low-emission zone
LIDAR	Light detection and ranging
LKA	Lane-keeping assistance
LSE	Least-squares estimator
LUT	Look-up table
MF	Membership function
ML	Machine learning.
MPC	Model predictive control
MSDV	Motion sickness dose value
NDS	Naturalistic driving study
NF	Neuro-fuzzy
NoC	Network-on-chip
non-NDS	Non-naturalistic driving study

OEM	Original equipment manufacturer
PCA	Principal component analysis
PCB	Printed circuit board
PCC	Pearson correlation coefficient
PGP	Percent gas pedal
PL	Programmable logic
PR	Pitch rate
PS	Processing system
PSoC	Programmable system-on-chip
RAM	Random access memory
RBF	Radial-basis function
RGB	Red-green-blue
RLS	Recursive least-squares
RMS	Root-mean-square
ROM	Read-only memory
RPU	Real-time processing unit
RR	Roll rate
SAE	Society of Automotive Engineers
SDK	Software development kit
SLFN	Single-layer feedforward network
SoC	System-on-chip
SOM	Self-organizing map
SVM	Support vector machines
SW	Software
SWA	Steering wheel angle
SWS	Steering wheel speed
TCS	Traction control system
TETH	Time-exposed time-headway
THW	Time-headway
TITH	Time-integrated time-headway
TSI/TSR	Traffic-sign identification/recognition
TTCi	Inverse of the time-to-collision
U-matrix	Unified distance matrix
USA	United States of America
USB	Universal serial bus
VHDL	VHSIC hardware description language
VR	Vomit rate
VS	Vehicle speed
VTTI	Virginia Tech Transportation Institute
X/Y/ZACC	X/Y/Z-axis acceleration
YR	Yaw rate

Resumen

Los automóviles han evolucionado significativamente desde que fueran comercializados por primera vez, y, junto a ellos, la conducción [1]. Para afrontar dichos cambios, aparecen los sistemas de asistencia a la conducción (DAS: *driving assistance systems*) [2] inicialmente como una forma de liberar a los conductores de llevar a cabo ciertas tareas repetitivas, tales como mantener una velocidad constante. Estos sistemas fueron evolucionando gradualmente para actuar en caso de que ocurra algún evento que comprometa la seguridad, como puede ser la pérdida del control de la trayectoria del vehículo, viéndose drásticamente mejorada la seguridad [3].

En ese sentido, la tecnología implementada en los automóviles actuales se encuentra en un estado de avance y madurez tal que permite el despliegue de sistemas avanzados de asistencia a la conducción (ADAS: *advanced driving assistance systems*), que, a diferencia de los DAS más convencionales, también tienen en cuenta variables externas al vehículo, como pueden ser las condiciones de la carretera o del propio ambiente, proporcionando funciones más avanzadas [4]. Así, estos sistemas pueden leer señales de tráfico, mantener la distancia de seguridad con el vehículo anterior e, incluso, frenar automáticamente en caso de riesgo inminente de colisión.

Sin embargo, a pesar de su complejidad y nivel de refinamiento, estos ADAS normalmente carecen de funciones de personalización por parte del conductor, llegando incluso a mostrar comportamientos que muchos automovilistas podrían considerar poco naturales [5]. Estas respuestas, por un lado, pueden perjudicar la propia experiencia de conducción, y, por el otro, pueden ocasionar una sensación de falta de confort en los ocupantes del coche. Por estas razones, el nivel de utilización de los ADAS por parte de los conductores es menor de lo que podría ser, reduciendo el impacto positivo de estos sistemas en la mejora de la seguridad.

Del mismo modo, la influencia del automóvil no solamente se refleja en el entorno de la carretera, sino también en el medio ambiente, impactando notablemente en el calentamiento global. Es sabido que los

vehículos con motor de combustión interna se encuentran entre los mayores contribuyentes en las emisiones de gases de efecto invernadero (GHG: *greenhouse gasses*), y que el estilo de conducción (DS: *driving style*) ejerce un efecto directo en dichas emisiones [6]. Por ello, parece razonable pensar que, si de alguna manera pudiese analizarse el estilo de conducción de forma automática, este podría corregirse para mejorar los niveles de emisiones y la conducción ecológica. Este enfoque, desarrollado para los coches de conducción manual, es también válido para generar directrices de eficiencia energética orientadas a los futuros coches autónomos.

En lo que respecta a la falta de confort, es de destacar que, además de por el comportamiento poco natural de algunos ADAS, también puede originarse en ciertos tipos concretos de DS [7]. Así, del mismo modo que para la conducción ecológica, el análisis y diagnóstico automáticos se postulan como una alternativa prometedora para mejorar ciertas tendencias en la conducción que perjudican el confort.

Por estas razones, el objetivo principal de este trabajo es contribuir al desarrollo y perfeccionamiento de diversos ADAS. El enfoque seguido se centra particularmente en la personalización de ADAS basados en datos de conducción. Para ello, se utilizan algoritmos de *machine learning* tales como redes neuronales artificiales (ANN: *artificial neural networks*) [8] y sistemas neuro-borrosos [9], que, conjuntamente con bases de datos de conducción en situaciones reales, sirven para modelar estilos de conducción. Este modelado tiene como propósito obtener parámetros característicos del estilo de cada conductor para personalizar ADAS ya existentes, como el control de crucero adaptativo (ACC: *adaptive cruise control*) [10]. Además, es también útil para proponer nuevos ADAS que ayuden a los conductores a mejorar sus estilos de conducción. Todas estas contribuciones están diseñadas para aumentar el nivel de confianza en los sistemas automatizados, influyendo de manera positiva en la seguridad, la conducción ecológica [11] y en el confort de marcha [12]. Para terminar, se desarrollan y prueban soluciones hardware que puedan ejecutar en tiempo real las aplicaciones para ADAS propuestas.

En el Capítulo 1 se lleva a cabo una contextualización histórica, desde el mismo comienzo de la automoción hasta las tecnologías más innovadoras. Además, se repasan los algoritmos de *machine learning*, haciendo especial hincapié en los algoritmos utilizados en este trabajo, como son las redes neuronales, los sistemas neuro-borrosos y los mapas auto-organizados (SOM: *self-organizing maps*). Seguidamente, se introduce el concepto de estudio de conducción naturalista (obtenido con conductores no-controlados, conduciendo sus propios vehículos) y no-

naturalista (obtenido bajo un estricto control experimental, y utilizando un mismo coche altamente instrumentalizado). Se utilizan asimismo bases de datos de ambos tipos (SHRP2 de Virginia Tech [13] y Uyanik, de la Universidad de Sabançi, Estambul [14]). Estos datos se utilizarán para entrenar los ADAS propuestos. Para terminar el capítulo, se realiza un repaso de diferentes soluciones hardware para implementar ADAS, como las FPGA (*field-programmable gate arrays*), los PSoC (*programmable system-on-chips*) o los recientes ACAP (*adaptive compute acceleration platform*).

El Capítulo 2 introduce un sistema para la personalización de la respuesta de los sistemas ACC. Este sistema se desarrolla utilizando datos de conducción naturalista, y un sistema neuro-borroso que identifique los comportamientos más deseables de seguimiento de vehículos.

Dado que la implementación de un sistema de personalización de la conducción para su integración en el automóvil requiere un modelo de agrupamiento y clasificación (*clustering*) de estilo de conducción con alta velocidad de respuesta, la solución adoptada se basa en la aproximación de alto rendimiento de los clusters mediante un sistema neuro-borroso de tipo ANFIS (ANFIS: *adaptive neuro-fuzzy inference system*). La capacidad de aproximación universal de los ANFIS, junto con su topología de capas inherentemente paralelizable, hacen que este modelo sea adecuado para una implementación eficiente en *hardware* (HW). El sensor neuro-borroso es implementado con éxito utilizando un dispositivo FPGA que proporciona alta velocidad y bajo consumo de energía para la ejecución de ADAS en tiempo real. Finalmente, se desarrolla un software que ajusta automáticamente el intervalo temporal con el vehículo precedente de acuerdo con el estilo de conducción de cada persona.

En el Capítulo 3, se analiza el consumo de combustible con el objetivo de fomentar la conducción ecológica. Para ello, se procesan datos de conducción reales con resultados de simulación de consumo de combustible, para corregir comportamientos contrarios a las directrices de conducción ecológica mediante la generación de recomendaciones didácticas. Seguidamente, se seleccionan las características con mayor correlación con el consumo de combustible y se agrupan mediante una red neuronal no-supervisada de tipo SOM. Este agrupamiento permite clasificar a los conductores y diagnosticar las causas de su elevado consumo de combustible.

Se desarrolla y testea además un acelerador hardware basado en PSoC que permite al sistema funcionar en tiempo real. Para concluir el capítulo, se describe el software que proveé recomendaciones basadas

en el estilo de conducción. Es importante resaltar, además, que estas recomendaciones están diseñadas para ser válidas para la gran mayoría de los conductores, utilizando un lenguaje natural y fácil de entender, abarcando desde el uso de los pedales de acelerador y freno hasta el manejo del cambio de marchas. Así, se obtiene una mejora en el rendimiento de los sistemas de ahorro de combustible ya existentes, con mejoras esperables en el consumo de combustible y en la emisión de gases de hasta el 31,5 %.

El Capítulo 4 describe un sistema para mejorar el confort de marcha. Así, con los mismos datos que en el anterior, se propone proporcionar consejos didácticos de conducción personalizados para corregir comportamientos al volante que puedan perjudicar la sensación de confort en el automóvil. Para ello, se seleccionan características de conducción que influyan en en confort de marcha, y se agrupan mediante una red neuronal no-supervisada de tipo SOM. Dicho agrupamiento ayudará a diagnosticar las causas que originan las perturbaciones de la comodidad de los ocupantes del vehículo. Se desarrollan recomendaciones didácticas de estilo de conducción basadas en lenguaje natural y se comprueba y verifica que sean compatibles con los consejos proporcionados en la solución descrita en el Capítulo 3. Al igual que previamente, las recomendaciones proporcionadas involucran el uso de los pedales, de la palanca de cambios y del volante, permitiendo potencialmente obtener una mejora de los parámetros de evaluación del confort de hasta el 57,7 %.

En el Capítulo 5, se propone una solución integrada para implementar ADAS en un único chip. Este enfoque utiliza un tipo novedoso de dispositivo hardware reconfigurable, conocido como ACAP, para mejorar el rendimiento de los dispositivos FPGA y PSoC, alcanzando rendimientos punteros tanto para la partición hardware como para la software. En este sistema se despliegan los módulos propuestos en los Capítulos 2, 3 y 4, para integrarlos en el mismo dispositivo y se rediseña la arquitectura de la solución completa con el fin de optimizar al máximo el rendimiento.

Con ese objetivo, se adaptan los módulos para ser compatibles con el sistema novedoso de interconexión de Xilinx, conocido como red en un chip (NoC: *network-on-a-chip*), que permite ligar diversos módulos con un rendimiento inédito en dispositivos reconfigurables. Es de destacar que esta matriz de interconexión, a pesar de su complejidad, se gestiona de manera transparente para el diseñador, lo que permite desarrollar potentes aplicaciones en un ciclo de tiempo relativamente corto. En lo que respecta al software, se efectúa una migración e integración de los

desarrollos de los capítulos anteriores.

Para terminar, se efectúa el análisis del rendimiento del sistema, y se compara con las implementaciones descritas en los apartados anteriores, obteniéndose frecuencias de reloj hasta un 42,2 % superiores. Estas frecuencias permiten tiempos de latencia extremadamente bajos que, junto a las capacidades de escalabilidad de los ACAP, implican la posibilidad de implementar la práctica mayoría de los ADAS de un vehículo en el dispositivo, con el ahorro de consumo y la simplicidad que esto supone. Finalmente, se analiza el consumo de potencia.

Finalmente, las conclusiones y trabajos futuros se indican en el Capítulo 6. Así, en esta tesis se presenta un marco de trabajo para el desarrollo de ADAS basados en el DS. Este marco, que se deriva de datos de conducción real, proporciona no solo herramientas para mejorar la conducción no-autónoma, sino también información conductual que puede ser de ayuda a la hora de mejorar el rendimiento y el comportamiento de los coches autónomos del futuro en términos de control longitudinal, ahorro de combustible y confort de marcha. Estos aspectos son de vital importancia para dar soluciones en aspectos como la seguridad vial y la eficiencia energética en el contexto actual de continuo aumento del tráfico y de las emisiones de contaminantes. Otros ADAS podrían beneficiarse también de esta aproximación basada en datos, imitando mejor la conducta humana, y así mejorando el nivel de confianza de los usuarios en estos sistemas, impulsando la adopción y la aceptación de los mismos.

En lo que respecta al HW y al *software* (SW), en este trabajo se demuestra que los ACAP son herramientas eficientes para implementar una amplia gama de nuevas funcionalidades, pudiendo incluso llegar a sustituir las unidades de control electrónico (ECU: *electronic control units*) actualmente en los coches, incrementando, por otro lado, la potencia computacional disponible y reduciendo el consumo energético. Además al utilizar HW reconfigurable, se abre la posibilidad de actualizar no solo el SW de los vehículos, sino también su HW, pudiendo incorporar nuevas funcionalidades incluso después de ser fabricados.

Á miña familia, por
coidardes sempre de min

Sen choiva,
non habería flores

Abstract

Automobiles have noticeably evolved since they were put into market firstly, and driving has changed by their side. To face these changes, driving assistance systems (DAS) appeared at the beginning as a manner to relieve drivers of performing repetitive tasks, such as keeping a fixed speed. These systems gradually evolved to act if a safety condition that might compromise the handling of the vehicle happened, drastically improving the general safety ratings of automobiles.

In that sense, the technology boarded in current automobiles is advanced and mature enough to implement more advanced driving assistance systems (ADAS), that, in contrast with the more typical DAS, also take into account external variables of the vehicle (i.e. environmental and road conditions) to provide more complete functionalities, such as reading traffic signs or keeping the safety distance with the precedent vehicle, and even braking when a collision is about to happen.

Nevertheless, these ADAS, despite their complexity and refinement, often lack driver-selected personalization functions, showing responses that look unnatural for many drivers. These responses, on the one hand, could impair the driving experience when using them, and, on the other hand, may cause discomfort in all the car occupants. Consequently, the level of engagement with these systems is lower than it could, reducing their effectiveness on increasing safety.

On the other hand, not only do cars influence the road environment, but also play an important role in global warming. It is well known that automobiles are one of the main contributors to the emissions of greenhouse-effect gases (GHG), and that driving style (DS) has a direct effect on those emissions. Thus, it seems reasonable that if the DS could somehow be automatically assessed and diagnosed, it could be corrected to improve emissions and eco-driving. This approach, developed for the current context of cars still being manually driven, is valid for generating fuel efficiency guidelines for the emerging autonomous driving paradigm.

Regarding discomfort, it should be remarked that not only may be caused by the unnatural response of ADAS, but also by certain types

of DS. Hence, in the same manner as for eco-driving, the automatic assessment and diagnostic seems a promising alternative to improve DS-related uncomfortable trends.

For those reasons, the main objective of this work is contributing to the development and refinement of ADAS. Particularly, the spotlight is put on the data-based personalization of ADAS. For that purpose machine learning algorithms such as artificial neural networks (ANNs) and neuro-fuzzy systems jointly with real-word driving databases have been used to model DSs with the aim of obtaining characteristic parameters for each individual driver. These parameters can be used to personalize already existing ADAS, such as adaptive cruise control (ACC), and to propose novel ADAS that help drivers to modify their DS. All these contributions are intended to increase the level of confidence on the automated systems, presumably influencing on car safety, eco-driving and ride-comfort positively. Finally, hardware solutions that can run the developed applications in real time with very high performance rates are developed and tested.

This work is organized as follows:

Chapter 1 puts this document in historical context, from the very beginning of the automotive era to the most innovative technologies. On the other hand, an overview on the machine learning algorithms, with the stress put over the algorithms used to carry out this work is performed. Additionally, the concepts of naturalistic and non-naturalistic driving studies are introduced, and datasets of both the types (i.e. SHRP2 and Uyanik instrumented car, respectively) are presented. These data are going to be used to train the proposed ADAS. Finally a review on several ADAS-intended hardware solutions, such as field-programmable gate arrays (FPGAs), programmable system-on-chips (PSoCs) and adaptive compute acceleration platfors (ACAPs), is performed.

Chapter 2 introduces a system to personalize the response of ACC. This system is developed by using real-word data. With these data, a neuro-fuzzy based system that identifies the desired car-following behavior is developed. Since this type of algorithms are computing intensive, a FPGA)-based hardware accelerator is developed and tested to improve the general performance. Finally, a piece of software that automatically tunes the desired time gap with the precedent vehicle according tho each individual's DS is developed.

In Chapter 3 fuel consumption is assessed with the aim of improving eco-driving. This system is also developed by using driving data, and is intended to correct eco-driving-compromising DSs by providing educational advice to drivers. For that purpose, fuel consumption-related

features are properly selected and clustered by an unsupervised neural network. This allows to group drivers and to diagnose the causes of their high level of fuel consumption. An FPGA-based hardware accelerator is developed and tested to enable this system to run in real time. Finally, a piece of software that automatically provides drivers with natural-language DS-based advice is described. This system could help drivers to achieve potential reductions of the 31.5% in fuel consumption and GHG.

Chapter 4 describes a system to improve ride comfort from a data-based approach. This solution is intended to correct ride comfort-compromising DS by providing educational advice to drivers. For that purpose, ride comfort-related features are properly selected and clustered by an unsupervised neural network to diagnose the causes of discomfort. Natural-language DS-based advice to improve ride comfort is described and its compatibility with the fuel consumption-improvement solution of Chapter 3 is tested and verified. This system could help drivers to achieve potential improvements of the 57.7% in ride comfort parameters.

Chapter 5 proposes an integrated solution on a single chip to implement ADAS. This approach uses a novel type of re-configurable device, called ACAP, to supersede the performance of the FPGA and PSoC devices, achieving state-of-the-art performance rates for both HW and SW developments. In this system, the modules proposed in Chapters 2, 3 and 4 are deployed in ta single chip and the timing performance and consumption of power and resources are checked.

Finally, the conclusions and future works are enumerated in Chapter 6.

Chapter 1

Introduction

Automobiles have drastically changed since Karl Benz patented his first vehicle in 1886, in Mannheim [1]. These first cars were clearly derived from the horse-drawn carriages they intended to substitute, with the same elements as in the previous centuries except for the animal traction. As a matter of fact, these initial cars were little more than an extravagance for the richest, with few advantages and many drawbacks when compared with the animal-drawn wagons, particularly regarding mechanical reliability. Concerning the advantages, the main one was the ability of reaching speeds of around 20 km/h, which do not appear to be very high for the today's standards, but they were ludicrous by the road standards of the late 19th century [15]. As a consequence, and due to the fact of pedestrians, carriages and automobiles shared the same space, even though cars were rare at the time, the combustion engine car-related pedestrian injuries and deaths began to rise and quickly surpassed those caused by traditional means of transportation, specially in urban roads and during night-countryside trips. These safety conditions were aggravated by the fact that no additional safety requirements, such as driving licenses or medical exams, were required for driving a car, so no minimal car-handling abilities were guaranteed.

Because of the aforementioned safety issues, authorities, majorly city councils, began to elaborate the very first set of road safety regulations, as soon as in 1879, with the *Red Flag laws*. These regulations imposed several restrictions while driving a non-animal-traction vehicle through urban areas, with the main objective of reducing the fatalities among pedestrians. For that reason, they mandated several exotic measures, such as notifying in advance that a car was going to go through the urban area, so that a waiter could be ready to escort the car while blowing a whistle and waving a red flag to warn the other road users of its presence [16]. This measure, despite its exoticism, can be considered as the first speed limit, clearly intended to minimize the road

fatalities. However, this was only practical when cars were a luxury for the very few and many days could pass between transits, so, when the revolution of the mass-produced automobile was carried out by the Ford Motor Company with the Model T [17], it was proven insufficient and further regulations had to be passed by authorities [18].

Regarding the improvement of both the task of driving and the automobiles themselves, the efforts at the dawn of the automotive era were focused on improving comfort, reliability and usefulness of a mean of transportation that suffered from many mechanical failures and lacks on ergonomics, with little attention paid to safety. Thus, despite automobiles becoming faster and more comfortable during the years, their safety measures did not really improve at all, and, consequently, road fatalities did not stop increasing. It was in the 1950s when the car safety paradigm experimented a turning point with the invention of the 3-point seat belt by Volvo [19]. This car maker freed the patent so that all manufacturers could implement it while paying no royalties, and consequently, they began incorporating this advance to their automobiles. This fact contributed to reducing the severity of the injuries resulting of car accidents, and, thus, the likeliness of dying in a crash [19].

However, despite the risk of severe injuries and the likeliness of dying as a result of a crash was drastically alleviated by the use of seatbelts, the number of fatalities did not stop increasing in the subsequent years. This fact, despite apparently being contradictory, becomes evident if we consider the sinistrality as a consequence of mobility [20]. Global mobility has grown non-stop since the end of the World War II due to the development of both automotive technology and the transportation network [21]. Nonetheless, although the automobiles clearly evolved in terms of comfort and mechanical reliability, the task of driving did not evolve too much since the consolidation of the standard commands of the car, such as the pedals, the gear stick, the steering wheel or the turning lights' lever. These controls have remained barely unchanged for almost a century, and consequently, driving has always been an eminently manual task exposed to the errors and distractions that human drivers are prone to commit, specially in the case of accidents with human fatalities [22]. For that reason, the automation of the most unpleasant driving tasks would, on the one hand, help to alleviate the human-related risks, and, on the other hand, increase the perceived comfort.

It was in the late 20th century when original equipment manufacturers (OEMs) decided to focus on measuring the internal status of the vehicle, such as wheels' angular velocity, accelerations and rotational

velocity. These variables allow to know how the vehicle is behaving at a given moment, and consequently, control the dynamics of the vehicle, mainly by selective actuations on braking and throttling systems.

Those actuations enabled the car manufacturers in the late 1970s and 1980s to effectively put the spotlight into the real improvement of safety. For that purpose, several systems were deployed in cars to, on the one hand, minimizing the injuries in the event of an accident (i.e. passive safety systems), such as pyrotechnical seat belt pretensioners or airbags; and, on the other hand, to alleviate the factors that could cause an accident to occur (i.e. active safety systems).

On this topic, the first, fully-functional driving assistance system (DAS) was successfully developed in 1978 by Bosch with the anti-lock braking system (ABS) [23], which, inherited from the landing gear braking system of airplanes, avoids the tires from blocking in very critical braking situations, preventing them from sliding and, subsequently, reducing the braking distance.

The following evolution of ABS was the traction control system (TCS - 1986) [24], which, using the same sensors as ABS, prevents the wheels from loosing grip when starting march by reducing torque, recovering adherence.

Nevertheless, one of the biggest breakthroughs on DAS is the electronic stability control (ESC - 1995), which, using an electronic gyroscope and the ABS sensors and actuators, can correct the path of a skidding vehicle, improving trajectory following and consequently improving safety [3]. This system is probably one of the major breakthroughs in the field of safety systems since the invention of the 3-point seat belt, and enabled vehicles to present dynamic characteristics of stability that previously could only be achieved by very complex and expensive suspension and geometric designs.

The aforementioned systems, being marketed before as optional equipment in top tier models, have drastically improved automotive safety, leading to the obligation of fitting these systems in every new car in the United States of America (USA) since 2011, and in the European Union (EU) since 2014.

1.1 Driving Tasks Automation

Despite DAS being helpful to ease the consequences of a variety of human-originated handling mistakes, they cannot be considered as task automation at all since they do not act until the mistake and the safety

condition has happened. Additionally, due to these systems having been intended to somehow correct the actuation of the driver on the commands of the vehicle, it means that the driver has to actually operate the car command so that the system can actuate to make corrections.

Task automation, on the other hand, began with the American grand-luxury cars of the 1950s which automated the most inconvenient driving tasks, such as keeping a fixed speed or switching the high beam lamps, by the development of cruise control (CC) and auto-switching high beam lamp systems. Nonetheless, they were very expensive options and very prone to malfunction, and, hence, the adoption was marginal until car microelectronics boomed in the late 20th century. These systems, specially CC, despite helping drivers on reducing mental load and fatigue during long trips, might bring about a variety of dangerous situations, such as rear end collisions. These situations occur because conventional CC systems are not aware of the distance from the preceding vehicle, that is, they only take into account the internal variables of the car.

1.1.1 Automated Driving or Autonomous Driving?

Nowadays, and maybe because of the aggressive marketing campaigns from the car makers, terms of automated driving and autonomous driving are usually confused. This is specially misleading for the current consumers of the highly automated cars and has already caused fatalities.

The main difference between an automated car and an autonomous car is, basically, the complexity of the tasks required to the driver. In this context, the Society of Automotive Engineers (SAE) has defined a scale of 5 automation levels suitable to classify if an automobile is conventional, automated or autonomous [25],[26]. This scale is depicted in Table 1.1.

Coherently with the ideas shown in [25], [26], most of the current cars fitted with the mandatory DAS fit into the 1st level: *driver assistance* since although the driver must carry out all the tasks, including the safety-critical ones, he/she is assisted mainly in traction loss scenarios, but without acceleration/deceleration and steering being performed simultaneously. Due to this definition, even adaptive cruise control (ACC)-fitted cars are placed in this first level.

When both acceleration/deceleration and steering are performed simultaneously in an automated way, the vehicle fits into the 2nd level: *partial automation*. In this level, cars are able to drive along a lane, recognizing speed limits and setting the speed according to the preceding vehicle and the maximum speeds, but they are not able to decide

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	**No Automation**	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	**Driver Assistance**	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	**Partial Automation**	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	**Conditional Automation**	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	**High Automation**	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	**Full Automation**	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

TABLE 1.1: SAE automation levels (from SAE International and J3016).

whether or not to perform an overtaking manoeuvre. Tesla AutoPilot and some Volvo and Mercedes models include automation enough to fit here. However, despite the marketing campaigns, specially for Teslas, there exists the misconception that the car assumes automatically the safety-critical functions. For this reason, some drivers overtrusted on the system, resulting on fatalities [27].

About level 3, no current commercially-available cars fit into this category, while companies such as Google or Uber have covered several millions of kilometers with their prototypes [28]. Cars in this level of automation can control all the environmental aspects and drive according to them in certain environments, such as highways. Human drivers have to perform monitoring tasks though, provided that some glitches may occur or the car may have to face scenarios unforeseen in its programming, and should be ready to intervene if the car requests it.

Level 4 automated cars can go from a starting point to an arrival point in almost every situations, except in very complex traffic situations or severe weather conditions. However, they can handle emergency stop in case of a system failure. Nevertheless, drivers can still handle the car if they feel like to or in off-road, non-mapped scenarios.

Finally, the higher level of automation is the 5th one. In this level, cars are not even fitted with physical controls and the users are simply passengers with the only task of selecting the desired destination. Consequently, they are designed around productivity and comfort of their passengers, looking like working offices or entertainment lounges.

With this hierarchy, the main difference between 4th and 5th level cars is, precisely, that whilst 4th level cars can be considered fully automated, they still need a human driver to solve some complex and infrequent situations, while the 5th level ones are fully autonomous. However, although maximum level of automation reached by production cars is the 2nd one, these cars are not affordable for the general public, so, it can be said that the current level of automation of the traffic is the first one.

1.1.2 Advanced Driving-Assistance Systems (ADAS)

To give response to the challenges of driving tasks automation, external world data-based systems arose as an alternative to provide enhanced safety and comfort functionalities. This type of systems, known as advanced driving-assistance systems (ADAS) [2], are based on complex intelligent sensors that measure external parameters through high-bandwidth signals such as radar, light detection and ranging (LIDAR), or video.

ADAS rely on a continuous stream of data from multiple sensors that measure internal and external variables to provide advanced functionalities [29]. In the same fashion as for safety systems, ADASs can be classified into two categories taking into account their actuation level: passive and active ADAS. The former provides advice or information to the driver. Examples of passive ADAS are blind-spot sensors that use ultrasound sensors to detect obstacles in the blind spot of the rear-view mirrors. Collision-avoidance systems use radar, and seldom LIDAR or video, to detect potential front collisions, warning the driver (front collision warning (FCW)) [30]. Lane-departure warning systems (LDW) [31] detect lane marks by video signals and inform the driver about lane-departure events. Finally, traffic sign identification/recognition systems (TSI/TSR) are able to detect speed limits displaying warnings in the dashboard of the vehicle. The latter, active ADAS, can perform actions on the car, which include systems such as autonomous emergency braking (AEB) [30], that can automatically stop the car. Lane-keeping assistance (LKA) [31] is a step forward from LDW systems; it can correct the trajectory of the vehicle by performing autosteering manoeuvres

by itself to avoid unintended lane changes. Other examples of active ADAS are the above introduced ACC and automatic speed assistants (ASA), which, based on TSI/TSR and jointly acting with global navigation satellite systems (GNSS) signal, automatically apply the speed limit of the road to the car [32].

Radar-Based ADAS

One of the most popular ADAS may be the ACC [33]. ACC is an evolution of the conventional CC system in terms of keeping a fixed speed, however, ACC uses radar technology (previously a very expensive and energy consuming technology) and electronic braking and throttle to keep the distance with the preceding vehicle at a user-defined maximum speed [4]. It is described in Figure 1.1. This development considers many other variables apart from distance with the precedent vehicle and speed, such as the slope of the terrain, being able to actuate on the braking system or switch to a shorter gear in order to keep a constant speed even in descending, steep slopes.

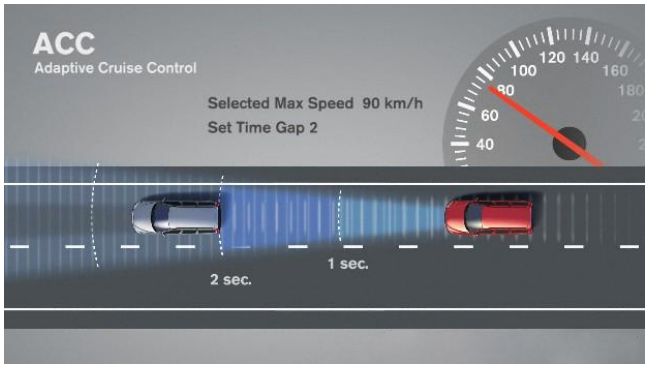


Figure 1.1: Representation of an ACC system. Through radar sensing, the red vehicle measures the time gap with the preceding one and adjust its speed to keep it constant.

Since radars are becoming a more frequent default equipment in cars of all ranges, new functionalities are being developed to take advantage of that feature, such as FCW [30], able to eventually stop the car when a frontal collision is about to happen.

LIDAR-Based ADAS

LIDAR is a pulsed laser beam based range detection method used to determine the distance from the emitter to an object. The distance to

the object is determined by measuring the time it takes to the pulsed beam to return to the emitter after being reflected on a surface. If the LIDAR is mounted on a rotary device, it enables to create a 3D cloud of points that allows to be aware of all the objects on its surroundings. Additionally, LIDAR can detect objects that radar struggles to correctly identify, such as pedestrians, trees or wildlife. This allows to improve the already existing radar-based ADAS, and to predict the behavior of all the agents implied in road transit, potentially increasing safety. Thus, LIDAR is a topic of interest for the development of ADAS [34],[35].

Video-Based ADAS

Another mainstream safety feature is LDW [31], which making use of a front camera, detects the lines delimiting the lane we are driving in, as shown in Figure 1.2. In the case of driving through a lane mark with the turn signals switched off, the system notifies the driver by a tone or a rumble [36]. This feature contributes to reduce the risk of front/side collision, specially in densely populated roads [37].

Additionally, in the line of lane marks detection [38], front cameras can be used to implement lane-tracking auto-steering. Using them in conjunction with the radar sensors needed for ACC [39], they provide some models with automation of the tasks of steering, throttling and braking, noticeably relieving drivers from the most tedious tasks. The ACC radar adds the feature of detecting vehicles beyond the camera's visual field [40],[41], preparing the field to autonomous driving. This builds an intermediate scenario of cars fitted with a kind of auto-pilot instead, such as Tesla AutoPilot [5]. More advanced systems, such as Tesla Navigate on AutoPilot [42], use video jointly with GNSS-based location to provide fully automated door-to-door navigation.

Front cameras contribute to ease the implementation of many other safety systems and amenities which can be based on image processing. Among these systems, we can include glare-free headlights [43], which, coupled with the front camera, automatically switch off a portion of the LED lamps to allow the driver to keep using the high beam lights but avoiding dazzling the incoming drivers. Pedestrian, cyclist and obstacle detection [44] are video-based safety applications being currently deployed in some selected models.

FIGURE 1.2: Video-based ADAS can detect lane marks in almost any visibility condition. Through border detection algorithms and the computation of the perspective, LDW systems can determine if the wheel of the vehicle is stepping over the lane mark by only using a front camera.

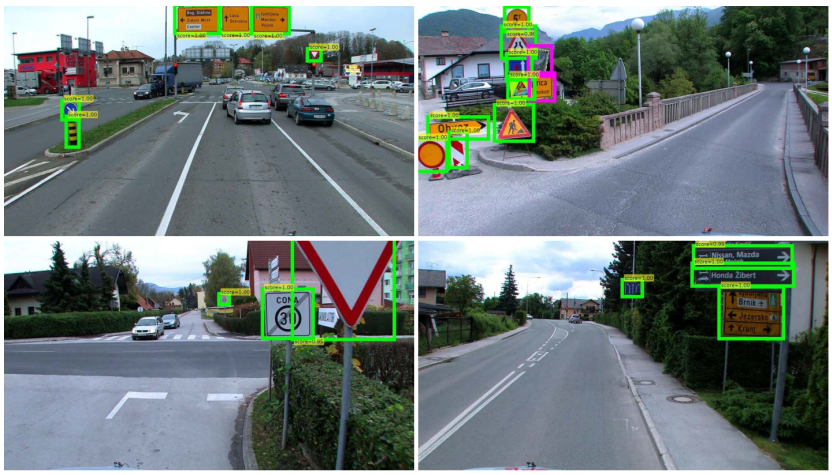


FIGURE 1.3: An example of traffic sign recognition. As shown, these systems can identify the location of one or several traffic signs, display them on the vehicle's dashboard and give advice or perform automated actions according to them.

TSR [32] is another video-based application. TSR relies on a variety of algorithms to fit the vehicle with the ability of recognizing, basically, speed limits, warnings and obligations and displaying them to the driver via on-screen advices (see Figure 1.3). It is worth noting that this system

will be mandatory for all new cars sold in the EU from 2024, under the name of intelligent speed assistant (ISA), and, in addition to the advice function itself, this system will be enabled to restrict the maximum speed of the vehicle according to the corresponding speed limit sign. This is expected to reduce the number of speed-related collisions by the 20% and the number of fatalities by the 30%.

Hyperspectral Imaging

An evolution of the conventional-video based techniques is hyperspectral imaging. This technique divides the visible spectrum in more bands than red-green-blue (RGB), including wavelengths beyond visible light [45]. Hyperspectral imaging is intended to obtain the spectrum of each individual pixel in order to find objects, identify materials or detecting processes [46],[47]. These applications take advantage of the fact that many objects and substances leave a characteristic and unique "spectral signature". With these signatures, it is easier and more reliable to identify specific phenomena.

For these reasons, hyperspectral imaging is a highly interesting field for the automotive industry, since it has the potential to outperform the current applications of rain detection, glare-free headlighting [48], anti-fog lighting, pedestrian detection, and sliding surface detection among others [49]. However, due to the high cost and complexity of the hyperspectral cameras currently marketed [50], it is not feasible to implement this feature in today's production cars, but it is expected to be possible in the future.

1.1.3 Driving Style Personalization

ADAS have been demonstrated to improve safety in cars since they relieve drivers from some of the most safety-critical tasks [51]. The conjunction of all of these intelligent sensors, together with high-speed wireless communications, have allowed car-makers for the first time to develop intelligent vehicles with automated driving systems [52]. Nonetheless, the acceptance of these systems by drivers mainly depends on the engineering factors of the system, predefined by technicians and implied by system functional specifications [53]. However, despite these systems being relatively easy to use and contribute to improving road safety, acceptance by the motorists has been limited [54]. This lack of acceptance is caused by drivers perceiving the car as not as natural and stable as expected, despite it being programmed to be as conservative

and safe as possible, making motorists prone to dismiss this advice or even to disengage the system [55]. This lack of use causes their effect on global sinistrality rates to not be as good as previously foreseen, despite ACC systems having been gradually installed in new production cars and demonstrated to contribute in reducing accidents' severity, even eliminating them [56].

Hence, it seems reasonable that if these systems showed a more adaptable behavior, they would be considerably more accepted by the users, and, consequently, their adoption would noticeably increase [10]. With that objective in mind, car-makers have introduced a slight level of ADAS customization, allowing users to manually select a variety of pre-engineered parameters from a knob in the steering wheel or a lever in the steering column. These selection of parameters is particularly available for ACC systems, where the time interval with the preceding vehicle, known as time headway (THW) can be selected from a set of memory-stored parameters. Nevertheless, not many drivers appreciate this option, and, particularly when drivers are not familiarized with these control systems, they are particularly hard to operate, so, this personalization option is often set aside.

For that reason, if this personalization could be automatically performed in some fashion, with no driver intervention at all, these systems would be much easier to operate, and, consequently, those drivers set aside by their complexity would be encouraged to use them [10], increasing with that the level of adoption. Regarding trust, several studies point out that ADAS that mimic the behavior of real human drivers tend to be more trusted by the users than more artificial ones, no matter if those are designed with better-than-human standards [57]

1.1.4 Eco-Driving

Pollution has been an issue since the popularization of the automobile, and it has become an even more concerning point in recent years due to the global warming, and because of the harmful effects on people's health and lives [58].

Jointly with the appearing of the first DAS, it was in the 1980s when pollution became a main issue for both the public and for the developed countries' health systems [59], [60]. Several pieces of research observed that pollution mainly contributed to climate change and worsened the health condition of people suffering from breathing system pathologies in cities with high emission levels [61].

For those reasons, alternative energy-based means of transportation, which mainly rely on electric energy, emerged as a way to mitigate those harmful effects. However, because electricity-powered automobiles needed an evolution in both power storage and power train technologies, hybrid cars arose as a trade-off solution between electric and petrol cars. Despite hybrid automobiles still being a polluting means of transportation, the addition of electric engines into their power train contributes to increasing their power efficiency and, consequently, their eco-friendliness [62]. Nevertheless, the outstanding advances in battery technology and energy converters in recent years have changed the paradigm of non-polluting transportation, since they made electric cars a market reality [63], [64]. Since the early 2010s, electric cars' sales have consistently risen in countries, such as Norway, where this kind of automobile is mainly sold [65]. These cars, though, still have several problems, such as the impact of their complete life cycle. Though advances in recycling several components, such as batteries [66], have been made, the use of energy-intensive materials causes a noticeable impact on greenhouse gasses (GHG) emissions [64], [67]. On the other hand, at least for the current decade, fuel-powered cars are expected to continue as the mobility standard [68], [69]. Thus, the development of systems to change low-efficiency driving behaviors, such as driving at high regimes, seems to be a good alternative in efforts to increase the eco-friendliness of the current fleet of fuel-powered cars.

Driving Style and Eco-Driving

On the other hand, as found in several studies, the manner motorists operate the throttle and brake pedals, their desired rate of acceleration, speed control, and control stability play a major role in fuel consumption, regardless of the driven vehicle. Thus, we can learn how some drivers have a higher energy cost than others by studying the impact of their driving behavior on fuel consumption, thus helping high-energy-cost drivers to achieve energy-efficient driving style (DS). Factors, such as personality, ability and skills, attitudes, perceptions, socio-economic characteristics, age, gender, and experience, among others, have been identified to be related to riskier and more aggressive driving events, such as extreme accelerations, excess revolutions per minute (RPM), extreme braking, and hard starts, events that cause high fuel consumption [6]. In consequence, eco-driving emerges as a set of rules and parameters to be followed with the aim of improving the energy efficiency while reducing the GHG emissions [70]. Hence, any autonomous car

related development should take into account these rules to minimize the impact of pollution.

1.1.5 Ride Comfort

The paradigm of transportation is experiencing technological advances, causing new challenges to emerge [71]. The breakthrough of the autonomous driving scenario aims to relieve motorists from the tasks and risks of driving a car, with the subsequent improvement of safety and perceived comfort [72], [73]. Thus, while the former is expected to be clearly enhanced since most of the road accident-associated injuries and fatalities depend on the human factor [74], mostly due to recklessness and distractions [75], the latter requires further attention. This change of the driving scenario will turn drivers into mere passengers, and, hence, new challenges to their wellness will appear [76], being passenger comfort one of them.

There exist many factors that contribute to the global ride comfort. These aspects are derived from both the environment within the vehicle (e.g. smell, temperature, humidity, etc.), as well as from the characteristics of the passenger (e.g. gender, age, health conditions, etc.) along with his/her behavior (keeping or not his/her gaze on the road). Besides, vibrations affect the overall comfort perception. Several pieces of research link perceived distress, motion sickness and the frequency components of the vibrations with the resonance frequencies of the organs of the human body [77], [78]. Although some vibrations are caused by the constructive characteristics of either the roadway or the vehicle itself including the handling of the automobile, those derived from the driving behavior and handling skills of the driver have a noticeable influence in compromising ride comfort.

Motion sickness is the most severe vibration-originated comfort condition, causing from mild effects such as cold sweating and dizziness to nausea and vomit, which strongly impair passengers [79], [80]. This condition, highly related to low-frequency vibrations, happens because the brain perceives a mismatch between the expected movements and the real accelerations detected by the vestibular apparatus of the inner ear [81]. For that reason, the vast majority of drivers do not get motion-sick, since they receive a more complete range of stimuli that helps the brain to have a more accurate insight into the actual dynamics of the car [82]. Motion sickness is very frequent among passengers due to the fact of not perceiving as many stimuli as drivers do, consequently making the sensory conflict between the sight and the hearing senses more notice-

able. It is worth noting that these effects are worsened as passengers get involved into secondary tasks, such as surfing the internet, reading or working on a PC [83].

Driving Style and Ride Comfort

Several works have been carried out on DS as a main conditionant for both the driver and the occupants in terms such as lack of comfort, energy efficiency and even the increase on the risk of accident [84]. The aforementioned behaviors are mainly influenced by the experience/inexperience of the drivers, as well as his/her age, gender and general health state [85].

In the same fashion of eco-driving, if we assume that most of the discomfort causing vibrations are derived from each individual's DS [7], drivers might be classified among a variety of DSs, cycles and scenarios, regarding their driving patterns' characteristics [12]. Consequently, since the autonomous driving paradigm encourages passengers to perform the secondary tasks, they are expected be more prone to suffer motion sickness [86], [87], becoming a public health concern, and, thus, any autonomous driving-intended development should consider the impact of vibrations on passengers' discomfort.

1.2 Machine Learning Algorithms

Machine learning (ML) is a term coined in the year 1959 at the IBM company [88]. It refers to giving the computation systems the ability to learn how to perform a task without the need of being directly programmed to do so. This area of Computer Science evolved from the study of pattern recognition and computational learning theory in artificial intelligence (AI) with the aim of allowing the computers learn automatically with no human intervention. After the learning procedure has been completed, the machine is able to carry out predictions, in short, it becomes able to generalize the previously learned "concepts" or behaviors.

The main motivation of using this kind of techniques in the field of driving automation are the situations to be dealt with when deploying ADAS, which require the processing of big amounts of data that shows strong non-linearities and complex behaviors, as well as a very high level of uncertainty. Due to these characteristics, classic computing techniques may sometimes not be appropriate because they require their inputs and outputs to be strongly determined, otherwise, the system could fail.

There exist several manners of classifying ML algorithms [89], [90] attending to a variety of criteria. One of the most popular criteria is the availability of labels for training data. In case of not-labeled training data (data which are not previously classified), the algorithms used to classify or approximate them are known as unsupervised ML methods [91], [92], while the supervised ML methods [93] are focused on labeled training data. In the following description, the methods utilized in this work are remarked in bold typeface.

1.2.1 Supervised Algorithms

Supervised ML groups the techniques that use an input training dataset with an associated response values (labels) dataset to make predictions. The supervised learning algorithm searches for building a model able to fit the predicted outputs for the input training data with the labels of the response values dataset. Once the model is properly trained, it is able to predict the response values of a new input dataset of the same type as the training one. These techniques can be used for classification and regression:

- *Support vector machines (SVM)* [94]: they are a kind of binary classifier. They split the input data classes into several spaces as broad as possible by means of a separating hyperplane defined in terms of the vector joining the nearest points of the adjacent spaces. This vector is known as the support vector. Using, among others, Gaussian kernels is a valid approach as well.

- ***Artificial neural networks (ANN)*** [8]: emulating the structures in biological brains, they use a set of interconnected nodes to model complex behaviors. These networks can rely on several topologies, regarding either the arrangement of the elements of nodes or the disposition of the nodes themselves in layers. Each node modifies the state of activation of the adjacent ones by the manipulation of their input values by a set of weights. Features of the ANN algorithms will be detailed in Section 1.2.3.

- *Naïve Bayes classifier* [95]: these classifiers do not take for granted relationships between characteristics of the input data that, for the human experience, are known to be implicit for each object. This characteristic is highly desirable because each probability threshold can be tuned individually in order to provide good classification rates.

- *Decision trees* [96]: all the possibilities are contained in a tree-type structure. Thus, depending on the characteristics of each sample, some branches are activated, leading to the class it belongs to.

- *Discriminant analysis* [97]: is a method used to find a linear combination of features to characterize or separate the input objects into two or more different classes. These linear combinations can be interpreted as dimension reduction steps that help to simplify further classification procedures.

- ***Nearest neighbors (k-NN)*** [98]: the input consists of the k closest training examples in the feature space. Thus, for classification purposes, the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors.

- *Regression models* [99]–[101]: these classic algorithms are based on modeling the relationship between two or more variables. Thus, they link the value of a scalar response (dependent variable) with one or more explanatory variables (independent variables). Basically, they predict the expected value of the output of a black-box system by computing the combination of its input variables.

- ***Fuzzy logic-based systems*** [102]: based on human logic, in which not only must a logical statement be true or false, but also it can take intermediate values. For that purpose, first a fuzzyfication of the inputs, which consists on assigning a value within the interval $[0, 1]$, has to be performed according to a fuzzy set. After that, the fuzzyfied values are operated by fuzzy operators and, finally, IF - THEN linguistic rules are used to map the computed values into the outputs, that are defuzzified again from the range $[0, 1]$. More details of these systems are provided in Section 1.2.4.

1.2.2 Unsupervised Algorithms

Unsupervised ML groups the techniques able to infer patterns among non-labeled data. Thus, the main goal of this type of learning methods is modeling the implicit relationships between samples. Differing from supervised ML, there is not a singular-correct solution and there is not real teaching, but rather algorithms are left on their own to discover these hidden structures. These methods can be grouped into two classes: clustering and association.

- **Clustering:** in these problems, data is analyzed to discover common patterns among the experiments, such as grouping drivers by their driving behavior. Thus, objects in the same group (cluster) are more similar to each other than to those in other groups. Into this category, the most common algorithms are:
  - *Hierarchical agglomerative clustering (HAC)* [103]: it builds a tree (hierarchy) of clusters attending to the measure of dissimilarity between sets of data. Clusters are usually presented on a dendrogram.
  - ***k-means clustering*** [104]: it consists on building k clusters among which, observations will be distributed. Each observation will be located in the cluster with the nearest mean value.
  - *Gaussian mixture models (GMM)* [105]: these models are based on the probability distribution of the observations. In these particular models, the probabilities are modelled by means of the Gaussian (normal) distribution.
  - ***Self-organizing maps (SOM)*** [106]: they use ANNs to learn the topology and distribution of the data by producing a low-dimensional representation map. They differ from other ANN topologies by the fact that they use a neighborhood function to preserve the properties of the input space. These ANNs, with a proper size, allow to extract characteristics of the input space on the map itself, enabling them as a powerful clustering tool. A further insight on this topology is shown in Section 1.2.5.
  - *Hidden Markov models (HMM)* [107]: assume that the system intended to be modeled is a Markov chain with hidden states. It is worth to remark that a Markov chain is an stochastic model describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event.
- **Association:** these methods are intended to find correlations between large datasets, identifying strong rules discovered in databases using some measures of interest [108]. Some well-known association algorithms are Apriori [109], Eclat [110] and FP-Growth [111].

1.2.3 Artificial Neural Networks (ANNs)

An artificial neural network is set of interconnected nodes, known as artificial neurons, able to model and represent high complexity problems. In contrast with human brain (with up to 100 billions of neurons) [112], ANNs rely on a more reduced set, ranging from tens to a few millions depending on each topology.

The Artificial Neuron

As displayed in Figure 1.4, there is a parallelism between the structure of a biological neuron and that of an artificial one. These similitudes occur in the following way [112]:

- **Dendrites:** weighted inputs.
- **Nucleus:** adder and activation function.
- **Axon:** output of the activation function to the weighted inputs (dentrites) of other neurons.

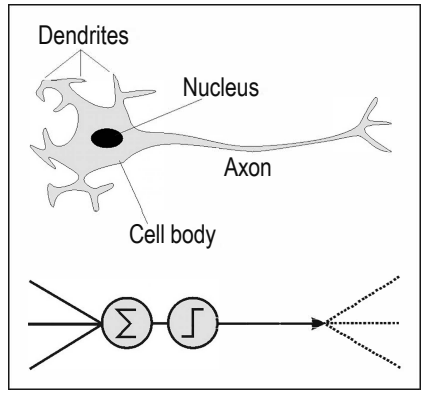


FIGURE 1.4: Structure of a biological neuron vs. an artificial neuron.

For each input, a weight is applied, then, each weighted input is added by the adder, and the result is trimmed by an activation function. These calculi are performed in the subsequent manner:

$$H(\mathbf{w}_i, \mathbf{b}_i) = g\left(\sum_{j=1}^{N} x_j w_j + b_i\right), \tag{1.1}$$

where x_j is each input term, w_j its associated weighting factor, b_i is the bias constant of each neuron i, $g(x)$ an activation function and $j =$

1, ..., N the number of inputs. Thus, in matrix form, given a set of weights $\mathbf{w}_i = (w_1, ..., w_N)^T$ for neuron i, a vector of inputs $\mathbf{x} = (x_1, ..., x_N)$ and an activation function $g(x)$, the output of neuron i $H(\mathbf{w}_i)$ of equation 1.1 is reformulated such that:

$$H(\mathbf{w}_i, b_i) = g\left(\mathbf{x} \cdot \mathbf{w}_i + b_i\right) \tag{1.2}$$

Artificial Neural Network Topologies

Once the mathematical model of the artificial neuron is defined, they must be interconnected in order to solve a variety of complex tasks. Generally, ANNs' structure is organized into layers. Therefore, we can distinguish two main types of ANNs depending on how their layers are organized: single layer (Figure 1.5) [113] and multi-layer (Figure 1.6) [114]. Independently of their topology, each neuron from a given layer is connected to all the neurons on its preceding and subsequent layers. However, the names of these topologies may be misleading since they refer to the amount of layers between the input layer and the output layer (hidden layers). Thus, a single-layer feedforward ANN (SLFN), also known as shallow neural network, has actually 3 layers: input layer, hidden layer and output layer; while multi-layer ANNs have more than 3 layers of nodes: input layer, 2 or more hidden layers and output layer.

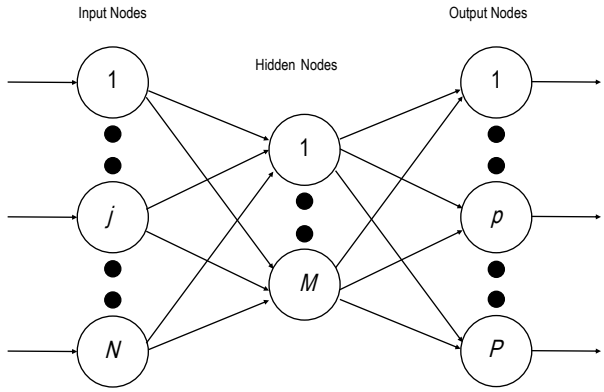


Figure 1.5: Structure of a single-layer artificial neural network (SLFN) or shallow neural network with N input nodes, M hidden nodes and P output nodes.

According to Figure 1.5 and Equation 1.2, the input nodes correspond with the dendritical input vector $\mathbf{x}$ and the outputs of the hidden nodes are grouped in a matrix $\mathbf{H}(\mathbf{w_i}, \mathbf{b_i})$ with a number of hidden nodes

$i = 1, ..., M$. Finally, for the output nodes, a vector $\boldsymbol{\beta} = (\beta_1, ..., \beta_M)^T$ of output weights is defined. Once the neural network is described on its matrix form, the output product of the ANN is computed in the subsequent manner:

$$\mathbf{Y} = \mathbf{H}(\mathbf{w}_i, ..., \mathbf{w}_M; b_i, ..., b_M) \cdot \boldsymbol{\beta} \tag{1.3}$$

Multi-layer neural networks, also known as deep neural networks, (see Figure 1.6) rely on hidden neurons organized in multiple layers. Each neuron in a layer is connected with all the neurons in the previous layer. Thus, a multi-layer neural network consists of a set of input nodes, multiple hidden neurons organized in multiple layers, and a set of output nodes.

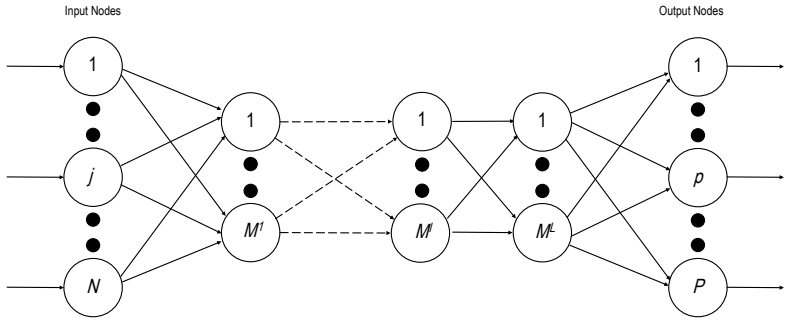


Figure 1.6: Structure of a multi-layer artificial neural network with N input nodes, L hidden layers with M^l hidden nodes each and P output nodes.

These ANNs' weights can be tuned by means of a wide variety of methods, while the most widely used one is the gradient-descent back-propagation (BP) algorithm.

Gradient-Descent Backpropagation

This algorithm is based on propagating the error, i.e., the difference of the desired output with the actual, in the reverse direction of the ANN dataflow to calculate a cost function that needs to be optimized to find its minimum. [115]

The approach stated in the latter paragraph is contained in the traditional, well-known *gradient-based solution of SLFN* on which, intending to train a SLFN, and given a set of weights and biases $\mathbf{W}(\mathbf{w}_i, \beta_i, b_i)$, with $i = 1, ..., M$, the proposed objective is to find a specific set of weights

and biases $(\hat{\mathbf{w}}_i, \hat{\beta}, \hat{b}_i)$ such that:

$$\mathbf{T} - \mathbf{H}\left(\hat{\mathbf{w}}_i, ..., \hat{\mathbf{w}}_M, \hat{b}_i, ..., \hat{b}_M\right)\hat{\boldsymbol{\beta}} = \min_{\mathbf{w}_i, \boldsymbol{\beta}, b_i} \mathbf{T} - \mathbf{H}\left(\mathbf{w}_i, ..., \mathbf{w}_M, b_i, ..., b_M\right)\boldsymbol{\beta}, \tag{1.4}$$

where $\mathbf{T} - \mathbf{H}\beta$ to be minimized is a cost function with an expanded form:

$$E = \frac{1}{2}\sum_{j=1}^{N}\left(\mathbf{t}_j - \sum_{i=1}^{M}\beta_i g\left(\mathbf{w}_i \cdot \mathbf{x}_j + b_i\right)\right)^2. \tag{1.5}$$

$\mathbf{H}$ is usually unknown, so, gradient-based learning algorithms are generally used to find the minimum of $\mathbf{T} - \mathbf{H}\beta$. The set $\mathbf{W}(\mathbf{w}_i, \beta_i, b_i)$, with $i = 1, ..., M$ of weights and biases is iteratively updated in the following way:

$$\mathbf{W}_k = \mathbf{W}_{k-1} - \eta\frac{\partial E\left(\mathbf{W}\right)}{\partial \mathbf{W}}, \tag{1.6}$$

where η is a learning rate. This method, known as BP learning algorithm, is able to compute gradients efficiently by propagation from the output to the input.

Nevertheless, although the efficiency in the computation of the gradients is the main advantage of the BP algorithm, there are several issues that must be considered before deploying this technique such as the adjustment of the learning rate η. For instance, since a too large learning rate may destabilize the convergence of the cost function and a very small η can cause the algorithm to slow down excessively, the adequate value of the rate should be a trade-off solution between speed and stability [116]. In addition, since the iterative updating algorithm only uses the first derivative of the cost function, calculations might stop if a local minimum is found far enough from the global minimum, which is an undesirable behavior.

Extreme Learning Machines (ELM)

As stated in the previous section, gradient-descent BP training algorithms are an utterly consolidated form of optimizing cost functions in order to tune the parameters of a variety of ANN topologies. However, they have some critical problems, such as their sensitivity to local minima and the difficulty of adjusting the learning rate η to its optimal value. Although the mathematical problem to be solved is still the same, the approach to finding a minimal solution must be changed in order to reduce, or even, eliminate the influence of the challenges indicated above.

The new approach, a type of SLFN known as extreme learning machine (ELM) [117], pursues the same objective as BP on SLFNs, but instead of adjusting the weights and biases of all layers, $\mathbf{w}_i$ and b_i are initialized randomly, and only the weights β of the output layer are tuned. This drastically simplifies the mathematical problem to the solution of a system of linear equations, reducing training times. Additionally, it has been observed that not only does ELM reduce SLFN training times, but also improves the generalization performance with respect to BP-trained SLFNs [118].

Thus, only a set of output nodes' weights $\hat{\boldsymbol{\beta}}$ must be found such that Equation 1.4 is simplified to:

$$\mathbf{T} - \mathbf{H}\left(\mathbf{w}_i, ..., \mathbf{w}_M; b_1, ..., b_M\right)\hat{\boldsymbol{\beta}} = \min_{\boldsymbol{\beta}} \mathbf{T} - \mathbf{H}\left(\mathbf{w}_i, ..., \mathbf{w}_M; b_1, ..., b_M\right)\boldsymbol{\beta}. \tag{1.7}$$

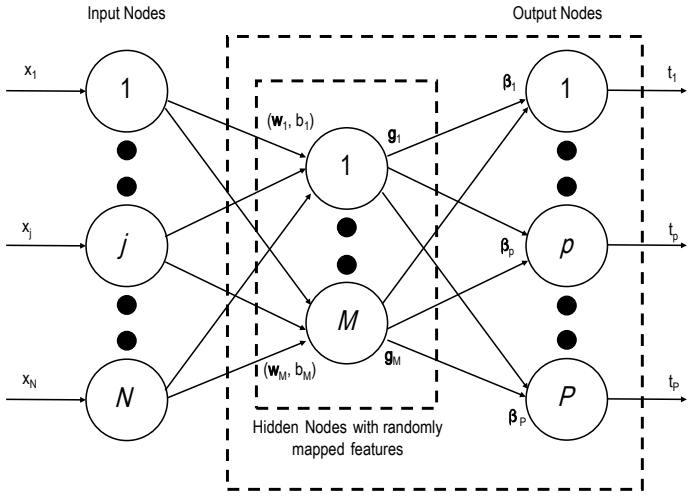


Figure 1.7: ELMs are SLFNs with the particularity of having random weights $\boldsymbol{w}_i$ and biases b_i for the hidden layer.

Extreme Learning Machine training algorithm

This section presents the algorithm for computing the matrix β of output nodes' weights in Equation 1.7.

Thus, given a number of samples K, a number of hidden nodes M, a number of input nodes N, a number of output nodes P, and an activation function $g(x)$, a training set $\aleph = (\mathbf{X}, \mathbf{T})$ is defined, such

that $\mathbf{X} = (\mathbf{x}_k, ..., \mathbf{x}_K)$ and $\mathbf{T} = (\mathbf{t}_k, ..., \mathbf{t}_K)$, with $\mathbf{x}_k = (x_1, ..., x_N)$ and $\mathbf{t}_k = (t_1, ..., t_P)$.

Once all variables taking part in this process are defined, the algorithm can be displayed:

1. Initialize matrices $\mathbf{w}$ and $\mathbf{b}$ with random values.
2. Calculate the hidden layer output matrix $\mathbf{H}(\mathbf{w}_i, b_i)$ with $i = 1, ..., M$ as follows:
$$\mathbf{H}(\mathbf{w}_i, b_i) = g(\mathbf{X} \cdot \mathbf{w} + \mathbf{b}) \tag{1.8}$$
3. Compute the output weight matrix β as:
$$\beta = \mathbf{H}^{\dagger}\mathbf{T} \tag{1.9}$$
To obtain an array $\boldsymbol{\beta}$ such that:
$$\mathbf{H}\boldsymbol{\beta} = \mathbf{T} \tag{1.10}$$
with $\mathbf{H}^{\dagger}$ the Moore-Penrose generalized inverse matrix of $\mathbf{H}$ [117].

1.2.4 Fuzzy Systems

Fuzzy systems are based on the fuzzy set theory proposed by Zadeh in the 1960s [9]. These are systems that can deal with imprecision, vagueness or incomplete information. Fuzzy logic is used as a means of representing and manipulating imprecise data. Fuzzy logic provides an inference morphology that enables approximate human reasoning capabilities to be applied to knowledge-based systems [119]. The theory of fuzzy logic provides a mathematical framework for capturing the uncertainties associated with human cognitive processes, such as thinking and reasoning. Conventional approaches to knowledge representation lack the means for representing the meaning of fuzzy concepts. As a consequence, the approaches based on first order logic and classical probability theory do not provide an appropriate conceptual framework for dealing with the representation of common sense knowledge, as such knowledge is, by its nature, both lexically imprecise and non-categorical.

Fuzzy Sets and Membership Functions

If X is a collection of objects, each denoted generally by x, then, a fuzzy set A in X is defined as a set of ordered pairs:

$$A = \{(x, \mu_A(x)) \,|\, x \in X\}, \tag{1.11}$$

where $\mu_A(x)$ is the membership function (MF) for the fuzzy set A. The MF maps each element of X to a membership grade between 0 and 1.

There are a number of common MFs as shown in Figure 1.8:

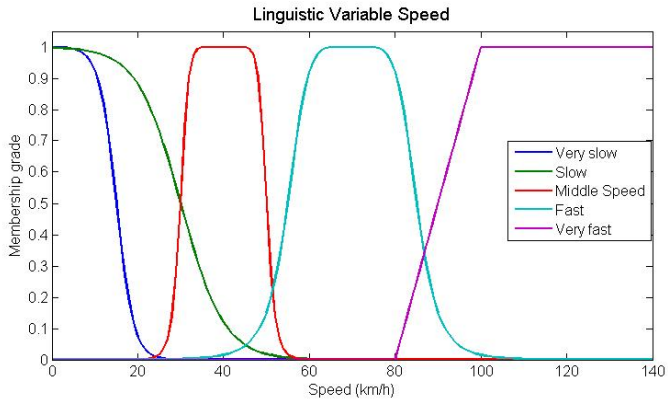


FIGURE 1.8: Typical MFs, such as generalized bells (red and light blue), sigmoidals (dark blue and green) or trapezoidal (purple), are displayed. In this case, they correspond to of a term set $T(speed)$.

Linguistic Variables

Linguistic variables are those variables whose values are not numbers but words or sentences in a natural or artificial language [120]. The motivation for the use of words or sentences rather than numbers is that linguistic characterizations are, in general, less specific than numerical ones. For example, "speed" is interpreted as a linguistic variable, which can take the values as "slow", "fast", "very fast", and so on.

A linguistic variable is characterized by a quintuple $(x, T(x), X, G, \mu)$ where x is the name of the variable; $T(x)$ is the term set of x (that is, the set of its linguistic values); X is the universe of discourse; G is a syntactic rule that generates the terms in $T(x)$; and μ is a semantic rule which associates terms in $T(x)$ to fuzzy sets in X. In the previous example, if "speed" is interpreted as a linguistic variable, then its term set $T(speed)$ could be:

$$\begin{aligned} T(speed) = \{&slow,\ fast,\ very\ slow,\ not\ very\ fast,\\ &too\ fast,\ not\ slow,\ not\ very\ slow\ and\\ &not\ very\ fast\}, \end{aligned} \tag{1.12}$$

where each term in $T(speed)$ is characterized by a fuzzy set of a universe of discourse X, as for example $X = [0, 120]$, as shown in Figure 1.8. In the expression "speed is fast", the linguistic value "fast" is applied to the linguistic variable "speed". On the other hand, in the expression "speed= 85", speed is interpreted as a numerical variable, assigning a numerical value of 85. The syntactic rule refers to the way the linguistic values in the term set $T(speed)$ are generated. The semantic rule defines the MF of each linguistic value of the term set. The five MFs for defining the linguistic variable "speed" are shown in Figure 1.8.

Fuzzy Rules

For a particular case of 2 linguistic variables, an "IF-THEN" fuzzy rule is an expression of the form:

$$\text{IF } x_1 \text{ IS } A \text{ AND } x_2 \text{ IS } B \text{ THEN } y \text{ IS } C, \tag{1.13}$$

where A and B are linguistic values defined by fuzzy sets. Typically, "x_1 IS A AND x_2 IS B" is known as the antecedent or premise, while "y IS C" is the consequence or conclusion. Several example of fuzzy rules are:

- IF speed is "very fast", THEN driving is "dangerous".
- IF temperature is "cold" THEN heater is "high".
- IF temperature is "high" AND humidity is "high" THEN room is "hot".

Fuzzy Inference Systems

Fuzzy inference systems (FISs) are based on the concepts of fuzzy set theory. Fuzzy inference is the process of formulating mapping from a given input to an output using fuzzy logic. Consider a set of rules such that:

$$\begin{aligned} R_i : \text{IF } & x_1 \text{ IS } A_{1i}(x_1) \text{ AND } x_2 \text{ IS } A_{2i}(x_2) \text{ AND } \cdots x_n \text{ IS } A_{ni}(x_n) \\ & \text{THEN } y \text{ IS } c_i, \end{aligned} \tag{1.14}$$

where R_i is the i-th rule ($1 \leq i \leq M$), x_j ($1 \leq j \leq N$) are input variables, y is the output, c_i is a constant consequent, and $A_{ji}(x_j)$ are linguistic labels, each one associated with an MF, $\mu_{ji}(x_j)$. In one of the most utilized FISs, the zero-order Takagi–Sugeno fuzzy model [121], the

inference procedure used to derive the conclusion for a specific input $\mathbf{x} = \left(x_1^0, x_2^0, \cdots, x_N^0\right)$ consists of two main steps. First, the firing strength or weight w_i of each rule is calculated as

$$w_i = \prod_{j=1}^{N} \mu_{ji}\left(x_j^0\right). \tag{1.15}$$

Next, overall inference result y is obtained by means of the weighted average of the consequents

$$y = \frac{\sum_{i=1}^{M} w_i c_i}{\sum_{i=1}^{M} w_i} = \frac{Num}{Den}. \tag{1.16}$$

Neuro-Fuzzy Systems

The term neuro-fuzzy systems (NFs) refers to a combination of techniques from neural networks and fuzzy systems [122]. Ever since fuzzy systems were applied industrially, the community has perceived that the development of a fuzzy system with good performance is not an easy task. The problem of finding MFs and appropriate rules is frequently a tiring process of trial and error. Therefore the idea of applying learning algorithms to fuzzy systems was considered early on. One of the first studies that proposed a combination of neural network learning methods with fuzzy system concepts was published in 1985 [123]. Several other approaches date from the beginning of the 1990s, including those of Jang [124]–[126], Lin and Lee in 1991 [127], Berenji in 1992 [128], and Nauck in 1993 [129], [130]. The majority of the first applications were in the field of process control. Gradually, it began to be applied to all areas of knowledge, including data analysis, data classification, imperfection detection, and support to decision-making. The aim of NF systems is to combine the advantages of ANNs and FISs. Knowledge of the system is expressed as a linguistic fuzzy relationship while neural network learning schemes, capable of learning nonlinear mappings of numerical data, are used to train the system by tuning the parameters of the MFs. Furthermore, an NF system is capable of extracting fuzzy knowledge from numerical data. In this particular work, the well-known adaptive neuro-fuzzy inference system (ANFIS) algorithm proposed by Jang in 1993 was used [125].

On the other hand, as detailed in Figure 1.8, MFs μ_{ij} associated to these labels can be, for example, bell-shaped functions, which are defined as follows,

$$\mu_{ji}(x; a, b, e) = \frac{1}{1 + \left|\frac{x-e}{a}\right|^{2b}}, \tag{1.17}$$

where a defines the width of the MF, b is the steepness of the curves at each side of the center plateau, and e is the center of the function.

To train the ANFIS, a combination of a least-squares estimator (LSE) and a gradient-descent method were used. Each epoch of this learning process was composed of a forward pass and a backward pass. In the forward pass, consequent parameters, c_i, were identified by the LSE method, and in the backward pass, antecedent parameters a, b, and e were updated by the gradient-descent method.

1.2.5 Self-Organizing Maps (SOMs)

SOMs are a particular type of unsupervised ANN suitable for clustering and visualization of complex multi-dimensional data [131]–[133]. It defines a mapping or projection from a set of high-dimensional input data onto a regular low-dimensional discrete grid. This grid, which is known as a feature map, preserves the principal features of the input data. Unlike conventional feed-forward ANNs, which are generally trained using the supervised back-propagation learning algorithm, SOMs are trained through an unsupervised strategy; that is to say, in a SOM, there are no known target outputs that are associated with the input samples. On the contrary, during the training phase, SOMs process a collection of data, only input data, in order to discover unknown clusters hidden in the data.

The architecture of a SOM consists of a single layer neural network with neurons set along a regular grid: the output layer. Each input to the SOM is fully connected to every neuron in the output layer. Figure 1.9 depicts two typical two-dimensional output layers with $M = 25$ neurons set along a rectangular grid (a) and a hexagonal grid (b). Although most of the SOMs are based on a two-dimensional grid, many applications also use three or more dimensional spaces.

Each neuron in the output layer has a double representation: an N-dimensional vector $\mathbf{m}_i$, known as the weight vector, and its position in the grid. The number of components of the vector is equal to the number of input features N. Figure 1.10 shows the structure of the SOM that

was used in this work, and it is based on a two-dimensional hexagonal grid.

$$\mathbf{m}_i = (m_{i1}, m_{i2}, \cdots, m_{iN}), 1 \leq i \leq M. \tag{1.18}$$

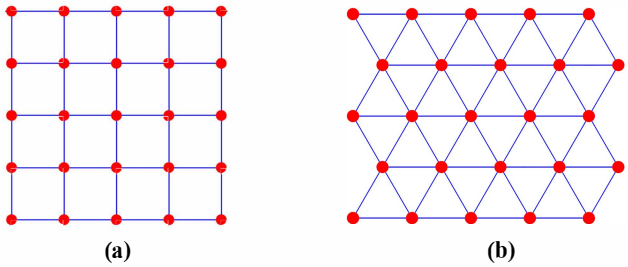


FIGURE 1.9: Typical SOM topologies: a rectangular output grid (**a**) and a hexagonal output grid (**b**).

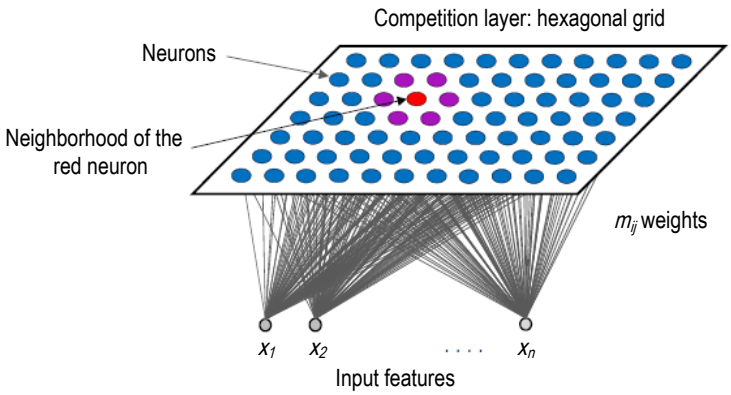


FIGURE 1.10: Typical SOM topology: structure of an N-input SOM, $\mathbf{x} = (x_1, x_2, \cdots, x_N)$, and M=77 output neurons distributed into a 7x11 two-dimensional hexagonal grid.

Clustering a dataset by means of a SOM paradigm is carried out using a two-level approach: first, the SOM is trained; afterwards, the prototype vectors of the SOM are clustered. The advantage of using this approach, instead of clustering the data directly, is that the computational load decreases considerably, which makes it suitable for analyzing different pre-processing and initialization strategies in a short time. In addition, a two-level approach is less sensitive to noise than a single-level strategy. Obviously, this solution is only valid if the clusters found using the SOM are representative of the original data [134].

Training Self-Organizing Maps

First, an initial weight is assigned to each neuron connection. There are simple initialization approaches, such as using random numbers or using input samples randomly selected from the dataset. Although sophisticated algorithms that are based on data analysis (e.g., principal component analysis (PCA)) can also be used, it was observed that random initialization performed rather well for non-linear datasets [135].

Thus, in each training step, one input sample $\mathbf{x}^k = (x_1^k, x_2^k, \cdots, x_N^k)$, $1 \leq k \leq K$, from the dataset is chosen randomly, and the distances between this sample and all of the neuron weights of the SOM are computed. The most popular distance measure in real applications is the Euclidean distance $\|\cdot\|$.

$$\left\|\mathbf{x}^k - \mathbf{m}_i\right\|^2 = \sum_{j=1}^{N} (\mathbf{x}_j^k - \mathbf{m}_{ij})^2. \tag{1.19}$$

The output neuron whose weight vector is closest to the k-th input sample, according to Equation (1.19), is the best matching unit (BMU) or the winner neuron, which is usually denoted by c.

$$\left\|\mathbf{x}^k - \mathbf{m}_c^k\right\| = \min_i \left\|\mathbf{x}^k - \mathbf{m}_i\right\|. \tag{1.20}$$

The BMU is used to update the weight vectors of the SOM. In this process, the BMU and its neighbors are moved towards the k-th input sample, bringing them closer. For each neuron of the SOM, the weight vector is updated, as follows:

$$\mathbf{m}_i(n+1) = \mathbf{m}_i(n) + \alpha(n) h_{ci}(n) \left\|\mathbf{x}^k(n) - \mathbf{m}_i(n)\right\| \tag{1.21}$$

where n denotes the iteration step, $\mathbf{x}^k(n)$ is an input sample randomly selected from the training dataset at iteration n, $h_{ci}(n)$ is a neighborhood function or kernel around the BMU, and $\alpha(n)$ is the learning rate. Both $\alpha(n)$ and $h_{ci}(n)$ are decreasing functions approaching zero with each iteration in order to guarantee the convergence and stability of the training process. The neighborhood function specifies how much the i-th neuron has to move toward the input sample at iteration step n. It is a radial basis function, usually a Gaussian function that is centered at the BMU:

$$h_{ci}(n) = exp\left(-\frac{\left\|\mathbf{m}_c(n) - \mathbf{m}_i(n)\right\|^2}{2\sigma^2(n)}\right). \tag{1.22}$$

Equation (1.22) defines the region of influence of the current input sample, with $\sigma^2(n)$ being the neighborhood radius.

Concerning the learning rate in Equation (1.21), different functions have been proposed, such as linear or exponential functions. In this work, a decaying exponential function, with initial learning rate α_0, has been selected:

$$\alpha(n) = \alpha_0 e^{\left(-\frac{n}{T}\right)}, \tag{1.23}$$

where T is the number of iterations or training length. It is worth noting that Equation (1.21) is also suitable for the online training of the SOM by substituting n by t (i.e., discrete time).

In sum, the sequential training of SOMs involves the following steps:

1. *Initialization.* Initial weights are randomly selected from the dataset.

2. *Competition.* For a randomly selected input sample, all of the neurons in the output layer compete with each other to be the BMU (Equation (1.20)). The neuron that is closer to the input sample is the winner.

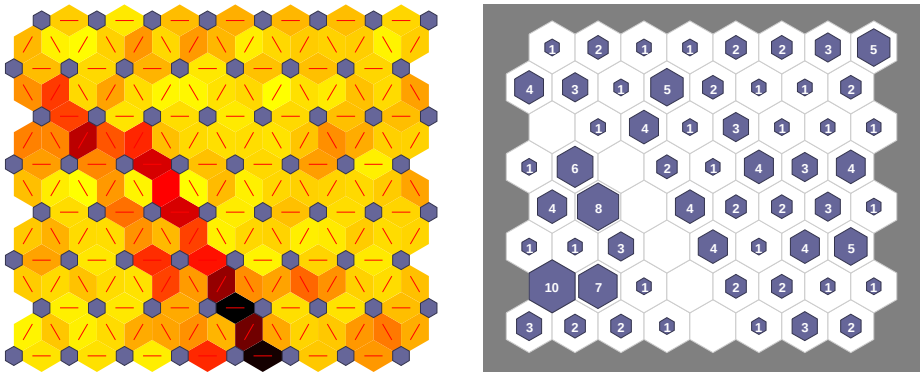
3. *Cooperation.* The BMU also excites the neurons in its topological neighborhood. This cooperative process decays as neurons are further away from the winning neuron (Equation (1.22)).

4. *Adaptation.* The BMU and its neighboring neurons are pulled closer to the input sample. For each neuron in the SOM, the weight vector is updated according to Equation (1.21).

After initialization, the remainder training steps are repeated until a stop criterion is achieved.

Clustering of Self-Organizing Maps

The above unsupervised learning algorithm preserves data topology, that is to say, samples that are close together in the high-dimensional input space have close positions in the map. After training, the SOM provides a nonlinear mapping of the dataset onto a two-dimensional grid that allows for identifying groups of samples with similar characteristics (i.e., clusters) by taking all features into account simultaneously.



(a) **(b)**

FIGURE 1.11: SOM organized into an 8 × 8 neuron grid. (**a**) Neighbor weight distances. The blue hexagons represent the output neurons, while the red lines are neuron connections. Darker colors represent larger distances between neighboring neurons, and lighter colors represent smaller ones. (**b**) Sample hits. This image shows how many training samples are associated with each neuron.

The unified distance matrix (U-matrix) provides the distances of the weight vectors to each of its immediate neighbors in the grid. It can be used with the aim of displaying the distance structures, while using a color scale in the two-dimensional array of neurons, maintaining the topology, and allowing for the identification of the clusters, boundaries, and representative neurons. See, for example, Figure 1.11(a), where the U-matrix of an 8 × 8 hexagonal SOM topology is shown. The U-matrix is not only a useful visualization tool but also a powerful analysis tool suitable for mathematically identifying clusters. Another useful visualization method consists in displaying the number of hits in each neuron of the map. This information can also be applied in clustering the SOM using low-hit neurons to locate cluster borders (see Figure 1.11(b)). In this work, the U-matrix based clustering algorithm was used [136].

1.3 Data Information Sources and Data Mining for Intelligent Vehicles

Since ADAS depend on ML algorithms, a reliable data source is needed, on the one hand, to model real-world contexts properly, and, on the

other hand, to select and understand the best sensors and actuators for each driving situation.

Many research works have been conducted in the field of intelligent vehicles, and their sensors, to understand how drivers and traffic behave and to determine which sensors are suitable for each situation. Most of these studies rely on cars instrumented with controller area network (CAN)-bus loggers, inertial measurement units (IMUs), radars, LIDARs, and video cameras to collect meaningful data. There exist two main branches in driving studies: non-naturalistic and naturalistic.

Non-naturalistic driving studies (non-NDS), such as NU-Drive [137], UYANIK [14], and UTDrive [138] make use of dedicated instrumented cars, which simplifies data collection and logistics by increasing the number and complexity of the boarded sensors. The selected human subjects (motorists) do not drive their own cars, but, on the other hand, the driving conditions can be controlled, assuring the good quality of the results. This approach also involves simulator-based road-safety studies, self-report studies, statistical analysis, and authority-investigated crashes. However, despite the fact that these methods have greatly contributed to understanding how road users behave and which factors involving crashes are the most important, they do not reflect completely realistic situations. Therefore, naturalistic driving studies (NDSs) have been conducted to compile data that faithfully reflects driver behavior in every-day traffic situations. NDSs, pioneered by the Virginia Tech Transportation Institute (VTTI) [139], are the most recent trend in traffic safety and ADAS research.

Meaningful examples of NDSs are UDRIVE [140], carried out in the EU, and the 100-Car NDS [141] and SHRP2-NDS [142], carried out by the VTTI. These studies focused on collecting data from drivers (human subjects) in their own vehicles and environment in everyday trips without interfering in any normal behavioral patterns, that is to say, with no experiment control. Thus, NDSs allow for the observance of normal driver situations, providing much better feedback to correctly understand drivers' behavior in normal, unguided traffic situations, as participants do not have the feeling of being involved in an experiment.

In this chapter, the non-NDS Uyanik and the SHRP2-NDS datasets are introduced. These datasets allow us to explore the differences on the concrete characteristics of NDSs and non-NDSs to decide whether they are adequate or not for the concrete applications to be developed, and their advantages and drawbacks when used to develop ML-based ADAS.

1.3.1 Uyanik Non-NDS Description

Uyanik [14], [143] is a Renault Mégane sedan, fitted with a reinforced front bumper, a high-power battery, a 1500 W DC-AC power converter, a CAN-bus output socket, an instrument bench at the navigator's seat, and power and signaling rewiring (see Figure 1.12). The complete dataset that was compiled by the vehicle on each test drive comprises three channels of uncompressed video from the left and right sides of the driver and the road ahead.

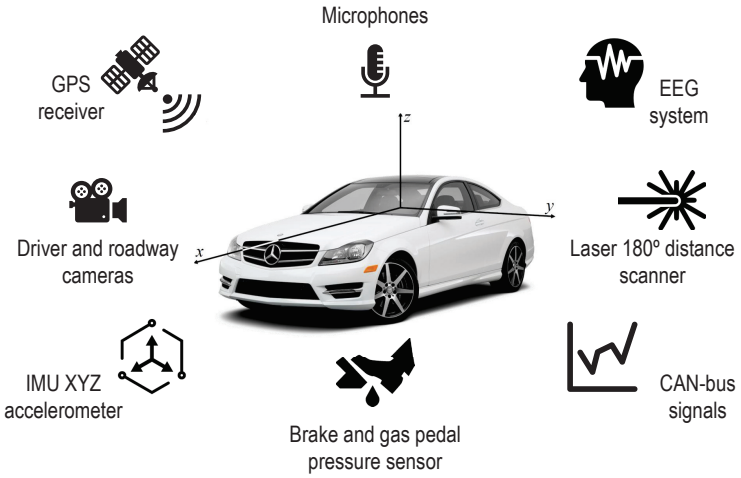


Figure 1.12: Data-acquisition systems and sensors installed in the Uyanik car [14].

It also includes three audio recordings, GPS, and CAN-bus readings, including vehicle speed (VS), engine RPM (ERPM), steering wheel angle (SWA), and brake pedal status (pressed or idle) (see Table 1.2). Gas pedal engagement percent (PGP) is sampled at either 10 or 32 Hz, whereas brake pedal and gas pedal pressure sensors (BP and GP, respectively) are sampled at the same CAN-bus rate. Finally, a laser distance measuring device was fitted in the front bumper jointly with an IMU XYZ Acceleration measuring sensor set-up and an electroencephalogram (EEG) monitor. All of the signals were handled jointly, which requires a re-sampling of the data streams to the highest frequency of 32 Hz.

Video channels were collected by means of a digital video recorder, audio by a data acquisition system, and digital data by a merge of all the signals with RS-232 and USB buses into a laptop computer running custom software (SW) that was developed by the Technical University

Table 1.2: Significative variables of the Uyanik dataset [14].

Features	Signals (Time Series)
CAN bus	Steering wheel angle (SWA)
	Steering wheel speed (SWS)
	Vehicle speed (VS)
	Percent gas pedal (PGP)
	Engine RPM (ERPM)
Pressure sensors	Brake pedal pressure (BP)
	Gas pedal pressure (GP)
IMU unit	Roll rate (RR)
	Pitch rate (PR)
	Yaw rate (YR)
	X axis accelerometer (XACC)
	Y axis accelerometer (YACC)
	Z axis accelerometer (ZACC)
Laser	Distance to obstacle (d_90)
GPS	Coordinates (GPS)

of Istanbul [14]. It is worth noting that the audio and video feeds, as well as the digital data, were properly synchronized.

This re-synchronization was carried out by displaying the video feeds jointly with the plain data of Uyanik, resulting in a set of spreadsheet-like data chunks that can be easily processed automatically.

The driving behavior data collection was performed in Istanbul. The car route is around 25 km (about 40 minutes), and includes different kinds of sections: city, highway, secondary roads, and a university campus. The age range for female drivers was 21–48, and the corresponding male range was 22–61. The route was the same for all 20 drivers, however, the road conditions differ depending on traffic and weather.

1.3.2 SHRP2-NDS Description

The main objective of the SHRP2 project is to address the influence of driver performance and behavior on traffic safety. This involves under-

standing the way the driver handles and adapts to a vehicle, roadway characteristics, traffic lights, signs, infrastructure, and other environmental features. SHRP2-NDS offers two key advantages: detailed and accurate precrash information, including objective information about driving behavior, and exposure information, including the frequency of behaviors in normal driving. To take part in this study, the participants' vehicles are checked for their suitability to fit the systems used in the NDS. Next, while the data acquisition system and instrumentation are installed, the participant serves several driver-assessment tests as well as medical examinations for a total of 2–3 h.

SHRP2-NDS is the largest NDS ever conducted, involving 2360 participants as of the ending moment of the study (September 2012). These participants were recruited in six different locations across the United States of America. Each location hosted 150 to 450 vehicles, and these locations with their coordinating groups were:

- *Bloomington, Indiana*—Indiana University, 150 vehicles.
- *Central Pennsylvania*—Pennsylvania State University, 150 vehicles.
- *Tampa Bay, Florida*—CUBRC and University of South Florida, 441 vehicles.
- *Buffalo, New York*—CUBRC, 441 vehicles.
- *Durham, North Carolina*—Westat, 300 vehicles.
- *Seattle, Washington*—Battelle, 409 vehicles.

The study, designed by the VTTI, required the same number of participants for each of age and gender groups. SHRP2-NDS adheres to the principles of informed consent and privacy requirements and each institution operated under the monitoring of either their own Institutional Review Board (IRB) or the VTTI IRB. The participants receive an annual incentive of $500, and they must authorize access to the vehicle so as to replace the hard disk in which the data are recorded every 4–6 months.

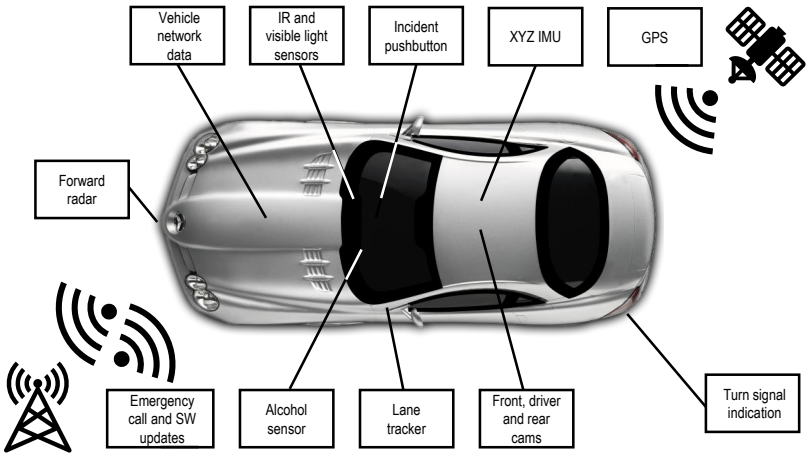


Figure 1.13: Data acquisition systems and sensors installed in the vehicles that participated in the SHRP2-NDS. IR: infrared; SW: software.

Table 1.3: Significative variables of the SHRP2 NDS dataset.

Features	**Signals (Time Series)**
CAN bus	Steering wheel angle (SWA)
	Steering wheel speed (SWS)
	Vehicle speed (VS)
	Percent gas pedal (PGP)
	Engine RPM (ERPM)
IMU unit	Roll rate (RR)
	Pitch rate (PR)
	Yaw rate (YR)
	X axis accelerometer (XACC)
	Y axis accelerometer (YACC)
	Z axis accelerometer (ZACC)
Radar	Distance to front vehicle (d_90)
GPS	Coordinates (GPS)

The VTTI developed a custom data acquisition system for the SHRP2-NDS [142]. This system, whose main blocks are displayed in Figure 1.13, was manufactured by American Computer Development Inc. and includes a forward radar, four video cameras (with a forward-facing one, color, and wide-angle view), an IMU with XYZ accelerometers and gyroscopes, vehicle network (CAN bus) data logging, GPS-based location, computer vision-based lane tracking, and data-storage capability (refer to Table 1.3). Additionally, the data acquisition system has cellular connectivity to provide emergency-call functionalities, system health checks, and SW updates. The captured data, including the radar, are uniformly sampled at a rate of 10 Hz, and all the different sources are properly synchronized.

1.4 Hardware Solutions

The deployment of the aforementioned ADAS requires of the development of high-performance embedded systems. These systems must allow, on the one hand, to achieve all the required specifications regarding timing and throughput and, on the other hand, they must have a energy consumption low enough to not having a significant impact on the energy efficiency of the vehicle, which can be critical in the emerging scenario of electric automobiles.

Currently, ADAS-intended system-on-chips (SoCs) are being offered by semiconductor manufacturers. However, these systems typically are not optimized enough, increasing both the costs and the energy consumption. This happens because they are ADAS prototyping platforms, ready to run systems that are not completely debugged. Conversely, application-specific integrated circuits, despite meeting all the specifications regarding utilization and energy efficiency, are expensive to design and produce, particularly for short life-cycles and small series. In that sense, on-demand optimized solutions based on field-programmable gate array (FPGA)+CPU, known as programmable system-on-chip (PSoC) can be considered as a trade-off solution that provides the strictly necessary resources for each application, enabling cost reduction and power efficiency. Finally, in recent years, due to the size of the PSoCs available in the market growing non-stop, application scalability issues began being identified. These issues, on the one hand, were strongly related to the internal interconnections, and, on the other hand, to the available resources not being heterogeneous enough. For

those reasons, adaptive compute acceleration platforms (ACAPs) were released to market to improve the milestones already reached by PSoCs.

1.4.1 Field-Programmable Gate Arrays

FPGAs are a trade-off solution between the high performance of an application specific integrated circuit (ASIC) and the attempt to bring high performance, cost-effective solutions [144]. As ASICs, they still allow to perform very different operations all together and in parallel. Basically a FPGA is an integrated circuit that is configured after being manufactured [145]. For that purpose, FPGAs rely on structures known as configurable logic blocks (CLBs) as their fundamental blocks. These CLBs are divided into slices, each of them containing flip-flops (FFs), look-up tables (LUTs), multiplexers, etc., as shown in Figure 1.14.

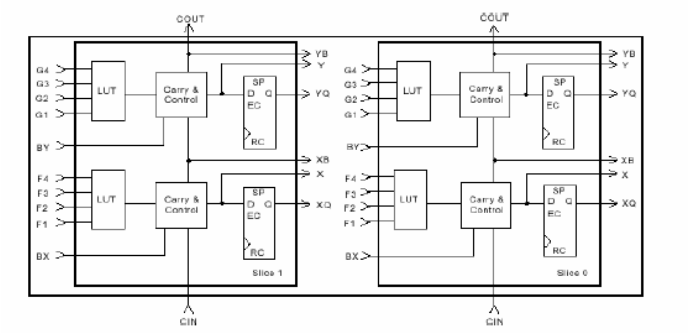


Figure 1.14: Typical architecture of a CLB.

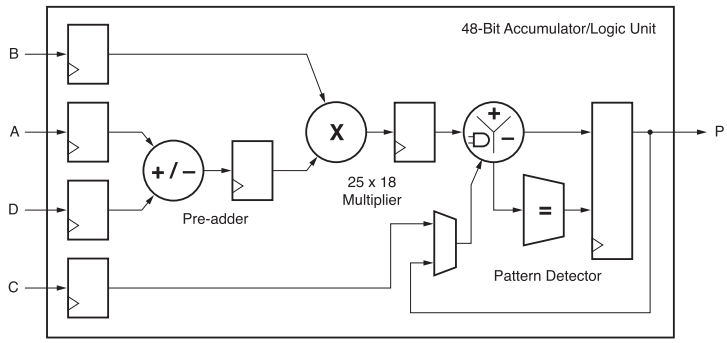


Figure 1.15: Block diagram of a *hard* Xilinx DSP Block.

In addition, since there are high-resource consuming applications, such as long word-length products, accumulators and comparisons, FPGA

manufacturers include "hard" resources such as dedicated multipliers, comparers and accumulators in order to free up CLBs for other purposes. The most known *hard* blocks are digital signal processing (DSP)-dedicated ones [146], with block diagrams like the one in Figure 1.15. In this Figure, it can be seen that DSP blocks include resources to implement all the operators described in this paragraph, operators that if they were implemented by using only CLB logic would consume a very high number of them, impairing both energy efficiency and performance.

Several manufacturers have developed their FPGA families regarding several considerations and applications. On the one hand, Intel FPGA, formerly known as Altera, and Xilinx offer similar solutions in terms of size and variety of resources, being focused on general purpose FPGAs, handling almost the entire market. Well known solutions of these manufacturers are the Intel Cyclone, Arria and Stratix, and the Xilinx Spartan, Artix and Kintex. On the other hand, manufacturers like Lattice Semiconductor are focused on application-specific FPGAs, with a portfolio of ultra-low power consumption, small form factor devices intended for applications where size and battery life are a critical design specification.

The last iteration of *hard* blocks is the incorporation of microcontroller-like peripherals, such as UARTs, Ethernet switches and even full microprocessor systems, turning FPGAs into PSoCs [147].

1.4.2 Programmable System-on-Chips (PSoCs)

PSoCs are, essentially, FPGAs fitted with *hard* microprocessor systems. Therefore, besides hardware (HW) implementations, they are suitable for running SW applications as well as mixed HW/SW designs [147]. This is specially to discharge the microprocessor system from the most intensive, parallelizable, calculation stages, with the application running on SW but using FPGA-deployed accelerators to perform operations that can be paralleled, drastically reducing computation times if used properly. Furthermore, since the microprocessor system and the FPGA logic are built in the same silicon wafer, printed circuit board (PCB) layouts turn simpler and timing gets improved, topping the performance. Additionally, manufacturers include rugged, automotive-standard compliant models that enable developers to reach more easily the reliability standards.

PSoCs are suitable for the implementation of ADAS. PSoCs combine Microprocessors and FPGAs, hence, this range of devices is specially suitable for these mixed-approach developments since they allow to

implement complex algorithms taking advantage of the C language on their microprocessor while the most calculation-intensive stages can be HW accelerated.

Mainly two manufacturers have developed their PSoC families regarding several considerations and applications: Intel FPGA and Xilinx. These manufacturers offer quite similar solutions in terms of size and variety of resources, being focused on HW-based acceleration of SW applications. Well known solutions of these manufacturers are the Intel Cyclone, Arria and Stratix, and the Xilinx ZynQ-7000, ZynQ Ultrascale and ZynQ Ultrascale+. In the case of Xilinx, the FPGA included in the PSoC is derived of any of the FPGA families, despite having different names.

The individual implementations of this work will be deployed on a ZynQ-7000 XC7Z045 FFG900 – 2 PSoC [147], embedded in a ZynQ-7000 SoC ZC706 Evaluation Kit [148], which includes all the JTAG programming interface, peripherals and other accessories needed for the SoC to work properly.

This FPGA, derived from the Xilinx Kintex-7 family, has the following logic resources:

- 54 650 logic blocks, each one conformed by four six-input LUTs and FFs;
- 19.2 Mbits of high-speed RAM blocks;
- 900 DSP blocks; and,
- An analog-to-digital converter (ADC).

ZynQ-7000 SoC include in the same silicon wafer a microprocessor, called processing system (PS); and a FPGA, called programmable logic (PL). The PS is connected to the PL as well as its peripherals through a variant of the Advanced Microcontroller Bus Architecture (AMBA), known as Advanced eXtensible Interface-4 (AXI4) [147], [149].

The AXI4 interface has become the standard specification for Xilinx to interconnect the IP cores deployed in the PL between them and with the PS. The main advantage of using this interconnection methodology is, basically, its consistence and standardization. In addition, it is fully specified and parametrized, which makes the users able to adopt it directly, with no adaptations needed to be performed.

The general architecture of ZynQ-7000 SoCs is displayed on Figure 1.16. As it can be seen, the block named as MPCore relies on a Dual-Core ARM Cortex-A9 RISC microprocessor to perform the logic

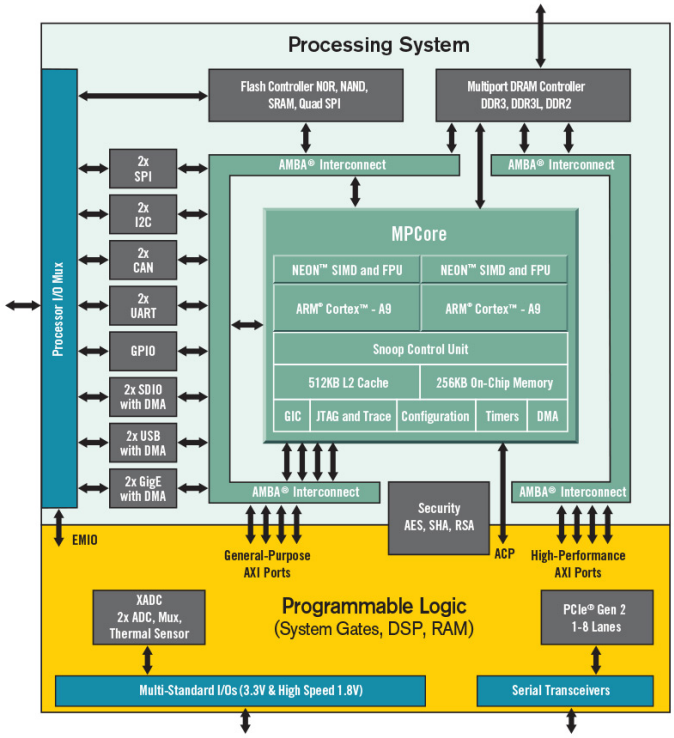


FIGURE 1.16: ZynQ-7000 SoCs' architecture.

operations. In addition, this MPCore has a JTAG interface, a global interrupt controller (GIC), timers, DMA, etc. The rest of peripherals are connected to the MPCore by means of AMBA-AXI4, which has a dual purpose: this allows the peripherals to be accessed by both the microprocessor and the PL. Thus, for instance, if a designer wanted to add a CAN-bus interface to an hypothetical PL design, it would not be not necessary to implement it on FPGA-logic, because the CAN-bus PS peripheral could be accessed through AMBA-AXI4.

1.4.3 Adaptive Compute Acceleration Platforms (ACAP)

ACAPs are the next step on the field of PSoCs. Based on a novel heterogeneous compute architecture, they combine the following elements [150]:

- *Scalar processing elements*: they are based on CPU cores, which are very efficient at complex algorithms with diverse decision trees

and a broad set of libraries. However, they are limited in performance scaling.

- *Vector processing elements*: they are based on DSP and graphic processing unit (GPU) architectures. These elements are more efficient at parallelizable compute functions. On the other hand, they experience latency and efficiency lacks as a trade-off due to the inflexible hierarchy of their memories.

- *Programmable logic (PL)*: this element is basically an FPGA. This architecture can be precisely parameterized and customized to a particular compute function or user-designed topology. This fact makes them the best choice in terms of low latency and real-time applications, such as those for ADAS. However, these implementations require of expert-level knowledge of the algorithms to be implemented, and each change may take several hours of compilation.

As can be seen, these platforms show a level of heterogeneity that, despite not differing very much from PSoCs, enables to perform a segmentation of the different types of operations that could be necessary to implement a concrete application as well as extreme data throughput.

In the same maner as for PSoCs, Intel FPGA and Xilinx are the companies that control the market of these particularly novel devices, particularly with the Intel Agilex and Xilinx Versal families.

The integration of all the developed applications in a single-chip solution will be performed on a Xilinx Versal ACAP, more precisely, the Versal XCVM1802-2MSEVSVA2197 [150], embedded in a Versal VMK180 Evaluation Kit [151]. This kit includes all the peripherals and accesories needed for the ACAP to work properly.

The PL section of the ACAP, that is to say, the embedded FPGA, relies on the following list of resources:

- 28 120 logic blocks, each one conformed by thirty-two six-input LUTs and sixty-four FFs;

- 33.5 Mbits of high-speed RAM blocks;

- 130.2 Mbits of ultra-high-speed RAM blocks; and,

- 1 968 single precision floating point-enabled DSP engines.

The selected device is part of the Versal Prime series, which means that it does not count on the GPU-like vector processing elements,

known as AI engines by Xilinx, but, on the other hand, it has an enormous set of logic resources that allows to implement virtually any HW/SW development.

So as to increase the performance of the SW partition, the scalar processing elements are grouped in two subsets, the APU, based on a dual-core ARM Cortex-A72 RISC microprocessor able to run very complex top-level applications; and the RPU, based on a dual-core ARM Cortex-R5F RISC microprocessor to run time-critical applications in real-time [150]. The block interconnection runs on an evolution of the AMBA AXI4

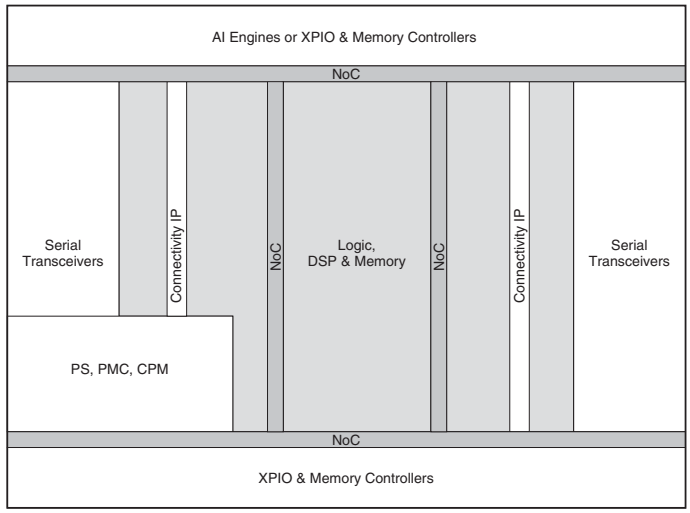


Figure 1.17: Versal ACAP's architecture.

interface used in the ZynQ series, known as the programmable network-on-chip (NoC). The NoC easily enables high-bandwidth connections to be routed around the device, to connect together areas of the device that demand and use large amounts of data, alleviating the possible bottlenecks that might occur [150]. In Figure 1.17 the architecture of a Versal device along with the paths through the NoC runs is displayed.

As can be seen, and, as the main difference with the ZynQ architecture of Figure 1.16, when compared with the AXI4 bus, besides surrounding the Logic, DSP and Memory gray block, which are nothing else than the PL of the device, the NoC goes through the middle of it, achieving drastic improvements of the timing performance of the interconnection of the implemented blocks. This happens because, with this topology, the logic path of the blocks to the NoC is minimized due to the synthesis tools placing them by its sides. On the other hand, it can

also be observed that the NoC extends along the upper and lower edges of the PL to bring the most direct access to the remaining blocks of the system, which, jointly with the 7 nm manufacturing process, drastically improves the performance of the entire solution.

Chapter 2

An FPGA-Based Neuro-Fuzzy Sensor for Personalized Driving Assistance

2.1 Overview

This chapter describes an FPGA-based neuro-fuzzy system to personalize ADAS with the aim of providing DS-based, human-like driving assistance, particularly for the case of ACC. To achieve this objective, firstly, a real-world driving database was used to get access to DS knowledge obtained in real-life situations. Since ACC is designed for car-following scenarios, the search of data stretches was limited to those instants when car-following-related driving behavior was happening. For that purpose, in this chapter, definitions about car-following are provided. Once the stretches are delimited according to the definitions, different driving behaviors are identified and labeled through a k-means clustering algorithm. A Takagi-Sugeno ANFIS was trained with the labeled data to provide real-time, high-performance DS identification. To guarantee its high-performance execution, it was deployed on a Xilinx PSoC. Finally, different following profiles were developed and configured to enable the system to automatically mimic the time gap kept by human drivers at car-following situations.

Thus, this chapter is organized as follows. Section 2.2 introduces the concept of personalization for ADAS. Section 2.3 provides an outline of the proposed DS personalization system. In Section 2.4, the DS characterization methods in car-following scenarios are presented. The neuro-fuzzy modeling approach of DS groups is provided in Sections 2.5. Then, Section 2.6 exposes the implementation of the FPGA-based intelligent sensor and provides experiment results for a personalized

ACC in a steady car-following situation. Finally, concluding remarks are summarized in Section 2.7.

2.2 Personalization Approaches in ACC

ACC systems are a reality in an increasing number of recently manufactured cars [152]. These systems, as they are provided, help drivers, on the one hand, to keep a constant speed, just in the way that conventional CC systems do, and, on the other hand, to keep the time gap with the preceding vehicle, known as THW. Thus, not only do these systems keep a constant speed, with the decreasing of the workload that implies, but also force to maintain a safety distance with the preceding vehicle, reducing the risk of a rear-end collision [153].

However, not all drivers are familiarized with these systems, and many car owners even ignore that their car fits any of these features. On the other hand, among those that are aware, only a marginal rate uses ACC frequently. This fact might respond to many factors, but the main one is related to the acceptance of the system. The main problem that forces drivers to neglect the use of ACC is by itself the experience of using it [54]. This happens because many ACC systems are too conservative at keeping the distance with the preceding car, prioritizing a very rigid safety gap at the cost of impairing the constant speed feature. This causes constant accelerations and decelerations that many car occupants find annoying, making them prone to disengage the system. For that reason, and with the aim of improving the driver experience, a variety of car manufacturers allow to adjust some system parameters [154].

In recent years, several car-makers have introduced a slight level of ADAS customization by allowing users to manually select the THW parameter from a knob or a lever in the steering wheel, always within the pre-engineered parameters. However, despite a group of drivers appreciating the freedom that manually adjustable parameters provide them in automatic longitudinal control modes; this personalization process, including the operation of the system itself, can be complicated for other drivers not so familiarized with control systems in automobiles. Therefore, it seems reasonable to introduce personalization strategies that need no driver intervention to make the adoption of these systems easier for that sector of the automotive community [155]. ADAS personalization embeds characteristics of motorists' DS into the system. The DS is the manner the driver operates a vehicle in terms of steering,

acceleration and braking, and how this driver relates to the other ones in terms of predictability and aggressiveness [57]. There exist two approaches to personalize ADAS depending on DS: individual-based and group-based.

Individual-based personalization strategies try to reproduce or identify the DS of a given individual using ML techniques or mathematical models. Within this scope, in the works by the authors of [156], [157], ACC was adapted to individual drivers in real time based on DS observations, achieved by the recursive least-squares (RLS)-based fitting of a linear car model. The model reproduced the time gaps observed in a short manual-driving session (learning mode) and mimicked these learned time gaps when the personalized ACC was enabled (running mode). In of [158], an ACC was developed with the same approach as previous, but introducing a forgetting factor with the learning of the driver parameters occurring when the driver is manually controlling the vehicle while following a lead vehicle. A different approach was chosen in [159], where a learning, HMM-based driver model, combined with a model predictive control (MPC) algorithm, was used to create personalized driver assistance able to imitate different DSs. Regarding FCW/AEB, in [160], the personal DS was statistically modeled to estimate driver-specific probability distribution of danger level to determine the activation threshold of the system. Individual-based strategies have the main advantage of entirely mimicking the DS but, despite this feature being very desirable, it generally requires intensive computation, hard to be achieved in real-time. Moreover, the modeled behaviors would require safety verification since not all drivers handle vehicles in a correct way. To mitigate these drawbacks, group-based approaches have emerged.

Group-based personalization strategies locate drivers in a small number of representative DSs for which a control strategy is implemented. In [161], a group-based approach of the driver's ACC preferred time gap is presented. The drivers were clustered to create three general driver profiles to be used, together with demographic information, to predict the gap by using a regression model and decision trees. The authors of [162] describe a stop-and-go ACC system that groups drivers into three clusters depending on their DS, with the cluster membership determining the reference acceleration profile to adjust the ACC controller. Recently, in [163], an SVM-based approach was used to classify driving behavior into two different clusters in order to select a personalization parameter for an ACC. These strategies, despite not entirely mimicking DS, are computationally more efficient, requiring less com-

putation since they work on a previously offline-trained classification algorithm, allowing online, real-time computation. On the other hand, as they represent a class-averaged DS, they can easily be validated and verified to always operate in safe margins.

In this chapter, a hybrid personalization strategy for DS modeling is proposed. To perform system development, data from a subset of real-world trips of the SHRP2 NDS [13] described in Section 1.3.2, are used. First, a group-based technique was used with the aim of building a three-cluster DS classifier. Then, each the clusters was approximated by means of an ANFIS obtaining identification rates higher than 85.7% for the three clusters. Finally, an individual-based algorithm was used to adapt the behavior of the group to a particular driver. The whole system was successfully implemented using an FPGA device of the Xilinx's ZynQ PSoC. The system can mimic the typical timing parameters of a group of drivers as well as tune these typical parameters to model individual DSs. The neuro-fuzzy intelligent sensor provided high speed for real-time active ADAS implementation and could personalize its behavior into safe margins without driver intervention. In particular, the personalization procedure of the THW parameter for an ACC in steady car-following scenarios is described.

2.3 Outline of Driving-Style Personalization System

The intelligent sensor developed relies in two well-differentiated stages: an offline design stage and the online in-car operation stage. The sequence of tasks involved in the offline design stage are depicted in Figure 2.1.

The first task comprises the segmentation of SHRP2-NDS trips with the aim of selecting car-following scenarios, and the computation of a set of meaningful car-following parameters or features for each segment of the trip (e.g., THW, time-exposed THW (TETH) and time-integrated THW (TITH) [56], which will be defined later in Section 2.4.1). Next, an unsupervised clustering technique, the k-means algorithm, already introduced in Section 1.2.2, was used to group DSs into a number of clusters in car-following circumstances. Then, an ANFIS, described in Section 1.2.4, was trained in order to develop a high-performance model of the DS classifier. The main advantage of using an ANFIS-like model is that it was suitable for the development of high-speed parallel-HW architectures, allowing in-car DS classification for ADAS personaliza-

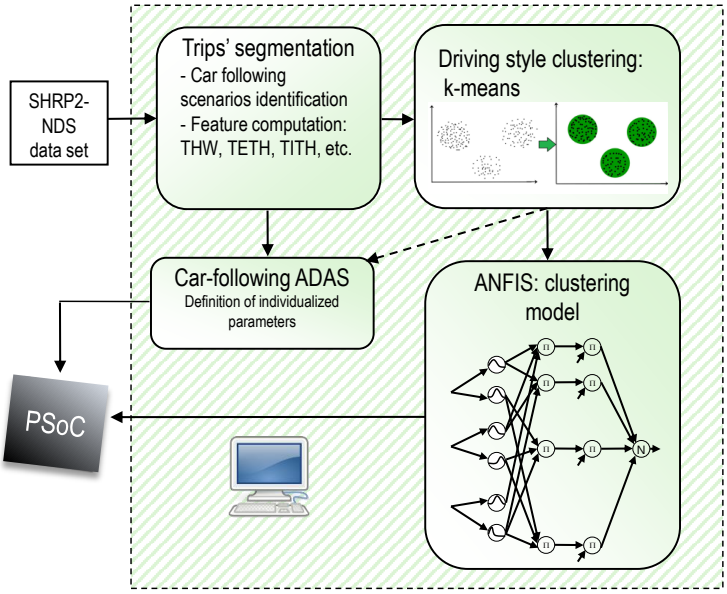


FIGURE 2.1: Offline sequence of tasks involved in the design and development of a neuro-fuzzy sensor for ADAS personalization.

tion in the online stage. Thus, the ANFIS model was implemented using an FPGA of the Xilinx ZynQ device family. More precisely, as displayed in Section 1.4.2, it is a PSoC that integrates microprocessors and their peripherals with PL.

The FPGA-based DS classifier acted as an intelligent sensor able to adapt the ADAS response to driver preferences in longitudinal car-following scenarios. In particular, a steady car-following application was developed: a personalized ACC.

2.4 Driving Style Characterization in Car-Following Scenarios

Car-following describes a driver following another driver in a traffic stream. For that purpose, drivers operate throttle and brake pedals to maintain a desired range of distance from the preceding vehicle. The main objective of modeling car-following behavior is predicting the following VS and distance based on stimuli provided by the preceding vehicle for a set of road and driver characteristics. A retroactive approach can be applied in this topic, that is, based on DS and speed,

predicting the desired distance between the following and the leading vehicle. This is the base of several ADAS, such as ACC, FCW, or AEB.

2.4.1 Driving Parameters and Driving-Style Characterization

The main step to correctly identify DSs is determining the adequate variables to provide a robust enough identification. The signal choice depends, however, on the desired application, which is crucial as any further processing of that data will entirely depend on that choice. Therefore, due to the plurality of applications, ranging from ADAS personalization [164] or driving correction for safety and comfort improvement [165] to fuel economy advice, there is no recommended set of parameters. Thus, for identifying aggressive drivers, high accelerations must be monitored. On the other hand, speed profiles should be monitored to analyze fuel efficiency. Additionally, the acceleration variable is generally combined with brake activation and speed measurements. In other works, pressure on the brake and throttle pedals are used as reliable indicators to identify DSs [166], [167]. Furthermore, the selection of which signals are the most adequate for each application might be guided by the identification or apparition of some circumstances, or the restriction of the identification problem to some contexts or driving events, such as braking, distance-keeping, roundabouts, cornering, lane changes, or even car-following.

In this Chapter, the DS characterization problem was restricted to steady car-following scenarios where clear distance-keeping behavior was observed. In car-following scenarios, the most relevant variables are the speed of the host vehicle (v), the relative speed of the host to the leading vehicle (v_r), and the distance between host and leading vehicle (d). With these variables as a starting point, we derived features to parameterize car-following scenarios such as THW and the inverse of time-to-collision (TTCi), which are commonly used.

$$THW = \frac{d}{v} \tag{2.1}$$

$$TTCi = \frac{v_r}{d}. \tag{2.2}$$

THW (Equation (2.1)) is the time difference between two successive vehicles when they cross a given point, whereas TTCi (Equation (2.2)) is the inverse of the time two vehicles would require to crash if they kept the same speed and trajectory. These parameters are often used to assess car-following styles. Nevertheless, other parameters can be used

to provide more complete insight into DS in car-following scenarios. These parameters are TETH and TITH [56]. Thus, TETH (Equation (2.3)) represents the time exposure to a THW lower than a predefined safety threshold during a ride.

$$TETH = \sum_{t \leq T} \delta_i(t)\tau_s$$
$$\delta_i = \begin{cases} 1 & \forall\ 0 \leq THW_i(t) \leq THW^* \\ 0 & \text{else,} \end{cases} \tag{2.3}$$

where T is the total time interval considered, $\delta_i(t)$ is a binary activation parameter, τ_s is the sampling period, $THW_i(t)$ is the instantaneous value of THW at a given moment t, and THW^* is the predefined safety threshold value (Figure 2.2(a)). On the other hand, TITH (Equation (2.4)), is the summation of the difference between THW^* and $THW_i(t)$ restricted to time intervals when $THW_i(t) < THW^*$ (Figure 2.2(b))

$$TITH = \sum_{t \leq T} [THW^* - THW_i(t)]\,\delta_i(t)\tau_s$$
$$\delta_i = \begin{cases} 1 & \forall\ 0 \leq THW_i(t) \leq THW^* \\ 0 & \text{else.} \end{cases} \tag{2.4}$$

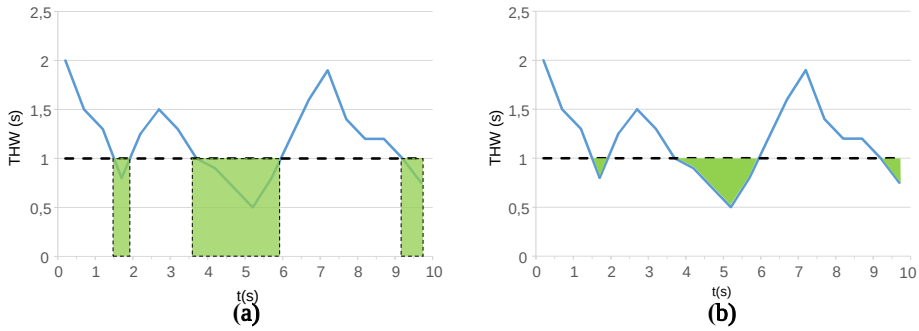


FIGURE 2.2: Representative example of car-following features: (**a**) TETH and (**b**) TITH.

The above parameters have been found to characterize the car-following behavior in both motorists used to activate the ACC mode and non-ACC users [56]. Thus, as TETH is the time a driver rides behind a leader car below a certain THW threshold, we can determine which percentage of that trip occurs at a time distance below the recommended

values. Simultaneously, TITH enables to measure how close a following vehicle has got to its leader during the TETH. Consequently, the use of these parameters jointly with the THW root-mean-square (RMS) value THW_{RMS} provides a good measure of driver behavior. As seen later, this set of features is very helpful to identify car-following DSs.

2.4.2 Steady Car-Following Premises

Car-following scenarios include accelerating, braking, approaching, and steady following [168]. Steady car-following circumstances occur when the relative speed between vehicles is low and $|TTCi| \leq 0.05\ s^{-1}$ [168]. The statements used to segment the trips into steady car-following stretches are the same as those used in [169]. Thus, it was assumed that there was a lead vehicle in front of the host vehicle and this leader traveled in the same lane. Additionally, the lead vehicle must have been at a maximum distance of 120 m and the host vehicle must have been traveling at 20 km/h at least. The maximum distance was constrained because the radar sensor could detect targets beyond that 120 m range, and following behaviors with vehicles at such distance are negligible. On the other hand, a minimum-speed constraint was applied to filter traffic jams, which cannot be considered real car-following. Additionally, the segments of interest were restricted to those lasting more than 30 s.

2.4.3 Car-Following Stretches in the SHRP2-NDS Trips

The selected car-following stretches were extracted from 48 different trips of 40 different drivers [170]. Each driver was identified with a numeric code that eased identification of the driving data while preserving their privacy. Most of the trips contained mixed-environment driving, ranging from parking lots and streets to motorways and highways. These trips were selected so as to involve different traffic situations as well. Different traffic situations enable researchers to better understand driver behavior and how drivers relate to each other in complex contingencies in both regular transit and safety compromising events. The segmentation of trips into car-following stretches is not trivial, and many parameters should be considered to perform it.

Steady Car-Following Segments

After applying the premises of Section 2.4.2 over the 48 trips, a total of 115 continuous car-following stretches were segmented. Neverthe-

less, these 115 stretches were extracted from 28 of the 48 selected trips, as the 20 remaining trips did not contain stretches that gathered the characteristics delimited in steady car-following premises. It is worth noting that the segmented stretches were not evenly distributed among the 28 trips. Thus, in order to uniformize the length of the segmented stretches and consequently reduce standard deviation to increase comparability, all stretches were split into smaller ones lasting between 30 and 59.9 s. Additionally, those with $THW_{RMS} > 4.5$ s were discarded because with this THW we could not assure significant car-following events. This new partition was composed of 176 uniformized segments with a duration of $T_{RMS} = 37.3$ s and a standard deviation of $\sigma = 8.18$ s.

2.5 Neuro-Fuzzy Modeling of Driving-Style Clusters

As depicted in Figure 2.1, the first task involved in the design of the neuro-fuzzy sensor was the segmentation of the SHRP2-NDS trips into the set of steady car-following segments introduced in Section 2.4.3 and the computation of the selected features: THW_{RMS} (Equation (2.1)), TETH (Equation (2.3)), and TITH (Equation (2.4)) for each one of the 176 segments. These parameters are representative of a longitudinal DS in steady car-following situations; therefore, they could be used to personalize ADAS. Consequently, this task consists in grouping together similar DSs using a clustering approach.

2.5.1 Driving-Style Clustering

Throughout bibliography, several clustering algorithms have been used to distinguish DSs and DS-class labeling [171]. The selection of a concrete algorithm depends on the trade-off between complexity and performance. Mean-shift clustering is based on finding high-density data areas by means of a sliding window of a specified radius, aiming to locate the centroid of each area. This algorithm has the advantage of not needing to know the number of desired clusters, as the algorithm detects them by itself, but as a weakness, it should be pointed out that the selection of the window radius may be nontrivial. Another used clustering algorithm is density-based spatial clustering of applications with noise (DBSCAN) [172], which can filter outliers and find arbitrarily shaped and sized clusters, but does not perform well when clusters have variable density. Some of the following algorithms, already introduced

in Section 1.2.2, have been also used. Expectation–maximization (EM) clustering using GMM is a flexible algorithm in terms of cluster covariance [173], which bridges the restriction of distance-based solutions that only work on circular-shaped clusters. Additionally, since GMMs use a probability cluster, a given datapoint can belong to several data clusters with different probability, allowing mixed membership. HCA relies on building a tree depending on the similarity/dissimilarity between data points/clusters [174], and due to this characteristic, HCA does not need to preselect a number of clusters and it is particularly suitable to recover underlying hierarchical data structures. However, when there are particularities in some of the input data, HCA tends to group all of those particular points together, causing cluster unbalance. Finally, several research pieces in driver identification [175] and road condition monitoring [176] have successfully used the k-means algorithm, as it is useful for the proposed group-based DS identification application. The k-means algorithm is simple and quick since it is based on computing distances between each point and the groups' centroids. However, since the number of clusters must be previously specified, and the centroids of each cluster are randomly initialized, the repeatability of this type of clustering is not always assured. Nevertheless, it is quick enough to execute multiple runs in a reasonable period of time. Additionally, input data groups elaborated by k-means can easily be interpreted. Hence, due to prior characteristics, the k-means algorithm has been used to carry out DS grouping.

K-Means Clustering Results

First, the selected THW_{RMS}, TETH, and TITH features were computed for each of the 176 driving segments and normalized into the [0,1] range. According to the minimum following safety threshold of [53] and the selected THW in the work by the authors of [56], TETH and TITH were calculated for a critical value of $THW^* = 1.5$ s. After that, three-group k-means clustering of those segments was performed. The obtained cluster structure is depicted in Figure 2.3. Note that, for THW_{RMS} values higher than THW^* 1.5 s, TETH and TITH were always 0, generating the blank zone at the right TETH – THW_{RMS} semiplane of the figure.

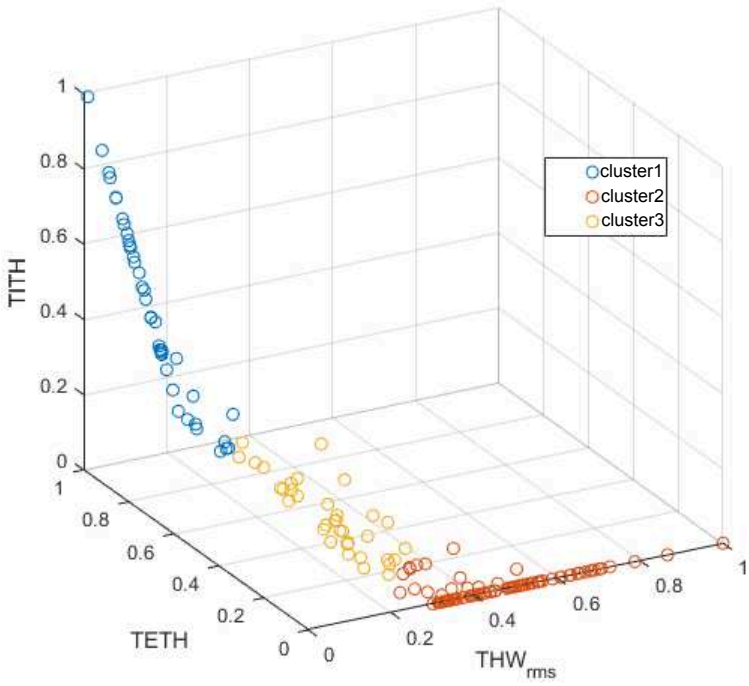


FIGURE 2.3: Clusters obtained applying the k-means algorithm to the car-following segments; THW_{RMS}, TETH, and TITH values were normalized.

DS groups were found to be stable and highly reproducible despite the randomness of the cluster centroid initialization. Therefore, within these data, a unique solid structure could be found. In Figure 2.4, the distribution of THW_{RMS}, TETH, and TITH values according to this normalized cluster structure is shown. Given the distributions displayed in the figure, the clusters could be described as follows:

- Cluster 1: groups the drivers with the lowest THW_{RMS} and the highest TETH and TITH. This cluster is representative of the most aggressive car followers.
- Cluster 2: groups the drivers with high THW_{RMS} values and minimum TETH and TITH. Thus, it incorporates the least aggressive car followers.
- Cluster 3: groups the drivers with low THW_{RMS} values, medium to low TETH and the lowest TITH, representing medium aggressive car followers.

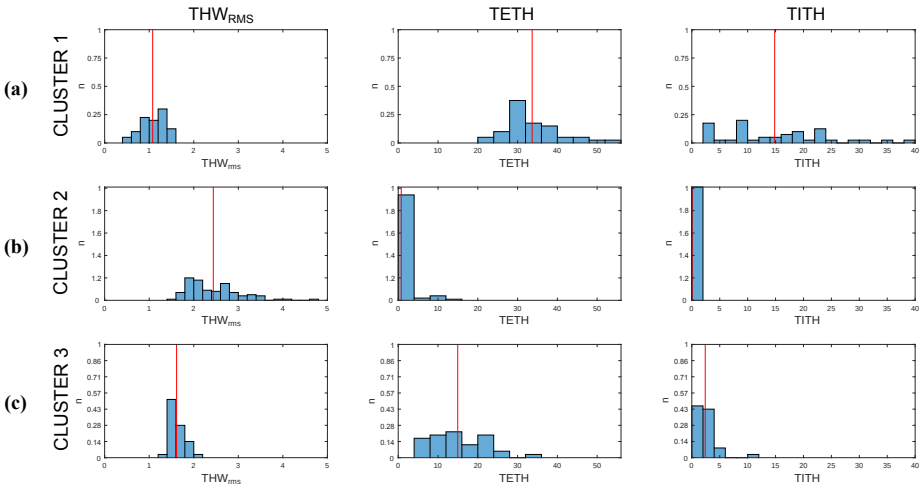


FIGURE 2.4: Histogram of THW_{RMS}, TETH, and TITH values distribution for (**a**) Cluster 1, (**b**) Cluster 2, and (**c**) Cluster 3. Red line represents average value of each distribution.

2.5.2 ANFIS-Based Identification

Following the workflow of Figure 2.1, the second task in the development of the proposed intelligent sensor was the development of a high-performance model of the DS classifier obtained above. As the proposed system was intended to accomplish online DS identification, clustering techniques could not be directly used. There is a variety of solutions suitable for efficient real-time HW implementation. Thus, ANNs, introduced in Section 1.2.3, were used in [177] to score drivers depending on the safety of their DS. In [178], finite-state machines (FSMs) were used to decide whether a driver belongs to a DS class depending on their driving decisions. Nevertheless, an outcoming line of work in DS identification is based on fuzzy-logic implementations [9] (refer to Section 1.2.4). Thus, in [179], anomalous DS was identified applying an FIS on accelerometer data, and an FIS-based dangerous DS identification application was proposed in [180]; in [57], fuzzy logic was applied to identify DSs in an online fashion.

To accomplish this task, an ANFIS was used since this system was suitable to model clusters with online performance. Thus, once the k-means clustering algorithm classified the steady car-following segments of Section 2.4.3 into three DS clusters depending on their THW_{RMS}, TETH, and TITH, an ANFIS was trained for each one of the three clusters. Each ANFIS model returned a continuous value indicating the

fitting of the prior input parameters into each of the clusters. Attending to those output values, the ANFIS model with the maximum output identified the cluster to which the inputs belong.

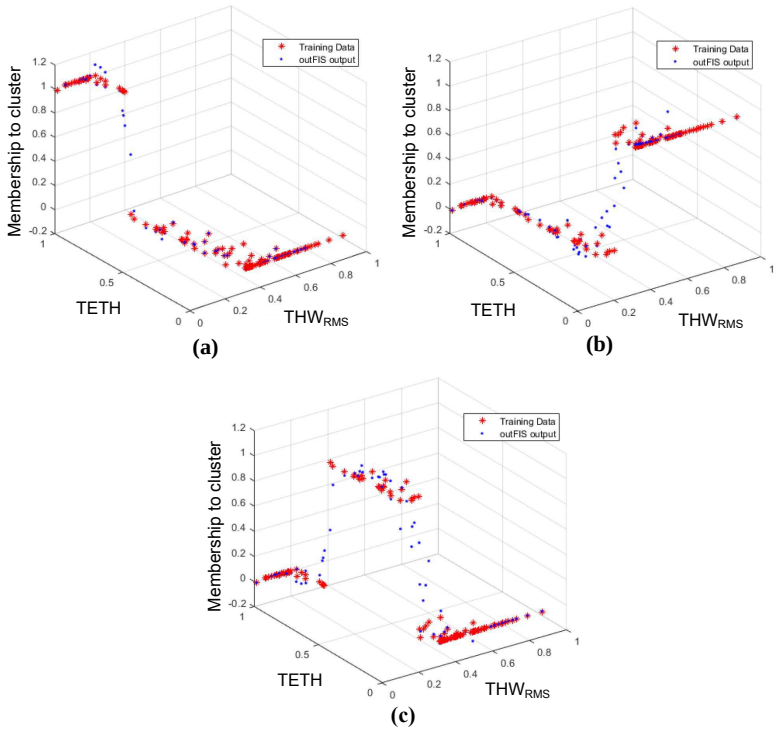


FIGURE 2.5: (**a**) ANFIS 1 (Cluster 1), (**b**) ANFIS 2 (Cluster 2), and (**c**) ANFIS 3 (Cluster 3). Training data shown in red; response of corresponding trained ANFIS shown in blue.

ANFIS Training

The 176 segments from Section 2.4.3 were labeled in accordance with the clustering described in Section 2.5.1 (see Figure 2.3). Once the segments were labeled, they were partitioned into a training set (75% of the samples) and a test set (25% of the samples). Each ANFIS cluster was trained and tested with the same set of data since they were designed to decide whether the input data belong to the cluster they represent. Consequently, membership was represented with a value of "1" if the data belonged to the cluster modeled by that ANFIS, and with "0" if not.

As can be seen in Figure 2.5, THW_{RMS} and TETH values were the same for all the clusters, and despite this figure being a 3D plot and TITH not being able to be displayed, it also coincided for all three ANFIS models. It could also be observed that the value of the Z-axis was "1" when data points belonged to the cluster to be modeled, and "0" when not.

ANFIS Testing and Identification Results

After the training stage, the remaining 25% of the steady car-following segments (see Section 2.4.3) are used to test the identification performance of the ANFIS-based DS identifier. Thus, the test segments were simultaneously input to the three ANFIS-clusters, and the neuro-fuzzy system with the highest output was considered to be the class to which the segment belonged. With this procedure, the outputs of the system were evaluated, showing an accuracy mark of 95.45%. This mark was much higher and the classification more detailed than those obtained in previous works from other authors, such as [163], where, using an SVM, a given driver's DS was classified between only two clusters with an accuracy of 85%.

TABLE 2.1: Confusion matrix of ANFIS-based DS identifier.

Actual/Identified	Cluster 1	Cluster 2	Cluster 3
Cluster 1	6	0	0
Cluster 2	0	27	0
Cluster 3	1	1	9

Additionally, the confusion matrix in Table 2.1 gives deeper insight into this accuracy result. By analyzing this matrix, accuracy rates of 85.71% for Cluster 1, 96.43% for Cluster 2, and 100% for Cluster 3 were reached by the final ANFIS. Nevertheless, confusions reflected in the matrix happened between contiguous clusters; hence, no erratic classification behavior happened while identifying DSs with this system.

2.6 Implementation of the FPGA-Based Intelligent Sensor

The block diagram of the FPGA-based implementation of the intelligent sensor for personalized ADAS is depicted in Figure 2.6.

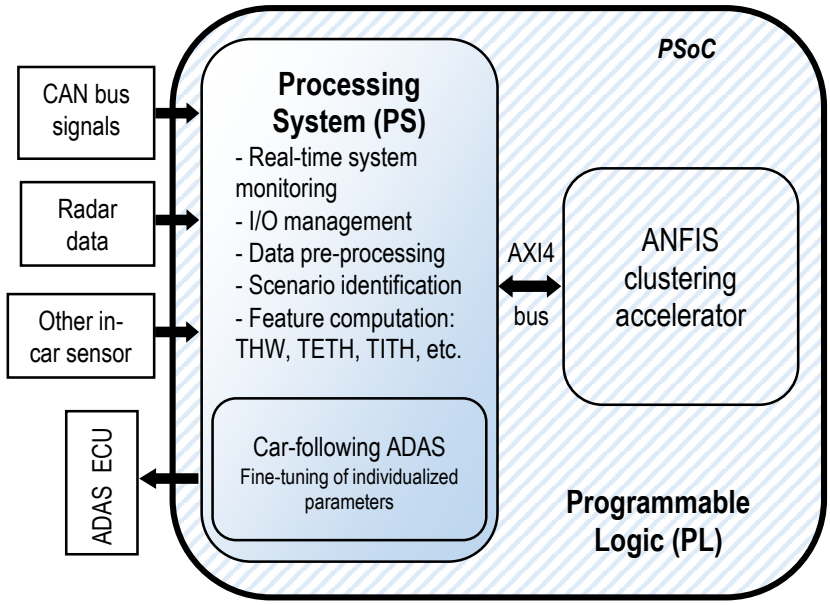


Figure 2.6: Block diagram of the FPGA-based intelligent sensor for online car-following ADAS.

It is a hybrid HW/SW architecture implemented on the Xilinx XC7Z045-2FFG900 PSoC [147] using the Xilinx ZC706 development board [148]. The entire HW partition of the system (deployed in the PL of the PSoC) was implemented using VHDL language and the Xilinx Vivado 2018.1 design suite. On the other hand, the remainder of the proposed system with its functionalities was programmed at the PS by developing a bare-metal C application that can acquire vehicle bus data; compute the THW, TETH, and TITH features; share them with the PL; retrieve the ANFIS accelerators' results; compute the personalization parameters; and send them to the ACC electronic control unit (ECU).

Considering the characteristics of PS and PL, the distribution of the tasks to be performed by the proposed system is explained in the following lines.

The PS, apart from performing global system monitoring, was intended to be connected to the vehicle systems through field buses, managing the I/O interface of the vehicle's systems with the proposed solution. Therefore, the PS was responsible for capturing the input data from both the radar sensor and the standard information from the CAN-bus. Regarding input data signals, the system is fed with the distance with the preceding vehicle and relative speed between the host vehicle and the preceding one (both from the radar sensor), and with the host

vehicle speed (standard information from the CAN-bus). With those signals, the PS computes the driving features that allow to perform DS classification. These features, stated in Figure 2.6, are computed according to Equations (2.1), (2.3) and (2.4) from Section 2.4.1. Inside the PSoC, once the features have been computed, they are sent from the PS to the PL by means of an internal bus, on which the entire PSoC architecture is based: the AXI4-bus. According to features sent from the PS to the PL through the AXI4 bus, the VHDL-based PL-implemented ANFIS clustering accelerator classifies DS and sends the classification results back to the PS through the AXI4-bus. With the received classification results, the PS computes the ACC personalization parameter and, finally, sends it through the vehicle's CAN-bus to its ECU, responsible for the ACC.

Thus, in summary, and according to Figure 2.6, the PS executes the tasks of interfacing with the vehicle buses to collect the input data, computes selected identification features, sends them to the ANFIS accelerator deployed in the PL through AXI4 bus, collects the outputs of the accelerator, computes the personalized ACC parameters, and sends them to the ACC module. On the other hand, as it is computationally intensive, the PL implements an ANFIS-based classification that needs to be executed as fast as possible.

2.6.1 Hardware Partition: ANFIS Accelerators

The building blocks on which the ANFIS HW accelerators rely were structured according to the Takagi–Sugeno Inference System (Equations (1.15)–(1.17)). After several tests, system inputs (THW^0_{RMS}, $TETH^0$ and $TITH^0$) were represented using an 8-bit fixed-point fractional data format, the bit widths of the intermediate operations were properly propagated and trimmed for not losing precision, and output y was trimmed to a 32-bit two-complement fixed-point representation, chosen to match with the AXI4 bus width.

As can be seen in Figure 2.7, the proposed ANFIS architecture was organized in four layers. In the first layer, the membership of the system inputs to the antecedents of the rules were evaluated. Then, in the second layer, rule activations were concurrently computed (Equation (1.15)). Next, in Layers 3 and 4, the weighted average of the consequents was calculated (Equation (1.16)). The HW partition was composed of three ANFIS accelerators, one per cluster.

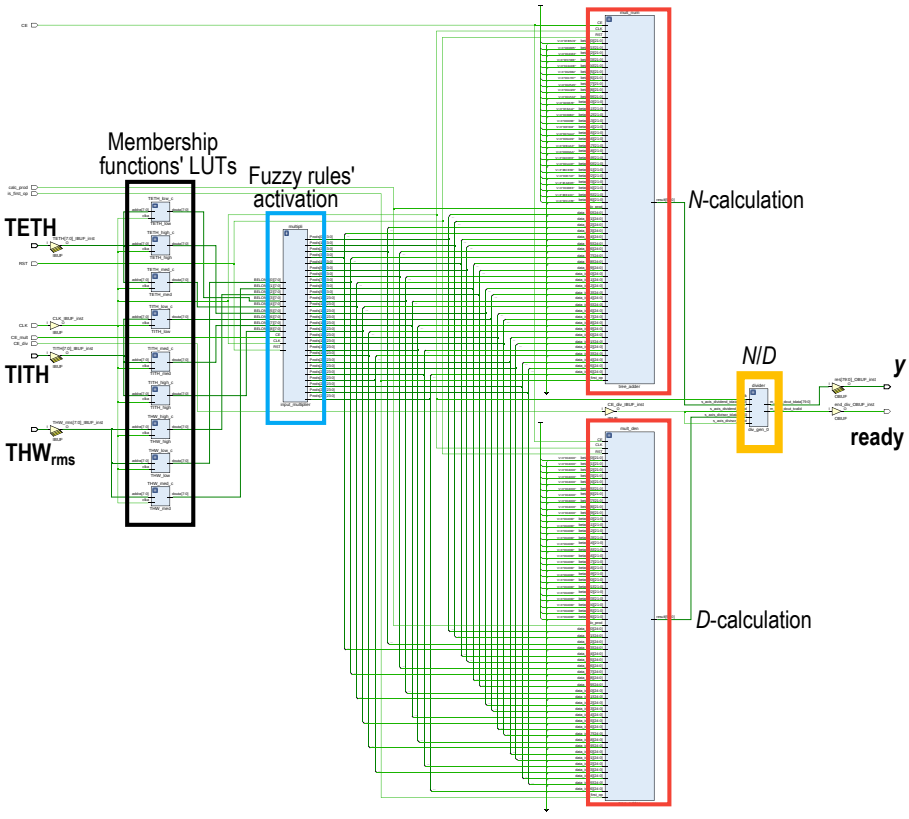


FIGURE 2.7: Block scheme of the parallel architecture of a three-input ANFIS implemented in the PL of the PSoC. Three ANFIS cores, one per cluster, were implemented in the HW partition.

Membership Function Evaluation and Fuzzy-Rule Computation

The generalized bell-shaped MFs were precalculated and stored as LUTs (remarked in black in Figure 2.7) at the PL block RAMs (BRAMs). Therefore, evaluation of the input membership to each antecedent was straightforwardly obtained by addressing those values, lasting only one clock cycle. Once the input MFs were evaluated, fuzzy-rule activations were calculated. As there were three MFs for each of the three inputs, 27 weights were to be computed. These fuzzy-rule activations were three-input products, computed by a fuzzy-rule activation module remarked in blue in Figure 2.7. These products were efficiently computed by a full-VHDL design intended to only use Xilinx DSP resources [146], improving timing performance. To achieve this DSP-only implementation, the three-input products were done two by two. Thus, first the

product of two of the inputs was calculated and stored in an intermediate result pipeline register and second, the stored partial product was multiplied by the remaining input and saved in the output register. This product pipeline required two clock cycles.

Computation of Sum and Weighted Sum of Rule Activation

In [181], a high-performance product sum architecture, developed by the authors, was described. This topology, shown in Figure 2.8, replaced the tree-adder architecture. It was intended to minimize latency, save resources, and minimize the number of used DSPs. Thus, for a given number M of products, the proposed architecture only used M multiplier/adder blocks, while a tree adder would spend $2M-1$ of the same HW resources. This architecture, with inputs $\mathbf{u} = (u_1, \cdots, u_M)$ and $\mathbf{v} = (v_1, \cdots, v_M)$, control signals `is_prod` and `CE`, and output p operates as follows.

1. Product signal `is_prod` set to "1" and all registers are reset.
2. Product $u_i \cdot v_i$ with $i = 1, ..., M$ computed and stored in each of the M accumulator registers.
3. Signal `is_prod` set back to "0" and first accumulation is performed. Thus, accumulator registers from 1 to $M/2$ contain the sum of $u_i \cdot v_i$ products. Registers from $M/2+1$ to M are now filled with zeros.
4. Successive $\lceil log_2 M \rceil$ accumulations are performed until valid result is present in register 0.

Two instances of this core, labeled D and N in Figure 2.7, are used to perform the computation of the sum and the weighted sum of rules' activation parallelly (D and N in Equation (1.16), respectively). In both modules, M equals the number of rules, that is, $M = 27$. For the N-module, $u_i = w_i$ and $v_i = c_i$, whereas for the D-module, $u_i = w_i$ and $v_i = 1$. A ROM storing the values of c_i is connected to the N-module. The latency of this architecture is $\lceil log_2 M \rceil + 2$; thus, with $M = 27$, the latency of both instances is seven clock cycles.

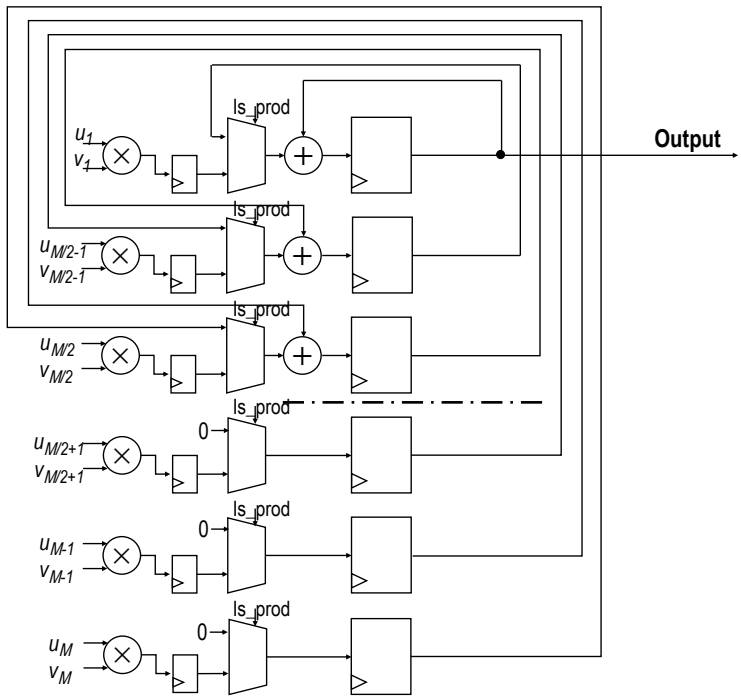


FIGURE 2.8: Scheme of proposed sum of product architecture that substitutes traditional tree-adder solution.

Divider Module

This is the last layer of the ANFIS accelerator. The divider module was elaborated by means of the Xilinx IPCore divider generator [182]. This IPCore was parameterized to match with the size of the N and D operands using the high-radix division implementation. This particular implementation could be pipelined to achieve good time performance and, as it depends on multiply–accumulate operations, it is optimally deployed in DSP blocks. With the selected word lengths and pipelined, this module requires 43 clock cycles to return valid results.

Parameterization and Control Signals

The complete structure of the ANFIS HW accelerator is parametric and fully customizable. LUT ROMs containing MFs as well as consequents were simultaneously initialized. Elements such as type depths, signal bit-widths, and number of inputs, number of membership functions, or number of fuzzy rules were defined on a standalone package. The

complete ANFIS was controlled by the sequence of control signals represented in the chronogram of Figure 2.9.

Control signals of the ANFIS HW accelerator were `rst`, `CE_mult`, `CE`, `is_prod`, and `CE_div` (see Figure 2.9); they worked as follows.

1. `rst` clears pipeline registers and multiplication–accumulation units.
2. `CE_mult` drives multipliers of fuzzy-rule calculation.
3. `CE` activates multiplication–accumulation units to iteratively compute N and D.
4. `is_prod`, in conjunction with the first cycle of `CE`, is used to indicate that the multiplier–accumulation unit must store the products of the fuzzy rules by their corresponding consequents instead of performing any accumulation.
5. `CE_div` triggers divider module calculating the output result of the ANFIS.

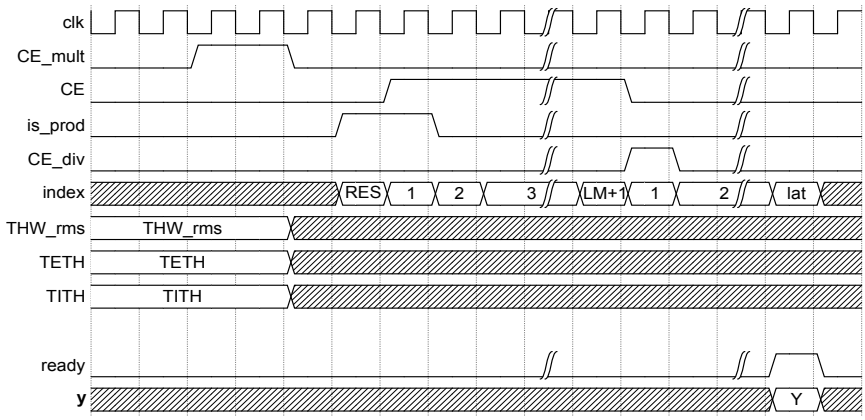


Figure 2.9: Chronogram of the control-signal sequence of the ANFIS core. LM stands for $\lceil log_2 M \rceil$

Finishing with the ANFIS HW accelerator implementation, it is worth noting that the three clusters of Section 2.5.1 must be modeled by this method. Consequently, three instances of this HW accelerator had to be deployed in the PL, each one configured with the parameters of the ANFIS cluster to which it corresponds.

2.6.2 Experiment Results

The three ANFIS cluster HW accelerators were implemented in the selected PSoC, achieving the subsequent results.

Resource Usage

The full HW system was successfully implemented, with the postimplementation results displayed in Table 3.9. The three-ANFIS system fit into the selected PSoC's logic, leaving enough resources available for further system applications, escalations, or improvements.

Table 2.2: Post-implementation resources report (Xilinx XC7Z045-2FFG900).

Resource	Utilization	Available	% Used
LUT	13 500	218 600	6.17
FFs	15 759	437 200	3.60
RAM blocks	15	545	2.76
DSP	294	900	32.76

Timing Performance

Before the deployment, the maximum operational frequency was calculated. For that purpose, three-ANFIS' architecture was implemented with a minimum clock period of 10 ns, obtaining a slack of 2.878 ns. Thus, the maximum operational frequency of the design can be calculated as follows:

$$F_{max} = \frac{1}{T_{imp} - d_{slack}} = \frac{1}{10\text{ ns} - 2.878\text{ ns}} = 140.41\text{ MHz}. \tag{2.5}$$

With the maximal clock frequency of Equation (2.5), the designed HW implementation could be used as an AXI4 peripheral dependent on an AXI4 bus clock frequency of 100 MHz. This design delayed 53 clock cycles (530 ns at $F_{CLK} = 100$ MHz) to return the computed outputs. These results outperformed the timing obtained for the full-SW PC-based (20-core Intel Xeon CPU E5-2630 v4 at 2.20 GHz with 32 GB of DDR4 RAM) MATLAB model design, with top performance peaks of 1.829 ms to compute the same set of 3 ANFIS, as well as a PC-based, C-coded prototype that achieved timing marks of 12.45 µs.

The obtained timing was better than in other FPGA-based ANFIS approaches, such as the work by the authors of [183], where timings

of ~12 µs were obtained in the computation of a system with the same number of inputs and outputs (three and one, respectively) as that developed in this chapter. On the other hand, in [184], a novel ANFIS HW architecture, able to reduce the timing mark of 530 ns achieved in the present work more than 50%, was presented. Recently, several innovative architectures on other ML algorithms have been proposed with the aim of achieving extreme timing performance results. Examples of these innovations are a HW implementation of a radial-basis function (RBF) network, for which operational frequencies of up to 450 MHz for high bit-width inputs were achieved [185], and an SVM implementation able to be run up to 20 times faster than other state-of-the-art techniques [186].

Consequently, the hybrid HW/SW implementation developed is an innovative solution between conventional SW-based approaches and novel FPGA-based, extreme performance architectures, which provides an adequate trade-off between complexity, performance, and development time.

ACC Personalization Application

The particular example of ANFIS Cluster 1 is displayed in Figure 2.10. In this figure, each column depicts the MFs for each input of the ANFIS system (THW^0_{RMS}, $TETH^0$, and $TITH^0$, and output y, respectively), whereas each row corresponds to a fuzzy rule. Thus, for the selected example, with $THW^0_{RMS} = 0$, $TETH^0 = 0.5$, and $TITH^0 = 0.18$, ANFIS Cluster 1 returned an output value $y = 0.965$. Since this value was close to 1, and the input data fulfilled the description of Cluster 1 in Section 2.5.1, this ANFIS correctly identified this value as a member of the cluster it modeled. Additionally, this datapoint was input to the ANFIS of Clusters 2 and 3, with returning output values of 0.017 and 0.31, respectively. As a result, considering the maximum output value of the three ANFIS, the system successfully classified the inputs as Cluster 1.

Regarding ANFIS HW accelerator verification, in Figure 2.11, a simulation of the ANFIS Cluster 1 HW accelerator is depicted. As can be seen in this figure, with the same input values, the system returned output $y = 0.958$. The results obtained with the HW accelerator agree with Figure 2.10. The outputs of the three ANFIS clusters were recovered through the AXI4 bus by the SW programmed in the PS partition of the PSoC. The PS determined which the highest recovered value was,

hence identifying the corresponding cluster.

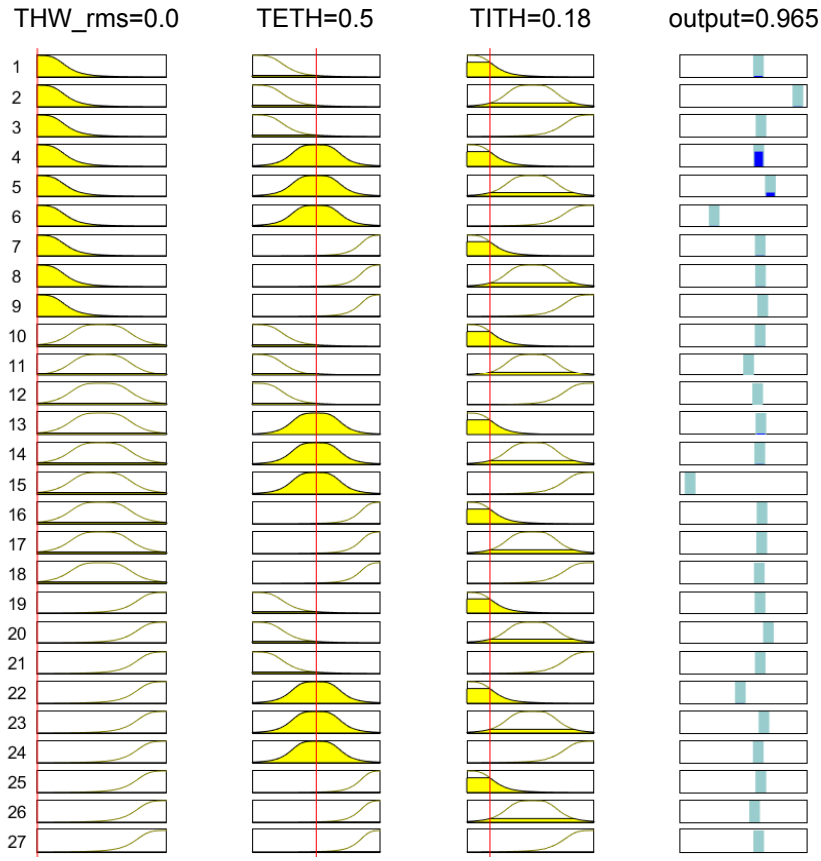


FIGURE 2.10: Rules and MFs of ANFIS Cluster 1 for a given input.

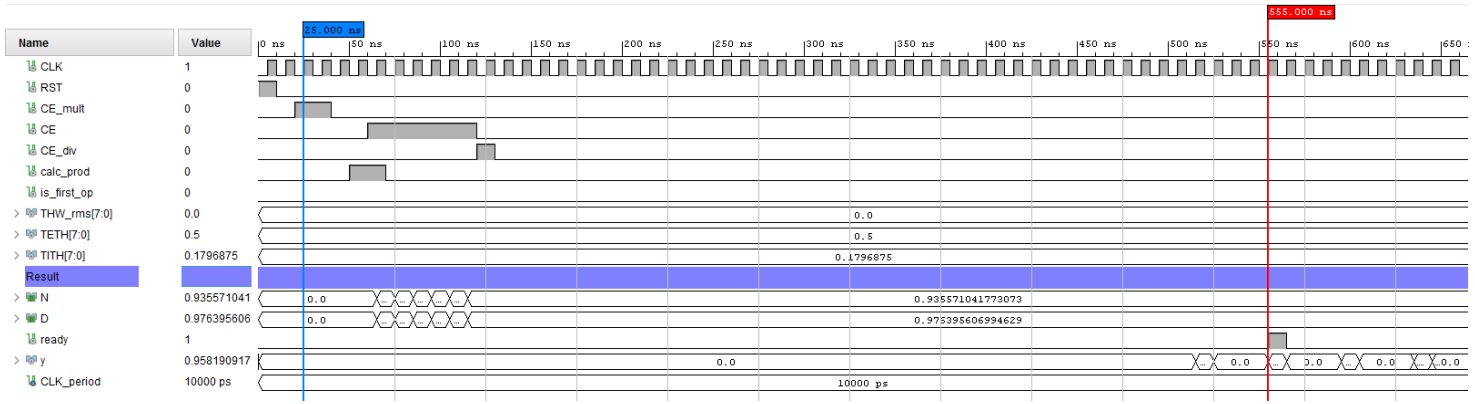


FIGURE 2.11: Simulation results of the ANFIS Cluster 1 HW accelerator obtained with the Vivado design suite. This simulation replicates the chronogram of Figure 2.9. The latency is measured from the blue marker to the red one. $F_{CLK} = 100$ MHz

Software Partition: Individual-Based Personalization for ACC ADAS

Once the DS classification of each individual has been performed, the last step to be performed is the automatic tuning of the THW each user wants to keep. For that purpose, the SW partition in the PS implemented a plane-shaped that models a continuous range of possible THW for each cluster depending of the characteristic car-following parameters of each driver. Thus, given three clusters $1 \leq i \leq 3$, the ith plane was defined, such that

$$\widehat{THW}_i = f\left(\overline{THW}_{RMS_i}, TITH_i\right), \tag{2.6}$$

where $\widehat{THW}_i$ is the individualized THW adjustment, $\overline{THW}_{RMS_i}$ is the average THW_{RMS} value observed during the learning period of the system for a particular driver in a steady car-following situation, and $TITH_i$ is the normalized TITH value for the same period.

These planes were defined by the three-point method depending on the minimal, maximal, and average values for THW_{RMS} and TITH for each cluster according to Figure 2.4. With these distributions, the $\overline{THW}_{RMS_i}$ and σ_i for each cluster were:

- Cluster 1: $\overline{THW}_{RMS_1} = 1.08$ s, with $\sigma_1 = 0.27$ s.
- Cluster 2: $\overline{THW}_{RMS_2} = 2.44$ s, with $\sigma_2 = 0.59$ s.
- Cluster 3: $\overline{THW}_{RMS_3} = 1.61$ s, with $\sigma_3 = 0.17$ s.

For each cluster, the point of minimum THW_{RMS} and maximum TITH were assigned with a value of $\widehat{THW}_{RMS_i} = \overline{THW}_{RMS_i} - \sigma_i$, as it corresponded to drivers from that cluster who like to drive with a shorter time gap. On the other hand, drivers who would rather drive with longer time gaps (that is, those who are represented by the point of maximum THW_{RMS} and minimum TITH), have a value of $\widehat{THW}_{RMS_i} = \overline{THW}_{RMS_i} + \sigma_i$ assigned. Finally, intermediate drivers (average values of THW_{RMS} and TITH), have a value of $\widehat{THW}_{RMS_i} = \overline{THW}_{RMS_i}$. Consequently, three points $(\overline{THW}_{RMS_i}, TITH_i, \widehat{THW}_i)$ that defined each THW-modeling plane i are as follows.

- $p_{i1} = (min(THW_{RMS_i}), max(TITH_i), \overline{THW}_{RMS_i} - \sigma_i)$
- $p_{i2} = (max(THW_{RMS_i}), min(TITH_i), \overline{THW}_{RMS_i} + \sigma_i)$
- $p_{i3} = (\overline{THW}_{RMS_i}, \overline{TITH}_i, \overline{THW}_{RMS_i})$

With these considerations, the planes modeling the individualized $\widehat{THW}_i$ for each cluster according to Equation (2.6) were computed and shown in Figure 2.12.

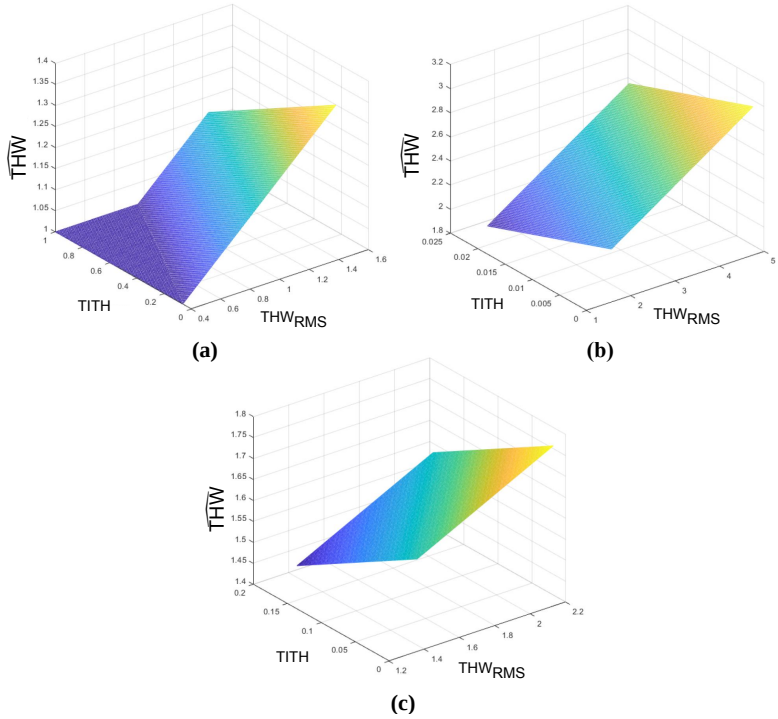


FIGURE 2.12: $\widehat{THW}_i$ model planes for (**a**) Cluster 1, (**b**) Cluster 2, and (**c**) Cluster 3.

As can be seen in Figure 2.12, for each of the models, the predicted $\widehat{THW}_i$ was directly proportional to $\overline{THW}_{RMS_i}$ and inversely proportional to TITH. Note that, for the Cluster 1 model, the plane was saturated to $\widehat{THW}_i = 1$ s to assure that the personalized THW value never took a value lower than the minimal safe THW values [53].

In sum, once the ANFIS accelerator identified the cluster for a given driver, one of the three models in Figure 2.12 was selected. Then, an individualization stage measured and computed the THW_{RMS} and TITH during a steady car-following period. Finally, with those measurements, the system evaluated the corresponding plane model and set a personalized THW value for the ACC system (see Figure 2.6).

2.7 Concluding Remarks

Throughout this Chapter, an ML approach to face the challenges of ADAS personalization was proposed. It is based on a hybrid personalization strategy for DS modeling that uses a group-based clustering technique, namely, a k-means clustering with an individual-based model that adapts the parameters of the clusters to an individual driver. This solution introduces personalization strategies that need no driver intervention with the aim of easing the use of ADAS and thus promoting their adoption in daily driving, with the ultimate goal of increasing road safety and reducing traffic accidents. The DS clusters developed in this piece of research are representative of car-following behavior obtained with a meaningful sample of drivers from an NDS in different kinds of roads, weather conditions, and lighting. Nevertheless, they can easily be extended to account for the requirements of particular groups of drivers, mainly the most vulnerable drivers (e.g., elderly or inexpert drivers). In addition, a similar approach could be used to personalize and improve current ADAS through different spotlights, such as the fuel economy or passenger comfort.

The implementation of a single-chip driving personalization system for in-car integration requires a high-speed clustering model. The adopted solution relied on high-performance approximation of the clusters using an ANFIS. The universal approximation capability of ANFIS with its inherently parallelizable layered topology make this model suitable for efficient HW implementation. The whole neuro-fuzzy sensor was successfully implemented using an FPGA device of a Xilinx ZynQ-7000 PSoC providing high speed and low-power consumption for real-time ADAS implementation. In addition, due to the reconfigurable nature of FPGAs, both the HW and the SW partition of the PSoC could be updated to cope with the continuous changes that new vehicle technologies introduce.

Chapter 3

Real-Time Assessment of Fuel Consumption to Promote Eco-Driving

3.1 Overview

In this chapter, we develop an eco-driving assistance system based on DS characteristics. These characteristics are extracted from data of the Uyanik instrumented car, as described in Section 1.3.1. Based on these data, a realistic fuel-consumption model is developed by means of the Gtisoft GT-Suite [187], for verification purposes. The automatic characterization of DS is performed by means of SOMs (see Section 1.2.5). For this application, this unsupervised ML algorithm is able to automatically group driving behaviors, since it relies on a two-dimensional representation of a high-dimensional complex system, known as a map, which is suitable for a qualitative evaluation of multiple driving behavior features. The car-boarded SOM assessment solution, able to give natural language-based DS improvement recommendations, is deployed utilizing hybrid HW/SW implementation based on a FPGA-based Xilinx ZynQ-7000 PSoC, which, according to the statements in Section 1.4.2, enables to reach real-time performance rates.

The remainder of this chapter is organized as follows. Section 3.2 introduces the concept of eco-driving and puts it into context. Section 3.3 describes the followed development strategy. In Section 3.4, the driving behavior characterization methods for fuel-consumption scenarios are presented, including the selection and obtainment of relevant driving features, while Section 3.5 presents experimental results concerning the fuel consumption assessment and emission reduction. Section 3.6 exposes the implementation and validation of a hybrid HW/SW

PSoC-based fuel-consumption reduction and eco-driving advice system. Section 3.7 summarizes concluding remarks.

3.2 Eco-Driving Approaches

In the current society, traffic restrictions and environmentally friendly means of transportation are a reality. However, despite authorities gradually passing more restrictive environmental regulations, such as low-emission zones (LEZs) [188], or automotive ecological ratings, their effects on reducing GHGs have been found to not be as significant as expected [67]. In that sense, it has been observed that individuals' DS plays a more important role in emitting polluting agents than the ecological rating of the vehicle itself, with studies showing that, in different situations, aggressive driving could increase energy consumption by 47% [189]. With these assertions in mind, it seems reasonable that if we could assess the fuel-consumption efficiency of individuals, their DS could be corrected in order to increase their ecological friendliness (refer to Section 1.1.4).

Eco-driving is mainly an operational decision that allows drivers to maximize fuel efficiency and reduce pollutants' emissions. It is characterized by the use of several techniques that help to maximize the vehicle's energy efficiency. Therefore, this concept can be seen as a set of rules that differ from the driving that motorists are used to performing, including calm driving, the avoidance of unnecessary stops, and the anticipation and elimination of idling when possible. Several authors remark that eco-driving could effectively contribute to reducing overall fuel consumption and CO_2 emissions if adequate education about strategic, tactical, and operational decisions were provided to drivers [190]–[192]. In this sense, during the trip, and when the trip has finished, providing practical recommendations might be useful.

In the most commonly used form of eco-driving measures, drivers are given advice in training sessions, and the organizers measure differences in fuel consumption and CO_2 emissions before and after training [193]. Another valid approach is providing a report of the strengths and weaknesses after each eco-driving session [194]. Nevertheless, a natural evolution on those lessons provides instantaneous feedback of the driver's operational decisions [194]. It has been found that on-trip eco-drive support is more efficient, with reductions of up to 10% on fuel consumption when compared to post-drive assessment (which only achieves a 5% reduction) [191]. However, the former is more expensive

and it requires complex algorithms as well as real-time technology dependence, while the latter can be provided through an end-of-trip fuel consumption assessment [191].

For that reason, several attempts of fuel economy-intended system implementations, acting on the aforementioned parameters, such as gear recommendation [195] or eco-driving scoring, have been deployed in cars [196]. These systems, despite achieving the objective of reducing the polluting agents' emissions, with rates of 1.63% in case of gear recommendation and 3.63% for eco-driving scoring, have not been proven to be effective enough [195]. Consequently, and given that a personalized assessment of ecological behavior might help motorists to achieve outstanding fuel consumption results, with reductions up to 18.4% [197], providing online DS recommendations seems reasonable. These recommendations must be based on each individual's driving behavior, and they are intended to re-educate drivers if they follow incorrect driving patterns (e.g., aggressive driving) like a human instructor would do.

To carry out that task, ML techniques, such as fuzzy-logic, introduced in the precedent Section 1.2.4, have been used to give coaching feedback to the driver about his/her performance [198]. Some other approaches uses ANNs (refer to Section 1.2.3) to differentiate drivers that are classified among a plethora of driving behaviors, cycles, and scenarios, successfully distinguishing between aggressive and defensive behaviors and urban and highway driving [199]–[202].

With all these considerations in mind, the main motivation of this chapter is that several limitations regarding eco-driving assessment systems have been identified. (1) Current in-car systems are intended for generic driving recommendations reporting reduced effectiveness. (2) Most personalized driving assessment systems are based on training sessions and fuel-consumption improvement tracking, normally after the driving sessions, achieving low percentages of fuel economy. (3) ML techniques have been successfully applied in order to classify DSs into several aggressiveness categories; however, the full potential of these techniques is still unexploited. (4) Most of the existing works in this field do not analyze the handling operations of the driver on the car commands that cause fuel consumption to rise. (5) Providing online DS-based handling recommendations to improve fuel economy is still a mostly unexplored path.

In contrast with the vast majority of works, not only does this proposal classify DS into two or three aggressiveness categories, but it also analyzes driving behavior by identifying up to five different DSs. This detailed analysis allows for an insight into the concrete causes of driver-

associated high fuel consumption and, consequently, provides personalized DS recommendations to re-educate drivers for eco-friendlier handling.

Thus, we present the following contributions to the development of an eco-driving assessment system that is able to provide real-time personalized advice:

- New applications of unsupervised neural networks to discover particular driving patterns and analyze the effect of driving patterns in fuel-consumption.
- A novel approach for the examination of the underlying causes of different types of non-optimal DSs from the eco-driving viewpoint and analysis of the fuel-economy-compromising command operations.
- Personalization of the provided advices when considering the aforementioned points. Those advices comprehend instructions to improve the use of the gas and brake pedals as well as advice on the shift of the selected gear.
- Improvement in the performance of the already-existing systems, with expected enhancements in both fuel consumption and emissions ranging from the 9.5% to the 31.5%, or even higher for drivers that are strongly engaged with the system.
- Development of a SoC with real-time responsiveness.

3.3 Outline of the Overall Eco-Driving System Design

A system that promotes behavioral adaptations leading to eco-driving is more desirable to encourage drivers to fulfill those requirements, as stated in the preceding sections of this text. For that purpose, a real-world data-driven approach was selected. The data, used to develop a strategy that identifies eco-driving classes regarding an individual DS, provide eco-advice based on learning generated from data.

Figure 3.1 shows the development strategy that was used in this chapter. It combines several algorithms and tools in a multi-stage fashion. Five stages can be clearly identified from the raw data itself to the obtainment of a hybrid HW/SW integrated system to recommend drivers about changing their DS.

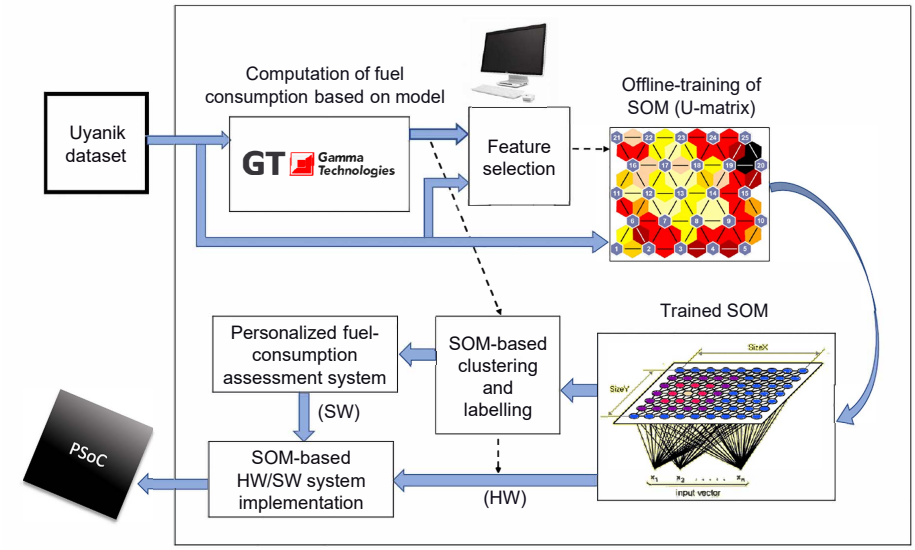


FIGURE 3.1: Offline sequence of tasks involved in the design and development of a self-organized map (SOM)-based intelligent system for fuel consumption assessment. The dotted arrows indicate that the simulated fuel consumption data are also used to label the SOM-based clustering for verification purposes and to elaborate the HW implementation of the SOM.

The first stage comprises the use of several resources from two different sources: the Uyanik car dataset and GT-Suite simulation tool [187]. This simulation allowed us to obtain the fuel consumption flows that the Uyanik dataset lacked. Afterwards, the most relevant features as well as the optimal data window size were selected. Then, a SOM was trained in a completely unsupervised manner, that is to say, the GT-Suite data were not used during the offline-training step, as detailed in Section 1.2.5. The third stage consists of performing both a quantitative and qualitative analysis of the trained SOM, so that several groups are identified and labeled according to the mean fuel consumption that was obtained by simulation with GT-Suite. After that, with the properly labeled clusters, meaningful three-dimensional plots of the selected features were analyzed with the aim of discovering the aspects that each DS group can improve so that several fuel-consumption-compromising circumstances were identified and, according to them, concrete actions were developed. Finally, the personalized fuel consumption assessment system was developed and implemented on a PSoC of the Xilinx ZynQ family by means of the VHDL HW description language and the Xilinx Vivado 2018.1 design suite [203].

3.4 Driving Behavior Characterization for Fuel-Consumption Scenarios

Energy consumption and carbon dioxide emissions of passenger cars are affected by a combination of human, environmental, and technological factors, according to a recent report of the Joint Research Centre of the EU [204]. Human factors refer to driving behavior, that is to say, the driving patterns that an individual driver or a group of drivers follows, such as acceleration, mean speed, and preferred engine gear. The main environmental factors include both weather conditions (i.e., ambient temperature, rain, and wind) and actual characteristics of the road (i.e., morphology, surface quality, and traffic conditions), while technological factors refer to the vehicle type and its characteristics.

In this chapter, we focused on the consequences of the DS on fuel consumption, so the human factor had to be isolated as much as possible from the remaining factors [205], [206]. With the aim of fulfilling the above requirement, we considered a group of drivers exhibiting different driving behaviors while driving the same car, along the same route, and in similar environmental conditions. It is worth noting that the latter factor, mainly traffic conditions, is the most difficult feature to reproduce in live traffic.

3.4.1 Selection of Relevant Features

The dataset used in our experiments was collected using an instrumented car traveling a fixed route around the city of Istanbul, as already introduced in Section 1.3.1. The route is little over 25 km and lasts about 40 min, depending on weather and traffic conditions. It includes different types of road sections: city, very busy city, highway, and a university campus. With the aim of minimizing the impact of environmental variations, all of the selected driving sessions were conducted in a short period of time, from August to October, and during the central part of the day, between 11 a.m. and 4 p.m. The driver population was composed of 20 drivers, 17 male and three female, whose ages ranged from 21 to 61. This is a reduced subset of a more comprehensive data collection (i.e., about 100 drivers and a single trip per driver) provided by the Drive-Safe Consortium [14].

Because instant fuel consumption was not available within the dataset, we developed a model of the Uyanik car and used the GT-Suite tool to obtain fuel consumption data during the driving sessions [207]. Afterwards, we computed two types of features: mean values and variances

of the Uyanik signals. The whole set of time series, more than 30 independent signals, was evaluated, including CAN-bus data, pedal pressure sensors, a laser scanner, and IMU unit readings. In particular, the treatment of the X-axis acceleration variables was divided into two parts, positive and negative values, since they have different consequences on fuel consumption. In fact, negative instantaneous values are associated with zero consumption.

Subsequently, the features that provide the highest relationship with fuel consumption were selected, while the irrelevant or redundant features were discarded. We computed both the Pearson correlation coefficients (PCCs) and the p-values of every feature. The former provides a measure of the relevance of each feature, while the latter is used for testing the hypothesis of no correlation (i.e., the probability of obtaining a correlation as large as the observed value by random chance, when the true correlation is zero). The features were computed over 8 s windows (i.e., 256 samples) with a 4 s shift. That is to say, the overlapping between consecutive windows is 4 s (i.e., 128 samples). The format of the windows was selected by exhaustively analyzing the consequences of both the window size and the shift on the PCC of the most relevant features.

Table 3.1 summarizes the set of low level signals (i.e., time series) that exhibit the strongest correlation with fuel consumption. Moreover, the p-values are less than 0.0001 for almost all of the features included in this table, thus guaranteeing the reliability of the correlations. The exceptions are the mean and variance of the negative X-axis acceleration, whose p-values are close to 0.05. These features were not selected because of their low PCCs.

The features that present the strongest correlations with fuel consumption are remarked in bold in Table 3.1. Four mean values (i.e., VS, PGP, ERPM, and GP) and the PGP variance have a strong positive correlation with fuel consumption, while the positive X-axis acceleration presents moderate correlations, as can be seen. On the other hand, BP and negative X-axis acceleration, both mean and variance, exhibit negative correlation coefficients. This means that an increase in BP or in X-axis deceleration is associated with a decrease in fuel consumption. Although these features are meaningful concerning driving behavior analysis, their correlations with fuel consumption are rather weak.

TABLE 3.1: Driving behavior signals and PCCs of fuel consumption with relevant features. Mean values and variances are computed using 8 s analysis windows.

Measurement Units	Signals (Time Series)	PCC: Mean	PCC: Var.
CAN bus	Vehicle speed (VS)	**0.59**	0.15
	Percent gas pedal (PGP)	**0.63**	**0.58**
	Engine RPM (ERPM)	**0.66**	0.18
Pressure sensors	Brake pedal pressure (BP)	-0.35	-0.23
	Gas pedal pressure (GP)	**0.52**	0.20
IMU unit	Positive X axis acceleration (XACC pos)	0.32	0.25
	Negative X axis acceleration (XACC neg)	-0.17	-0.11

Note: The boldface PCCs correspond to the strongest correlations.

Afterwards, we chose the most fuel-demanding sections of the Uyanik route, those that ran through highway and motorway, in order to develop the assessment system. Moreover, sections with traffic jams and slow traffic (i.e., mean speed below 60 km/h) were discarded with the aim of avoiding outliers during the training process, returning the itinerary shown in Figure 3.2. After limiting the type of road, the PCCs, as presented in Table 3.1, varied slightly. The most noticeable changes were a moderate reduction of the fuel consumption correlations with VS and a remarkable increase in the fuel consumption correlations with the mean and variance of positive X-axis acceleration. In view of these results, the latter features were also taken into account in a preliminary round of SOM training experiments. Thus, the following variables were selected as candidate features: mean values {VS, PGP, ERPM, GP, Pos XACC} and variances {PGP, Pos XACC}. A comprehensive series of training experiments revealed that a reduced subset of only four features is able to model the relationship between fuel consumption and

driving behavior in a very satisfactory way. These features were mean PGP, mean ERPM, mean GP, and Pos XACC variance.

Figure 3.2: Stretches of the Uyanik route used. An average of 9:27 minutes of driving is available for each driver.

3.4.2 Fuel-Consumption Obtainment by Simulation

An important step of this chapter is the obtainment of a meaningful set of fuel consumption data, as mentioned in precedent paragraphs. This step was found to be needed, since the Uyanik dataset lacked the ECU data regarding fuel injection or intake airflow.

Several alternatives were studied in order to obtain and measure fuel control unit data, finally choosing a simulation environment. We selected the Gamma Technologies GT-Suite environment [187], since it does not only allows element-by-element simulation of mechanical systems, but also enables users to run macroscopic approximated models of complete automobiles. Thus, while the former requires an exact parameterization of each mechanical element and link of the engine, the

latter allows us to fit a pre-elaborated model based on telemetry (such as speed, acceleration, brake, or selected gear) as well as on car manufacturer information, such as gear ratios, tire dimensions, or wheelbase (see Figure 3.3).

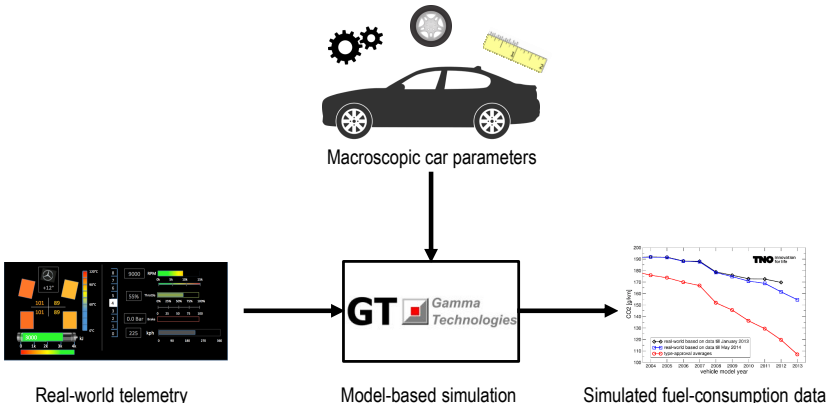


Figure 3.3: Flow of real-world telemetry-based fuel consumption simulation. It has macroscopic car parameters (gear ratios, tyre dimensions, and wheelbase) and telemetry (gas pedal, brake pedal, speed, selected gear, and accelerations) as inputs. The model returns the simulated fuel flow as output.

Finally, the gear ratios were computed while using data about RPM and VS, available for each instant. We computed the speed/rpm ratio sample-by-sample and matched it with each gear's ratio. When computed ratios did not match with any of the gear ratios, we assumed that the driver was operating the clutch pedal.

Once the car parameters were successfully extracted, we elaborated on the model that is displayed in Figure 3.4. In this model, four main elements can be identified for the car itself, the vehicle, transmission, engine, and ECU blocks, while the driver is modeled by another one. These blocks contain the characteristic parameters of their corresponding real-world counterparts.

- Vehicle comprises data regarding car wheelbase, wheel radius, friction coefficients, aerodynamics, weight, inertia, and final transmission ratio.
- Transmission incorporates individual ratios for each of the user-selectable gears, as well as clutch parameters.

- Engine consists of parameters such as engine displacement, engine type (4-stroke or 2-stroke), minimum operation speed, or fuel characteristics.
- ECU controls the maximum engine RPM, idle speed, and fuel injection cutoff and restart points.
- Driver wraps the telemetry data related to the handling of the car, such as selected gear, accelerator pedal state, brake pedal state, clutch, and desired speed.

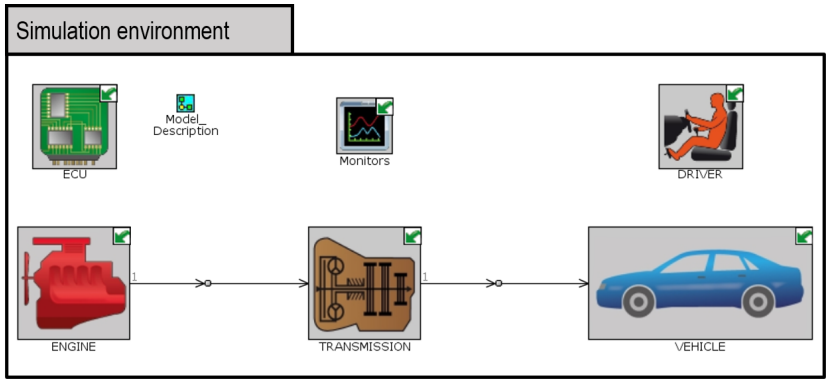


FIGURE 3.4: Block diagram of real-world telemetry-based fuel consumption simulation of Uyanik Renault Mégane 1.5 dCi Sedan 74 kW. It has macroscopic car parameters (gear ratios, tyre dimensions, and wheelbase) and telemetry (gas pedal, brake pedal, speed, selected gear, and accelerations) as inputs. The model returns the simulated fuel flow as output.

Several checks were performed on the simulation model in order to verify that the returned results provide an acceptable emulation of the real car performance. Thus, given a set of selected gears, as displayed in Figure 3.5(a), the application of the driver operation of the accelerator and brake pedals, along with the dynamics of the car restricted to a set of measured accelerations, brings out the simulated RPM and vehicle speed red curves of Figure 3.5(b),(c), respectively. As can be seen, these red curves are almost totally overlapped with the blue ones, which represent the real world-collected data, with relative errors of 1.83% for RPM and of 0.44% for speed. These low relative errors mean that the simulation faithfully emulates the real car behavior and, consequently, that the returned fuel consumption simulated data is useful for carrying out estimations in order to verify the proposed SOM-based models and extracting conclusions.

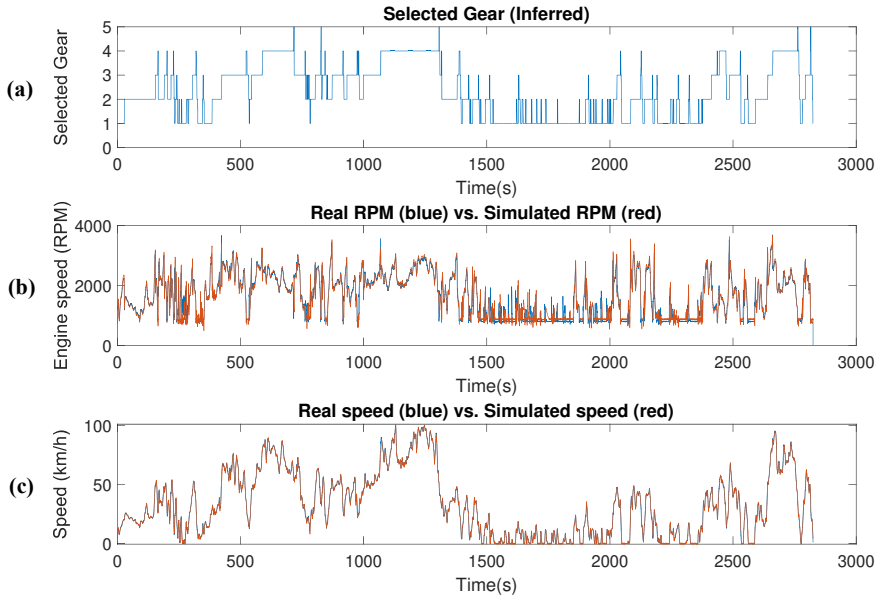
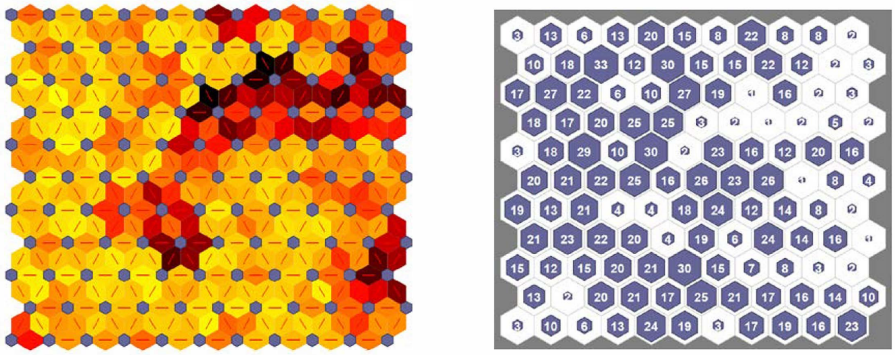


FIGURE 3.5: Comparison of measured data vs. simulation results. (**a**) The inferred gear considering the computed rpm/speed ratios. (**b**) The measured revolutions per minute (RPM) of the vehicle vs. the RPM simulated by the model. (**c**) The measured speed of the vehicle vs. the speed simulated by the model.

3.4.3 SOM-Based Drivers Grouping Regarding Fuel-Consumption

In Section 3.4.1, the most relevant features for fuel consumption characterization were selected: mean PGP, mean ERPM, mean GP, and the variance of positive XACC. Concurrently, window size and window shift were analyzed to preserve a high correlation between the above features and fuel consumption. The features were computed over 256-sample windows (i.e., 8 s) with 50% overlapping between consecutive windows. The number of available training windows per driver varies slightly between drivers, depending on their DS, traffic, and quality of the measurements. On average, there are 115 windows per driver, while the whole set of driving samples consists of 2290 windows (i.e., more than 2.5 driving hours). Three-quarters of the four-dimensional samples will be used to train an SOM, which is to say $K = 1717$, keeping the remaining quarter for testing purposes.



(a) (b)

FIGURE 3.6: This SOM organized into an 11 × 11 neuron grid. (**a**) Neighbor weight distances. The limit between neuron groups is reflected by a frontier of long distances (darker colors). (**b**) Sample hits. This image shows how many training samples are associated with each neuron. Neurons with higher distances to their neighbors show the lowest number of hits.

The number of output neurons of the SOM was initially set using Vesanto's rule [136], which defines the optimal number of neurons as $M = 5\sqrt{K}$. Thus, a 14 × 14 SOM topology (i.e., M=196) was defined and repeatedly trained. However, because the corresponding U-matrices showed overly smooth maps, the size of the SOM was gradually swept until a suitable map was obtained. Neuron maps ranging from 10 × 10 ($M = 100$) to 15 × 15 neurons ($M = 225$) were tried by using the Matlab Neural Network Clustering App [208]. The most robust and consistent results were obtained using 11 × 11 maps ($M = 121$). This training process, as detailed in Section 1.2.5, is completely unsupervised, that is to say, the GT-Suite data were not used during the offline-training step.

As the number of neurons in the map ($M = 121$) is less than the number of samples ($K = 1717$), most of the neurons in the map are the BMU or hit of several samples in the dataset (see Figure 3.6(b)). As can be seen, there are neurons across the map with 4 or fewer hits, which match the regions with dark neuron connections in Figure 3.6(a). These units could be considered to be interpolating neurons, smoothing the transitions between clusters.

SOM Classification Results

The above visualizations of the trained SOM can only be used to obtain qualitative information concerning driving behavior. Interesting groups of neurons (i.e., clusters) must be identified and labeled in order to develop meaningful quantitative descriptions of driving data, suitable for a real-time fuel consumption assessment. Although the clustering of the SOM can be performed by means of any unsupervised clustering method, such as K-means or hierarchical clustering, the U-matrix method was used in this chapter. This U-matrix can be evaluated mathematically. Thus, it is useful for identifying clusters both graphically and numerically (see Section 1.2.5). This tool helps to see the cluster structure of the map: high values of the U-matrix indicate cluster borders, while uniform areas of low values can be identified as potential clusters. The quantitative evaluation of the SOM was performed by means of the CIS SOM Toolbox for Matlab [136], which uses the k-nearest neighbors algorithm (k-NN) [136]. According to the identified clusters, as described in this section, several groups were identified and labeled according to the mean fuel consumption that was obtained by simulation with GT-Suite.

Three-Cluster Grouping First, we carried out a three-cluster grouping of the SOM neurons. Table 3.2 presents relevant statistical values of fuel consumption for each cluster: average value, variance, and maximum value. Taking these values into account, the clusters were labeled as Very Low, Low, and Medium-High. The classification results applied to the Uyanik dataset are shown in Figure 3.7, where three-dimensional views are provided.

The three displayed clusters are compact, their contained data are contiguous, and they are clearly grouped, as can be seen in Figure 3.7. Matching clusters with their associated consumption displayed in Table 3.2 by color, it is apparent that the green cluster corresponds to medium-high fuel consumption rides, the blue one to low consumption, and the red group represents very low fuel consumption rides. Additionally, by analyzing clusters' fuel consumption variances, it can be seen that the higher the average value, the higher the variance, with this correlation being a noticeable feature of the identified groups.

Further analysis of the relationships of the identified groups with the driving features displayed in each sub-figure can be performed. Regarding Figure 3.7(a),(b), the DS groups look similar, since the GP variable of (a) and the XACC var of (b) are highly correlated as a measure of swift operation of the gas pedal. On the other hand, Figure 3.7(c) shows

a different cluster distribution. In this graph, the Low and Very low consumption classes (blue and red, respectively) are interleaved. This happens because the correlation between XACC var and GP is strong, with GP vs. XACC var providing no additional meaningful information. In contrast, the PGP vs. GP and the PGP vs. XACC var planes show that the positioning of the clusters is interchanged with respect to Figure 3.7(a),(b). Nevertheless, this interchange is coherent with the precedent figures, since the green cluster is placed at the upper range of the PGP axis, while the other ones are at the lower range, the blue cluster is related to low GP, and the red one is related to medium GP. The same clusters' position interchange phenomenon can be observed in Figure 3.7(d), according to the aforementioned characteristics.

Table 3.2: Fuel consumption (L/100 km) parameters of the three-cluster classification.

Cluster Label	Avg. Value	Variance	Max. Value
Very low (red)	2.76	1.02	6.66
Low (blue)	3.04	1.53	7.80
Medium-High (green)	5.15	3.34	12.6

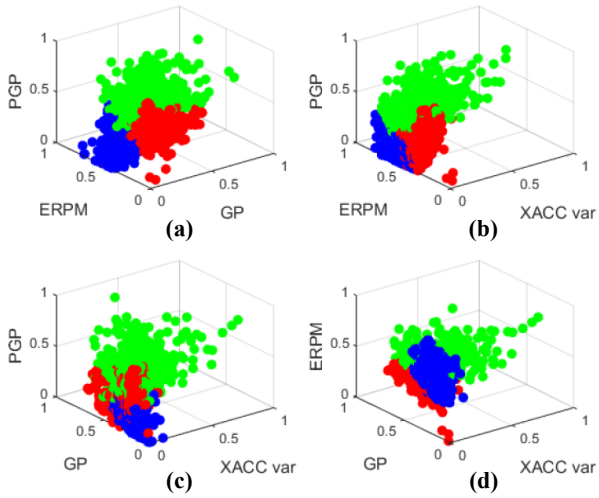


Figure 3.7: Three-dimensional views of the three-cluster fuel consumption classification results. The clusters were labeled as Very low (red), Low (blue), and Medium-High (green). **(a)** Displays the cluster distribution considering PGP, ERPM and GP. **(b)** Considers PGP, ERPM and XACC var. **(c)** Displays clusters regarding PGP, GP and XACC var, and **(d)** considers ERPM, GP and XACC var.

When considering the cluster distribution of Figure 3.7, and taking into account that aggressiveness and fuel consumption are well correlated, considering this figure as our baseline for further comparisons, we can assert that

- Very low fuel consumption (red) corresponds to drivers who keep the car running at its lowest regime (low PGP, low ERPM, and medium GP).
- Low fuel consumption (blue) corresponds to drivers who use the gas pedal gently and run the car at medium regimes (low PGP, low GP, and medium ERPM).
- Medium-High fuel consumption (green) corresponds to drivers who use the gas pedal extensively and run the car at high engine regimes (high PGP, disperse GP, and high ERPM).

Nevertheless, despite interesting DS-related fuel consumption profiles being extracted at a joint interpretation of the information that is depicted in Figure 3.7 and Table 3.3, driver classification cannot be kept uniform along an entire trip, since it is far from being a binary task. For that reason, due to driving circumstances changing during a trip, evaluation by time windows provides a better assessment of the fuel consumption trend. In Table 3.3, the distribution of DSs among clusters is displayed. As can be seen, each driver shows a unique cluster distribution for his/her trip. This distribution means that fuel-consumption-related DS is not a binary feature, but a composition of several cluster mixture ratios.

Four drivers stand out among the remaining ones: D1, D6, D11, and D14, as remarked in bold in Table 3.3. Thus, D6 and D11 spend a longer time classified with Very low consumption DS, with 80.2% and the 78.6% of the total ride time, respectively, so they can be considered as eco-friendly drivers. On the other hand, D1 and D14 are the opposite case, with 75.3% and 66.3% of the total ride time being classified as Medium-High fuel consumption drivers, totally compromising eco-friendliness. According to the clusters identified in Figure 3.7, while the former drivers operate the throttle pedal uniformly and keep engine RPMs low, being an ideal operation decision, the latter ones swiftly operate the gas pedal and keep engine RPMs at the upper range for most of the trip. Finally, it is worth remarking that most drivers' behavior evolves between contiguous classes, except D10, which exhibits a particular behavior, leaping between extreme classes (from Very low to Medium-High, and vice versa).

Table 3.3: Percentage of the route that each driver travels using different fuel-consumption DSs (three-cluster classification).

Driver	Very Low (%)	Low (%)	Medium-High (%)
D1	4.7	20.0	**75.3**
D2	49.4	10.4	40.2
D3	23.0	33.3	43.7
D4	7.8	51.1	41.1
D5	33.3	26.5	40.2
D6	**80.2**	10.4	9.4
D7	15.1	33.7	51.2
D8	16.2	29.7	54.1
D9	8.1	38.4	53.5
D10	56.3	0	43.7
D11	**78.6**	11.9	9.5
D12	10.6	33.0	56.4
D13	14.0	46.5	39.5
D14	8.4	25.3	**66.3**
D15	2.5	45.7	51.8
D16	14.3	40.5	45.2
D17	38.1	21.4	40.5
D18	2.7	48.0	49.3
D19	15.6	30.0	54.4
D20	32.2	26.9	40.9

Table 3.4 compiles the actions drivers should perform to modify their DS with the aim of reducing their fuel consumption attending to their current classification. Different actions are needed, depending on the group, as can be seen in this table. Thus, for example, since Medium-High fuel consumers typically operate the gas pedal swiftly, keeping RPMs high due to that aggressiveness, they are required to lower RPMs while trying to operate the gas pedal smoothly and to a lesser extent. On the other hand, low consumption drivers are required to switch to a higher gear because, despite their softly operating the gas pedal, they keep RPMs high due to the use of low gears. Finally, very low fuel consumers are required to keep their DSs with no changes.

Table 3.4: Actions that are associated to the three-cluster classification.

Current Cluster	Required Action
Very low (red)	Keep driving style
Low (blue)	Lower RPM/Switch to a higher gear
Medium-High (green)	Lower RPM/Keep gas steady/Lower PGP

Five-Cluster Grouping The recommendations that are indicated in Table 3.4 could be unclear for some drivers, especially those being classified into the green cluster (Medium-High consumption). For that reason, with the aim of personalizing the driving recommendations, SOM clustering of the trained map was recomputed using a lower threshold value in its U-matrix, so that more precise partitions could be obtained. Figure 3.8 and Table 3.5 show the five-cluster grouping obtained after this re-computation and the associated fuel consumption for each cluster, respectively. Nevertheless, despite the existence of a higher number of groups, the relationships identified in Figure 3.7 remain. Thus, the blue and red clusters (Low and Very low consumption) are kept barely unaltered both in position and number of elements. In contrast, three new classes appear from the former Medium-High consumption group, namely Medium, High, and Very High (yellow, green, and magenta, respectively).

Being the aforementioned partitions, extracted from Figure 3.8, jointly analyzed with the statistical data contained in Table 3.6, we can assert that this SOM-based grouping is finer, and more meaningful information can be extracted when compared to the three-cluster classification.

The fuel-consumption-associated DS ratios for each driver are recalculated in Table 3.6 in order to verify that the five-cluster classification adds information to the already existing groupings, mainly with the purpose of personalizing the assessment of the most aggressive drivers. Drivers D1, D6, D11, and D14 are analyzed again to inspect whether the new cluster classification changes the information contained in Table 3.3.

For these drivers, the percentages of Very low and Low consumption remain practically unchanged with respect to the three-cluster table. On the other hand, if we accumulate the percentages of medium, high, and very high consumption instants, we can observe that they practically match the Medium-High column of Table 3.3. This means that not only does this five-cluster classification provide comparable results, but it

also allows one to thoroughly examine the detailed behavior formerly grouped as medium-high consumption DS.

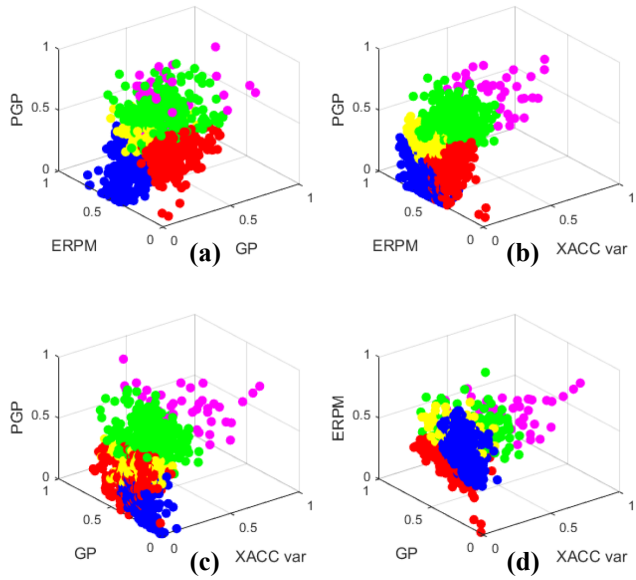


Figure 3.8: Three-dimensional views of the five-cluster fuel consumption classification results. The clusters were labeled as Very low (red), Low (blue), Medium (yellow), High (green), and Very high (magenta). **(a)** Displays the cluster distribution considering PGP, ERPM and GP. **(b)** Considers PGP, ERPM and XACC var. **(c)** Displays clusters regarding PGP, GP and XACC var, and **(d)** considers ERPM, GP and XACC var.

Table 3.5: Fuel consumption (L/100 km) parameters of the five-cluster classification.

Cluster Label	Avg. Value	Variance	Max. Value
Very low (red)	2.75	1.04	6.66
Low (blue)	3.04	1.54	7.80
Medium (yellow)	4.44	2.21	10.1
High (green)	5.42	3.13	11.4
Very high (magenta)	7.81	5.38	12.5

Table 3.6: Percentage of the route that each driver travels using different fuel-consumption DS (five-cluster classification).

Driver	V. Low (%)	Low (%)	Medium (%)	High (%)	V. High (%)
D1	4.7	20.0	**44.7**	**30.6**	0
D2	48.3	10.3	2.3	32.2	6.9
D3	21.8	33.3	13.8	30.0	1.1
D4	7.8	51.1	22.2	18.9	0
D5	33.3	26.5	10.3	25.3	4.6
D6	**71.9**	10.4	9.4	8.3	0
D7	14.0	33.7	18.6	29.1	4.6
D8	14.9	29.7	8.1	37.8	9.5
D9	8.1	38.4	31.4	22.1	0
D10	56.3	0	16.1	25.3	2.3
D11	**78.6**	11.9	1.2	8.3	0
D12	10.6	33.0	37.2	19.2	0
D13	14.0	46.5	15.1	24.4	0
D14	8.4	25.3	**31.3**	**35.0**	0
D15	2.5	45.7	24.7	23.4	3.7
D16	13.1	40.5	30.9	15.5	0
D17	38.1	21.4	10.7	21.5	8.3
D18	2.7	48.0	19.2	26.0	4.1
D19	15.6	30.00	21.1	32.2	1.1
D20	32.2	26.9	22.6	18.3	0

According to Figure 3.8, the Medium-High consumption cluster of Figure 3.7 can be detailed with the following groups:

- Medium fuel consumption (yellow): corresponds to drivers who run the car at engine regimes similar to those achieved for the low consumption cars, but with the difference of a more extensive use of gas pedal, (i.e., medium PGP, low GP and medium ERPM).
- High fuel consumption (green): corresponds to drivers who run the car at medium-high RPM, with moderate swiftness of the gas pedal operation (medium-high ERPM, medium-high PGP, and medium GP).
- Very high fuel consumption (magenta): corresponds to drivers who are slightly more aggressive that those from the preceding group (high ERPM, high PGP, and medium-high GP).

Additionally, with this new cluster distribution, more actions can be indicated to drivers to modify their DS. Thus, in contrast with Table 3.4, where Medium (yellow) to Very high (magenta) consumption classes were aggregated, in Table 3.7, actions were added for each individual newly identified cluster. In contrast, actions for Very low and Low consumption groups remain unchanged. The action Lower RPM/Keep gas steady was disaggregated into 3 different recommendations: Lower RPM/Operate gas softly, Lower PGP/Lower RPM and Lower PGP/Keep gas steady. This can be noticed when comparing Table 3.4 with Table 3.7. This happens because, differing from the uniform DS of the big Medium-High consumption group of the three-cluster classification (green group in Figure 3.7), the Very high (magenta) fuel-consumers are required to lower both RPM and PGP, because they drive at high RPM. In contrast, High (green) consumers run engines at moderated RPM rates. However, the latter swiftly operate the gas pedal at moderately high percentages, consequently being required to use it less and more smoothly. Finally, Medium (yellow) consumers operate the gas pedal smoothly but their usage percentage is still high, so they are required to further reduce gas pedal usage.

Table 3.7: Actions associated to the five-cluster classification.

Current Cluster	**Required Action**
Very low (red)	Keep driving style
Low (blue)	Lower RPM/Switch to a higher gear
Medium (yellow)	Lower RPM/Operate gas softly
High (green)	Lower PGP/Lower RPM
Very high (magenta)	Lower PGP/Keep gas steady

Should drivers follow the recommendations that are displayed in Table 3.7, a noticeable reduction in fuel consumption is expected to occur. However, because the level of engagement of motorists with the provided advice may vary depending on behavioral characteristics of each individual, the expected improvement on fuel economy must be cautiously analyzed. For that reason, Table 3.8 was elaborated to estimate the expected improvement when considering a minimal level of engagement with the system that would allow drivers to modify their DS to the immediately adjacent cluster.

Table 3.8: Expected fuel-consumption reduction between contiguous clusters.

Current Cluster	Target Cluster	Reduction (%)
Low	Very low	9.5
Medium	Low	31.5
High	Medium	18.1

As can be seen in Table 3.8, obtained from the values displayed in Table 3.5, if drivers could only improve their DS to the best adjacent class, reductions in fuel consumption ranging from 9.54% up 31.5% are expected, with even higher performances for strongly-engaged drivers, showing that a significant reduction in polluting agents could be expected if this system was implemented in cars. It is worth to remark that the consumption reduction from the Very high class is not analyzed because not all the drivers show it. This potential reduction is significantly greater than the already existing systems, which were exposed in Section 3.1, making this approach a promising solution.

3.5 Fuel-Consumption Assessment Results

Most drivers' behaviors vary among different clusters, and, the Very low consumption class being the ideal one, indications should be addressed to drivers to modify their DS if they fall into the other classes at any moment of the ongoing ride, as has been seen in the previous section. In the following, the driving behavior of two particular drivers, D1 and D11, will be analyzed with the aim of verifying the suitability of the advice provided by the fuel-consumption assessment system. In addition, the potential impact on emissions that this advice system can achieve is quantitatively evaluated.

3.5.1 Drivers' Advice

Figure 3.9 depicts the Uyanik measurement of relevant CAN-bus and IMU signals corresponding to D1 (green) and D11 (red) during five uninterrupted minutes of the route. DS evaluation times from 8 s to 292 s were considered for the performance analysis, as can be seen in Figure 3.10. It can be observed that the longer the evaluation time, the lower the number of involved DS classes for each evaluation period. For many drivers, this number of classes reached a local minimum at a length of 100 s, while overshooting for longer times until the low-pass filtering

effects of a very long evaluation time happened, which would eliminate the details that are needed for a correct DS evaluation.

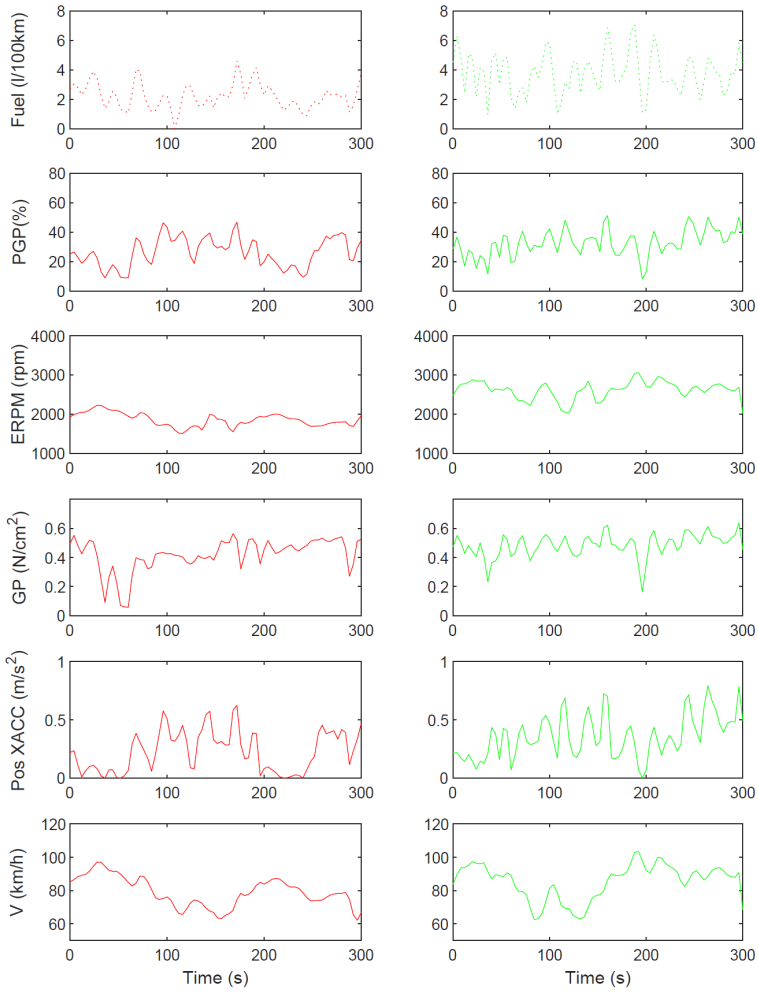


Figure 3.9: Uyanik measurement of relevant CAN-bus and IMU signals corresponding to D1 (green) and D11 (red) during five uninterrupted minutes of the route. The driving behavior was evaluated every 100 s, and the cluster with the maximum percentage was selected. Both D11 and D1 were classified into a single cluster during the whole segment of the trip: D11 drives according to the Very low cluster, and D1's DS is mostly into the Medium cluster. GT-Suite simulations of fuel consumption are also displayed.

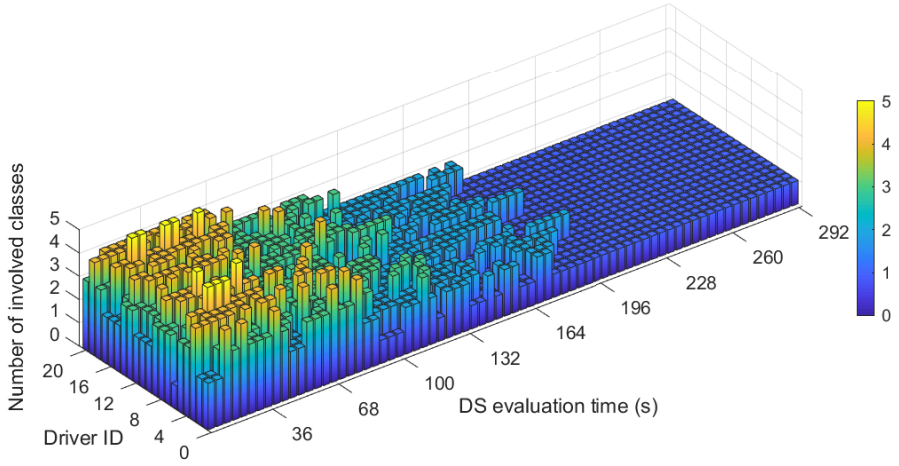


FIGURE 3.10: Three-dimensional bar diagram of the number of classes identified, depending on the evaluation time for each driver.

For that reason and, according to [209], the corresponding driving behavior was evaluated every 100 s in order to be correctly assessed and to provide useful advice to the drivers, with the last 100 s stretch being test data (i.e., unseen by the system). Taking into account the evolution of the DS during each 100 s stretch, the cluster with the maximum percentage was selected. D11 and D1 were both classified into a single cluster during the whole segment of the trip: D11 drove according to the Very low cluster, and D1's DS was mostly in the Medium cluster. GT-Suite simulations of fuel consumption are also displayed. As can be seen, the average fuel consumption of D11 is much lower than D1, as expected. Again, the measured RPMs are lower for D11 than for D1, and the same is the case for the variance of the positive XACC, as in the cluster distribution of Figure 3.8. On the other hand, measured speeds are not significantly different, proving that fuel consumption under similar conditions has more to do with the car handling itself than with speed. In consequence, the eco-driving system would provide the following advice: D11: "Keep driving style"; D1: "Lower RPM/Operate gas softly".

In sum, most drivers' classifiable behaviors vary among different clusters and, with the Very low consumption class being the ideal one, indications should be addressed to drivers to modify their DS if they fall into the other four classes at any moment of the ongoing ride. The eco-driving system provides advice to the driver according to a user-configurable time interval.

3.5.2 Fuel Consumption and Emissions Reduction

The same two drivers (D1 and D11) shown in Figure 3.9 were selected in order to indicate the potential reduction on emissions that this advice system can achieve. In this 300 s highway driving stretch, speeds for both drivers are kept above 79 km/h (79.1 km/h and 85.4 km/h, respectively). In these conditions, the DS identification system classifies D1 mostly into the Medium consumption class, while D11 is classified into the Very low consumption class. Additionally, the GT-Suite simulation data show that the mean fuel consumption measurements in that stretch for D1 and D11 are 4.46 L/100 km and 2.61 L/100 km, respectively. Assuming that the average composition for diesel fuel corresponds to the formula $C_{12}H_{23}$, with a density of 0.835 g/L [210], the stoichiometric combustion of this fuel type follows the equation

$$4C_{12}H_{23}+71O_2 \rightarrow 48CO_2 + 46H_2O. \tag{3.1}$$

With the chemical reaction of Equation (3.1), the average CO_2 generation rate can be calculated, with the CO_2 emissions for D1 and D11 being 128.4 g/km and 75.2 g/km, respectively. That is to say, D11's CO_2 emissions are 41.4% lower than those of D1 for similar road stretches at similar speeds. With these results, we can assert that, if the recommendations of Tables 3.4 and 3.7 were provided to D1, with the aim of being classified into the Very low cluster, then a noticeable reduction in fuel consumption and emission rates could happen.

In sum, by using SOM algorithms, the clusters of Figures 3.7 and 3.8 were discovered and fuel-consumption-associated DS features were extracted. Those clusters and features, jointly with the analysis of the cluster distribution ratio for each driver of Tables 3.3 and 3.6, allowed for the identification of complex behaviors. Finally, several actions were described to modify individual DSs with the aim of improving fuel economy while taking the clustering as well as the distributions and the features into account, consequently encouraging eco-driving.

3.6 Implementation of the PSoC-Based Intelligent System

The intelligent system for real-time assessment of fuel consumption and eco-driving was properly evaluated and tested through a specific PC-based model. After that, the whole system was implemented, such that it can be executed in real-time. For that purpose, the device that this

task is implemented in must be capable of performing high-speed data computing, while providing high throughput data outputs. For those reasons, a hybrid HW/SW architecture was developed and implemented on the Xilinx XC7Z045-2FFG900 Programmable PSoC [147] using the Xilinx ZC706 development board [148].

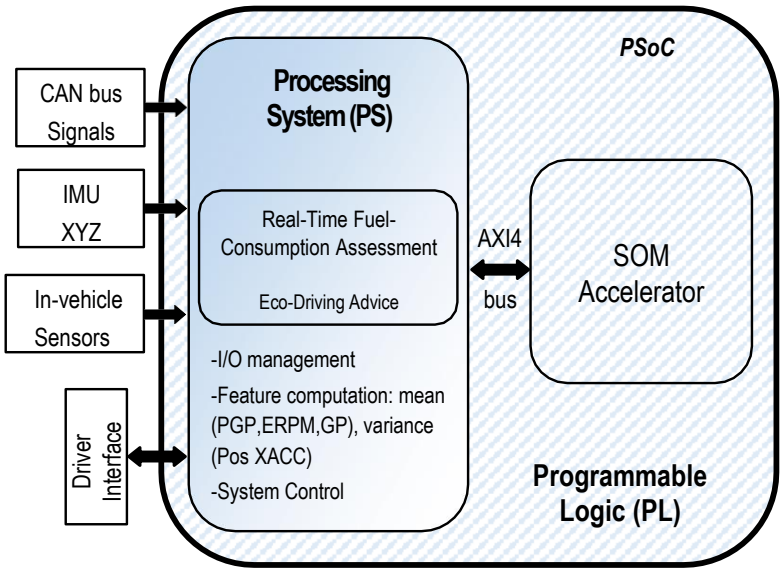


Figure 3.11: Block diagram of the programmable system-on-a-chip (PSoC) for real-time fuel consumption assessment and eco-driving.

Figure 3.11 depicts a block diagram of the proposed solution. Details on PSoC architectures are provided in Section 1.4.2. The entire HW partition of the system, consisting of an SOM accelerator, was deployed in the FPGA of the PSoC using VHDL language and the Xilinx Vivado 2018.1 design suite [203]. On the other hand, the remainder of the proposed system functionalities were programmed at the microprocessor (SW partition depicted in Figure 3.11) by developing a bare-metal C application that can acquire data from the buses of the vehicle, compute the windows of the ERPM, GP, PGP, and pos XACC features, share them with the FPGA, retrieve the SOM accelerator results, and provide advice to drivers.

3.6.1 Hardware Partition: SOM Accelerator

The HW partition is based on a digital electronic system, whose architecture, deployed within the PSoC's FPGA, can be described as follows.

The SOM HW accelerator is composed of four main modules: input registers, neurons, comparers, and internal ROMs (see Figure 3.12), as well as a controller unit. This architecture has been designed to be totally parallel with the aim of returning a correct response in the minimum time lapse. The VHDL language is used in order to create a fully scalable architecture regarding the number of input features (N) and the number of neurons of the SOM (M).

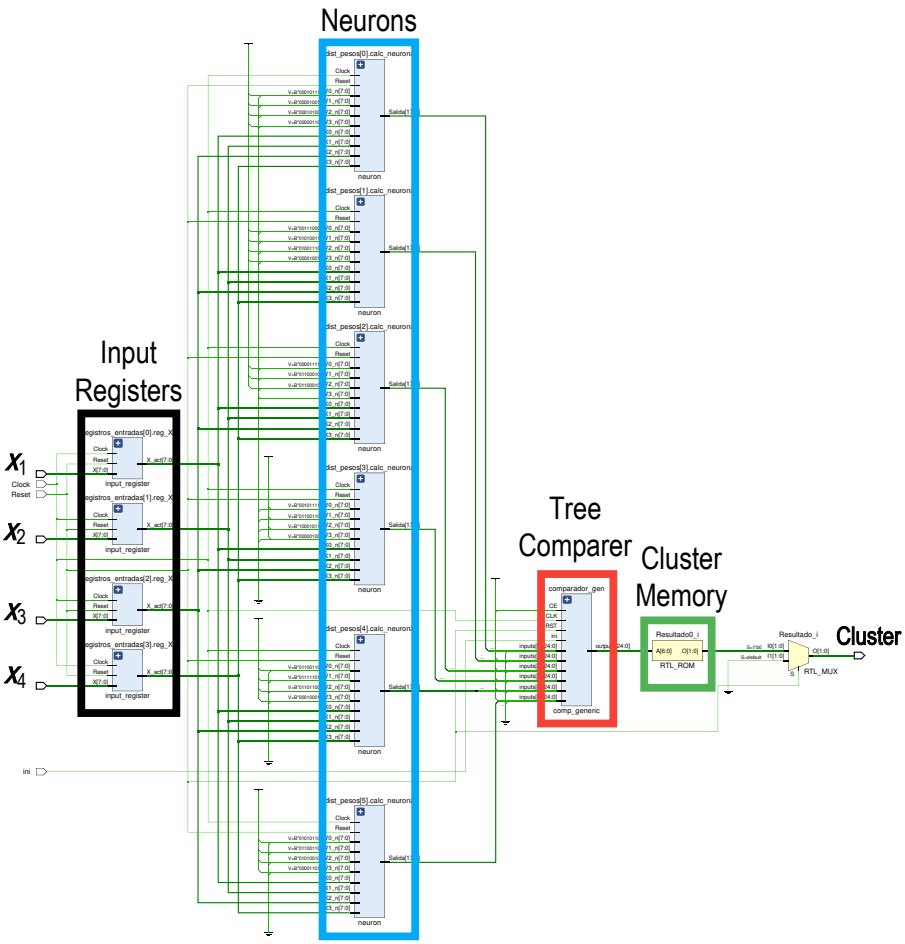


FIGURE 3.12: Scheme of the SOM HW accelerator. A four-input SOM topology with six output neurons is shown as a case example.

Input Registers

The input registers (the black box in Figure 3.12) are used in order to feed the input samples $\mathbf{x} = (x_1, x_2, \ldots, x_N)$ into the SOM accelerator

synchronously, with each rising edge of the clock signal. The number of input registers depends on the number of input features, N, since each feature needs a separate register. These registers' inputs are read from the AXI4 interface.

Neurons

The neuron components (the blue box in Figure 3.12) compute the squared Euclidean distance between the input sample and a given neuron's weight (see Equation (1.19)). Each neuron block is shaped by two types of components: the distance module and the adder module. Thus, while the former computes how far each input feature x_j^k is from the corresponding neuron weight m_{ij} and squares that difference (squared euclidian distance), the latter, which is based on a typical tree-adder, sums the N individual squared distances to compute the total distance from the input sample to the i-th neuron weights.

Besides, a neuron pointer, i, is added to the output. It indicates which neuron each computed distance belongs to, with the aim of easily accessing the cluster memory once the neuron with the minimum distance (i.e., the BMU) is found. Finally, each neuron block stores its corresponding weights into a small ROM.

Tree-Comparer

The tree comparer (the red box in Figure 3.12) computes the BMU of the input sample (see Equation (1.20)). The input to the module is the array of outputs of the neuron blocks, that is to say, the distances between each neuron with the input sample, concatenated with the neuron pointers. It returns the BMU along with its corresponding pointer (i.e., the BMU index).

For this design, a recursive tree-comparer was developed based on the previous work of the authors [181]. This topology, as shown in Figure 3.13, was adapted so as to provide latency rates that were comparable to those from traditional binary tree comparers, while minimizing resource usage.

Thus, the proposed architecture of Figure 3.13 only uses $M/2$ comparer blocks, while a binary tree comparer would spend $M-1$ of the same HW resources. Given an input array $\mathbf{u} = (u_1, u_2, \ldots, u_M)$, and a control control signal `ini`, the module operates, as follows:

1. Signal `ini` is set to "0" and all registers are reset.

2. First comparisons, $u_{2j-1} < u_{2j}$ with $j = 1, \ldots, M/2$, are computed and stored in each of the $M/2$ comparer registers.

3. Signal `ini` is set back to "1" and the next comparison is performed. Consequently, comparisons are stored in registers 1 to $M/4$, while registers $M/4 + 1$ to $M/2$ are now filled with ones.

4. Succesive $\lceil log_2 M \rceil - 1$ comparisons are computed until a valid result is obtained.

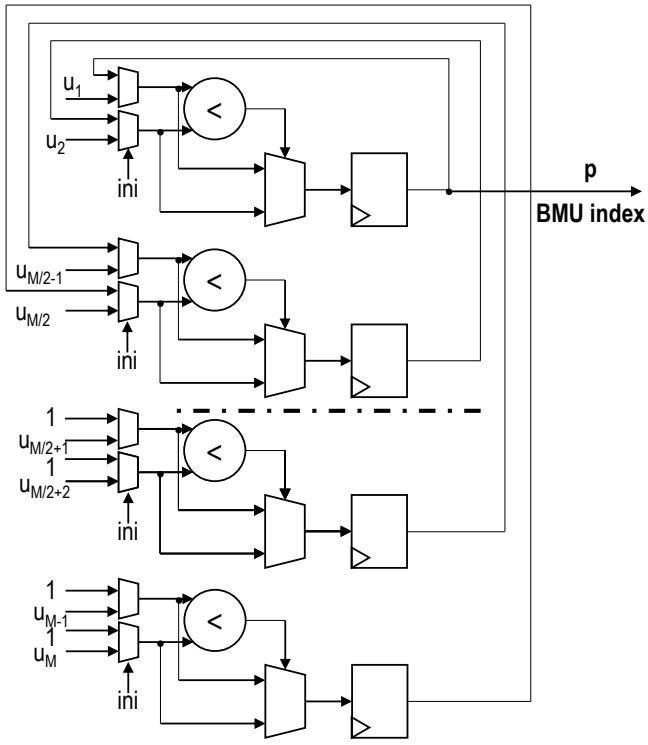


FIGURE 3.13: Scheme of the proposed recursive tree comparer architecture that substitutes a traditional comparer solution.

Internal ROM

The ROM, as marked in green in Figure 3.12, stores the cluster to which each neuron belongs. The cluster identification is performed by addressing the ROM while using the index of the BMU identified in the tree comparer module, consequently allowing one to know which cluster the input sample fits the most. The ROM is implemented using LUTs

with the aim of improving the circuit speed. LUTs are typical FPGA resources that reduce the propagation delays when compared to block BRAM-based memories.

Parameterization and Control Signals

The complete structure of the SOM HW accelerator is parametric and fully customizable. ROMs containing the neurons' weights as well as the clusters that are associated with each neuron are simultaneously initialized. Elements such as type depths, signal bit-widths, and the number of inputs and neurons were defined on a standalone package.

The architecture's latency depends on two factors: a fixed time that always delays the same number of clock cycles and a variable time that depends on the number of features (N) and the number of neurons (M). Thus, one clock cycle is needed to load the input registers and two clock cycles for the computation of Euclidean distance within the neuron's distance modules.

On the other hand, the neurons' adder module computes its outputs 2-by-2. Consequently, the clock cycles that are required to obtain the output of the adder module are calculated, as follows:

$$t_{tree} = \lceil log_2 N \rceil = min\{k \in Z | (log_2 N) \leq k\}. \tag{3.2}$$

As in the case of the adder module, the tree comparer decides which neuron holds the shortest distance recursively 2-by-2. For that reason, its latency is computed in the same way as in Equation (3.2), but using M instead of N, since it has the array of the outputs of the neurons as inputs. Therefore, the number of clock cycles elapsed since the input signal arrives in the architecture until a valid output is provided is computed by the following expression:

$$N_{cycles} = 3 + \lceil log_2 N \rceil + \lceil log_2 M \rceil. \tag{3.3}$$

On the other hand, the control signals of the SOM HW accelerator are `rst`, `launch`, and `ini` (see Figure 3.14); they work, as follows:

1. `rst` clears all of the architecture registers and prepares the modules for a new input array.
2. `launch` loads the input data into the neurons' input registers. Neurons' outputs are ready after $3 + \lceil log_2 N \rceil$ clock cycles.
3. In the next clock cycle, the neurons' outputs are loaded into the input registers of the recursive tree comparer module.

4. `Ini` is triggered to indicate to trigger the recursive comparisons of the tree adder. The result is ready after $\lceil log_2 M \rceil$ clock cycles.

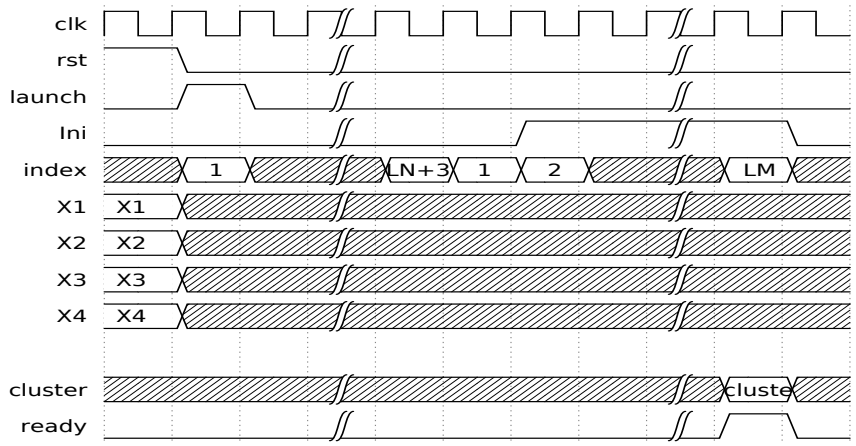


FIGURE 3.14: Chronogram of the control-signal sequence of the SOM HW accelerator. LN stands for $\lceil log_2 N \rceil$, and LM for $\lceil log_2 M \rceil$.

3.6.2 Experiment Results

In Section 1.2.5, the SOM training as well as a classification method were explained. Additionally, the classification was carried out on the Uyanik dataset completed by the GT-Suite simulation data, obtaining the weights and clusters to which the neurons belong. With that data, the SOM network was implemented in the FPGA in order to classify the Uyanik drivers by storing the neuron weights and their corresponding clusters into the internal ROM of the architecture.

A PSoC of the Xilinx ZynQ-7000 family (XC7Z045-2FFG900 PSoC) [147] is used in order to implement the SOM accelerator.

The architecture described in Section 3.6.1 has a total of $M = 121$ neurons and $N = 4$ input features. We used fixed-point binary arithmetic to represent data. Both input data and neurons' weights have been represented with 8 bits, with all the 8 bits representing the decimal part, because both are unsigned positive numbers. The intermediate operations' bitwidths were selected such that neither overflows nor rollovers could occur.

Simulation Results

In order to verify that the developed SOM HW accelerator works as expected, several input arrays are fed to the architecture, the control signals displayed in Figure 3.14 are applied to the circuit, and the identified clusters are verified, as can be seen in Figure 3.15.

In this Figure, a latency of 12 clock cycles (0.12 µs at a 100-MHz clock frequency) is shown. As will be seen, it matches with the timing indicated in Section 3.6.2. Additionally, the BMU's distance is checked and compared against an equivalent MATLAB SOM model (with a value of 3.52×10^{-3}), jointly with the cluster ROM position (position = 42), which the HW, the fixed-point implementation, and the MATLAB floating-point model results totally match.

Resource Usage

The full HW system was successfully implemented, with the post-implementation results displayed in Table 3.9. The SOM network acceleration system fit into the selected PSoC's logic, leaving enough resources available for further system applications, scalations, or improvements.

TABLE 3.9: Post-implementation resources report (Xilinx XC7Z045-2FFG900).

Resource	Utilization	Available	% Used
LUT	21 107	218 600	9.66
FFs	13 337	437 200	3.05

Timing Performance

Before the deployment, the maximum operational frequency was calculated. For that purpose, the architecture was implemented with a minimum clock period of 10 ns, obtaining a slack of 2.292 ns. Thus, the maximum operational frequency of the design can be calculated as indicated in Equation (2.5), obtaining F_{max} = 129.74 MHz. Hence, the designed HW implementation could be used as an AXI4 peripheral dependent on an AXI4 bus clock frequency of 100 MHz. With this operational frequency, and applying Equation (3.3) with $N = 4$ features and $M = 121$ neurons, this design delayed 12 clock cycles (0.12 µs at F_{CLK} = 100 MHz) to return a valid result.

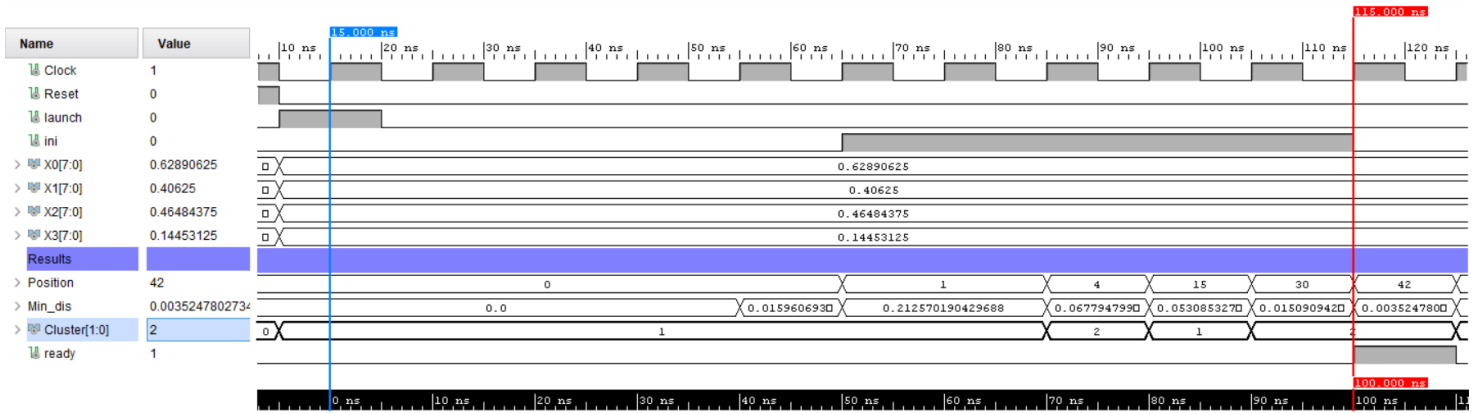


FIGURE 3.15: Simulation results of the SOM HW accelerator obtained with the Vivado design suite. This simulation replicates the chronogram of Figure 3.14. The latency is measured from the blue marker to the red one. $F_{CLK} = 100$ MHz

These results outperformed the timing that was obtained for a full-SW PC-based MATLAB model design (20-core Intel Xeon E5-2630 v4 CPU at 2.20 GHz with 32 GB of DDR4 RAM), with a timing performance of 128.03 ± 12.01 µs to compute the same SOM, as well as a PC-based, C-coded prototype that achieved timing marks of 1.34 ± 0.28 µs.

The obtained timing performance was better than in other FPGA-based SOM applications, such as the work by the authors of [211], where the highest operational frequency obtained was 101.54 MHz for a considerably smaller SOM network. On the other hand, in [212], an absolutely novel architecture is presented with the aim of improving the general performance of the minimum's finding procedure by sidelining the use of comparers. This architecture, despite operating at even lower frequencies (a maximum of 19.6 MHz), achieves very high throughput at the cost of drastically increasing complexity.

In contrast, in [213], outstanding frequencies of 188.9 MHz are achieved for a bigger SOM implementation by carrying out some simplifications that might compromise the accuracy of smaller networks.

Consequently, we can assert that the HW partition developed in this chapter is an appropriate solution between conventional SW-based approaches and novel FPGA-based, extreme performance architectures, which provides an adequate trade-off between complexity, performance, and development time.

3.6.3 Software Partition

The PS of the PSoC (SW partition) is built around a dual-core ARM Cortex-A9 microprocessor, as described in Section 1.4. With this architecture in mind, the SW application was developed, enabling the full operation of the hybrid HW/SW system. These functionalities are as follows:

- I/O management: the system retrieves the driving features from the buses of the vehicle (e.g., CAN-bus) and outputs the natural-language driver recommendations.
- Data logging and windowing: the microprocessor stores 8 s of data (256 samples at 32 Hz) to compute the data windows used to extract the driving features. Each window has an overlapping of 4 s with its preceding one; thus, with an 8 s size, a new window is generated every 4 s.

- Feature computation: for each data window, the microprocessor computes the average values of ERPM, GP, and PGP, and the variance of Pos XACC.
- Data exchange: the SW partition sends the features to the HW accelerator through the AXI4 bus and retrieves the identification results (DS clusters) from the HW partition.
- Cluster distribution computation: the cluster distribution of a set of windows, evaluated during a certain time of uninterrupted driving above a certain speed threshold, is computed.
- Driver advice: natural-language advice is provided, depending on the cluster distribution, that is to say, according to the cluster in which the driver spends the longest time. This advice is intended to be provided by means of non-invasive text messages in the instrument cluster, and shown as needed.

In sum, with the aforementioned data logging, windowing, and feature computation, once the data are fed to the FPGA, the SOM HW accelerator identifies the DS cluster for that window. With those results, the SW partition computes a cluster distribution, decides which cluster registered the highest number of hits, and, according to that maximum, provides eco-driving advice to drivers.

3.7 Concluding Remarks

The main motivation of this chapter was the development of ADAS on the board vehicle contributing to the encouragement of eco-driving by providing real-time personalized advice to drivers. With this aim, a holistic approach, based on ML techniques and FPGA technology, was proposed. It uses a data-based focus to identify relevant, fuel-consumption-associated features. For that purpose, a mix of real-world data, which were obtained with an instrumented car and fuel consumption simulation results, was jointly processed. This analysis had the goal of providing informative data to train an SOM, which, after a clustering process, allows to classify fuel-consumption-compromising DSs.

The DS recommendations developed are designed to be valid for the majority of drivers. They are provided using natural language, and can be easily understood and followed by most drivers. If a given driver follows the advice, he/she will increase in ecological awareness, modify his/her DS, and consequently reduce fuel consumption and pollutant

emissions, with the expected results ranging from the 9.5% to the 31.5%, or even better in the case of a driver with a high level of engagement with the advices that the system provides. In addition, current implementations of efficient driving strategies for autonomous vehicles could also benefit from these results by incorporating the proposed system at the development stage, or even after it, to improve and verify the eco-friendliness of the developed model.

The solution adopted in this chapter relies on a high-performance, fully parallel SOM implementation. This architecture is inherently parallelizable for high-performance HW implementation due to its layered topology, and easily scalable due to its extensive parametrization. The entire SOM-based classification system was successfully implemented while using an FPGA device of the Xilinx ZynQ-7000 PSoC family, with the HW partition providing high speed and low-power consumption for real-time implementation, while its microprocessor executed complementary tasks. Moreover, due to the reconfigurable nature of FPGAs, both the HW and SW partitions of the PSoC can be updated to cope with the continuous changes that new vehicle technologies introduce.

Chapter 4

A Data-Based Approach for Ride Comfort Improvement

4.1 Overview

In this Chapter, a method to improve ride comfort is described. This approach, that could be used to develop ADAS, aims to reduce the DS-related discomfort in car occupants. Thus, not only driving features that can trigger ride discomfort are identified, but also personalized advice according to this field is provided. For that purpose, real-world data from the Uyanik-instrumented car (refer to Section 1.3.1), such as the CAN-bus and IMU data streams were used to model ride comfort features without setting aside the fuel consumption characteristics. Concretely, the data stretches selected in Section 3.4.1 are used. After this, significant variables are selected and clustered by using SOMs (described in Section 1.2.5). The main purpose of this solution is to identify the causes of discomfort and to provide educational advice to drivers with the aim of correcting the wellness-compromising conditions.

The remainder of this chapter is organized as follows. Section 4.2 introduces ride comfort concepts. Section 4.3 provides an overview of the proposed approach and describes the utilized instrumented car dataset. In addition, the most relevant features concerning ride comfort are analyzed and selected. In Section 4.4 the development of SOMs for ride comfort classification is provided. Different DS clusters are identified and natural language-based handling advice is developed in Section 4.5. In Section 4.6 system validation and analysis is presented, and its compatibility with the eco-driving advice system presented in Section 3 is assessed. Finally, Section 4.7 summarizes the achieved improvements and exposes some final considerations.

4.2 SOM-Based Ride Comfort Characterization

Due to the main part of the proposed approach being based on the characterization of DS taking into account ride comfort through the use of unsupervised ML algorithms, some basic theory on this concept is introduced in this section.

4.2.1 Ride Comfort Parameters

As introduced in Section 1.1.5, two types of discomfort can be distinguished when we analyze the ride comfort during a given trip. In [214] the general feeling of malaise is called average discomfort, while motion sickness is associated with dizziness, fatigue and nausea. The synergy between these two sensations causes the feeling of discomfort.

Two complementary types of approaches can be followed to assess the ride-quality experienced by passengers: qualitative and quantitative. Regarding qualitative methods, subjective tests [215] can be used to rate a variety of parameters from the viewpoint of the individual experience of the passenger. Conversely, several methods can be used to quantify the ride-quality during a given trip. In this line, the sensations caused by vibrations on the human body strongly depend on the signal direction and its spectral content. Hence, the ISO elaborated one of the mainly used standards: International Standard 2631 (ISO-2631-1) [216]. This standard describes ways to evaluate vibration exposure to the human body, defining methods to measure vibrations as well as how to process measurement data to standardized quantified performance measures concerning health, perception, comfort and motion sickness.

In this standard, measurements are based on the frequency weighted RMS computations of acceleration data for each axis. This norm defines several filter shapes that delimit the frequency bands where different components of discomfort are present: filters w_f, w_d, and w_k, where filter w_f is representative of motion sickness discomfort, while the filters w_d and w_k model the horizontal and vertical components of global discomfort, respectively.

As shown in Figure 4.1, the frequencies that mainly cause motion sickness are those between $w_{f_1} = 0.1$ Hz and $w_{f_2} = 0.3$ Hz (blue curve), so, the motion-sickness-associated measures have to be carried out for the input data filtered by the blue curve w_f. On the other hand, the green filter w_d evaluates the sensation of general discomfort for a seated passenger when accelerations lie in longitudinal or lateral directions. Finally, the w_k red filter is related with vertical accelerations.

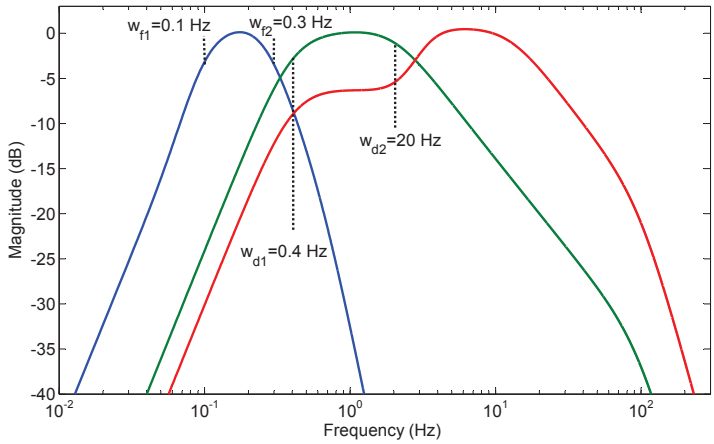


FIGURE 4.1: Amplitude responses of different weighting filters in ISO 2631, w_f: Motion sickness (blue), w_d: Global comfort horizontal-component (green), and w_k: Global comfort vertical-component (red)

Regarding the time persistent discomfort, the measured accelerations can be weighted and filtered in the way that ISO 2631-1 determines, where a_{wxd}, a_{wyd} and a_{wzk} are the results of being filtered by w_f, w_d and w_k, respectively (see Fig. 4.1). The weighted RMS acceleration for each axis is expressed as,

$$a_{wij} = \sqrt{\frac{1}{K}\sum_{k=1}^{K} a_i^2, w_j(k)}, \tag{4.1}$$

where i determines the direction, w_j is the corresponding filter, and K is the number of samples of the acceleration data.

Other significant parameter to be assessed is the likeliness of nausea by means of the computation of the motion sickness dose value ($MSDV_i$) for each axis by particularizing the Equation 4.1 for $w_j = w_f$ (with w_f being the motion sickness filter represented by the blue curve in Figure 4.1), such that,

$$MSDV_i = \sum_{k=1}^{K} a_i^2, w_f(k), \tag{4.2}$$

which is a standardized measure of motion sickness.

There are several works related to the analysis of those parameters suggesting some variants. Concerning motion sickness, although the international standard pays attention to vertical accelerations, there are

later works that prove the influence of lateral accelerations in motion sickness. Thus, in [217], the likeliness of nausea is represented by the Equation 4.2 applied to lateral accelerations. Additionally, in [218], the ISO weighting filters shown in Figure 4.1 are slightly modified to better match with the real sensations caused by the transverse forces, being $w_{f1} = 0.02\ Hz$ and $w_{f2} = 0.3\ Hz$. On the other hand, in [219], the probability of a car occupant to get motion sick enough to vomit is suggested, as well as several weighting parameters, both for vertical and horizontal accelerations.

In this chapter we chose the filter proposed in [218], and we analyzed motion sickness parameter using lateral accelerations, since they depend more on driving than the vertical ones. In addition, based on [219], we combined the contribution of both vertical and horizontal accelerations by means of the new VR parameter presented in Equation 4.3. Moreover, despite those parameters being accumulative (depends on travel length), in this chapter a window-based averaging was used.

$$VR = \sqrt{\left(\frac{1}{3}\right)^2 MSDV_z{}^2 + \left(\frac{\sqrt{2}}{3}\right)^2 MSDV_y{}^2}. \tag{4.3}$$

Finally, acceleration and jerk peaks are evaluated using a methodology based on acceleration thresholds [220]. A high value of acceleration or jerk can cause discomfort even during shorter periods of time. When the levels get too high the passenger will find it difficult to maintain posture. Limit values vary between the studies. In [221] a maximum acceleration value of 1.47 m/s^2 is determined whereas in [214] it is argued that since an automobile only carries seated passengers it is expected that the thresholds should be set on the higher side than in a train or on a bus and the limit is set closer to 2 m/s^2.

Thus, we can assess the transient discomfort, by counting the acceleration peaks with values above a certain threshold, such that:

$$n_i = n_i + 1 \text{ when } a_i > \text{Threshold}, \tag{4.4}$$

where i determines the direction of the acceleration in each of the XYZ axes and the threshold was fixed at 1.75 m/s^2.

4.2.2 Ride Comfort Characterization

As detailed in Chapter 3, where several approaches can be followed to alleviate and improve the scenario of high emission levels, the DS is

found to be the main conditionant of this aspect of driving. In the same fashion, DS plays an important role in ride comfort too.

However, in some scenarios, the eco-driving rules can interfere with the ride comfort viewpoint. As an example, in [222] the tradeoff between ride comfort and fuel efficiency are studied for the pulse and glide strategies. These strategies require the clutch to be disengaged between engine pulses to ride the car by inertia while the traction chain is not linked to the wheels. However, this may severely condition the passengers' comfort, since high values of vibration and jerk can happen, and, consequently, several levels of calibration have to be applied to not disturbing the occupants [222]. In the same fashion, some sources propose minimizing the required time to achieve the speed setpoint so as to minimize the fuel consumption [223], [224], which returns outstanding economy rates by the cost of conditioning the passenger comfort in both the danger sense and sensation terms (i.e., jerkiness, noise, accelerations, etc.). For those reasons, a trade-off solution has to be found to guarantee both the ride comfort for the vehicle occupants and fuel economy. To better analyze the aforementioned trade-off, among many of the previously mentioned methods, SOMs have been successfully used to online cluster DS regarding meaningful features and to provide advice to modify the undesirable aspects of car handling [225].

We used the SOM unsupervised classifier to characterize the DS [131], [132], [226], [227], as in Chapter 3. Based on this characterization, and taking into account its interpretation, some DS recommendations are set to be provided by the system. Moreover, this proposal has the main advantage of the aforementioned advice being provided by means of natural language, which makes it friendlier for the driver, always considering his/her individual characteristics. This fact is expected to encourage the engagement of the motorists with the system, who, otherwise, might not notice which points of their DS could trigger discomfort and motion sickness in their companions. Thus, we present the following contributions:

- A novel application of unsupervised neural networks to discover patterns that compromise ride comfort.
- Analysis of an approach to the examination of the underlying causes of different types of non-optimal DSs for ride comfort.
- Personalization of the provided advices when considering the aforementioned points. Those recommendations involve pedal, gear stick and steering wheel operation.

- Improvement in the ride comfort parameters, with potential enhancements of up to the 57.7%.

These results stand out amongst those achieved by the traditional methods of gear recommendation [195] or scoring [196] and have the additional advantage of facing the problematic putting the spotlight on the driver, instead of the vehicle.

4.3 System Overview and DS Features

To achieve the objective of developing an ADAS to improve ride comfort, we performed an in-depth analysis of the DS of 20 drivers of different age-groups and driving experience levels. We used data collected from the real-world using an instrumented car. This information was used to discover underlying DS characteristics, which are to be decoded from the raw data by means of the unsupervised SOM clustering algorithm.

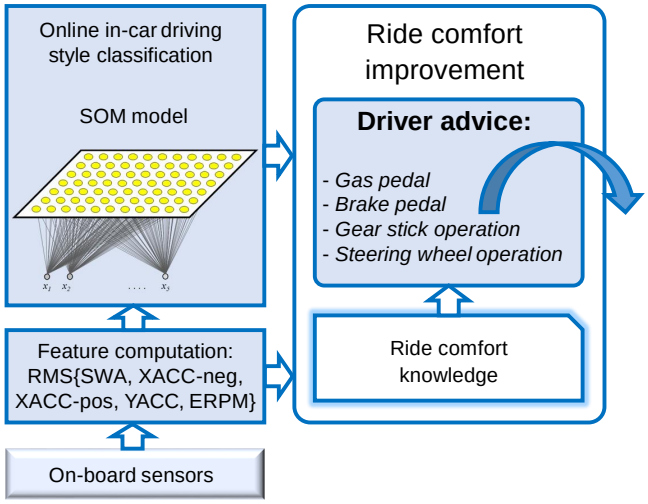


Figure 4.2: Block diagram of the proposed ride comfort assessment system.

Given a set of driving signals' measurements, the devised model is able to perform online classification and to provide the driver with personalized driving advice in real time, with the aim of improving his/her global ride comfort. Figure 4.2 shows a block diagram of the ride comfort ADAS. The system is composed of a feature computation block, a SOM-based DS classifier, and the driver advice module.

The development of the proposed ADAS for ride comfort improvement was performed in three stages. First, the selection of a set of meaningful

features, able to account for the ride comfort viewpoint, was performed by means of analyzing the magnitude of their correlation coefficients. Next, the SOM unsupervised clustering technique was applied to discover different DSs. A SOM was trained, to classify the driving data into a number of classes that account fundamentally for ride comfort. After a quantitative and qualitative evaluation of the trained SOM, well differentiated DS clusters were identified and labeled according to their ride comfort measures. After that, with these pieces of information, particular actions on the car controls (i.e. gas pedal, brake pedal, steering wheel and gear stick) were developed in order to provide advice to the drivers.

4.3.1 Dataset

As in Chapter 3, we used the Uyanik dataset [14], from the University of Sabançi at Istanbul (refer to Section 1.3.1). Additionally, to monitor fuel consumption, the data obtained by with the GT-Suite simulator [187], [225] as described in Section 3.4.2 is used.

It is worth noting that, for this development, we selected the stretches of route remarked in Figure 3.2, which exclusively comprehend highway-type roads. While riding along these roads, which show fluent traffic, vehicles acquire high mean speeds with low deviations. Additionally, these highway stretches were selected so that their mean slopes were lower than the 2%, so drivers had to uninterruptedly operate the gas pedal to adjust their vehicles' speeds to the traffic flow.

4.3.2 Statistical Analysis of Driving Behavior

With the aim of comparing the DS of individual drivers concerning ride comfort, a statistical analysis of comfort parameters was performed. Figure 4.3 depicts mean parameter values corresponding to each driver while completing the same route. As can be seen, the mean values of all the parameters differ among drivers.

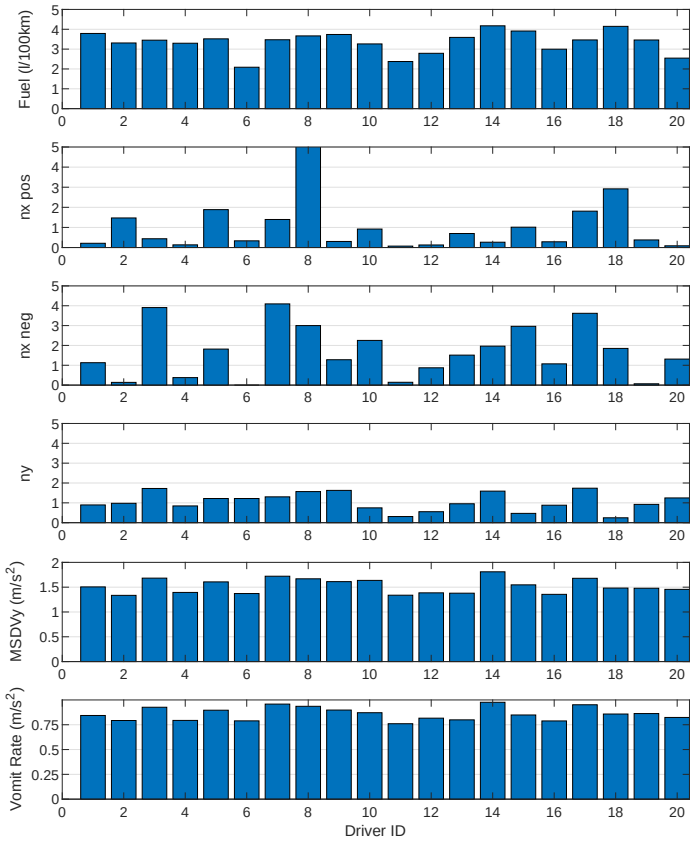


FIGURE 4.3: Average ride comfort parameters and fuel consumption for each driver.

The first group of drivers, namely calm drivers, exhibit low positive and negative acceleration peaks in x axis (i.e. n_x pos and nx neg), as well as relatively low y axis acceleration peaks (i.e. n_y). These drivers show also low MSDVs and VRs. For example, average VRs are: 0.79 m/s^2 (Driver 6), 0.76 m/s^2 (Driver 11), and 0.82 m/s^2 (drivers 12 and 20). In consequence, it can be concluded that the most calm drivers from the comfort viewpoint present also a ecologic DS. On the contrary, the drivers that consume the most fuel are not necessarily the most uncomfortable ones (e.g. Driver 18 shows high fuel consumption: 4.15 L/100 km and low VR: 0.86 m/s^2). Thus, it can be seen that there is a certain relationship between driving comfort and fuel consumption, but this synergy must be analyzed in depth. Moreover, it can be highlighted that drivers 6, 11, 12, and 20 consume less than 3 L/100 km, while drivers 14 and 18 are slightly above 4 L/100 km, being the remainder drivers between

those values.

Next, a different statistical approach to comfort and fuel consumption parameters is provided. The kernel density estimation (KDE) is used to shed light on the nonlinear relationship between the driving behavior of selected drivers and these parameters. The KDE technique, unlike histogram, produces smooth estimate of the probability density function and is able to suggest multimodality [228]. It is useful to estimate the probability density function of datasets difficult to be modeled by parametric density functions.

Figure 4.4 depicts the bivariate kernel density function of VR and fuel consumption corresponding to six representative drivers that exhibit different statistical driving behavior. The probability density function shows different shapes by each one of the selected drivers. The positions of the surface peaks indicate maximum likelihood of VR and fuel consumption. Furthermore, the sharper the surface, the more regular the DS. To carry out this selection, the kernel density function for each driver was elaborated, and consequently, two main trends were identified between all the drivers.

As can be seen, both Driver 6 and Driver 20 present a rather regular and comfortable DS, while Driver 6 peaks at the lowest values both in VR and fuel. In contrast the peak values corresponding to Driver 20 are slightly greater than the former ones. However, concerning fuel consumption, Driver 6 shows a more disperse trend than Driver 20.

Opposite, the coordinates of the peak for Drivers 8, 14 and 18 show high average fuel consumption and irregular driving patterns (i.e. flattened surfaces). Moreover, Driver 18 exhibits a clear bimodal DS. It is worth noting that although the secondary peak is unlikely, it increases the average fuel consumption.

These results are consistent with those shown in Figure 4.3. In sum, the driving data construct different KDE surfaces according to the drivers' preferred DS. Some drivers are more regular than others, although external factors such as weather or road conditions, could also be in part responsible of these differences.

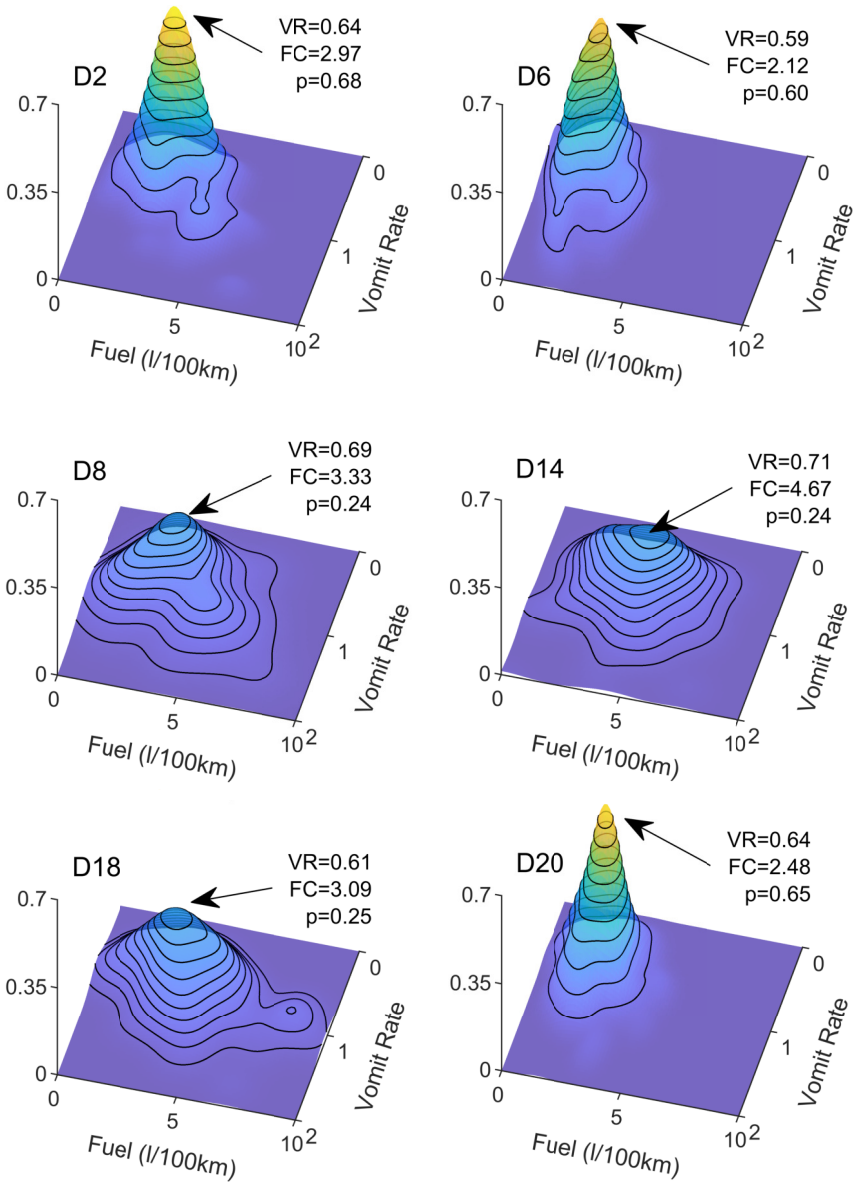


Figure 4.4: 3D KDE, computed on the driving windows, links VR and fuel consumption for some of the most representative drivers of the sample population. Wider projections onto the horizontal plane mean higher variances of the VR and the fuel consumption measured for the driver.

TABLE 4.1: PCCs between the ride-comfort variables and the real-world data from the instrumented car. RMS and variances are computed using 2290, 8s analysis windows.

	SWA RMS	SWA Var	VS RMS	VS Var	XACC RMS	XACC Var	XACC neg RMS	XACC neg Var
nx pos	0.12	0.15	-0.07	0.18	0.33	0.37	-0.08	-0.01
nx neg	0.09	0.10	-0.01	**0.49**	**0.61**	**0.53**	**0.65**	**0.79**
ny	0.28	0.28	0.12	0.03	0.11	0.18	0.06	0.13
$MSDV_y$ (m/s^2)	**0.68**	**0.60**	0.05	0.21	0.19	0.21	0.05	0.14
VR (m/s^2)	**0.63**	**0.56**	0.09	0.20	0.20	0.23	0.04	0.14

	XACC pos RMS	XACC pos Var	YACC RMS	YACC Var	ERPM RMS	ERPM Var
nx pos	**0.45**	**0.61**	0.16	0.20	0.07	0.16
nx neg	-0.10	-0.04	0.09	0.10	-0.08	0.20
ny	0.09	0.11	**0.65**	**0.70**	0.02	-0.02
$MSDV_y$ (m/s^2)	0.16	0.16	**0.73**	**0.60**	0.02	0.07
VR (m/s^2)	0.19	0.19	**0.76**	**0.65**	0.02	0.07

Note: The boldface PCCs correspond to the strongest correlations.

4.3.3 Feature Selection

With the aim of choosing DS variables with a significant level of relationship with the comfort parameters introduced in Section 4.2, we performed a correlation analysis with the real-world driving signals. The features were computed over 256-sample windows (i.e., 8 s at a 32 Hz sample rate) with 50% overlapping between consecutive windows (i.e., 128 samples, or 4 s). It is worth noting that the same driving sections as in Chapter 3 were selected, that is to say, those that ran through highway and motorway. Moreover, sections with traffic jams and slow traffic (i.e., mean speed below 60 km/h) were discarded with the aim of avoiding outliers.

Table 4.1 summarizes the PCCs of the set of selected driving characteristics with different discomfort parameters. The first column of each variable represents the RMS of the signal whereas the second represents the variance. As can be seen, the features with the strongest correlations with the selected discomfort measures are highlighted. For those features concerning the X-axis of the car (longitudinal direction, see Figure 1.12), we can see that both the RMS and variance of XACC pos and XACC neg signals present strong correlation with nx pos and nx neg, respectively. Moreover, the variance of VS and both RMS and variance of XACC correlate with nx neg positively.

In the same way, for those features concerning the Y-axis of the car (transverse direction, see Figure 1.12), both RMS and variance of YACC and SWA are strongly correlated with $MSDV_y$ and VR. It is worth noting that the Z-axis features, despite having a noticeable contribution on worsening the motion sickness felt by the passengers, were discarded since they do not directly depend on the DS, but on the road characteristics.

In sum, the results displayed in Table 4.1 show that the features with the strongest correlation with ride comfort are: RMS{SWA, XACC, XACC neg, XACC pos, YACC} and variance{SWA, VS, XACC, XACC neg, XACC pos, YACC}.

4.4 Development of the SOM-Based Classifier

According to the scheme depicted in Figure 4.2, an unsupervised SOM clustering algorithm was used to classify drivers regarding their ride comfort ratings. With this purpose, we selected the entire set of samples which comprises driving data of 20 drivers. As an average of 115 windows is available per individual, the whole set of driving samples

consists of 2290 windows (i.e., more than 2.5 driving hours). The 75% of the five-dimensional samples will be used to train a SOM, which is to say $K = 1717$, keeping the remaining quarter for testing purposes. The data were normalized before training in order to avoid distortion in the results due to the use of Euclidean distances (Equation 1.19).

A comprehensive series of training experiments revealed that a reduced subset of only five independent features is able to model the relationship of ride-comfort with driving signals in a very satisfactory way. These features are: RMS{SWA, XACC neg, XACC pos, YACC}. Additionally, RMS{ERPM} was also incorporated due to the trends displayed in Figure 4.4 to consider the fuel-consumption perspective. In contrast, despite a number of variances show high correlation coefficients, they were found less suitable for modeling driving behavior than the corresponding RMS values. The above selected features will be used to develop a SOM-based DS model and provide drivers with specific recommendations.

The number of output neurons of the SOM was initially selected with the Vesanto's rule [136], which defines the optimal number of neurons as $M = 5\sqrt{K}$. Thus, a 14 × 14 SOM topology (i.e., $M = 196$) was defined and recursively trained. However, we carried out experiments ranging from 13×13 to 15×15 maps, achieving the best results for the latter, so, a 15 × 15 map was built and trained to improve the feature extraction capabilities.

In Figure 4.5 the SOM results are analyzed by means of displaying the input weight planes. These planes depict the weights associated with each input for each neuron, and reflect the input magnitudes that are expected to cause a hit for each neuron (i.e., the neuron is the best matching unit for that particular input). Thus, considering that lighter colors represent the higher values, several assertions can be made by visually analyzing the weight values. As can be seen, all the 5 planes are clearly different, which indicates that the inputs show no correlation between them.

It can also be seen that drivers with high values of SWA tend to show also high levels of YACC, fact that conditions the ride comfort of the car occupants (see top left corner). On the other hand, several neurons of the bottom left part of both ERPM and XACC pos show high values as well, which also might show a relationship with discomfort. Regarding XACC neg, high values for this variable are displayed at the top-right part of its corresponding map, which could show a certain level of relationship with negative acceleration peaks, derived of spurious braking. Finally, it is worth noting that the weight map for ERPM is particularly uniform,

which indicates that the addition of this variable does not contribute to the modeling of fuel-consumption.

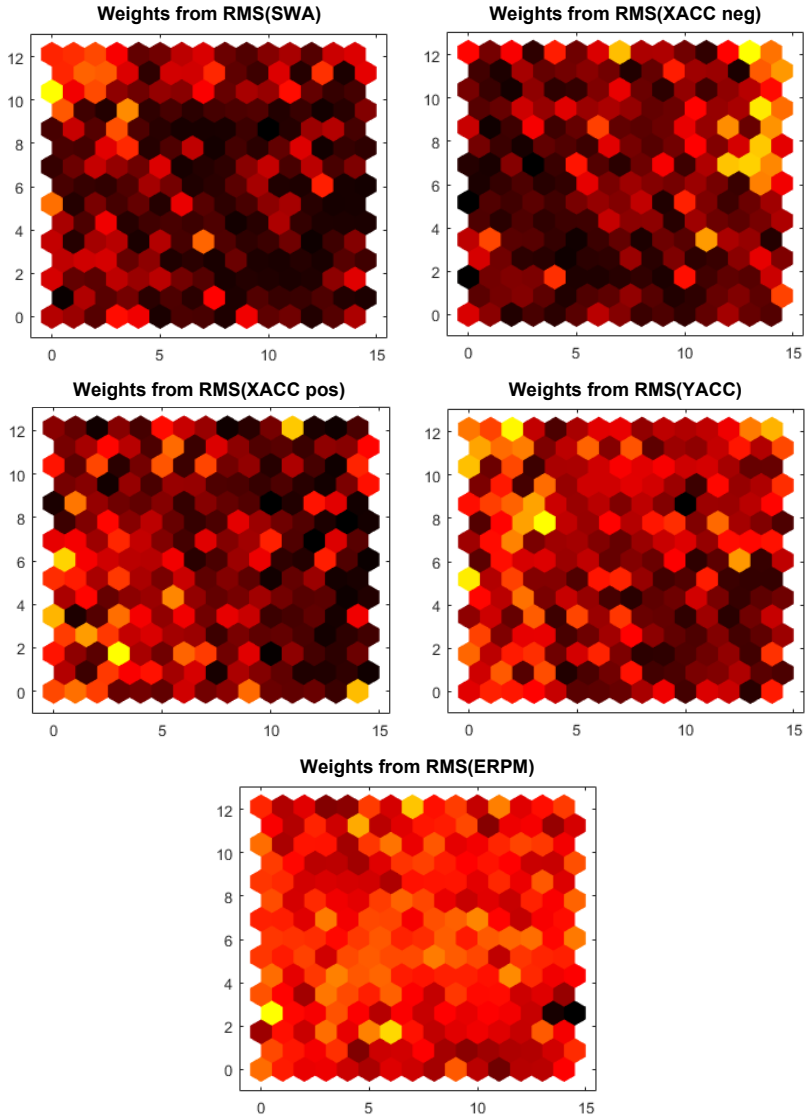


FIGURE 4.5: Input weight maps for the five selected input features, RMS{SWA,XACC neg,XACC pos} and RMS{YACC,ERPM}, from left to right and top to bottom. Lighter colors represent higher values of the corresponding features for each of the 15×15 neurons of the SOM.

To extract more solid conclusions that those provided by the visual inspection of the input planes, several partitions can be made on the map

in order to exploit the granularity of the algorithm and to group similar neurons into defined clusters. It should be remarked that, according to the k-NN algorithm referred to in Section 1.2.5, generally, the finer the partition, the greater the number of clusters found in the SOM. In this case, a 3-cluster partition was used as an starting point to assess the relationship with the ride comfortof drivers. This partition is shown in Figure 4.6.

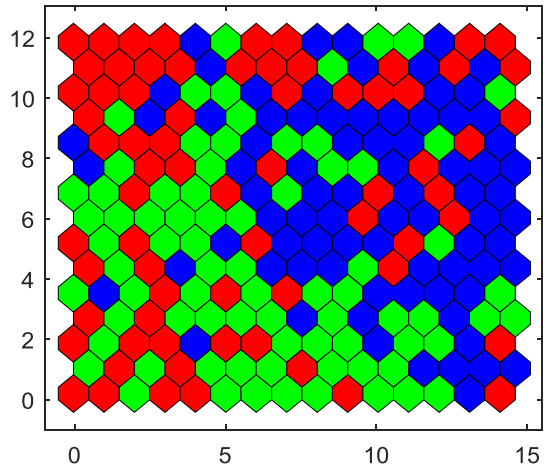


Figure 4.6: SOM partitioned to separate neurons between 3 clusters of ride comfort. This partition is based on the weights obtained during the training of the 15 × 15-neuron SOM and produced by carefully selecting the k of the k-NN partitioning algorithm.

When comparing Figure 4.6 with Figure 4.5, we can see several similarities, particularly with the input maps of SWA and YACC. Thus, if we compare again the top left section of the input map of SWA, we can see that the light neurons surrounded by the dark ones of the upper left corner match with the red neurons of the upper left corner of Figure 4.6. The same happens with the bottom left corner of YACC, for which, when comparing with the bottom left corner of Figure 4.6, it can be seen that the lighter neurons correspond with the elements of the red cluster, while the darkest ones correspond with the blue colored elements.

Once the 3 different clusters are displayed and analyzed, a quantitative evaluation of the ride comfort is performed. For that purpose, the mean values and variances of the selected variables are computed and displayed in Table 4.2.

Table 4.2: Average values and variances for discomfort for the 3-cluster classification.

Variable / Cluster		Blue	Green	Red
$MSDV_y$	Avg.	1.27	1.36	3.00
(m/s^2)	Var.	0.31	0.33	1.24
Vomit rate	Avg.	0.73	0.81	1.50
(m/s^2)	Var.	0.07	0.07	0.27
nx pos	Avg.	0.05	1.40	2.73
	Var.	0.19	36.8	94.2
nx neg	Avg.	2.30	0.42	3.69
	Var.	76.7	11.3	175.7
ny	Avg.	0.42	0.68	4.78
	Var.	1.95	3.87	48.0

As can be seen, the red cluster, which according to the description of the input planes has high values of SWA and YACC, depicts average values of the continuous discomfort variables that double those for the green and blue clusters, which compile much lower SWA and YACC values. In the same fashion, Table 4.2 shows that nx pos and ny, which model transient discomfort peaks, are noticeably higher for the red cluster than for the blue one. This trend, however, is not kept for nx neg, which has a minimum for the green cluster, increasing for the blue one. Regarding variances, a general increasing trend of variance jointly with the average values can be seen. This trend is related to the KDE graphs on Figure 4.4, where low motion sickness, low consumption drivers show more pointy surfaces with their probabilities lying in a much smaller area, reflected in the low variance of the low motion sickness drivers. Thus, in the case of the motion sickness variables, their variances are relatively low, which means that the generated clusters are fairly compact and well separated for the assessment of this feature. In contrast, the transient discomfort peaks' variance values are high due to some drivers being more likely to present spurious acceleration peaks above a certain threshold, as displayed in Figure 4.3.

It is worth to remark that the average values of comfort variables are slightly lower for the blue cluster than for the green one, being the difference between the average values of nx pos the major determinant for their separation. For these reasons, this partition can be considered valid for ride comfort. Nonetheless, with the aim of uniforming the gra-

dient of the average values for each group, the fineness of the clustering was further increased until a 5-class grouping was achieved.

Table 4.3: Average values and variances for discomfort for the 5-cluster classification.

Variable / Cluster		Blue	Green	Yellow	Magen.	Red
$MSDV_y$	Avg.	1.09	1.19	1.65	1.85	3.25
(m/s^2)	Var.	0.18	0.22	0.56	0.46	1.58
Vomit rate	Avg.	0.64	0.72	0.94	1.01	1.62
(m/s^2)	Var.	0.04	0.05	0.11	0.09	0.34
nx pos	Avg.	0.02	0.18	3.70	0.39	0.98
	Var.	0.05	0.78	116.3	1.84	17.3
nx neg	Avg.	1.20	0.21	0.37	5.27	2.21
	Var.	26.9	1.87	7.18	229.8	61.6
ny	Avg.	0.05	0.23	1.02	2.02	5.74
	Var.	0.16	0.89	5.43	10.5	69.2

To assess how the finer clustering affects to the discomfort ratings, the average and variance values were extracted and displayed in Table 4.3. In this case, the red cluster stands out among the remaining 4 clusters in terms of VR and $MSDV_y$, which are, by far, the highest. With respect to the other clusters, it can be seen that a full spectrum of classes, ranging from intermediate to low VR and $MSDV_y$, has been found. Regarding the discontinuous discomfort variables, the trend displayed for 3-clusters is kept. Thus, from the maximum displayed for the magenta class, nx neg uniformly decreases jointly with VR and $MSDV_y$ until the blue cluster is reached, increasing. Nevertheless, the yellow cluster stands out among the others, since, despite having an intermediate level of VR, it shows an increase on nx pos. This means that the continuous ride discomfort is not always related to the spuriusness of drivers.

Nonetheless, when inspected 2-by-2, the addressed clusters, particularly the couple blue-green and the couple yellow-magenta are very similar in terms of ride comfort. This resembles a 3-cluster classification, so, instead of using the latter 5-class clustering, the former 3-class one seems simpler and still useful to extract conclusions about ride comfort.

4.5 Deployment of the Driver Advice Module

With the SOM-based in-car DS modules properly developed and trained, achieving meaningful separation of the input samples into clusters, theride comfort improvement module is to be deployed, according to Figure 4.2. For this purpose, the causes of the ride comfort characteristics of the selected clusters are assessed and meaningful recommendations are provided according to them.

4.5.1 Ride Comfort Clusters' Characteristics

According to Table 4.2 in Section 4.4, three ride comfort clusters are enough to distinguish drivers regarding their VR and $MSDV_y$, being the red one the cluster with the highest rate, while the blue one represents the lowest.

Intermediate motion sickness data is compiled into the green cluster. Thus, with the clustering corresponding with Table 4.2, Figure 4.7 is elaborated, and, with the depicted characteristics, we can classify the clusters as follows:

- *High discomfort (red)* corresponds to drivers with high values of VR and $MSDV_y$, and, consequently, with elevated SWA and YACC, which means that the main feature of these motorists is that they tend to use the steering wheel aggressively, following swift cornering and overtaking strategies.

- *Medium discomfort (green)* drivers show moderate to low values of VR and $MSDV_y$, and, consequently, with moderate to low SWA and YACC, while their values of XACC pos are medium-high, pointing out that the extensive use of the gas pedal is the main conditionant of ride comfort.

- *Low discomfort (blue)* class shows the lowest values of VR and $MSDV_y$, so SWA is also kept in the lower range, XACC pos is kept low too, which means that these drivers tend to operate the gas pedal smoothly. Conversely, the values of XACC neg are higher than for the green class, which, jointly with the high correlation with nx neg, suggest that these drivers could use the brake pedal more thoroughly with spurious braking peaks.

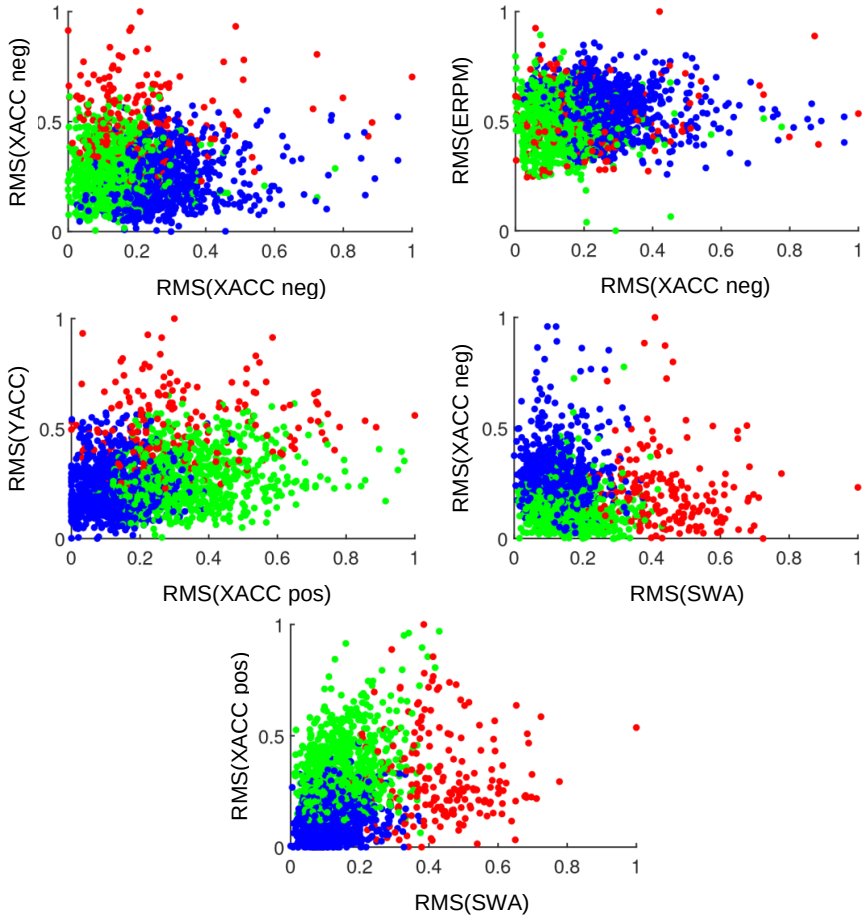


FIGURE 4.7: Two-dimensional views of the three-cluster distribution for ride comfort. The clusters were labeled as high discomfort (red), medium discomfort (green) and low discomfort (blue).

Additionally, by the inspection of Figure 4.7, it can be seen that no separation is achieved for ERPM, which is coherent with the input planes displayed in Figure 4.5.

Thus, according to the enumerated characteristics, as well as with the comfort variables displayed in Table 4.2 taken into consideration, the advice shown in Table 4.4 could be provided to drivers to improve their DS regarding ride comfort.

Table 4.4: Suggested actions to improve ride-comfort.

Comfort cluster	Driver Advice
High (red)	Operate steering wheel more smoothly
Medium (green)	Release gas pedal
Low (blue)	Avoid braking peaks*

*__Note__: This advice is only provided when a braking peak above a certain threshold occurs.

With these recommendations, and with the foregoing considerations in mind, we can elaborate Table 4.5 to show up the potential decrease on the likelihood of occupants to get motion sick in case of the DS recommendations were completely followed by the drivers.

Table 4.5: Expected VR reduction between clusters.

Current Cluster	Target Cluster	Vomit rate Reduction (%)	$MSDV_y$ Reduction (%)
Medium (green)	*Low (blue)*	9.88	6.62
High (red)	*Medium (green)*	46.0	54.7
High (red)	*Low (blue)*	51.3	57.7

As depicted in Table 4.5, a reduction of up to the 57.7% can be expected if a highly discomfortable driver could modify his/her DS so as to mimic a low motion-sickness motorist. Nevertheless, if he/she could only make it to drive like a medium motion-sickness driver, a promising reduction of up to the 54.7% in ride discomfort parameters could be achieved.

This system, which responds to a 5-input 15×15 SOM, can be easily accelerated by using the HW core developed in Section 3.6.1, which, according to Equation 3.3, would return its evaluation results in 14 clock cycles, allowing the very-high-performance evaluation of ride comfort parameters and the integration of the system with the existing electronics of the vehicle.

4.6 System Robustness Verification

Throughout this chapter it is shown that, with an appropriate selection of features, unsupervised ML methods can be used to extract conclu-

sions according to ride comfort regarding each driver's characteristics. For that purpose after the window size was properly sized and the driving signals were carefully selected regarding their PCCs, a SOM for the assessment of the ride comfort was properly sized and trained. Finally, the classes identified by the SOM were thoroughly examined to check that they did match with the already existing knowledge, and, analyzing their members' values, recommendations according to their characteristics were elaborated to create 3 comfort clusters.

To prove that the ride-comfort classification is robust, coherent with the previous knowledge, and tends to classify drivers according to the KDE displayed in Figure 4.4, it is cross compared with the 3-cluster, fuel-consumption classification described in Section 3.4.3. Thus, heat maps for drivers 2, 6, 8, 14, 18 and 20 were elaborated. Figure 4.8 represents the distribution of the cross classification for each of the drivers. If these distributions are compared with their corresponding KDE surfaces (Figure 4.4), it can be seen that drivers with the sharpest kernel density distribution, such as Driver 2 and Driver 6, tend to show a predominant medium discomfort and low fuel consumption cluster, driving during more than the half of the samples in this class. In contrast, for drivers such as Driver 8, Driver 14, or Driver 18 the distribution is more disperse, with several clusters having a high predominance, lacking a single predominant driving cluster.

On the other hand, conclusions about either ride comfort (i.e. by rows) or fuel consumption (i.e. by columns) can be extracted by analyzing Figure 4.8. Thus, the predominant cluster for Driver 2 is medium, with the 72.5% of his/her windows laying into this category. Regarding fuel consumption, it can be seen that low is the winning classification, with 67.4% of his/her windows being classified for this category. Driver 6 behaves similarly to Driver 2, the most frequent combination is also medium-low, with 74.8% and 56.8%, respectively. For Driver 8, medium-low, with 43.5% and 53.9%, is again the winner combination. But, in this case, the very low consumption class is also remarkable, with the 27%. Driver 14 and Driver 18 are very similar, with the highly discomfortable class outstanding among the others, with the 59.5% and the 60.9% of the driving windows, while most of the times they are classified as very low consumption drivers (45.7% and 54.8%).

Finally, Driver 20 shows a fairly uniform distribution, so, this driver cannot be clearly classified separately for ride comfort or fuel consumption. In contrast, the cluster intersection shows 9 clearly separated categories that can be easily distinguished, proving that the followed approach is useful to perform a deeper insight into driving behavior,

specially for drivers with no predominant class.

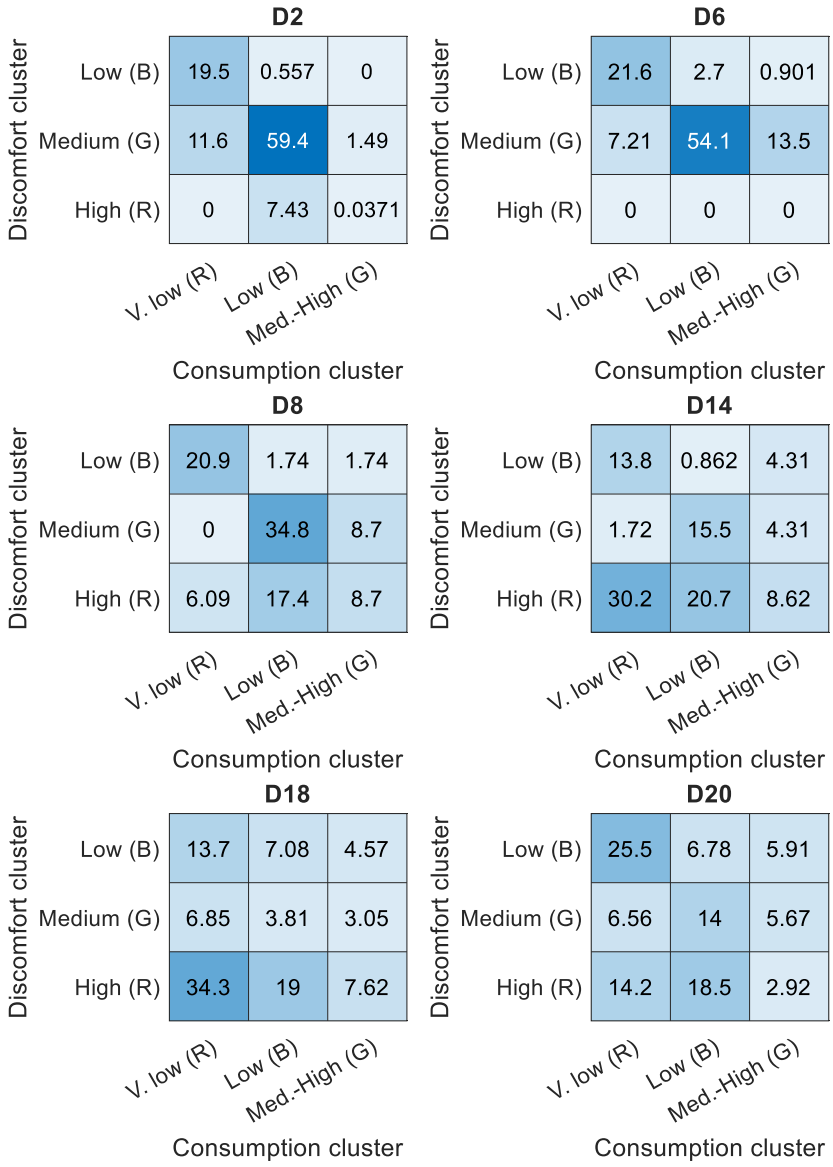


Figure 4.8: Heatmaps of the percentage distribution of the intersection of the ride comfort and fuel consumption clusters for drivers 2, 6, 8, 14, 18 and 20. B stands for blue, G for green, and R for red.

Regarding recommendations, this section suggests that advice for both eco-driving and ride comfort could be provided simultaneously, due to the coherence of the intersection of the two types of classification.

4.7 Concluding Remarks

The main motivation of this chapter was the development of an ADAS to increase ride comfort of the passengers, while taking into account the eco-driving viewpoint. The proposed solution provides the driver with a set of recommendations, in natural language, with the aim of improving his/her DS. The system is composed of two main subsystems: a SOM-based DS classifier and the driver advice module. The first one, consists of a SOM that was trained with real-world driving data. The classifier subsystem is able to group drivers regarding their ride comfort characteristics. The second subsystem identifies the underlying causes of the DS-associated lack of comfort, and provides advice according to them.

The aforementioned recommendations are designed to be easily understandable by most of the drivers, and, if they were completely followed, noticeable reductions in the comfort compromising parameters could happen. As shown, if a driver could modify his/her DS with the help of the recommendations from the most discomfortable group to the most comfortable one, the discomfort indicators would improve up to the 57.7%, improving the comfort perception of the vehicle occupants drastically and being compatible with the previously existing eco-driving knowledge.

The 5-input 15×15 SOM of this system, can also be easily accelerated by using the development described in Section 3.6.1, to return a valid solution in 14 clock cycles, allowing the very-high-performance evaluation of ride comfort parameters and the integration of the system with current vehicle designs.

Chapter 5

ACAP-Based HW/SW Solution for ADAS

5.1 Overview

In this chapter, a HW/SW solution to deploy a variety of ADAS in automobiles is proposed. This solution allows to integrate heterogeneous HW and SW applications in the same device while keeping high performance rates. For that purpose, all the pieces of HW developed along the precedent chapters of the document are deployed in the proposed architecture. To shape a very high-performance solution, an ACAP from the Xilinx Versal family (refer to Section 1.4) has been used.

This chapter is organized as follows. In Section 5.2, the proposed solution is displayed and described. Section 5.3 presents how the HW blocks have been created with the spotlight put on the design. The development of the complete HW/SW block design is displayed in Section 5.4. Finally, resource consumption, timing performance and power consumption are assessed in Section 5.5.

5.2 Proposed Solution

The block diagram of the ACAP-based implementation of the HW/SW solution for ADAS is depicted in Figure 5.1. It is a hybrid HW/SW architecture implemented on the Xilinx Versal XCVM1802-2MSEVSVA2197 ACAP [150] using the Xilinx VMK180 development board [151]. The HW partition of the system, displayed over the gray zone corresponding to the logic, DSP and memory resources, was implemented using VHDL language and the graphic tools of the Xilinx Vivado 2021.2 design suite. This partition includes the HW accelerators of the applications developed in Chapters 2, 3 and 4, for ACC personalization, eco-driving

advice, and ride-comfort advice, respectively. On the other hand, the SW partition, developed by means of the Xilinx Vitis 2021.2 SDK [229], is proposed to be deployed through a bare-metal C application running in the PS of the device. This SW is intended to acquire the vehicle bus data, compute all the ADAS significant features, share them with the HW partition, retrieve the responses of the HW blocks, compute the personalization parameters and DS advice, and send these personalization parameters to the vehicle's ECU to adjust the THW setpoint of ACC and to provide DS advices to drivers.

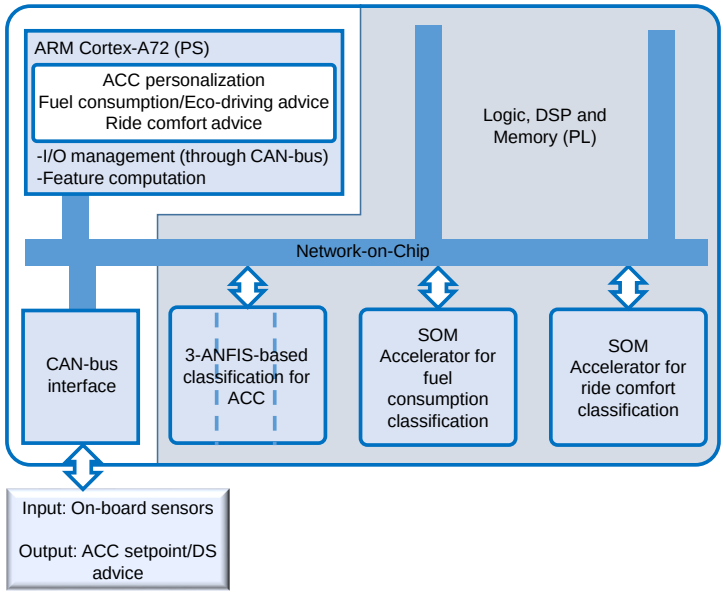


Figure 5.1: Block diagram of the ACAP-based HW/SW solution for ADAS. The HW accelerators are deployed in the PL partition, remarked in gray, and connected to the NoC. The SW partition is ran in the PS.

The PS, apart from all the tasks above, performs the system monitoring and is the responsible of the interfacing between the outer side (i.e. the vehicle's bus) and the inner side (i.e., the HW/SW solution itself). According to Figure 5.1, all the blocks are connected through the NoC, which, as described in Section 1.4, allows extremely high-throughput communications between blocks by seamlessly connecting the different domains of the device.

5.3 HW Partition Blocks

With the design displayed, the blocks of the HW partition have been developed by adapting their already developed PSoC-based architectures to the ACAP paradigm. As shown in Figure 5.1, two types of HW blocks needed to be deployed: the ANFIS block (whose internal architecture has been thoroughly described in Section 2.6), and the SOM block (refer to Section 3.6). These blocks are fully parameterized and re-configurable.

To enable the modules to be connected to the internal NoC of the Versal device, they have been wrapped as AXI4 bus peripherals. This means that their inputs and outputs are linked to registers that interface with the bus. After that, the Vivado design suite transparently manages the interfacing with the NoC. These peripherals are shown in Figure 5.2.

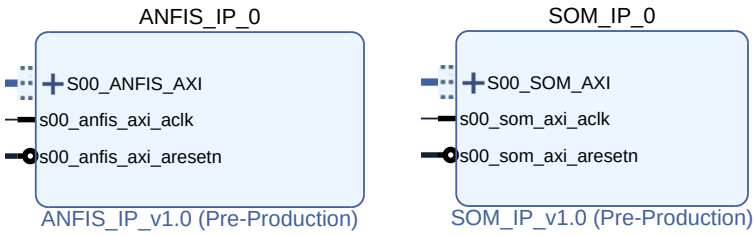


Figure 5.2: View of the ready-to-connect ANFIS and SOM HW blocks. `S00_XX_AXI` are the AXI4 connections of the blocks that link them to the NoC. `s00_xx_axi_aclk` are the bus clocks and the clock signals of the blocks. `s00_xx_axi_aresetn` are the reset signals.

It is worth noting that, in the case of the ANFIS block, the parameters of number of inputs, number of fuzzy rules and their structure, the number of MFs and their shapes, and the input and output resolutions can be adjusted. As for the SOM block, the number of inputs and outputs, the number of neurons and their weights, and the input and output resolutions can be adjusted. This fact makes these modules highly versatile and flexible enough for a wide range of applications. Thus, for the application described in Figure 5.1, three ANFIS blocks for the ACC personalization solution of Chapter 2 are required. In addition, two SOM blocks for the eco-driving and ride comfort solutions of Chapters 3 and 4, respectively, had to be configured and deployed.

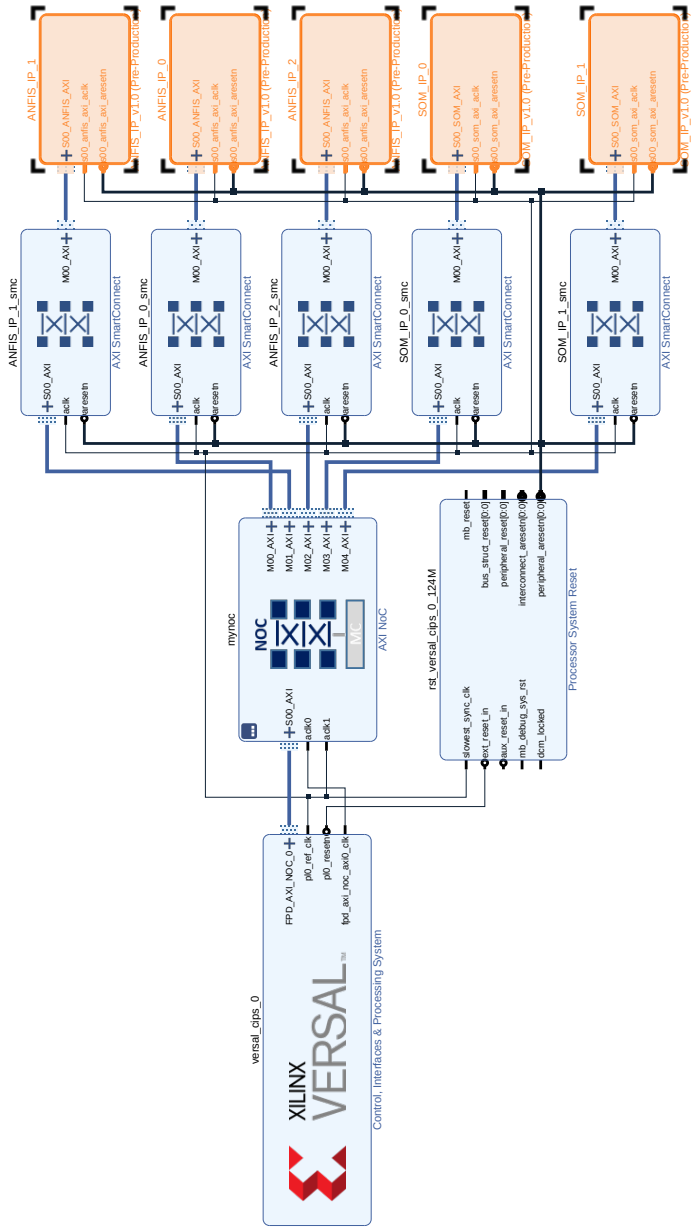


FIGURE 5.3: Block diagram of the entire system integration. The elaborated HW peripherals are remarked in orange. The SW partition is implemented and ran at one of the cores the Xilinx Versal PS block on the left. The NoC, represented by the blocks showing a network-like pattern, is managed transparently.

5.4 HW/SW Complete Design

Once the AXI peripherals are correctly wrapped and validated, the design that follows the guidelines of Figure 5.1 to incorporate these blocks is made by means of the Vivado IP integrator. The design shown in Figure 5.3 is next-to-totally automated, with only needing to perform the aggregation of the ANFIS and SOM AXI4 peripherals described in Section 5.3, labeled as `ANFIS_IP_0`, `ANFIS_IP_1`, `ANFIS_IP_2`, `SOM_IP_0` and `SOM_IP_1`.

These modules are remarked in orange in Figure 5.3. The remaining blocks apart from the Versal's SW core (which corresponds with the PS) are automatically added to setup the interface between the HW and the SW partitions through the NoC.

After this step is performed, the tool automatically allocates the addresses of the AXI registers of the elaborated peripherals in the memory space of the Versal's SW core in order to be accessed by the microprocessor and, consequently, enabling the information to be exchanged transparently from/to the the HW and the SW partition. The automatically elaborated address map is displayed in Figure 5.4.

Figure 5.4: Structure of the address map of the peripherals integrated in the system. The range of addresses reserved for each peripheral is displayed in blue, while the blank addresses are shown in gray.

It is worth to remark that despite the developed HW cores relying on a AXI4 interface to be connected with the SW partition of the system, the real connection between these elements does not happen through an AXI4 bus anymore, but through a NoC, as described in Section 1.4.

Hence, the previously described memory map peripherals are physically connected and allocated as described in Figure 5.5.

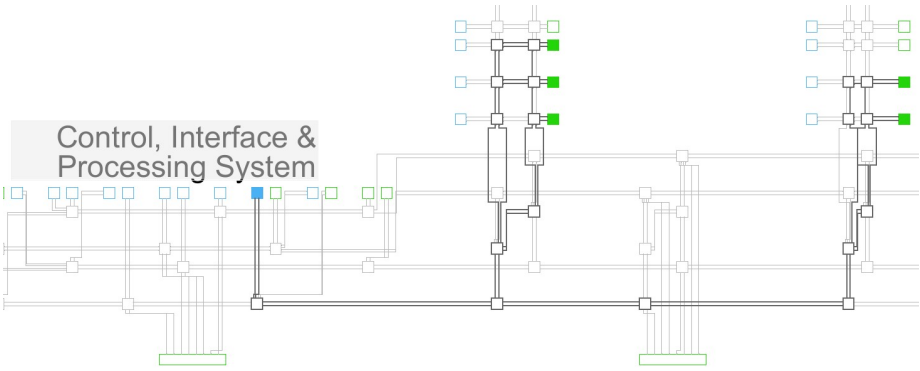


Figure 5.5: Overview of the implemented NoC with 5 wrapped cores in green. The blue block represents the gateway between the PS and the NoC, and the NoC's paths are visible in dark gray. The free paths, ready to feed more HW blocks (with green lines), are displayed in light gray.

In Figure 5.5, 6 different blocks can be seen: 1 blue block and 5 green blocks. The blue block corresponds with the connection of the PS of the Versal platform to the NoC, and the green ones with the physical interface between the NoC and the wrapped peripherals. As can be seen, they are connected by a system of pipes that represents the chip-embedded NoC. The management and routing of this network is automatically performed by the NoC master and is completely transparent. This means that the HW/SW communications can be performed in the same way as for the traditional AXI4 bus, reducing the level of complexity for the user.

Finally, the design is validated and the synthesis and implementation stages are run to deploy the development in the Versal platform.

As for the Xilinx Vitis-developed SW, the computation of data windows, sampling time and driving features, as well as the ADAS applications, are migrated directly from the SW partitions of Chapters2 and 3, with no additional work apart from the integration needed to be performed. Additionally, the sequence of operations needed to integrate the HW core that accelerates the ride comfort-dedicated SOM is incorporated. It is worth noting that, to communicate the HW and SW partitions, the Vivado 2021.2 design suite and the Vitis 2021.2 SDK automatically elaborate the board support package (BSP), that incorporates functions and definitions that ease the reading of the results and the input of the driving-related data from and to the AXI peripherals.

To finish with the complete application, a CAN bus peripheral, that would receive and send the ADASs-related information from and to the already existing systems of the car, would be used.

5.5 Resource Consumption and Performance

In this Section, the usage of the resources of the gray partition that represents the PS in the proposed architecture of Figure 5.1 is analyzed. Additionally, the timing performance of the entire solution is compared with that of the individual blocks described in Sections 2.6 and 3.6. Finally, the power consumption is analyzed.

Logic, DSP and Memory Resources Usage

The full HW platform was successfully implemented, with the post-implementation results displayed in Table 5.1. The three-ANFIS plus the two SOM integration platform fit into the selected ACAP's logic, leaving enough resources available for further system applications, escalations, or improvements.

TABLE 5.1: Post-implementation resources report (Xilinx Versal XCVM1802-2MSEVSVA2197).

Resource	Utilization	Available	% Used
LUT	53 408	899 840	5.94
FFs	39 331	1 799 680	2.19
RAM blocks	1.5	967	0.15
DSP	372	1968	18.90

Timing Performance

Before the deployment, the maximum operational frequency was calculated. For that purpose, the three-ANFIS plus two SOM architecture was implemented with a minimum clock period of 8 ns. This period selection was performed according to the maximum operation frequencies of the peripherals developed in Sections 2.6.2 and 3.6.2, where maximum clock frequencies of 140.41 MHz and 129.74 MHz were obtained for the ANFIS and the SOM developments, respectively. Thus, the initial period selection of $t = 8$ ns $\Rightarrow F = 125$ MHz, was selected so that it matched with the lowest time performance peripheral.

After implementation, a slack of 2.479 ns was obtained, which means that, according to Equation 2.5, the maximum clock frequency this HW integration can work at is $F_{max} = 181.12$ MHz. This frequency is much higher than those displayed in Sections 2.6.2 and 3.6.2 and is an evidence that, due to both the NoC and the 7 nm manufacturing process (both mentioned in Section 1.4), the performance of the Xilinx Versal platforms clearly supersedes that of ZynQ architectures, allowing computing on the edge.

The three ANFIS blocks present the same latency of 53 clock cycles (see Section 2.6.2), which means that, for the new maximum clock frequency of 181.12 MHz, these modules can return their solution in 292.61 ns. On the other hand, the latency for the 11×11, 4-input SOM developed in Chapter 3, according to Equation 3.3, is 12 clock cycles, returning its results in 66.25 ns. In the same manner, for the 15×15, 5-input SOM of Section 4.4, the latency is 14 clock cycles, that is to say, 77.29 ns. These results imply that the timing performance has been improved by the 28.99% for the ANFIS cores and by the 42.20% for the SOM cores when compared with the ZynQ family of devices. These processing times are fast enough for virtually any ADAS application and, due to the extraordinary scalability of the ACAPs, allow to even implement solutions that require very fast reaction times, such as safety-critical ADAS.

Power Consumption

In the light of Figure 5.6, the entire integrated system requires 12.166 W to operate, corresponding the 22% of this consumption to the dynamic power and the remaining 78%, to the static power. Thus, while the former corresponds to the energy spent by the developed design itself, the latter happens due to the leakage currents of the ACAP architecture, which is significant due to its size when compared to other solutions.

This power consumption information can be put into the context of current cars by considering that an average car relies on a range of 70 to 100 ECUs, and each of them consumes a current of 100 mA at 12 V [230]. This means that the electronics of a current car spent from 84 to 120 W to run its normal functionalities. On the other hand, in [231] the power consumption of several ADAS-intended sensors and processing elements is displayed, with rates from 12 W for LIDAR to 80 W to the ADAS ECU. Thus the 12.166 W of power consumption of the developed solution is in the line of the already existing systems, and even better if the comparison is limited to the ECU.

Consequently, the hybrid HW/SW implementation developed is an innovative solution between conventional SW-based approaches and novel FPGA-based, extreme performance architectures, which, provides an adequate trade-off between complexity, performance, and development time.

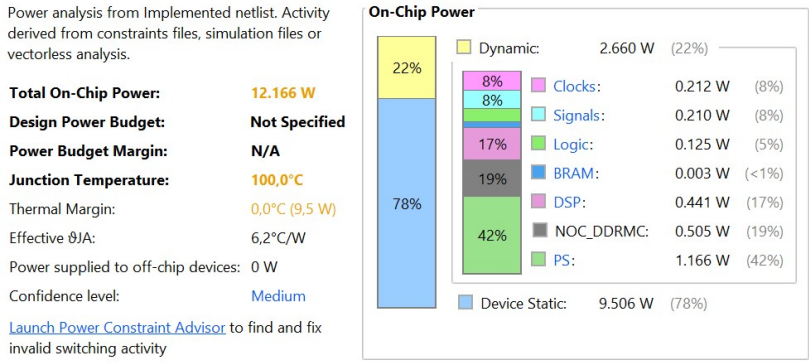


Figure 5.6: Power consumption post-implementation results, extracted from Vivado 2021.2. The maximum power consumption is 12.166 W.

5.6 Conclusions

In this chapter, a HW/SW design that integrates an heterogeneous set of ADAS-intended HW architectures has been described. The whole system has been successfully developed by means of a Xilinx Versal ACAP, a state-of-the-art device for high performance electronic implementations. This design joins both the HW and the SW domains to provide very high performance assessment of DS characteristics to personalize ADAS in real-time. It allows to adapt the accelerator architectures developed in Sections 2.6 and 3.6 to virtually any application that requires ANFIS or SOM algorithms with little design effort paid in only adjusting some parameters. Additionally, this development demonstrates the power of the Xilinx Versal devices to achieve timing performances that supersede those of the previously existing device families. Finally, this chapter shows the possibility of integrating several systems of an automobile into a re-configurable device, so that new features and HW updates could be implemented without physically changing any of the elements of the car systems' layout.

It is also worth to remark the adaptability of this type of platform not only for ADASs, but also for general purpose ML techniques, due to the HW allowing an unforeseen level of parallelism, with multiple cores

running at several speeds adapted to each concrete application. The SW partition plays an important role tool, with heterogeneous ARM cores for both high demanding applications and real-time processing, enabling SW scalability. This means that ACAPs could potentially substitute many of the systems of current cars, increasing the performance of the boarded electronics and reducing its power consumption.

Chapter 6

Conclusions and Future Work

In this work, several contributions to the state of the art of ADAS have been proposed and successfully developed. These systems, since they are intended to be implemented in automobiles, must be compact and energy efficient. On the other hand, the must also be powerful enough to manage and process the input data flow and to execute the deployed algorithms in real time. Hence, application-specific HW/SW architectures have been developed.

For the development of these systems, a data-based development strategy has been used, were data from naturalistic and non-naturalistic driving studies has been mined. This data mining-based strategy has allowed us to identify significant characteristics of human driving and, on the other hand, to isolate features of each individual's DS.

Several types of ML algorithms have been used to identify DS characteristics, thus, each system is able to infer the driving details by itself, and to group individual drivers regarding their handling features.

Thus, for the personalization of ACC systems, the SHRP2 NDS data base has been used to develop an ANFIS-based approach that classifies DS among three car-following styles. For that purpose, each DS is modeled by its corresponding ANFIS network, and the classification is done by identifying the one with the strongest activation. This method shows an state-of-the-art level of accuracy, with barely no confusions between groups and allows to select the desired THW for each individual.

Regarding eco-driving considerations, the Uyanik instrumented car non-NDS dataset has been used to develop a SOM-based eco-driving assessment system. To properly select the training variables of the system, a correlation analysis with simulated fuel consumption-data has been performed. With these highly correlated variables, the size of the SOM network has been selected and meaningful fuel-consumption driving behaviors have been clustered so that DS features automatically emerged. This has allowed to diagnose the weaknesses and strengths of

human drivers, and to provide them with instructional advice according to their particular characteristics, potentially allowing state-of-the-art fuel consumption and GHG reduction rates.

As for ride comfort, the same non-naturalistic dataset has been used to assess the DS-related compromising features by the use of SOMs. The training variables have been selected through a correlation analysis with significant ride-comfort compromising variables, and an SOM has been sized according to them. This SOM has been clustered and DS-related comfort compromising features emerged, enabling to assess the conflictive points of individual DS regarding discomfort, so that advice to correct these drawbacks could be provided.

Both the eco-driving and ride comfort advice systems have been jointly tested and evaluated to check if they are compatible with each other, showing that these systems have a high level of coherence when used together, which allows to provide simultaneous advice. Both systems' influence in potentially improving DS for eco-driving and ride comfort supersede the already existing techniques.

As for electronic implementation, VHDL-coded, PSoC-based generic HW cores for ANFIS and SOM neural networks have been designed. These cores have been parametrically coded so that their architecture automatically re-configures according to the particular parameters of each application. These cores allow very high performance, real-time DS evaluation and stand out when compared to other proposed solutions.

Regarding the final implementation, an integrated HW/SW solution for ADAS is proposed. This solution, based on a Xilinx Versal ACAP, enables to integrate a variety of heterogeneous HW and SW architectures within the same chip, ensuring high performance. This device also allows to seamlessly connect the designed architecture to the vehicle's buses through its communication peripherals. To develop the HW architecture, the PSoC-based generic HW-cores are adapted to the ACAP architecture, enabling them to connect to the NoC. The usage of the NoC allows vey high performance communication between the HW and the pieces of SW developed in the microprocessor of the ACAP. Additionally, it is shown that the Versal-based implementations outperform those carried out with other devices.

In sum, in this work a framework for the development of DS-based ADASs is provided. This framework, derived from human driving data, not only provides us with tools to enhance the non-autonomous driving scenarios, but also with behavioral information that could help to improve the attitude of the future autonomous cars in terms of longitu-

dinal control, eco-driving and ride comfort. These aspects are extremely important in the current context of steadily increasing road traffic and GHG emissions as a solution to improve road safety and energy efficiency. The former concept has a positive impact on potentially reducing the number of road fatalities, while the latter, has a positive impact on energy consumption and, consequently on economy, in a context of non-stop increasing energy prices. Many other ADAS, such as LKA/LDW, distracted driving advice or automated navigation could benefit of this data-based approach, showing more human-like behaviors that would increase the trust of car occupants in the system, encouraging the acceptance.

Regarding HW/SW development, in this work ACAPs are proven to be an efficient tool to implement a plethora of new functionalities in automobiles. These systems, due to their generous resources might potentially substitute the ECUs present in the currently marketed cars, increasing, on the other hand, the computational capabilities and reducing the consumption of energy. It is also worth to remark that, due to these devices being completely re-configurable, manufacturers could not only update the SW of the vehicle, but also the HW, allowing the implementation of new after-market functionalities.

Future Works

In future works, neuro-fuzzy sensor-based ACC capabilities could be enhanced by broadening the diversity of car-following scenarios. Both acceleration and braking will be analyzed. Moreover, a finer clustering approach could be investigated with the aim of categorizing driving scenarios according to, among others, weather conditions or lighting.

Regarding eco-driving, more SOM-based intelligent system applications could be investigated, adding more driving scenarios to those already researched for DS-related fuel consumption on highways and roads. It is worth remarking that further work can be done to decide which advice should be provided to drivers whose predominant DS is divided between non-contiguous clusters.

As for ride-comfort, SOM-based applications will be broadened, adding more driving scenarios to those already researched for DS-related ride comfort on highways and roads. On the other hand, the improvement on passenger ride comfort could be enhanced by using complementary questionnaires.

Additionally, we plan to deploy the ACAP-based HW/SW solution for ADAS in an actual car, so as to test the engagement of real drivers with the recommendations and personalizations provided by the system, as well as their effects on actual fuel consumption and discomfort reduction compared to built-in driving recommendation modes. Finally, for eco-driving, estimations of NOx, CO, and HC emissions could be performed to further analyze the effects of the system presented in this work.

Annex: Publications derived of this work

WOS-Indexed Journal Papers

[1] Ó. Mata-Carballeira, I. del Campo and E. Asua, "An eco-driving approach for ride comfort improvement," *IET Intelligent Transport Systems*, vol. 16, no. 2, pp. 186–205, 2021.

[2] Ó. Mata-Carballeira, M. Díaz-Rodríguez, I. del Campo and V. Martínez, "An intelligent system-on-a-chip for a real-time assessment of fuel consumption to promote eco-driving," *Applied Sciences (Switzerland)*, vol. 10, no. 18, p. 6549, 2020.

[3] Ó. Mata-Carballeira, J. Gutiérrez-Zaballa, I. del Campo and V. Martínez, "An FPGA-based neuro-fuzzy sensor for personalized driving assistance," *Sensors*, vol. 19, no. 18, p. 4011, 2019.

WOS-Indexed Conference Contributions

[4] Ó. Mata-Carballeira, I. del Campo, V. Martínez, and J. Echanobe, "A hardware/software extreme learning machine solution for improved ride comfort in automobiles," in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019.

[5] E. Sanz-Madoz, J. Echanobe, Ó. Mata-Carballeira, I. del Campo and V. Martínez, "Reduced Kernel Extreme Learning Machine for Traffic Sign Recognition," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 4101-4106.

[6] I. del Campo, E. Asua, V. Martínez, Ó. Mata-Carballeira, and J. Echanobe, "Driving style recognition based on ride comfort using a hybrid machine learning algorithm," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3251-3258.

[7] Ó. Mata-Carballeira, I. del Campo, V. Martínez, and J. Echanobe, "Deep Extreme Learning Machines with Auto Encoder for Speed Limit Signs Recognition," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 965-972.

Bibliography

[1] M. Iguchi, "Evolution of automobiles," in *Intelligent Vehicles Symposium, 1996., Proceedings of the 1996 IEEE*, IEEE, 1996, pp. 306–308.

[2] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[3] C. M. Farmer, "Effect of electronic stability control on automobile crash risk," *Traffic injury prevention*, vol. 5, no. 4, pp. 317–325, 2004.

[4] A. F. Larsson, "Driver usage and understanding of adaptive cruise control," *Applied ergonomics*, vol. 43, no. 3, pp. 501–506, 2012.

[5] M. Dikmen and C. M. Burns, "Autonomous driving in the real world: Experiences with tesla autopilot and summon," in *Proceedings of the 8th International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, ACM, 2016, pp. 225–228.

[6] C. Rolim, P. Baptista, G. Duarte, T. Farias, and J. Pereira, "Real-time feedback impacts on eco-driving behavior and influential variables in fuel consumption in a lisbon urban bus operator," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3061–3071, 2017.

[7] M. L. Jones, V. C. Le, S. M. Ebert, K. H. Sienko, M. P. Reed, and J. R. Sayer, "Motion sickness in passenger vehicles during test track operations," *Ergonomics*, vol. 62, no. 10, pp. 1357–1371, 2019.

[8] J. J. Hopfield, "Artificial neural networks," *IEEE Circuits and Devices Magazine*, vol. 4, no. 5, pp. 3–10, 1988.

[9] L. A. Zadeh, "Fuzzy logic," *Computer*, vol. 21, no. 4, pp. 83–93, 1988.

[10] Ó. Mata-Carballeira, J. Gutiérrez-Zaballa, I. Del Campo, and V. Martínez, "An fpga-based neuro-fuzzy sensor for personalized driving assistance," *Sensors*, vol. 19, no. 18, p. 4011, 2019.

[11] Ó. Mata-Carballeira, M. Díaz-Rodríguez, I. del Campo, and V. Martínez, "An intelligent system-on-a-chip for a real-time assessment of fuel consumption to promote eco-driving," *Applied Sciences*, vol. 10, no. 18, p. 6549, 2020.

[12] Ó Mata-Carballeira, I del Campo, and E Asua, "An eco-driving approach for ride comfort improvement," *IET Intelligent Transport Systems*, vol. 16, no. 2, pp. 186–205, 2022.

[13] *Shrp 2 - strategic highway research program 2 (shrp 2)*, `http://www.trb.org/StrategicHighwayResearchProgram2SHRP2/Blank2.aspx`, (Accessed on 29 June 2019).

[14] H. Abut, H. Erdoğan, A. Erçil, *et al.*, "Real-world data collection with "uyanik"," in *In-Vehicle Corpus and Signal Processing for Driver Behavior*, Springer, 2009, pp. 23–43.

[15] B. Loomis, "1900-1930: The years of driving dangerously," *Detroit News*, 2015. [Online]. Available: `https://eu.detroitnews.com/story/news/local/michigan-history/2015/04/26/auto-traffic-history-detroit/26312107/`.

[16] A. Glen, *The Highway Acts, 1862-1878, the Locomotive Acts, 1861-1878, and the General Provisions of the Turnpike Continuance Acts, 1863-1878*. Knight and Company, 1879.

[17] L. Brooke, *Ford model T: the car that put the world on wheels*. Motorbooks, 2008.

[18] L. Drummond, "How the car won the road: The surrender of atlanta's city streets, 1920-1929," Dissertation. Georgia State University, May 2021. [Online]. Available: `https://scholarworks.gsu.edu/history_diss/86`.

[19] M. Reichenbach, "Fasten your seat belt," *ATZ worldwide 2016 118:7*, vol. 118, pp. 3–3, 7 Jul. 2016. [Online]. Available: `https://link.springer.com/article/10.1007/s38311-016-0087-4`.

[20] E. F. van Beeck, G. J. Borsboom, and J. P. Mackenbach, "Economic development and traffic accident mortality in the industrialized world, 1962–1990," *International journal of epidemiology*, vol. 29, no. 3, pp. 503–509, 2000.

[21] E. Van Beckhoven, G. Bolt, and R. Van Kempen, "Theories of neighbourhood change and decline: Their significance for post-wwii large housing estates in european cities," in *Mass housing in Europe*, Springer, 2009, pp. 20–50.

[22] J. F. Müller and J. Gogoll, "Should manual driving be (eventually) outlawed?" *Science and engineering ethics*, vol. 26, no. 3, pp. 1549–1567, 2020.

[23] H. Leiber, A. Czinczel, and J. Anlauf, "Antiblockiersystem (abs) für personenkraftwagen," *BOSCH TECH BER*, no. 2, 1980.

[24] H.-W. Bleckmann, H. Fennel, J. Gräber, and W. W. Seibert, "Traction control system with teves abs mark ii," SAE Technical Paper, Tech. Rep., 1986.

[25] B. W. Smith, "Sae levels of driving automation," *Center for Internet and Society. Stanford Law School. http://cyberlaw. stanford. edu/blog/2013/12/sae-levels-drivingautomation*, 2013.

[26] "Automated vehicles for safety," *NHTSA*, 2018. [Online]. Available: `https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety`.

[27] B. Brown and E. Laurier, "The trouble with autopilots: Assisted and autonomous driving on the social road," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ACM, 2017, pp. 416–429.

[28] T. Litman, *Autonomous vehicle implementation predictions*. Victoria Transport Policy Institute Victoria, Canada, 2017.

[29] M. Feilhauer, J. Haering, and S. Wyatt, "Current approaches in hil-based adas testing," *SAE International Journal of Commercial Vehicles*, vol. 9, no. 2016-01-8013, pp. 63–69, 2016.

[30] C. Ho, N. Reed, and C. Spence, "Multisensory in-car warning signals for collision avoidance," *Human factors*, vol. 49, no. 6, pp. 1107–1114, 2007.

[31] R. N. Mahajan and A Patil, "Lane departure warning system," *International Journal of Engineering and Technical Research*, vol. 3, no. 1, pp. 120–123, 2015.

[32] S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad, "Comparative survey on traffic sign detection and recognition: A review," *Przegląd Elektrotechniczny, ISSN*, pp. 0033–2097, 2015.

[33] H. Winner, S. Witte, W. Uhler, and B. Lichtenberg, "Adaptive cruise control system aspects and development trends," SAE Technical Paper, Tech. Rep., 1996.

[34] M. Szarvas, U. Sakai, and J. Ogata, "Real-time pedestrian detection using lidar and convolutional neural networks," in *Intelligent Vehicles Symposium, 2006 IEEE*, IEEE, 2006, pp. 213–218.

[35] J. K. Kim, J. W. Kim, J. H. Kim, *et al.*, "Experimental studies of autonomous driving of a vehicle on the road using lidar and dgps," in *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, IEEE, 2015, pp. 1366–1369.

[36] N. M. Enache, M. Netto, S. Mammar, and B. Lusetti, "Driver steering assistance for lane departure avoidance," *Control engineering practice*, vol. 17, no. 6, pp. 642–651, 2009.

[37] B. Donmez, L. N. Boyle, and J. D. Lee, "Safety implications of providing real-time feedback to distracted drivers," *Accident Analysis & Prevention*, vol. 39, no. 3, pp. 581–590, 2007.

[38] D. Ding, J. Yoo, J. Jung, S. Jin, and S. Kwon, "Various lane marking detection and classification for vision-based navigation system?" In *Consumer Electronics (ICCE), 2015 IEEE International Conference on*, IEEE, 2015, pp. 491–492.

[39] H. Rohling, M.-M. Meinecke, K Mott, and L Urs, "Research activities in automotive radar," in *Physics and Engineering of Millimeter and Sub-Millimeter Waves, 2001. The Fourth International Kharkov Symposium on*, IEEE, vol. 1, 2001, pp. 48–51.

[40] M. Filipiak and J. Jajczyk, "Tests of the acc radar system in traffic conditions," *Computer Applications in Electrical Engineering*, vol. 14, 2016.

[41] A. Carvalho, A. Williams, S. Lefevre, and F. Borrelli, "Autonomous cruise control with cut-in target vehicle detection," in *Advanced Vehicle Control: Proceedings of the 13th International Symposium on Advanced Vehicle Control (AVEC'16), September 13-16, 2016, Munich, Germany*, CRC Press, 2016, p. 93.

[42] R. Mit, Y. Zangvil, and D. Katalan, "Analyzing tesla's level 2 autonomous driving system under different gnss spoofing scenarios and implementing connected services for authentication and reliability of gnss data," in *Proceedings of the 33rd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2020)*, 2020, pp. 621–646.

[43] C. J. Reinert-Weiss, H. Baur, S. A. Al Nusayer, D. Duhme, and N. Frühauf, "47-3: Distinguished student paper: Development of active matrix lcd for use in high resolution adaptive headlights," in *SID Symposium Digest of Technical Papers*, Wiley Online Library, vol. 48, 2017, pp. 696–699.

[44] P. Dollar, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 4, pp. 743–761, 2012.

[45] C.-I. Chang, *Hyperspectral imaging: techniques for spectral detection and classification*. Springer Science & Business Media, 2003, vol. 1.

[46] A. W. Nolin, "Recent advances in remote sensing of seasonal snow," *Journal of Glaciology*, vol. 56, no. 200, pp. 1141–1150, 2010.

[47] Y. Huang, S. J. Thomson, W. C. Hoffmann, Y. Lan, and B. K. Fritz, "Development and prospect of unmanned aerial vehicle technologies for agricultural production management," *International Journal of Agricultural and Biological Engineering*, vol. 6, no. 3, pp. 1–10, 2013.

[48] H. Kim, S. Kwon, and S. Kim, "Hyperspectral image-based nighttime vehicle light detection using spectral normalization and distance mapper for intelligent headlight control," *Sensors*, vol. 16, no. 7, p. 1058, 2016.

[49] A. W. Nolin and J. Dozier, "A hyperspectral method for remotely sensing the grain size of snow," *Remote sensing of Environment*, vol. 74, no. 2, pp. 207–216, 2000.

[50] A. Lambrechts, P. Gonzalez, B. Geelen, P. Soussan, K. Tack, and M. Jayapala, "A cmos-compatible, integrated approach to hyper- and multispectral imaging," in *Electron Devices Meeting (IEDM), 2014 IEEE International*, IEEE, 2014, pp. 10–5.

[51] N. Caber, P. Langdon, and P. J. Clarkson, "Designing adaptation in cars: An exploratory survey on drivers' usage of adas and car adaptations," in *International Conference on Applied Human Factors and Ergonomics*, Springer, 2019, pp. 95–106.

[52] L. Murray, "Inside the future cars [technology driverless cars]," *Engineering & Technology*, vol. 12, no. 10, 2017.

[53] J. M. Fleming, C. K. Allison, X. Yan, N. A. Stanton, and R. Lot, "Adaptive driver modelling in adas to improve user acceptance: A study using naturalistic data," *Safety Science*, 2018.

[54] M. Beggiato, M. Pereira, T. Petzoldt, and J. Krems, "Learning and development of trust, acceptance and the mental model of acc. a longitudinal on-road study," *Transportation research part F: traffic psychology and behaviour*, vol. 35, pp. 75–84, 2015.

[55] M. C. Panou, "Intelligent personalized adas warnings," *European Transport Research Review*, vol. 10, no. 2, p. 59, 2018.

[56] G. F. B. Piccinini, C. M. Rodrigues, M. Leitão, and A. Simões, "Driver's behavioral adaptation to adaptive cruise control (acc): The case of speed and time headway," *Journal of safety research*, vol. 49, 77–e1, 2014.

[57] D. Dörr, D. Grabengiesser, and F. Gauterin, "Online driving style recognition using fuzzy logic," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2014, pp. 1021–1026.

[58] A. C. Markham, *A brief history of pollution*. Routledge, 2019.

[59] M. Chappie and L. Lave, "The health effects of air pollution: A reanalysis," *Journal of Urban Economics*, vol. 12, no. 3, pp. 346–376, 1982.

[60] Z. J. Andersen, "Air pollution epidemiology," in *Traffic-Related Air Pollution*, Elsevier, 2020, pp. 163–182.

[61] G. W. Evans and S. V. Jacobs, "Air Pollution and Human Behavior," *Journal of Social Issues*, vol. 37, no. 1, pp. 95–125, 1981.

[62] N. C. Onat, M. Kucukvar, and O. Tatari, "Conventional, hybrid, plug-in hybrid or electric vehicles? State-based comparative carbon and energy footprint analysis in the United States," *Applied Energy*, vol. 150, pp. 36–49, 2015.

[63] A. Ahmad, Z. A. Khan, M. Saad Alam, and S. Khateeb, "A Review of the Electric Vehicle Charging Techniques, Standards, Progression and Evolution of EV Technologies in Germany," *Smart Science*, vol. 6, no. 1, pp. 36–53, 2018.

[64] G. Haddadian, M. Khodayar, and M. Shahidehpour, "Accelerating the Global Adoption of Electric Vehicles: Barriers and Drivers," *Electricity Journal*, vol. 28, no. 10, pp. 53–68, 2015.

[65] J. Aarstad and O. A. Kvitastein, "Has the popularity of battery electric vehicles in norway affected total new car sales? a synthetic control method study," *Applied Economics Letters*, pp. 1–4, 2020.

[66] C. Hanisch, T. Loellhoeffel, J. Diekmann, K. J. Markley, W. Haselrieder, and A. Kwade, "Recycling of lithium-ion batteries: A novel method to separate coating and foil of electrodes," *Journal of Cleaner Production*, vol. 108, pp. 301–311, 2015.

[67] O. Zehner, "Unclean at any speed," *IEEE Spectrum*, vol. 50, no. 7, pp. 40–45, 2013.

[68] N. Ortar and M. Ryghaug, "Should all cars be electric by 2025? The electric car debate in Europe," *Sustainability (Switzerland)*, vol. 11, no. 7, p. 1868, 2019.

[69] S Morgan, "Why Europe's roads are not yet swarming with electric cars," *Engineering Technology*, vol. 14, no. 4, pp. 12–13, 2019.

[70] X. Qi, M. J. Barth, G. Wu, K. Boriboonsomsin, and P. Wang, "Energy impact of connected eco-driving on electric vehicles," in *Road Vehicle Automation 4*, Springer, 2018, pp. 97–111.

[71] I. Lana, J. Del Ser, M. Velez, and E. I. Vlahogianni, "Road traffic forecasting: Recent advances and new challenges," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 2, pp. 93–109, 2018.

[72] L. Ye and T. Yamamoto, "Evaluating the impact of connected and autonomous vehicles on traffic safety," *Physica A: Statistical Mechanics and its Applications*, vol. 526, p. 121 009, 2019.

[73] Y. Qin, H. Wang, and B. Ran, "Impact of connected and automated vehicles on passenger comfort of traffic flow with vehicle-to-vehicle communications," *KSCE Journal of Civil Engineering*, vol. 23, no. 2, pp. 821–832, 2019.

[74] O. Oviedo-Trespalacios, V. Truelove, B. Watson, and J. A. Hinton, "The impact of road advertising signs on driver behaviour and implications for road safety: A critical systematic review," *Transportation research part A: policy and practice*, vol. 122, pp. 85–98, 2019.

[75] M. Elbanhawi, M. Simic, and R. Jazar, "In the passenger seat: Investigating ride comfort measures in autonomous cars," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 3, pp. 4–17, 2015.

[76] I. Saniee, S. Kamat, S. Prakash, and M. Weldon, "Will productivity growth return in the new digital era," *Bell Labs Technical Journal*, vol. 22, pp. 1–18, 2017.

[77] L. Svensson and J. Eriksson, *Tuning for ride quality in autonomous vehicle: Application to linear quadratic path planning algorithm*, 2015.

[78] N Karlsson and H Tjärnbro, *Motion sickness in cars, department of product and production development*, 2012.

[79] K. E. Money, "Motion sickness.," *Physiological reviews*, vol. 50, no. 1, pp. 1–39, 1970.

[80] N. J. Mansfield, *Human response to vibration*. CRC press, 2004.

[81] G. Bertolini and D. Straumann, "Moving in a moving world: A review on vestibular motion sickness," *Frontiers in neurology*, vol. 7, p. 14, 2016.

[82] B. Keshavarz, A. E. Philipp-Muller, W. Hemmerich, B. E. Riecke, and J. L. Campos, "The effect of visual motion stimulus characteristics on vection and visually induced motion sickness," *Displays*, vol. 58, pp. 71–81, 2019.

[83] N. M. Yusof, J. Karjanto, S. Sulaiman, J. Terken, F. Delbressine, and M. Rauterberg, "Effect of improving gravito-inertial force of the vehicle occupants in reducing severity of motion sickness," *Proceedings of Mechanical Engineering Research Day*, vol. 2020, pp. 6–7, 2020.

[84] F. Alonso, C. Esteban, L. Montoro, and A. Serge, "Conceptualization of aggressive driving behaviors through a perception of aggressive driving scale (pad)," *Transportation research part F: traffic psychology and behaviour*, vol. 60, pp. 415–426, 2019.

[85] V. Perepjolkina and V. Renge, "Drivers' age, gender, driving experience, and aggressiveness as predictors of aggressive driving behaviour," *Signum Temporis*, vol. 4, no. 1, p. 62, 2011.

[86] P. Hedrich, E. Lenz, and P. F. Pelz, "Minimizing of kinetosis during autonomous driving," *ATZ worldwide*, vol. 120, no. 7, pp. 68–75, 2018.

[87] J. Iskander, M. Attia, K. Saleh, *et al.*, "From car sickness to autonomous car sickness: A review," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 62, pp. 716–726, 2019.

[88] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM Journal of Research and Development*, vol. 3, no. 3, pp. 210–229, 1959.

[89] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, 2015.

[90] Z. John Lu, "The elements of statistical learning: Data mining, inference, and prediction," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, vol. 173, no. 3, pp. 693–694, 2010.

[91] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[92] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[93] S. B. Kotsiantis, I Zaharakis, and P Pintelas, "Supervised machine learning: A review of classification techniques," *Emerging artificial intelligence applications in computer engineering*, vol. 160, pp. 3–24, 2007.

[94] V. Vapnik, I. Guyon, and T. Hastie, "Support vector machines," *Mach. Learn*, vol. 20, no. 3, pp. 273–297, 1995.

[95] K. P. Murphy *et al.*, "Naive bayes classifiers," *University of British Columbia*, vol. 18, 2006.

[96] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[97] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," *Journal of the American statistical Association*, vol. 97, no. 458, pp. 611–631, 2002.

[98] L. E. Peterson, "K-nearest neighbor," *Scholarpedia*, vol. 4, no. 2, p. 1883, 2009.

[99] G. A. Seber and A. J. Lee, *Linear regression analysis*. John Wiley & Sons, 2012, vol. 329.

[100] R. E. Kass, "Nonlinear regression analysis and its applications," *Journal of the American Statistical Association*, vol. 85, no. 410, pp. 594–596, 1990.

[101] J. A. Nelder and R. J. Baker, "Generalized linear models," *Encyclopedia of statistical sciences*, vol. 4, 2004.

[102] Z. Lofti, "Fuzzy sets," *Journal of Information and Control*, vol. 8, no. 3, pp. 338–353, 1965.

[103] M. L. Zepeda-Mendoza and O. Resendis-Antonio, "Hierarchical agglomerative clustering," in *Encyclopedia of Systems Biology*, Springer, 2013, pp. 886–887.

[104] J. Burkardt, "K-means clustering," *Virginia Tech, Advanced Research Computing, Interdisciplinary Center for Applied Mathematics*, 2009.

[105] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015.

[106] M. M. Van Hulle, "Self-organizing maps," in *Handbook of Natural Computing*, Springer, 2012, pp. 585–622.

[107] L. R. Rabiner and B.-H. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986.

[108] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Acm sigmod record*, ACM, vol. 22, 1993, pp. 207–216.

[109] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.

[110] M. J. Zaki, "Scalable algorithms for association mining," *IEEE transactions on knowledge and data engineering*, vol. 12, no. 3, pp. 372–390, 2000.

[111] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, ACM, vol. 29, 2000, pp. 1–12.

[112] S Agatonovic-Kustrin and R Beresford, "Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research," *Journal of pharmaceutical and biomedical analysis*, vol. 22, no. 5, pp. 717–727, 2000.

[113] G.-B. Huang, Y.-Q. Chen, and H. A. Babri, "Classification ability of single hidden layer feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799–801, 2000.

[114] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[115] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, p. 533, 1986.

[116] D. F. Specht, "A general regression neural network," *IEEE transactions on neural networks*, vol. 2, no. 6, pp. 568–576, 1991.

[117] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489 –501, 2006.

[118] ——, "Real-time learning capability of neural networks," *IEEE Trans. Neural Networks*, vol. 17, no. 4, pp. 863–878, 2006.

[119] C. Carlsson and R. Fullér, "Fuzzy multiple criteria decision making: Recent developments," *Fuzzy sets and systems*, vol. 78, no. 2, pp. 139–153, 1996.

[120] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—i," *Information sciences*, vol. 8, no. 3, pp. 199–249, 1975.

[121] K. Mehran, "Takagi-sugeno fuzzy modeling for process control," *Industrial Automation, Robotics and Artificial Intelligence (EEE8005)*, vol. 262, pp. 1–31, 2008.

[122] D. Nauck, F. Klawonn, and R. Kruse, *Foundations of neuro-fuzzy systems*. John Wiley & Sons, Inc., 1997.

[123] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 6, pp. 693–699, 1985.

[124] J.-S. R. Jang *et al.*, "Self-learning fuzzy controllers based on temporal backpropagation," *IEEE Transactions on neural networks*, vol. 3, no. 5, pp. 714–723, 1992.

[125] J.-S. Jang, "Anfis: Adaptive-network-based fuzzy inference system," *IEEE transactions on systems, man, and cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[126] J.-S. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 378–406, 1995.

[127] C.-T. Lin, C. S. G. Lee, *et al.*, "Neural-network-based fuzzy logic control and decision system," *IEEE Transactions on computers*, vol. 40, no. 12, pp. 1320–1336, 1991.

[128] H. R. Berenji, P. Khedkar, *et al.*, "Learning and tuning fuzzy logic controllers through reinforcements," *IEEE Transactions on neural networks*, vol. 3, no. 5, pp. 724–740, 1992.

[129] D. Nauck and R. Kruse, "A neuro-fuzzy method to learn fuzzy classification rules from data," *Fuzzy sets and Systems*, vol. 89, no. 3, pp. 277–288, 1997.

[130] ——, "Neuro-fuzzy systems for function approximation," *Fuzzy sets and systems*, vol. 101, no. 2, pp. 261–271, 1999.

[131] T. Kohonen, E. Oja, O. Simula, A. Visa, and J. Kangas, "Engineering applications of the self-organizing map," *Proceedings of the IEEE*, vol. 84, no. 10, pp. 1358–1384, 1996.

[132] T. Kohonen, M. R. Schroeder, and T. S. Huang, *Self-Organizing Maps*, 3rd. Berlin, Heidelberg: Springer-Verlag, 2001.

[133] D. Miljković, "Brief review of self-organizing maps," in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2017, pp. 1061–1066.

[134] J. Vesanto and E. Alhoniemi, "Clustering of the self-organizing map," *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, vol. 11, pp. 586–600, Feb. 2000.

[135] A. A. Akinduko and E. M. Mirkes, "Initialization of Self-Organizing Maps: Principal Components Versus Random Initialization. A Case Study," 2012. arXiv: 1210.5873.

[136] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas, "Self-organizing map in matlab: The som toolbox," in *In Proceedings of the Matlab DSP Conference*, 1999, pp. 35–40.

[137] G. Meiring and H. Myburgh, "A review of intelligent driving style analysis systems and related artificial intelligence algorithms," *Sensors*, vol. 15, no. 12, pp. 30 653–30 682, 2015.

[138] P. Angkititrakul, M. Petracca, A. Sathyanarayana, and J. H. Hansen, "Utdrive: Driver behavior and speech interactive systems for in-vehicle environments," in *2007 IEEE Intelligent Vehicles Symposium*, IEEE, 2007, pp. 566–569.

[139] M. Regan, A Williamson, R Grzebieta, and L Tao, "Naturalistic driving studies: Literature review and planning for the australian naturalistic driving study," in *Australasian college of road safety conference 2012, Sydney, New South Wales, Australia*, 2012.

[140] R. Eenink, Y. Barnard, M. Baumann, X. Augros, and F. Utesch, "Udrive: The european naturalistic driving study," in *Proceedings of Transport Research Arena*, IFSTTAR, 2014.

[141] V. L. Neale, T. A. Dingus, S. G. Klauer, J. Sudweeks, and M. Goodman, "An overview of the 100-car naturalistic study and findings," *National Highway Traffic Safety Administration, Paper*, vol. 5, p. 0400, 2005.

[142] T. A. Dingus, J. M. Hankey, J. F. Antin, *et al.*, *Naturalistic driving study: Technical coordination and quality control*, SHRP 2 Report S2-S06-RW-1. 2015.

[143] H. Abut, H. Erdoğan, A. Erçil, *et al.*, "Data Collection with "UYANIK": Too Much Pain; But Gains Are Coming," in *DSPincars, Istanbul*, Sabancı University, 2007.

[144] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," *IEEE Transactions on computer-aided design of integrated circuits and systems*, vol. 26, no. 2, pp. 203–215, 2007.

[145] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, "Field-programmable gate arrays," 1992. [Online]. Available: `http://link.springer.com/10.1007/978-1-4615-3572-0`.

[146] Xilinx, *7 series dsp48e1 slice (ug479), v1.10*, 2018. [Online]. Available: `https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf`.

[147] ——, *Zynq-7000 soc data sheet: Overview (ds190), v1.11.1*, 2018. [Online]. Available: `https://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf`.

[148] ——, *Zc706 evaluation board for the zynq-7000 xc7z045 soc (ug954), v1.7*, 2018. [Online]. Available: `https://www.xilinx.com/support/documentation/boards_and_kits/zc706/ug954-zc706-eval-board-xc7z045-ap-soc.pdf`.

[149] ——, *Amba axi4 interface protocol*, 2017. [Online]. Available: `https://www.xilinx.com/products/intellectual-property/axi.html#details`.

[150] ——, *Versal architecture and product data sheet: Overview (ds950), v1.14*, 2021. [Online]. Available: `https://www.xilinx.com/support/documentation/data_sheets/ds950-versal-overview.pdf`.

[151] ——, *Vmk180 evaluation board (ug1411), v1.0*, 2021. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documentation/boards_and_kits/vmk180/ug1411-vmk180-eval-bd.pdf.

[152] B. Reimer, "Driver assistance systems and the transition to automated vehicles: A path to increase older adult safety and mobility?" *Public Policy & Aging Report*, vol. 24, no. 1, pp. 27–31, 2014.

[153] Y. Li, H. Wang, W. Wang, L. Xing, S. Liu, and X. Wei, "Evaluation of the impacts of cooperative adaptive cruise control on reducing rear-end collision risks on freeways," *Accident Analysis & Prevention*, vol. 98, pp. 87–95, 2017.

[154] M. Makridis, K. Mattas, B. Ciuffo, *et al.*, "Empirical study on the properties of adaptive cruise control systems and their impact on traffic flow and string stability," *Transportation research record*, vol. 2674, no. 4, pp. 471–484, 2020.

[155] M. Hasenjäger and H. Wersing, "Personalization in advanced driver assistance systems and autonomous vehicles: A review," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2017, pp. 1–7.

[156] G. N. Bifulco, F. Simonelli, and R. Di Pace, "Experiments toward an human-like adaptive cruise control," in *2008 IEEE Intelligent Vehicles Symposium*, IEEE, 2008, pp. 919–924.

[157] G. N. Bifulco, L. Pariota, F. Simonelli, and R. Di Pace, "Development and testing of a fully adaptive cruise control system," *Transportation Research Part C: Emerging Technologies*, vol. 29, pp. 156–170, 2013.

[158] J. Wang, L. Zhang, D. Zhang, and K. Li, "An adaptive longitudinal driving assistance system based on driver characteristics," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 1–12, 2012.

[159] S. Lefèvre, A. Carvalho, Y. Gao, H. E. Tseng, and F. Borrelli, "Driver models for personalised driving assistance," *Vehicle System Dynamics*, vol. 53, no. 12, pp. 1705–1720, 2015.

[160] F. Muehlfeld, I. Doric, R. Ertlmeier, and T. Brandmeier, "Statistical behavior modeling for driver-adaptive precrash systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1764–1772, 2013.

[161] A. Rosenfeld, Z. Bareket, C. V. Goldman, D. J. LeBlanc, and O. Tsimhoni, "Learning drivers' behavior to improve adaptive cruise control," *Journal of Intelligent Transportation Systems*, vol. 19, no. 1, pp. 18–31, 2015.

[162] M Canale, S Malan, and V Murdocco, "Personalization of acc stop and go task based on human driver behaviour analysis," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 357–362, 2002.

[163] E. de Gelder, I. Cara, J. Uittenbogaard, L. Kroon, S. van Iersel, and J. Hogema, "Towards personalised automated driving: Prediction of preferred acc behaviour based on manual driving," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 1211–1216.

[164] M. V. Martínez, I. Del Campo, J. Echanobe, and K. Basterretxea, "Driving behavior signals and machine learning: A personalized driver assistance system," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, IEEE, 2015, pp. 2933–2940.

[165] I. del Campo, E. Asua, V. Martínez, Ó. Mata-Carballeira, and J. Echanobe, "Driving style recognition based on ride comfort using a hybrid machine learning algorithm," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2018, pp. 3251–3258.

[166] Y. L. Murphey, R. Milton, and L. Kiliaris, "Driver's style classification using jerk analysis," in *2009 IEEE Workshop on Computational Intelligence in Vehicles and Vehicular Systems*, IEEE, 2009, pp. 23–28.

[167] C. Miyajima, Y. Nishiwaki, K. Ozawa, *et al.*, "Driver modeling based on driving behavior and its evaluation in driver identification," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 427–437, 2007.

[168] J. Wang, K. Li, and X.-Y. Lu, "Chapter 6 - comparative analysis and modeling of driver behavior characteristics," in *Advances in Intelligent Vehicles*, Y. Chen and L. Li, Eds., Boston: Academic Press, 2014, pp. 159 –198.

[169] B. Higgs and M. Abbas, "Segmentation and clustering of car-following behavior: Recognition of driving patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 81–90, 2014.

[170] K. Custer, J. Sudweeks, M. A. Perez, *et al.*, "PSoC for Real-time Driver Assistance Based on Machine Learning IP Cores," VTTI, 2019. [Online]. Available: `https://doi.org/10.15787/VTT1/JFVNZN`.

[171] G. Castignani, T. Derrmann, R. Frank, and T. Engel, "Driver behavior profiling using smartphones: A low-cost platform for driver monitoring," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 91–102, 2015.

[172] J. Shen, X. Hao, Z. Liang, Y. Liu, W. Wang, and L. Shao, "Real-time superpixel segmentation by dbscan clustering algorithm," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5933–5942, 2016.

[173] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[174] Y. Zhao and G. Karypis, "Evaluation of hierarchical clustering algorithms for document datasets," in *Proceedings of the eleventh international conference on Information and knowledge management*, ACM, 2002, pp. 515–524.

[175] R. Kalsoom and Z. Halim, "Clustering the driving features based on data streams," in *INMIC*, IEEE, 2013, pp. 89–94.

[176] R. Bhoraskar, N. Vankadhara, B. Raman, and P. Kulkarni, "Wolverine: Traffic and road condition estimation using smartphone sensors," in *2012 Fourth International Conference on Communication Systems and Networks (COMSNETS 2012)*, IEEE, 2012, pp. 1–6.

[177] P. Brombacher, J. Masino, M. Frey, and F. Gauterin, "Driving event detection and driving style classification using artificial neural networks," in *2017 IEEE International Conference on Industrial Technology (ICIT)*, IEEE, 2017, pp. 997–1002.

[178] A. Kurt and Ü. Özgüner, "A probabilistic model of a set of driving decisions," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2011, pp. 570–575.

[179] A. Aljaafreh, N. Alshabatat, and M. S. N. Al-Din, "Driving style recognition using fuzzy logic," in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES 2012)*, IEEE, 2012, pp. 460–463.

[180] A. K. Choudhary and P. K. Ingole, "Smart phone based approach to monitor driving behavior and sharing of statistic," in *2014 Fourth International Conference on Communication Systems and Network Technologies*, IEEE, 2014, pp. 279–282.

[181] Ó. Mata-Carballeira, I. del Campo, V. Martínez, and J. Echanobe, "A hardware/software extreme learning machine solution for improved ride comfort in automobiles," in *2019 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2019.

[182] Xilinx, *Divider generator v5.1 (pg151)*, 2016. [Online]. Available: `https://www.xilinx.com/support/documentation/ip_documentation/div_gen/v5_1/pg151-div-gen.pdf`.

[183] H. J. B. Saldaña and C. Silva-Cárdenas, "A digital hardware architecture for a three-input one-output zero-order anfis," in *2012 IEEE 3rd Latin American Symposium on Circuits and Systems (LASCAS)*, 2012, pp. 1–4.

[184] J. Darvill, A. Tisan, and M. Cirstea, "A novel anfis algorithm architecture for fpga implementation," in *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*, 2017, pp. 1243–1248.

[185] A. de Souza and M. Fernandes, "Parallel fixed point implementation of a radial basis function network in an fpga," *Sensors*, vol. 14, no. 10, pp. 18 223–18 243, 2014.

[186] F. F. Lopes, J. C. Ferreira, and M. A. Fernandes, "Parallel implementation on fpga of support vector machines using stochastic gradient descent," *Electronics*, vol. 8, no. 6, p. 631, 2019.

[187] Gtisoft, *GT-SUITE Overview | Gamma Technologies*, 2017. [Online]. Available: `https://www.gtisoft.com/gt-suite/gt-suite-overview/` (visited on 04/21/2020).

[188] R. Salas, M. J. Pérez Villadóniga, J. Prieto Rodríguez, and A. Russo, "Restricting Traffic into the City Centre: Has Madrid Central Been Effective to Reduce NO_2 Levels?" *SSRN Electronic Journal*, 2019.

[189] S. Javanmardi, E. Bideaux, J. Trégouët, R. Trigui, H. Tattegrain, and E. N. Bourles, "Driving style modelling for eco-driving applications," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 13866 –13 871, 2017, 20th IFAC World Congress.

[190] M. Sivak and B. Schoettle, "Eco-driving: Strategic, tactical, and operational decisions of the driver that influence vehicle fuel economy," *Transport Policy*, vol. 22, pp. 96–99, 2012.

[191] K. Ayyildiz, F. Cavallaro, S. Nocera, and R. Willenbrock, "Reducing fuel consumption and carbon emissions through eco-drive training," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 46, pp. 96–110, 2017.

[192] E. Gilman, G. V. Georgiev, P. Tikka, S. Pirttikangas, and J. Riekki, "How to support fuel-efficient driving?" *IET Intelligent Transport Systems*, vol. 12, no. 7, pp. 631–641, 2018.

[193] D. Barić, G. Zovak, and M. Periša, "Effects of eco-drive education on the reduction of fuel consumption and CO2 emissions," *Promet - Traffic - Traffico*, vol. 25, no. 3, pp. 265–272, 2013.

[194] C. K. Allison and N. A. Stanton, "Eco-driving: the role of feedback in reducing emissions from everyday driving behaviours," *Theoretical Issues in Ergonomics Science*, vol. 20, no. 2, pp. 85–104, 2019.

[195] S. Long, S. Qiuchen, N. Jun, X. Nan, and Z. Kai, "Experimental Analysis of the Influence of Gear Shift Indicator on Vehicle Fuel Consumption," in *Advances in Intelligent Systems and Computing*, vol. 929, Springer Verlag, 2019, pp. 883–892.

[196] J. Tulusan, T. Staake, and E. Fleisch, "Providing eco-driving feedback to corporate car drivers: What impact does a smartphone application have on their fuel efficiency?" In *UbiComp'12 - Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, New York, New York, USA: ACM Press, 2012, pp. 212–215.

[197] M. Staubach, N. Schebitz, F. Köster, and D. Kuck, "Evaluation of an eco-driving support system," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 27, no. PA, pp. 11–21, 2014.

[198] R. Massoud, F. Bellotti, S Poslad, R Berta, and A De Gloria, "Exploring fuzzy logic and random forest for car drivers' fuel consumption estimation in iot-enabled serious games," in *IEEE International Symposium on Autonomous Decentralized Systems (ISADS)*, 2019.

[199] W. Vaz, R. G. Landers, and U. O. Koylu, "Neural network strategy for driving behaviour and driving cycle classification," *International Journal of Electric and Hybrid Vehicles*, vol. 6, no. 3, pp. 255–275, 2014.

[200] G. Delnevo, P. Di Lena, S. Mirri, C. Prandi, and P. Salomoni, "On combining big data and machine learning to support eco-driving behaviours," *Journal of Big Data*, vol. 6, no. 1, p. 64, 2019.

[201] P. Ping, W. Qin, Y. Xu, C. Miyajima, and K. Takeda, "Impact of driver behavior on fuel consumption: Classification, evaluation and prediction using machine learning," *IEEE Access*, vol. 7, pp. 78 515–78 532, 2019.

[202] S. Lakshminarayanan, "Application of Self-Organizing Maps on Time Series Data for identifying interpretable Driving Manoeuvres," *European Transport Research Review*, vol. 12, no. 1, 2020.

[203] Xilinx, *Vivado Design Suite User Guide: Programming and Debugging (UG908)*, 2015. [Online]. Available: `www.xilinx.comhttp://www.xilinx.com/support/documentation/sw{\_}manuals/xilinx2015{\_}4/ug903-vivado-using-constraints.pdf`.

[204] N Zacharof, G Fontaras, B Ciuffo, *et al.*, "Review of in use factors affecting the fuel consumption and co2 emissions of passenger cars," *European Commission*, 2016.

[205] S. Tanvir, H. Frey, and N. Rouphail, "Effect of light duty vehicle performance on a driving style metric," *Transportation Research Record: Journal of the Transportation Research Board*, p. 036 119 811 879 607, Sep. 2018.

[206] S. Tanvir, R. Chase, and N. Roupahil, "Development and analysis of eco-driving metrics for naturalistic instrumented vehicles," *Journal of Intelligent Transportation Systems*, pp. 1–14, May 2019.

[207] M. S. Asfoor, A. M Sharaf, and S. Beyerlein, "USE OF GT-SUITE TO STUDY PERFORMANCE DIFFERENCES BETWEEN INTERNAL COMBUSTION ENGINE (ICE) AND HYBRID ELECTRIC VEHICLE (HEV) POWERTRAINS," in *The International Conference on Applied Mechanics and Mechanical Engineering*, Military Technical College, vol. 16, Egypts Presidential Specialized Council for Education and Scientific Research, 2014, pp. 1–16.

[208] MathWorks, *Cluster data by training a self-organizing maps network*. [Online]. Available: `https://es.mathworks.com/help/deeplearning/ref/neuralnetclustering-app.html{\#}d120e123485` (visited on 04/29/2020).

[209] M. N. Azadani and A. Boukerche, "Performance Evaluation of Driving Behavior Identification Models through CAN-BUS Data," in *IEEE Wireless Communications and Networking Conference, WCNC*, vol. 2020-May, Institute of Electrical and Electronics Engineers Inc., 2020.

[210] N. Prasanna, S. Manikandan, P. Guruprasath, and S. Mohan Baabu, "Nox reduction in IC engine using pressure swing adsorption technique," *International Journal of Scientific and Technology Research*, vol. 8, no. 12, pp. 2011–2017, 2019.

[211] A. Tisan and M. Cirstea, "SOM neural network design - A new Simulink library based approach targeting FPGA implementation," *Mathematics and Computers in Simulation*, vol. 91, pp. 134–149, 2013.

[212] W Kurdthongmee, "A hardware centric algorithm for the best matching unit searching stage of the som-based quantizer and its fpga implementation," *Journal of Real-Time Image Processing*, vol. 12, no. 1, pp. 71–80, 2016.

[213] H. Hikawa and K. Kaida, "Novel FPGA implementation of hand sign recognition system with SOM-Hebb classifier," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 1, pp. 153–166, 2015.

[214] L. Svensson and J. Eriksson, *Tuning for Ride Quality in Autonomous Vehicle: Application to Linear Quadratic Path Planning Algorithm (Doctoral thesis)*. Dissertation, 2015.

[215] H. R.A. and J. F.J., *Estudio del índice de confort del servicio de transporte público de pasajeros a través de la medición de aceleraciones*. 2015.

[216] *Mechanical vibration and shock – evaluation of human exposure to whole-body vibration – part 1: General requirements. iso 2631-1 international organisation for standarisation*, 1997.

[217] J. F. Golding, A. Mueller, and M. A. Gresty, "A motion sickness maximum around the 0.2 hz frequency range of horizontal translational oscillation.," *Aviation, space, and environmental medicine*, vol. 72, no. 3, pp. 188–192, 2001.

[218] B. E. Donohew and M. J. Griffin, "Motion sickness: Effect of the frequency of lateral oscillation," *Aviation, Space, and Environmental Medicine*, vol. 75, no. 8, pp. 649–656, 2004.

[219] M. J. Griffin and J. Erdreich, *Handbook of human vibration*, 1991.

[220] R. G., U. C., and S. F., "Embedded systems to evaluate the passenger comfort in public transportation based on dynamical vehivle behavior with user's feedback," *Measurement*, vol. 47, pp. 442–451, 2009.

[221] A. S. Kilinc and T. Baybura, "Determination of minimum horizontal curve radius used in the design of transportation structures, depending on the limit value of comfort criterion lateral jerk," in *TS06G-Engineering Surveying, Machine Control and Guidance*, 2012.

[222] C. Sohn, J. Andert, and R. N. Nanfah Manfouo, "A driveability study on automated longitudinal vehicle control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 8, pp. 3273–3280, 2020.

[223] J. Van Mierlo, G. Maggetto, E. Van de Burgwal, and R. Gense, "Driving style and traffic measures-influence on vehicle emissions and fuel consumption," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 218, no. 1, pp. 43–50, 2004.

[224] S. Innamaa and M. Penttinen, "Impacts of a green-driving application in city buses on fuel consumption, speeding and passenger comfort," *IET Intelligent Transport Systems*, vol. 8, no. 5, pp. 435–444, 2014.

[225] Ó. Mata-Carballeira, M. Díaz-Rodríguez, I. del Campo, and V. Martínez, "An intelligent system-on-a-chip for a real-time assessment of fuel consumption to promote eco-driving," *Applied Sciences (Switzerland)*, vol. 10, no. 18, p. 6549, 2020.

[226] X. Chen, H. Wang, and B. Tian, "Visualization model of big data based on self-organizing feature map neural network and graphic theory for smart cities," *Cluster Computing*, vol. 22, Nov. 2019.

[227] Y. Li, Z. Yang, and K. Han, "Research on the clustering algorithm of ocean big data based on self-organizing neural network," *Computational Intelligence*, Mar. 2020.

[228] S. Weglarczyk, "Kernel density estimation and its application," *ITM Web of Conferences*, vol. 23, p. 00 037, Jan. 2018.

[229] Xilinx, *Vitis unified software platform documentation (ug1416), v2021.2*, 2021. [Online]. Available: `https://docs.xilinx.com/v/u/en-US/ug1416-vitis-documentation`.

[230] J.-P. Vasseur and A. Dunkels, "Communication mechanisms for smart objects," *Interconnecting Smart Objects with IP*, pp. 147–165, 2010.

[231] J. A. Baxter, D. A. Merced, D. J. Costinett, L. M. Tolbert, and B. Ozpineci, "Review of electrical architectures and power requirements for automated vehicles," in *2018 IEEE Transportation Electrification Conference and Expo (ITEC)*, IEEE, 2018, pp. 944–949.