End-to-End Multiview Gesture Recognition for Autonomous Car Parking System

by

Hassene Ben Amara

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2019

Author's Declaration

Abstract

The use of hand gestures can be the most intuitive human-machine interaction medium. The early approaches for hand gesture recognition used device-based methods. These methods use mechanical or optical sensors attached to a glove or markers, which hinders the natural human-machine communication. On the other hand, vision-based methods are not restrictive and allow for a more spontaneous communication without the need of an intermediary between human and machine. Therefore, vision gesture recognition has been a popular area of research for the past thirty years.

Hand gesture recognition finds its application in many areas, particularly the automotive industry where advanced automotive human-machine interface (HMI) designers are using gesture recognition to improve driver and vehicle safety. However, technology advances go beyond active/passive safety and into convenience and comfort. In this context, one of America's big three automakers has partnered with the Centre of Pattern Analysis and Machine Intelligence (CPAMI) at the University of Waterloo to investigate expanding their product segment through machine learning to provide an increased driver convenience and comfort with the particular application of hand gesture recognition for autonomous car parking.

In this thesis, we leverage the state-of-the-art deep learning and optimization techniques to develop a vision-based multiview dynamic hand gesture recognizer for self-parking system. We propose a 3DCNN gesture model architecture that we train on a publicly available hand gesture database. We apply transfer learning methods to fine-tune the pre-trained gesture model on a custom made data, which significantly improved the proposed system performance in real world environment. We adapt the architecture of the end-to-end solution to expand the state of the art video classifier from a single image as input (fed by monocular camera) to a multiview 360 feed, offered by a six cameras module. Finally, we optimize the proposed solution to work on a limited resources embedded platform (Nvidia Jetson TX2) that is used by automakers for vehicle based features, without sacrificing the accuracy robustness and real time functionality of the system.

Acknowledgements

I would like to thank all the people who made this thesis possible. I thank my supervisor, Prof. Karray, for many insightful conversations during the development of the ideas in this thesis, and for helpful comments on the text. I would like to thank as well Shayonta Chowdhury and Divakar Kapil, two undergraduate students from the University of Waterloo for their support throughout this work. Wouldn't have been as easy without you! Finally, I would like to thank my wife for all she is, and all she has done to support my dreams and aspirations.

Dedication

To my beloved wife and mother.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, the most natural and intuitive means of communication for human-machine interaction is voice thanks to the recent advances in the areas of automatic speech recognition and natural language processing. However, humans are a visual species by nature, which means that the most human-to-human communication we use is gesture which can be manifested in our regular voluntary gestures and in our subconscious body language which conveys most of the meaning of what we say while communicating with other humans.

1.1 Background

Gesture is a communication channel in the field of Human Machine Interface (HMI) that has many recent advancements: the hand gesture can be used to reinforce speech in a noisy environment or to communicate with a computer. Furthermore, it can also be used in virtual or augmented reality applications (*e.g.* manipulate objects, replace mouse).

The use of hand gestures can be the most natural and intuitive human-machine interaction medium. The early approaches for hand gesture recognition used device-based methods. These methods rely on optical or mechanical sensors attached to a glove or markers that transform hand motions to electrical signals to determine posture. Using these methods, one can acquire different information namely angles, joints of the hand ..., etc. However, these methods require that the person is wearing a device that can be bulky and heavy, which hinders the natural interaction.

Vision-based methods, contrarily, are less restrictive and allow for a more natural communication without the need of an intermediary between human and machine. Therefore, vision gesture recognition has been a popular area of research for the past thirty years and, thanks to the success achieved in the field of artificial intelligence (cognitive computing, deep learning and multi-modal gesture recognition), several gesture recognition applications have been developed: sign language recognition[26, 80], Virtual Reality (a totally immersive experience that transports the user into a three-dimensional universe created by a computer program, and this, drawing as much as possible from reality) where the user uses his hands to interact with virtual objects and trigger actions[91], augmented reality (the physical world is augmented with virtual information, for example by back-projection)[68], multi-modal human-machine interaction (combining speech and gesture recognition)[85], biometry (personal recognition using hand shape and texture)[44], just to name a few.

Similar to voice recognition technology, gesture recognition is seeing its early days of real life application in the automotive industry. Choosing a car as an early adoption host for a new human-machine interaction technology is ideal. As the vehicle presents a controlled environment with a subset of possible interactions to test with. This makes the automotive industry a perfect test bed for testing gesture recognition in real life.

1.2 Context

As stated in a new report published in 2019 by Global Market Insights, Inc, the automotive gesture recognition market is estimated to reach the size of USD 13.6 billion by 2024 [37]. As modern cars continue to offer more and more functionalities that require a growing number of commands, the options to control those features present important flaws when it comes to driver vehicle interaction as they usually require visual attention of the user. The application of gesture recognition to advanced driver assistance systems allows the driver to use hand gestures to control various features of the car (*e.g.* infotainment system), thus reduces distracted driving risks and improves driving safety.

However, advances in technology are not limited to driving safety. Many recent innovations focus on offering more passenger convenience and comfort. Furthermore, automated driving is nowadays a hot topic in the car business with automakers competing to be the first to bring a self-driving car to the market. In this context, one of America's big three automakers has partnered with the Centre of Pattern Analysis and Machine Intelligence (CPAMI) at the University of Waterloo to investigate expanding their product segment through deep learning to provide an increased driver convenience and comfort with the particular application of hand gesture recognition for driver-less auto parking. This is an open-ended request that enables this thesis to explore a broad range of deep learning techniques and allowed the goals to best suit the applicability of deep learning techniques in embedded environments for gesture recognition. We denote this automotive partner as "the automaker" for the remainder of the thesis.

1.3 Problem Statement

Data released by Mercedes and BMW in 2015 show that while vehicle sizes have increased by up to 25 percent over the last 40 years, in many cases, the number of garages and parking spaces have remained constant [89]. In modern society, there is an ever-increasing number of vehicles, and this led to increasing difficulty to find large-enough spaces in busy parking lots. Parking can be a stressful endeavor, with the challenge of maneuvering into and out of a parking spot. Hence, drivers are in need of novel ways to park their cars without being behind the wheel. Being on the outside of the vehicle, the driver has a much better view of the hazards surrounding his car, hence he can park in tighter spots and it helps him avoid situations, such as illustrated in Fig.1.1, where his vehicle is boxed in which might cause a paint damage while trying to exit the car, or even putting the driver in embarrassing situation (*e.g.* exits through the trunk).

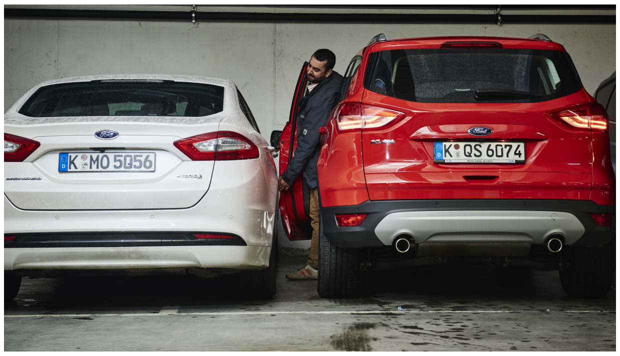


Figure 1.1: Driver squeezing himself out of his car parked in a small parking space

Several automakers have tackled this problem in late 2016 and have developed some solutions that only few of them are currently in production. The proposed systems make use of a mobile application on a smart phone to automatically park a car without a driver[88], a button on the vehicle display key to activates the remote-controlled parking feature or a smart watch [36] that recognizes a configurable wave gesture and transmit it to the vehicle over wireless connection to trigger the parking action.

All of the above mentioned solutions are device dependant (remote key, smart phone, smart watch). This represents a major drawback of these systems because of the disadvantages that come along with the usage of an additional hardware. In fact, any disfunctioning of the device (*e.g.* damage caused by water, low battery) or unfavorable conditions (*e.g.* rain or snowy weather) will render the self-parking feature unusable. Additionally, even though the above mentioned systems use a very common human-machine interaction medium (*e.g.* touch screen), it still presents an inconvenience to users as it requires an intermediary medium between them and the car.

To overcome the above mentioned weaknesses of the existent solutions, the automaker would like to, through this thesis work, investigate the potential of deep leaning techniques in developing a multiview vision-based system for real life vehicle self-parking.

1.4 Thesis Contributions

The main goal of this thesis is to develop an effective approach of a deep learning based framework for multiview dynamic hand gesture classification. In the context of this work, we partnered with the automaker to investigate the application of the latest advances in deep learning methods along with the state of the art techniques for gesture classification and develop a multiview hand gesture recognition prototype with the subject of vehicle self-parking in mind. Throughout the execution of this thesis work, many contributions were made. The following lists the most important ones:

- Expand the application of gesture recognition in the automotive sector from static to dynamic hand gesture recognition. We refer to static gesture recognition by classifying the posture or the form specific to the hand. While dynamic gesture recognition classifies a trajectory of the hand or a succession of postures of the hand subject to identification.
- Leverage a publicly available hand gesture videos dataset (20BN-JESTER) [84] to provision sufficient training and testing data that is required for a robust dynamic hand gesture classifier for real life applications.
- Use data augmentation techniques and transfer learning [1] to augment the publicly available dataset that represents a lab environment application with custom made video recordings to ensure robustness in real world environments.

[1]https://en.wikipedia.org/wiki/transfer_learning

- Identify, based on the use case at hand, the most appropriate deep learning architecture for the dynamic hand gesture recognition application. As concluded by the literature, the most suitable algorithms for video classification are 3D Convolutional Neural Networks (as it supports the third dimension *i.e.* time) and Long-term Recurrent Neural Networks as it is the most applied in sequential image data classification.

- Adapt the architecture of the end-to-end solution to expand the state of the art video classifier from a single image as input (fed by monocular camera) to a multiview 360 feed, offered by a six cameras system.

- Optimize the proposed end-to-end solution to work on a limited resources embedded platform that is used by automakers for vehicle based features, without sacrificing the accuracy and robustness of the system.

1.5 Thesis Outline

The remainder of this report is divided into five chapters. In the second chapter, we present a detailed description of the major approaches used for developing hand gesture recognition systems. Moreover, we will review the literature to identify the different applications of gesture recognition in general and then specifically in the automotive sector. We will end the second chapter with a review of existent vehicle self-parking solutions and their respective limitations that this thesis is addressing. Along the way, we outline the theoretical background on Deep Neural Networks (DNN) image and video classification tools that this work makes use of in chapter 3. Following the presentation of the background material, the proposed approach for dynamic hand gesture recognition and its evaluation against a state of the art method is then presented in chapter 4. Finally, this report will wrap up with a real-time system implementation of our proposed Multiview hand gesture recognition framework, and a chapter summarizing our conclusions that are connected to our initial analysis of the problem and our objectives of the thesis project.

Chapter 2

Literature Review and Applications of Gesture Recognition

2.1 Introduction

Gesture recognition is also called contact-less technology. Gestures describe body movements as a silent communication of language, conveying people's thoughts, emotions, and pedagogical information. The recognition of gestures can come from the movement of different parts of the body, but usually refers to the movement of the face and hands. Hand gesture recognition can be seen as a sub-problem of the very important video classification domain, in particular because of its related applications. In fact, many applications in audio and video processing, in multimedia indexing, in form recognition, in language processing, as well as in other areas such as biology or finance involve data in the form of sequences (*i.e.* an ordered sequence of information that can be numbers, symbols, or vectors). In this chapter, we present an elaborated review of various approaches found in the literature to resolve the problem of hand gesture recognition and their limitations and shortcomings. Also, we highlight the different applications of hand gesture recognition. Finally we conclude with a review of the existent vehicle self-parking solutions which led us to propose a better solution for the said problem.

2.2 Taxonomy

Before undertaking the overview of gesture recognition approaches, it is necessary to first, clarify what a gesture is. In reality, the meaning attributed to the word gesture (or action) varies according to the context of usage. Several definitions have been proposed whose main distinction lies in their level of semantic abstraction. In this manuscript, we particularly remember the distinction proposed in [58, 9] which defines the action according to two approaches, functional and phenomenological. On one hand, the functional approach refers to the functions that an action can execute in specific situations, which corresponds to the semantics of the action. On the other hand, the phenomenological approach is based on kinematic criteria, spatial and frequency that describe the movement associated with this action. The goal of a gesture recognition approach is to match the kinematics from an action to its semantics. Indeed, a recognition approach fills the so-called semantic gap that represents the difference in conceptual level between the machine, which operates on kinematic data (raw data), and the user who understands the meaning of an action. Such an approach therefore proposes a high level interpretation (the class of the action, the intention of the action) corresponding to the low level data from capture systems. Such recognition systems have a crucial role in several areas of application focused on humans, including video analysis [27], monitoring [38], robotics [18], human-machine interaction [78], augmented reality [25], assistance to the elderly [60], smart homes [8] and education [56].

2.3 Various Approaches in the Literature for Hand Gesture Recognition

2.3.1 Sensor-based Recognition

The early gesture recognition systems made use of electronic gloves equipped with sensors that provide the hand position and the angles of finger joints. A 3-D hand motion tracking as well as gesture recognition system was developed by Ji-Hwan Kim *et al.* which makes use of a data glove (KHU-1). The latter consists of one controller, one Bluetooth 3 tri-axis accelerometer sensors. [40]. The main limitation of the aforementioned approach is that these gloves are expensive and bulky, as well as requires a full hardware toolkit and less flexible in nature. Hence the growing interest in computer vision methods. Indeed, with technical progress and the appearance of cheap cameras, it is now possible to develop gesture recognition systems based on computer vision, operating in real time.

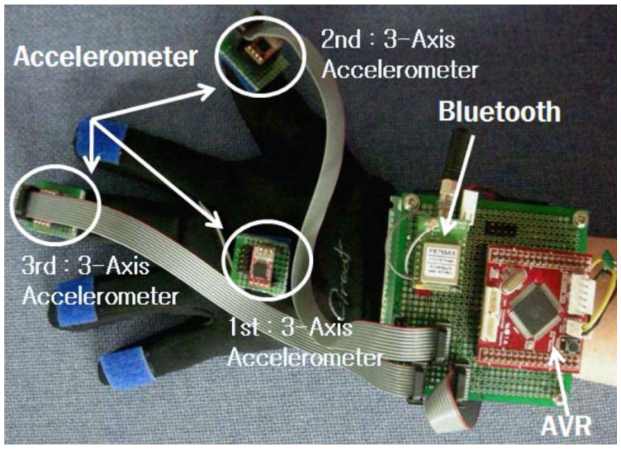

Accelerometer
2nd : 3-Axis Accelerometer
Bluetooth
3rd : 3-Axis Accelerometer
1st : 3-Axis Accelerometer
AVR

Figure 2.1: The KHU-1 data glove

2.3.2 Vision Based Approach: Probabilistic Models

Vision-based hand gesture recognition can be seen as a multi-class classification problem that can be modeled in a probabilistic way *i.e.* by defining a number of random variables, and looking for the best performance of these variables, according to a certain criterion. This is usually done via the maximization (or minimization, based on the use case) of a global objective function, which associates a probability or an energy with each output of all the variables. In practice, the estimation of these probabilities (so-called joint probabilities) remains very complex or not feasible in the absence of any constraints. This is due in particular to the fact that, without constraints, the random variables concerned can all be interdependent, both directly and conditionally. A solution to this problem then consists in imposing constraints on these conditional dependencies, in allowing certain variables to be conditionally independent (and not directly) from other variables, which makes it possible to factorize the joint probability (mentioned previously) into a product of factors, each involving a weak number of variables. Probabilistic graphical models are a family of models that present a practical solution to manage these conditional dependencies. They allow in effect to model them simply as a graph, in which the vertices represent the variables. Several probabilistic graphical models have been proposed, some of which are adapted to the processing of sequential data. The example best known in this category are the Hidden Markov Models(HMM), Conditional Random Fields (CRF) and Support Vector Machine (SVM). In this section, we present these methods with a focus on their application to the gesture recognition problem.

2.3.2.1 Hidden Markov Models (HMM)

Hidden Markov Models are among the most commonly used statistical techniques for dynamic hand gesture recognition, in particular because of their success in speech recognition problems and handwriting. The theoretical background of HMMs date back to the mid-1960s with the work of Baum et al.[4]. But these models became only more popular in the 1980s with the work of Bahl et al.[3], Poritz et al.[63] and especially those of Rabiner et al.[65] on speech recognition. Without going into details, HMMs can be seen as a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (i.e. hidden) states. An HMM models a state machine in which each state has a certain probability of transition to each of the other states. Each transition generates an observation, the observation follows a probability law associated with the current state. Let q_t the state of the system and y_t the output at the moment t. The probability $P(y_t|q_t)$ of each state depends only on the current state q_t. The transition between two states $P(q_t|q_{t-1})$ depends only on the previous state. This relationship is illustrated in Fig. 2.2. Training Hidden Markov Models can be achieved using the Expectation-Maximization (EM) algorithm [19].

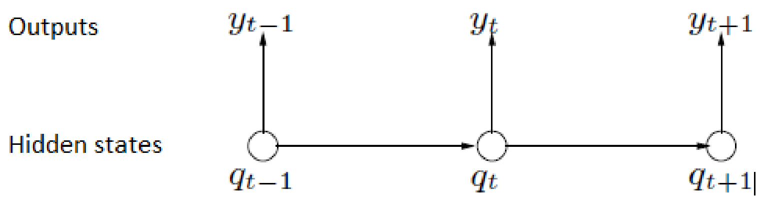


Figure 2.2: Example of three states HMM from left to right with dependencies between the output y and the hidden state q of the model

HMMs were successfully used in a considerable number of publications dealing with the recognition of human movements.

In 1998, Starner and Pentland [80] used a HMM to recognize real-time phrases expressed in ASL (American Sign Language). The basic data comes from 2D videos. Ellipses encompassing the shape of the hands are extracted at each "frame". A vocabulary comprising 6 pronouns, 9 verbs, 20 verbs and 5 adjectives has been defined, each word corresponds to a state. The Viterbi algorithm finds the most probable word sequence from of a sequence of observations.

Sylvain Calinon and Aude Billard [10] have exploited HMM in 2004 to learn several types of movements then make them recognized and imitated by a robot. The implemented system, shown in Fig.2.3, has been tested and validated in simulation and on a humanoid robot.

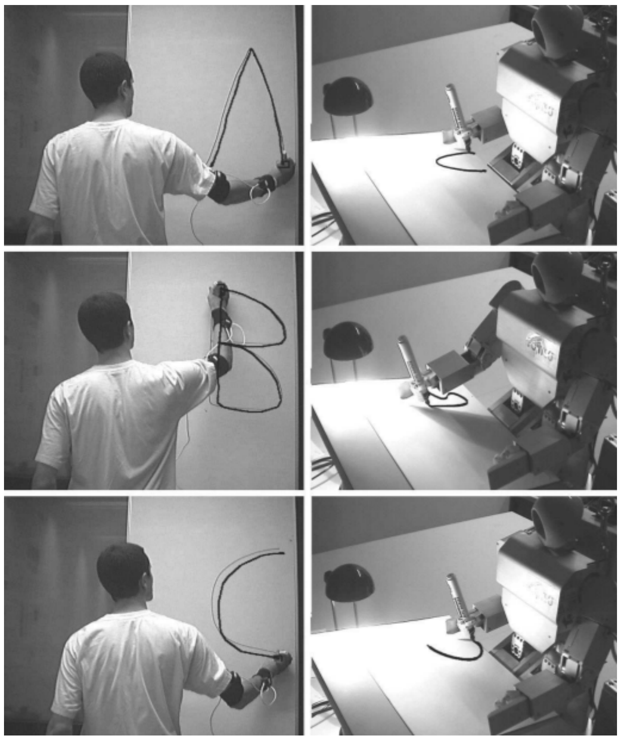


Figure 2.3: Demonstration (left column) and reproduction (right column) by a robot of drawing the three stylized alphabet letters A, B and C [10]

Other studies that used HMMs for gesture recognition were presented by Mitra and Acharya in their survey published in 2007 [54]. Further research studies introduced using Conditional Random Fields (CRF) which allowed to avoid the independence assumption between observations, and non-local dependencies between state and observations[87].

2.3.2.2 Conditional Random Fields (CRF)

Conditional Random Fields (CRF) presented in 2001 by Lafferty et al. [45] opened new doors for sequence analysis. Until then, Hidden Markov Model (HMM) was the most used for this type of analysis and different approaches in different fields have been proposed: detection, segmentation, classification ...etc. As for HMMs, CRFs have also been used to segment images. In its original definition, a CRF is a stochastic model that models the dependencies between a set of discrete observations made on a sequence (originally, a sequence of words) and a set of labels (Morphosyntactic Analysis). Conditional Random Fields or variants of CRFs were used for video classification problem in many works, these include the work of Lafferty et al. [45], Mendoza and Perez de la Blanca [52], Sminchisescu et al. [77], Wang and Suter [86]. The details of these works are available in the elaborated survey published by Poppe [62].

2.3.2.3 Support Vector Machine (SVM)

SVM belongs to the class of linear classifiers (which use a linear separation of data), and has its own method of finding the boundary (also called hyperplane) between different classes. The goal of an SVM is to find the linear hyperplane that separates the data and maximize the distance between the two classes. In the following diagram, we determine a hyperplane that separates the two sets of points:

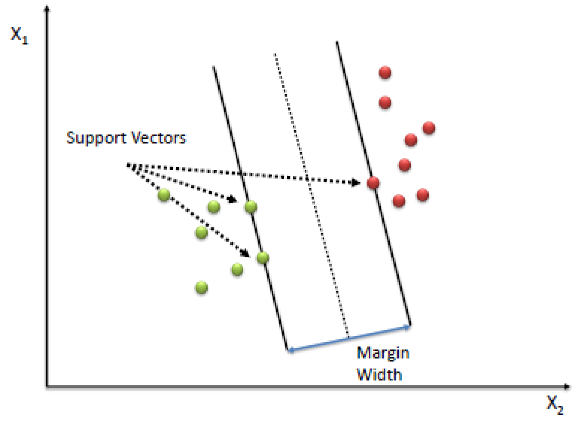
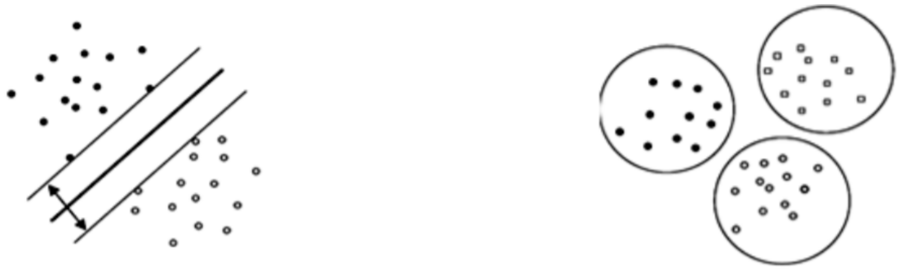


Figure 2.4: SVM classification: the vectors with the continuous line are the support vectors

Unlike binary classification, hand gesture recognition is a multi-class classification problem. Hence, the combination of multiple binary SVMs in a Classifier Tournament structure as illustrated in Fig.2.5 is required.



(a) Typical SVM model **(b)** Minimum Enclosing Ball MEB-SVM

Figure 2.5: Illustration of the typical SVM versus MEB-SVM model used for hand gestures classification, image source: [61]

The MEB-SVM (Minimum Enclosing Ball-SVM) outperformed other algorithms and proved efficient when implemented on the large scale data classification by [70]. Support Vector Machines (SVMs) have successfully been used in other works on recognizing human actions whether "by constructing video representations in terms of local space-time features and integrate such representations with SVM classification schemes for recognition" [46]. Or using a Support Vector Machine classifier on an existing spatio-temporal feature descriptor [12]. Also, Support Vector Machines were used in conjunction with 3D CNN which "was used to extract spatial and temporal features from adjacent video frames. Then, use the SVM to classify each instance based on previously extracted features" [47].

2.4 Applications of Hand Gesture Recognition

With the rapid development of computer technology, the interactive application between mechanical equipment and humans has grown exponentially. From technology initially limited to speech recognition and control, it has evolved into the recognition of gestures. More and more electronic products have increased gesture patterns, such as mobile phones and drones. In this section we present an overview of some of the main applications of gesture recognition in the literature.

2.4.1 Human-Computer Interaction (HCI)

Gestures can replace a traditional keyboard or a click on a mouse to control your own computer. This can make the interaction between people and machines smarter and more natural, and directly apply the experience people have in their daily lives. Various research studies have been published in the area of human-computer interaction.

Aashni Haria [29] in 2017 proposed a robust hand gesture recognition system that translates human gestures into action such as launching applications like PowerPoint and shuffle through the slides or opening a browser and navigate to a website.

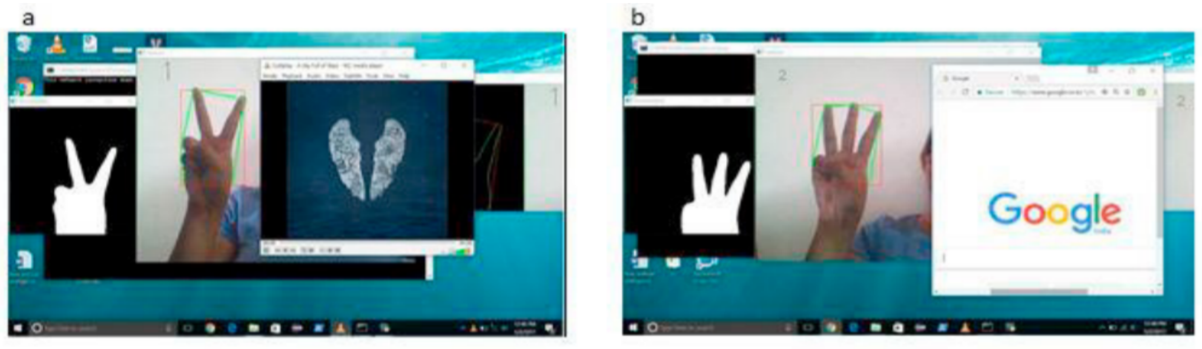


Figure 2.6: Hand gesture based system for computer control. (a) Gesture 'V' to launch VLC player (b) Gesture '3' to launch the browser[29]

2.4.2 Sign Language Recognition

Sign language is the main means of expression for deaf people, but for those who have not received sign language training, it is not easy to understand this language. The application of gesture recognition technology to sign language knowledge will greatly enhance communication between deaf people and ordinary people. Recently, there has been an increased interest in research work related to recognizing the alphabet and interpreting the Sign language words and phrases automatically. Recently, numerous research studies have been done around the interpretation of sign language [1, 74, 66].

Sign language commercialized solutions New applications facing real-world scenarios, although ambitious, are still experimental or in the beginning of development. They either face technical issues (recognition effectiveness) or logistical challenges (sensors congestion). There is currently a solution in the US market called MotionSavvy[1] which makes the use of a tablet with a leap motion controller. It performs both text-to-speech recognition and signs-to-words translation. Another commercial solution based on the Kinect

[1] www.motionsavvy.com, accessed: 01/20/2019

sensor: SignAll[2]. The development of more advanced solutions and devices dedicated to the assistance of deaf people or hearing-impaired is getting more interest currently and conferences such as m-enabling[3] are now dedicated to this topic.

(a) MotionSavvy (b) SignAll

Figure 2.7: Commercial solutions for vision based gesture recognition of sign languages

2.4.3 Augmented Reality

The concept of Augmented Reality is to coalesce the virtual world into the real world. Augmented reality leaves the user in the real world where the actions can be virtual. Currently, the areas of use of augmented reality are numerous. There has been an increase recently in research towards using hand gestures for interaction in the field of Augmented Reality (AR), "this include gesture-based interaction via finger tracking for mobile augmented reality"[35], or using 2D drawing annotations for augmented reality remote collaboration where users can draw 2D annotations that are then exchanged over the network in AR [21].

2.4.4 Virtual Reality

Virtual Reality (VR) is a system that gives the human the ability to interact with a virtual environment, a synthesized model generated by the computer that seems realistic. Virtual reality involves interactions across multiple sensory channels such as vision, touch and smell. Among the Virtual Reality (VR) devices that work on a PC, we can name HTC Vive and Oculus Rift, both named devices rely on a hand-held remote that tracks accurately positional hand data using button presses rather than gestures to interact with the virtual environment. Thus, the interest in using the natural hand gestures (*e.g.* camera-based method) as primary input in future VR systems is evolving. Vision based methods are

[2] www.signall.us, accessed: 01/20/2019
[3] www.m-enabling.com, accessed: 01/20/2019

therefore able to track the posture of individual fingers without encumbering users' hands with hand-held remote devices[16]. The hand gesture-manipulated virtual reality environment system presented in [16] uses hand gesture for navigation control and manipulation of 3D objects in a virtual world such is illustrated in Fig.2.8.

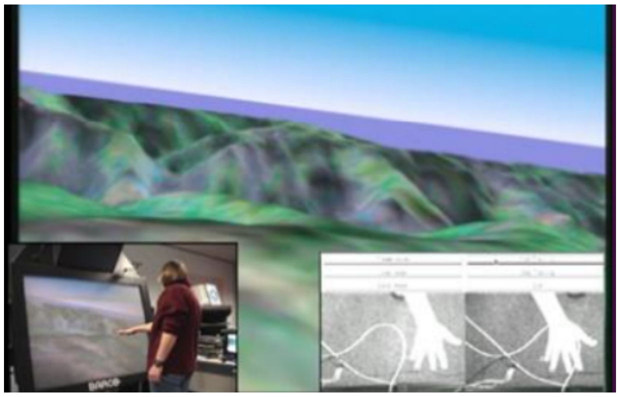


Figure 2.8: Hand gesture interface for a Virtual Environment

In this work, the user is able to move easily. His movements are controlled by the location of his hand in space. The segmentation of the hand is based on the skin color.

2.4.5 Applications of Hand Gesture Recognition in Automotive Sector

As stated in a new report published in 2019 by Global Market Insights, Inc, the automotive gesture recognition market is estimated to reach the size of USD 13.6 billion by 2024 [37]. The application of gesture recognition to advanced driver assistance systems can improve driving safety to some degree. The driver can use gestures to control various functions of the car or to modify various parameters of the car, hence pay more attention to reducing road accidents. Many research works were successfully published in this area, among others "a vision-based system that employs a combined RGB and depth descriptor to classify hand gestures for automotive interfaces" developed by Ohn-Bar [59]. A recent survey on hand gesture recognition in automotive human-machine interaction using depth cameras published late 2018 presents a thorough review of machine learning approaches to hand gesture recognition applied to human-machine interaction for automotive interfaces. At CES 2014 in Las Vegas, SoftKinetic, the Belgian company (acquired by Sony[4] in 2015)

[4] https://www.sony-depthsensing.com/Default.aspx

and Freescale (acquired by NXP Semiconductors[5]) unveiled their depth sensing and car gesture recognition solution. A simple camera posed and connected to the dashboard can directly recognize a wide range of gestures of the user. This solution, developed to work with most 3D cameras on the market, recognizes both finger and hand movements as well as wider body movements to control automotive infotainment systems in more intuitive and less distracting way. The solution developed could thus be used in many applications, such as changing the radio station with a snap of a finger, opening a door without a key, check changing traffic conditions (*e.g.* check for congestion, accident report, etc) or setting a destination on his GPS.

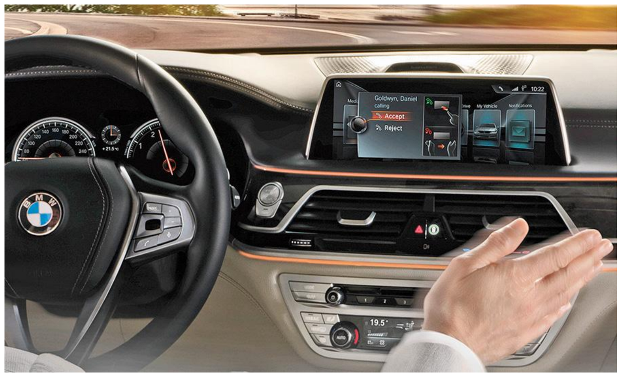


Figure 2.9: Car infotainment gesture control: SoftKinetic technology built in BMW car. The technology allows the driver to control the infotainment system with hand gestures.

2.5 Driver-less Vehicle Parking Systems

The automotive industry is now looking for the future of the driver-less parking systems. As part of our literature review, we will cover in the following subsections the device-based solutions for vehicles self-parking already offered by some automakers. We will highlight afterwards the drawbacks and disadvantages of these systems that drove our automaker partner's motivation to explore better approaches.

2.5.1 Remote-controlled Solutions

The Remote Control Parking assistance system is a system which lets the cars do their parking.The car parks itself by moving forward or in reverse directions in tight parking

[5] https://www.nxp.com/

space or in garage. The self-parking system can be activated using a button on the vehicle's remote key from outside the car and is monitored by the parking assistant and surround view sensors, also known as Park Distance Control. Fig. 2.10 shows the remote control parking system function of the 7-series sedan of BMW.

Figure 2.10: BMW 7-series remote-controlled driver-less self-parking system

2.5.2 Smartphone Controlled Solutions

Automakers of electric car like Tesla and Nissan have developed a smartphone controlled self-parking solution for their latest models. For example, in late 2016, the Japanese car manufacturer, Nissan, developed a prototype of a mobile app that allows users of the LEAF electric model to park their vehicles automatically using the app without being behind the wheel. Getting into the parking spot is fully automatic using installed sensors and cameras.[88].

2.5.3 Smartwatch-controlled Solutions

At the 2016 Consumer Electronic Show (CES), BMW presented its smartwatch-controlled gesture control parking feature deployed into the BMW i3 car model. The function is triggered by a configurable wave gesture that is recognized by the smart watch and transmitted to the vehicle over wireless connection. The transmitted hand gesture gives the vehicle the signal to initiate the parking operation. [36]. A full video demonstration of the system can be seen here: `https://goo.gl/ArWQxp`.

Figure 2.11: BMW i3 self-parking system: the vehicle recognizes gestures transmitted from an smart watch, and drives into and out of a parking space fully automatically. Image source: `https://goo.gl/CyeKJd`

2.5.4 Drawbacks of Device-based Solutions

Although the device-based solutions for vehicle driver-less parking systems we covered in the previous subsections seem promising and are already available in the market for end users, these systems, whether the remote-based or smartwatch-controlled, represents the following disadvantages:

- Device dependant: the presented solutions require an additional hardware to be fully functional. For the case of remote-controlled system like the BMW 7-series one, a smart key fob is required. On the other hand, a smart watch is also needed to trigger the self-parking system in the BMW i3. Therefore, any damage to the device (*e.g.* damage caused by water, low battery) or unfavorable conditions (*e.g.* rain or snowy weather) will render the feature unusable. In this context, several systems were developed to offer drivers with device-less solutions for some of the other car features (*e.g.* facial recognition for car unlock using Apple FaceID[6]).
- User experience/convenience: Even though the above mentioned systems use a very common human-machine interaction medium (touch screen), it still presents an inconvenience to the user as it requires an intermediary medium between the user and the car.

[6]https://futurism.com/apple-facial-recognition-car

2.6 Summary

In this chapter we elaborated a literature review of the research studies and approaches published in the area of dynamic hand gesture recognition. We covered the early sensor-based systems that uses data gloves, we highlighted then the vision based approaches that rely on both probabilistic models or deep learning techniques (will be presented in chapter 3). Furthermore, we looked at the applications of hand gesture recognition, including, but not limited to, human-computer interaction, sign language recognition, virtual reality and applications in the automotive sector, namely car infotainment gesture control. Finally, we studied the available solutions for driver-less vehicle parking, mainly developed by the German auto maker BMW, and discussed the disadvantages of these systems.

Chapter 3

Deep Learning in Computer Vision

3.1 Introduction

This thesis will only consider the use of vision based methods to solve the problem of dynamic hand gesture recognition. Therefore, the theoretical background presented in this chapter will focus only on the computer vision techniques. In general, deep learning techniques and in particular Convolutional Neural Networks (CNNs) have been the state of the art for image classification problems during the recent years . In this thesis we used a variant of CNNs called 3D CNN to solve the problem of video classification, particularly dynamic hand gesture recognition. Also, we evaluated alternative approach using Long-term Recurrent Convolutional Network (LRCN). The following sections in this chapter will go through the theoretical concepts behind the deep learning methods used in this thesis to solve the said problem.

3.2 Deep Learning Concepts

3.2.1 Artificial Neural Networks (ANNs)

Originally inspired by observations in neuroscience on the functioning of the brain, this type of model has existed for almost sixty years, with the invention of Perceptron [71]. Its initial formulation included important limitations [53], such as the impossibility of solving nonlinear problems. It was therefore put aside until the invention of multi-layered artificial neural networks and back-propagation [73]. The latter are still sensitive to the problem of local minima, they were again relegated to the background with the discovery of Support Vector Machines (SVMs) by [17], which can be used to solve nonlinear problems, using convex optimization. With deep learning [31, 64, 6] the Artificial Neural Networks were once again experiencing a boom. We will present the model in detail, starting from observations in neuroscience to highlight the important simplifications of neural networks used in machine learning with regard to their biological counterparts.

3.2.1.1 The Biological Neuron

The main computing unit of the nervous system, the neuron, is made of three parts: the dentritic tree, the soma and the axon; an illustration is presented in Fig. 3.1. At rest, the interior of the neuron is polarized relative to outside. This is due to a set of pumps controlling the concentrations of different ions, between the inside and the outside of the neuron. Synapses are the junction points between the axon and the dendrites of two neurons, a message is transmitted in the form of chemical compounds: neurotransmitters.

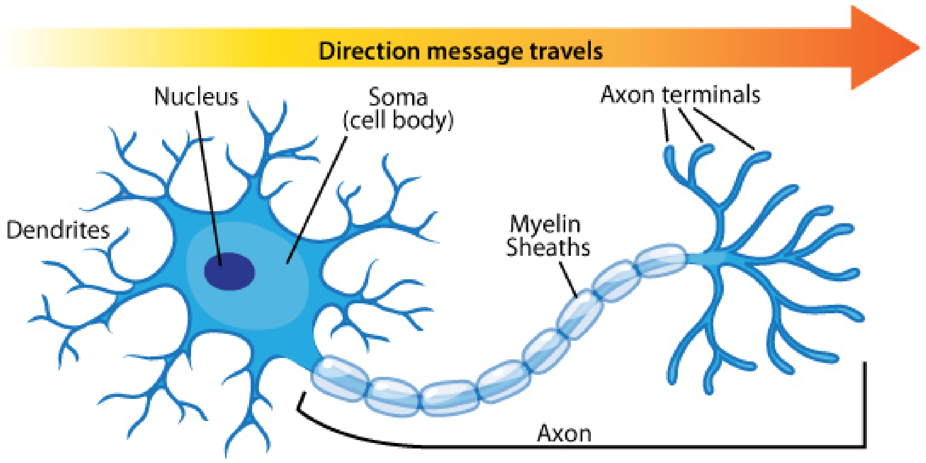


Figure 3.1: Schema of a biological neuron, source: `https://goo.gl/bRgF9E`

When these bind to postsynaptic receptors, ion channels open or close on the membrane and postsynaptic currents appear. These currents are then integrated in a complex and nonlinear way [42] in the dendritic tree. The potential of the soma will change and, if it exceeds a certain threshold, a potential action will be initiated and propagated through the axon, this will release neurotransmitters to the synapses to which it is connected. The mechanisms for creating and propagating the action potential are explained by the Hodgkin-Huxley model [33] based on a set of nonlinear differential equations.

3.2.1.2 Artificial Neural Network

The functioning of biological neurons is very complex, a faithful modeling requires monopolizing important computational resources. However, it is possible to define an efficient model inspired by biological neurons by admitting the following (false) hypotheses:

- The integration of postsynaptic currents is an affine transformation.
- The information of phase of the potentials of actions does not influence on the integration postsynaptic currents.
- The neuron response function can be assimilated to a function of our choice $g(x)$.
- All neurons can be formalized in the same way.

The output value of a neuron according to its inputs ($X = [x_1, ..., x_n]$) therefore becomes; with $w = [w_1, ..., w_n]$ the weights vector associated with each input and b the bias, or threshold of the artificial neuron:

$$z(X) = g(\sum_{i=1}^{n}(w_i \times x_i) + b) \tag{3.1}$$

Activation functions generally used are: Sigmoid and Hyperbolic Tangent [50] defined by the following equations:

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{3.2}$$

$$tanh(x) = \frac{e^{2x}-1}{e^{2x}+1} = 2 \times sigmoid(2x) - 1 \tag{3.3}$$

A Feed-forward Neural Network, illustrated in Fig. 3.2, is one of the most common neural network architectures. The input data, for instance an image, is fed to the network through multiple fully connected hidden layers. The algorithm forward-propagate the data until the output layer.

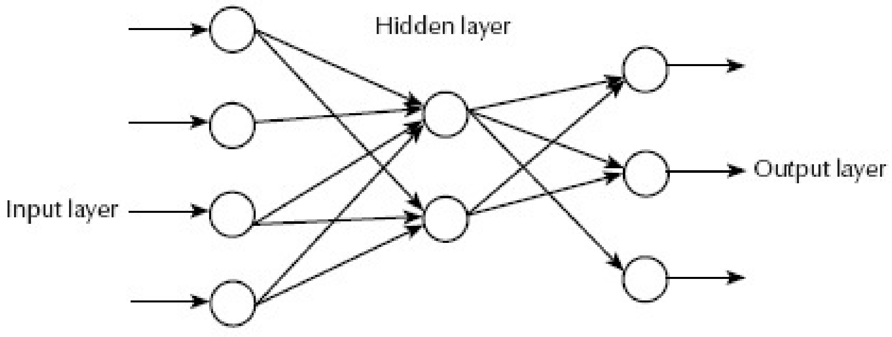


Figure 3.2: Typical multi-layer perceptron network

3.2.2 Learning Modes

Machine Learning algorithms can be grouped according to their learning mode. This categorization depends on the nature of the data included in the training set D. The choice of a learning mode influences the type of tasks that can be performed and the way the algorithm processes the data. There are several learning modes. For the sake of brevity, only those useful for understanding this document will be discussed: supervised learning (section 3.2.2.1) and unsupervised learning (section 3.2.2.2).

3.2.2.1 Supervised Learning

Supervised learning is considered the most intuitive form of learning. This means that the expected answer is observed in the training data. In a formal way, the training dataset D is composed of examples $z = (x, y)$ with x represents the "input" part, the data that the algorithm is allowed to use to make a prediction, and y the "label" part (*i.e.* the correct value for the prediction). For example, if the task is to determine the sex of a person from of an identity photo, x would be the photo and y be "man" or "woman" (assuming these are the only two possibilities). In such a case where the possible values of the label y are not interpreted as numbers, we are talking about a classification task. If, on the other hand, the goal was to predict the age of the person rather than his sex, which would be a number then we would speak about a regression problem. Fig. 3.3 illustrates these two situations.

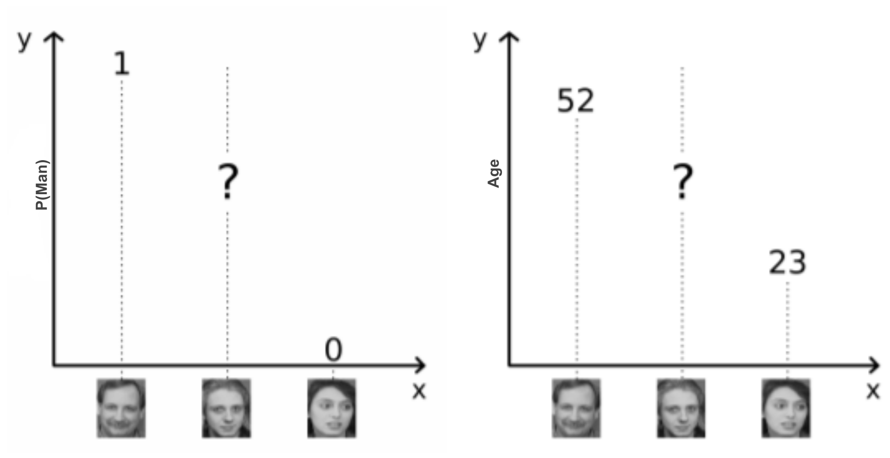


Figure 3.3: Typical examples of supervised learning: Classification (left) to predict sex, and regression (right) to predict age, photos to the extremities of the x axis are example training data points for which the label is known, while the middle photo is the one for which we want get a prediction.

The algorithm is trained on a training dataset of the form $D = z_1, ..., z_n$, with $z_i = (x_i, y_i)$. Many algorithms assume that these examples are taken independently and identically distributed (i.i.d.) from a distribution P, $z_i \sim P(X, Y)$ (where the exact form of P is not known in advance). According to the task we're trying to solve, a supervised learning algorithm prediction is generally an attempt to approximate one of the following three quantities:

- $P(y|x)$: in our classification example (find sex), it would be to predict the probability p that a photo is a photo of a man (the probability that it is the picture of a woman then being $1 - p$). For the regression example (finding age), the prediction would be a probability density over the interval $[0, +\infty]$.
- $argmax_y P(y|x)$: in our classification example, it would be a binary prediction of the most likely sex ("man" or "woman"). In our regression example, it would be the most likely age.

3.2.2.2 Unsupervised Learning

During unsupervised learning, the algorithm never observes the expected response. This learning mode does not require the data to be labeled. Without target, the algorithm must himself discover the latent structure of the data from the observations he receives. Formally, the training set D is only composed of non-labeled data examples $\mathbf{x} \in \mathbb{R}^n$. The advantage of unsupervised learning is the availability of unlabeled data. Indeed, there is an immense quantity and it is constantly growing. Moreover, it is not expensive to acquire them since they do not require any labeling. On the other hand, it is more difficult to directly evaluate the performance of these algorithms since there is no single solution compared to what is expected from the algorithm when presented with a training set. Unsupervised learning is used, among other things, for clustering problems, feature extraction and dimensionality reduction. Among the unsupervised algorithms are sparse coding, auto-encoders [5], Restricted Boltzmann machines (RBM) [30], Principal Component Analysis (PCA) and Independent Component Analysis (ICA) [57].

3.2.3 Optimization

The learning process of a model consists of updating its parameters so that he becomes better at accomplishing the desired task. Mathematically, this corresponds to the problem of optimization:

$$argmin\mathcal{L}(\theta, \mathcal{D}) \tag{3.4}$$

where we look for the parameters θ minimizing the **loss function** $\mathcal{L}(\theta, \mathcal{D})$ on the training set $\mathcal{D}$. To solve this problem of optimization, first order optimization algorithms are generally used. These are techniques that only require the information of the first derivative of $\mathcal{L}(\theta, \mathcal{D})$ (*i.e.* the gradient $\nabla_\theta \mathcal{L}(\theta, \mathcal{D})$). Despite their best convergence rate, higher order optimization algorithms are impractical because of the expensive calculation of higher order derivatives. Generally, the gradient descent algorithm or one of its variants is used (section 3.2.3.1).

3.2.3.1 Gradient Descent

The performance metric used by a machine learning model is called an "objective function". The lower the value, the better the model performs. Gradient descent is a technique used to minimize an objective function $J(\boldsymbol{\theta})$ by modifying the value of the parameters $\boldsymbol{\theta}$ of a model, to which we pass a training set D, allowing it to learn from examples. The gradient $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$

of the objective function relative to the parameters is used to apply a modification to the parameters $\boldsymbol{\theta}$. Fig. 3.4 is a simplified representation in one dimension of $J(\boldsymbol{\theta})$ which varies according to the value of the parameter $\boldsymbol{\theta}$. The orange arrow in the same figure represents the direction of the gradient for a given point (specific $\boldsymbol{\theta}$ value). Since the gradient is a vector indicating the direction of the steepest slope, just subtract it from the parameters $\boldsymbol{\theta}$ to move to the nearest local minimum. The learning rate η, that is multiplied by the gradient to determine the size of the step made in the opposite direction of the gradient. The process is repeated to approach the minimum each time. There are three gradient descent variants that differ in the amount of data used to estimate the gradient of the objective function. The first is Stochastic Gradient Descent (SGD) and uses only one example at a time. The second variant is the gradient descent by "Mini-Batch", which uses a small group of examples to calculate the gradient. Finally, the gradient descent by "Batch" is the variant that uses the complete set D to do its calculation. In this work we used SGD which is more efficient for large scale problems, such as video classification, than batch training because parameter updates are more frequent [41].

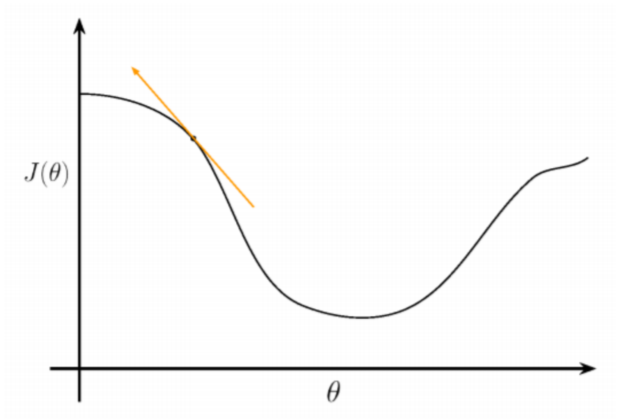


Figure 3.4: Direction of the gradient $\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$. The x axis represents the $\boldsymbol{\theta}$ parameter and the y axis represents the objective function to be minimized

3.2.3.2 Regularization

One of the major problems faced while training a machine learning model with large number of hyper-parameters, is that model can adjusts too much to the training data but does not respond well to the problem we want to solve. This is a phenomenon that is encountered in all classification problems is known as "overfitting". What happens in a situation of overfitting is that the network have learned to recognize almost perfectly the training data, in every detail (including defects, especially noise), but will not be able to generalize in a relevant way for unseen data. In order to prevent the overfitting when training an Artificial Neural Network (ANN), several methods of regularizing model parameters can be applied during training to reduce error generalization without affecting the training loss [49]. Among the most common regularization methods, that we used in this work, Early Stopping and Dropout which we cover in details in chapter 4 when presenting our gesture model training and parameters tuning.

3.3 Deep Networks for Video Classification

Deep Learning has many applications. It is this technology that is used for facial recognition of Facebook for example, to automatically identify your friends on photos. It is also this technology that allows Face ID facial recognition of Apple's iPhoneX to improve over time. As previously explained, machine learning is also the central technology of image recognition. To translate oral conversations in real time, software such as Skype or Google Translate also rely on machine learning. It is also thanks to this technology that the Google DeepMind, the division of Google dedicated to artificial intelligence, is continuously improving its products and services. In the present state-of-the-art for image classification tasks, deep convolutional neural networks have been successfully applied to multiple recognition challenges. In this context, Karpathy et al. observed the best results for large scale video classification when combining CNNs trained with two separate streams of the original and spatially cropped video frames [39]. On the other hand, Recurrent neural network and long short-term memory have also achieved great success in processing sequential multimedia data and yielded the state-of-the-art results when applied in digital signal processing, video processing, and speech recognition problems. Reported experimental results on different video classification tasks show significant improvements when using hybrid networks that combines recurrent neural network (RNN) and convolutional neural networks. In this thesis, we proposed a variant of 3D CNN model for dynamic hand gesture recognition and evaluated it against a state of the art's hybrid model LRCN (CNN+LSTM), thus in the following sections we will present these two deep networks and their variants used in our work.

3.3.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (shortened to CNNs) have been introduced for the first time by Fukushima [23], he derived a hierarchical neural network architecture introduced in Hubel's research work [48]. Lecun [50] has then generalized to successfully classify the handwritten digits using LeNet-5 network shown in Fig. 3.5. Ciresan [15] used convolutional neural networks for image recognition and realized state of the art performance on multiple image databases (*e.g.* MNIST, CIFAR10 and ImageNet).

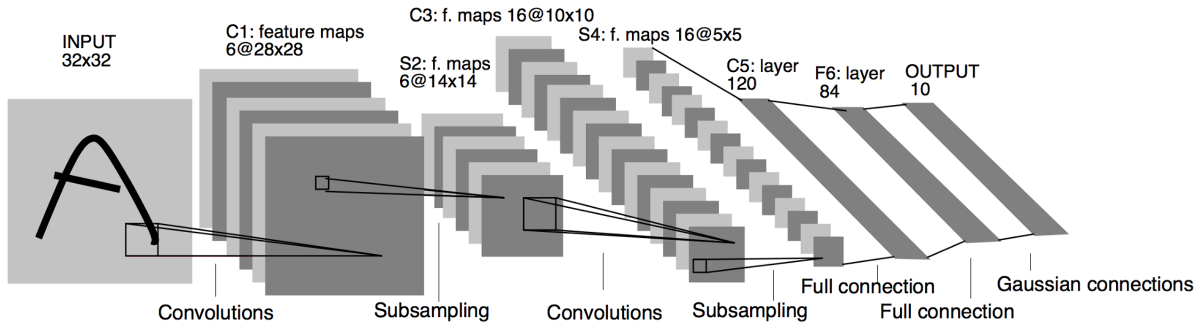


Figure 3.5: LeNet-5: A CNN network for handwritten digits recognition [50].

Convolutional neural networks are composed of two distinct parts. The input is an image provided in the form of a matrix of pixels. For a grayscale intensity image, it's a 2D matrix. The color is represented by a third dimension to represent the fundamental colors (RGB). The first part of a CNN is the actual convolutive part, it functions as a feature extractor from images. An image is passed through a succession of filters, or convolution kernels, creating new images called convolution maps. In the end, the convolution maps are concatenated into a single feature vector to pass through fully connected layers [93]. The different parts of a CNN are illustrated in Fig. 3.6.

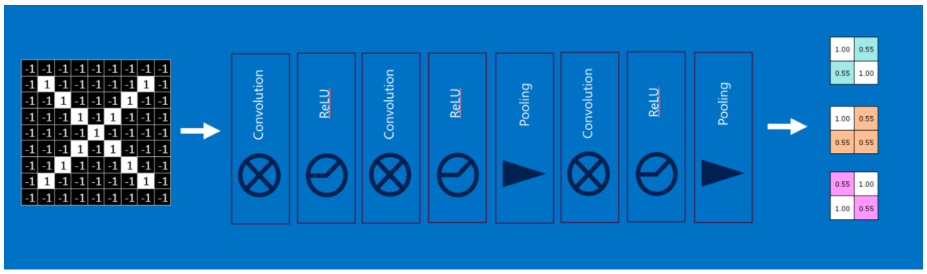


Figure 3.6: Different layers of a convolution neural network

Three types of layers are usually used to form a convolutional network: Convolution layer(s), pooling layer(s) and fully connected layer(s) which are often used as the network output.

3.3.1.1 Convolution Layers

The convolutional layers constitute the core of CNN. When presented with a new image, the CNN does not know exactly if the features will be present in the image and where they could be, so it seeks to find them in the whole image and in any position. By calculating in the whole image if a feature is present, we perform what's called filtering.

3.3.1.2 Pooling Layers

The pooling layer is a sub-sampling operation typically applied after a convolutional layer. In particular, the most popular types of pooling are max (each pooling operation selects the maximum value of the surface) and average pooling (each pooling operation selects the average value of the surface). Pooling is a method to take a large image and reduce its size while preserving the most important information it contains. Fig. 3.7 shows an example max pooling operation on a 2×2 window. In the end, a pooling layer is simply

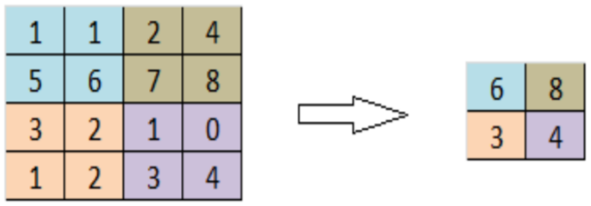


Figure 3.7: Max Pooling[22]

a pooling process on an image or collection of images. The output will have the same number of images but each image will have a lower number of pixels. This will reduce the computational burden. For example, by transforming an 8-megapixel image into a 2-megapixel image, which will make it much easier for the rest of the operations to be performed later.

3.3.1.3 Fully Connected Layers

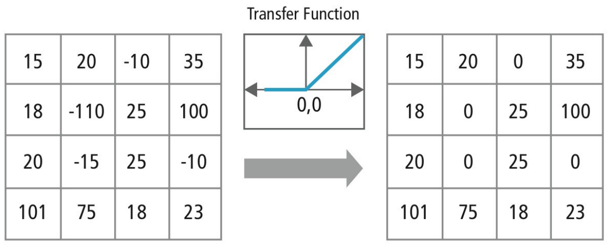
The fully connected layer (FC) is applied to a pre-flattened input where each input is connected to all the neurons. Fully connected layers are typically present at the end of CNN architectures and can be used to optimize objectives such as class scores.

3.3.1.4 Activation Function: Rectified Linear Units (ReLU)

An important element in the whole process is the Rectified Linear Unit or ReLU. The mathematics behind this concept are fairly simple once again: every time there is a negative value in a pixel, it is replaced by a 0. Thus, the CNN is allowed to stay healthy (mathematically speaking) by preventing values learned to stay trapped around 0 or explode to infinity. It is a fundamental tool without which the CNN would not really produce the state of the art results we know today. The ReLU is the following max function

$$f(x) = max(0, x)$$

with input x (*e.g.* matrix from a convolved image). This operation is called an activation function and considered the most popular activation function for deep neural networks. As



15	20	-10	35
18	-110	25	100
20	-15	25	-10
101	75	18	23

15	20	0	35
18	0	25	100
20	0	25	0
101	75	18	23

Figure 3.8: ReLU activation function

shown by the example in Fig. 3.8, the result of a ReLU layer is the same size as the input, with just all the negative values removed.

3.3.2 3D Convolutional Neural Networks for Video Classification

As an extension to 2D Convolution Neural Networks (CNNs) presented in section 3.3.1, which exploit spatial (width and height) features, 3D CNNs operates on features representing both spatial dimensions within a frame and the temporal dimension across frames. 3D CNN was recently intensively applied in video analysis and research works have reported good performance results.

The convolution kernel in 3D CNN is a cube that representing local spatial region in adjacent frames. Let take a kernel with size 2x2x2x1, then the dot product considers receptive field of 2x2 in two consecutive frames (Fig. 3.9) [82]. Mathematically, the 3D convolution formula is as follow:

$$y_{xyt} = f(\sum_i \sum_j \sum_k w_{ijk} v_{(x+i)(y+j)(t+k)} + b)$$

where:

— y_{xyt} is a feature map value at (x, y, t).
— f is the activation function.
— w_{ijk} is a kernel weight and $v_{(x+i)(y+j)(t+k)}$ is an input value at $(x+i, y+j, t+k)$, this way every scalar in the feature map has the information from multiple frames in that particular spatial region.

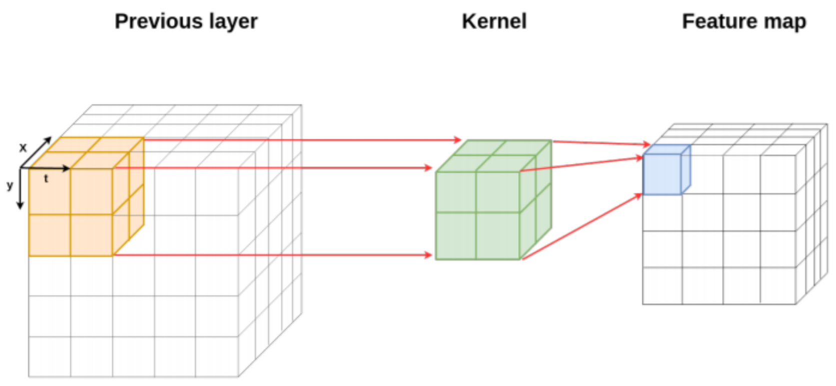


Figure 3.9: Sample 3D convolution operation[82]

3D CNN were used successfully in few research works related to video classification. Huang and al proposed a novel 3D convolutional neural network (CNN), which extracts discriminative spatial-temporal features from raw video stream, to tack the problem of Sign Language Recognition (SLR) [34]. Also, Molchanov and al from NVIDIA, achieved a correct classification rate of 77.5% on the VIVA challenge dataset using 3D Convolutional Neural Networks for Hand Gesture Recognition [55].

3.3.3 Recurrent Neural Networks (RNN)

The recurrent neural networks (RNNs) allow the processing of sequential data, a sequence of entries $x_0 ... x_m$. Indeed at the time t they calculate their output according to the input

x_t but also the state of the hidden layer at the previous time. So they evolve an internal state that acts as a short-term memory - and that allows to take into account the temporal dependencies that the entries show.

3.3.3.1 Vanilla RNNs

The simplest RNNs are described as follows:

— $n, k, p \in \mathbb{N}$
— $x_0 ... x_m$ with $x_t \in \mathbb{R}^n$ a series of inputs
— $W \in M_{k,n}$
— $W_h \in M_{k,k}$
— $O \in M_{p,k}$
— $h_{-1} \in \mathbb{R}^k$

Then the dynamics of the associated RNN is defined as:

$h_t = \sigma(W_{x_t} + W_h h_{t-1})$ $y_t = O h_t$

With $h_t \in \mathbb{R}^k$, the state of the hidden layer and $y_t \in \mathbb{R}^p$ the state of the output layer. In practice we often take $h_{-1} = 0$. We can visualize these RNNs in the form of the graph illustrated in Fig. 3.10:

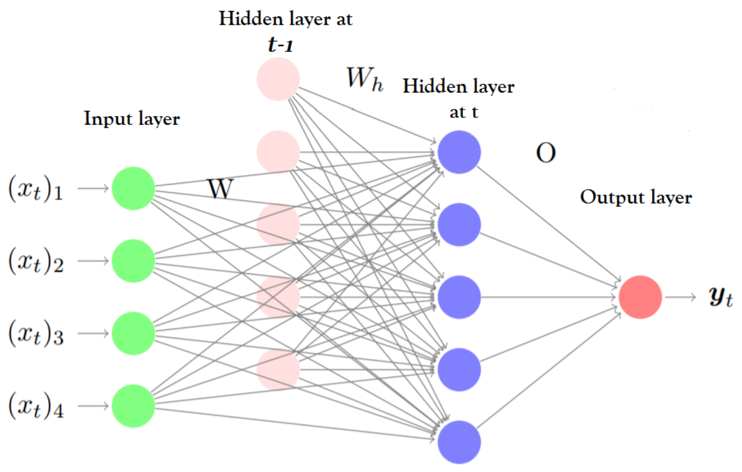


Figure 3.10: RNN with $n = 4$, $k = 5$ and $p = 1$

Theoretically these RNNs form a very powerful model [76], however in practice it is difficult to train them and reach their theoretical power. One of the reasons is presented in [7]: gradient descent adapted to RNNs (Backpropagation Through Time) encounters numerical computation difficulties also known as Vanishing and exploding gradient. This phenomenon is particularly problematic for the classification of videos (contrary to the classification of each image), since the network must wait until the end of the sequence before attributing it a label, and therefore, if the "short term memory" is negligible in front of the size of the sequence, the update of the weights during learning by back propagation will only take into account the first moments. It is to overcome these problems that Long Short-Term Memory (LSTM) model was introduced in 1997 [32].

3.3.3.2 Long Short-Term Memory (LSTM)

LSTMs are a special kind of RNN, capable of learning long-term dependencies. They were first introduced in 1997 by Hochreiter [32]. Intuitively, an LSTM can be seen as a neuron having, in addition to external connections, a recurrent self-connected connection with a fixed weight of 1. This allows to save successive states of the neuron from one iteration to another. In this thesis, the LSTM architecture used was proposed by Gers and al in [24] (Fig. 3.11).

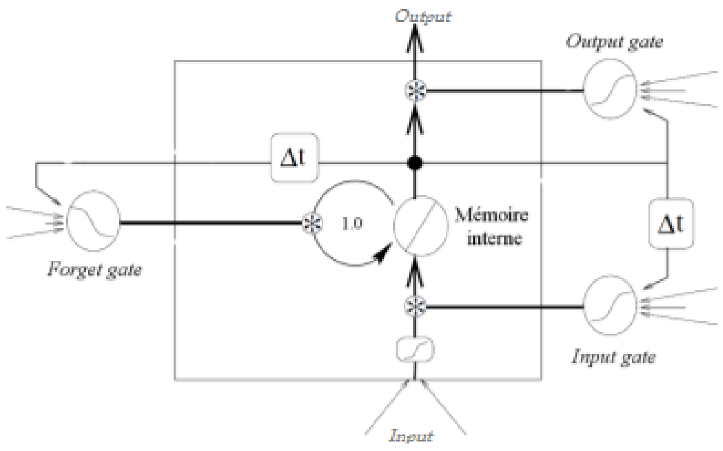


Figure 3.11: Example of an LSTM neuron: architecture proposed in [24]

3.4 Summary

In this chapter we presented the general background theory for deep learning approaches used to solve computer vision tasks, particularly gesture recognition. First, we went over 2D CNN architecture and then covered its extension to solve video classification problems, 3D CNN (*e.g.* [55, 34]). Furthermore, we highlighted the details of the Recurrent Neural Network architecture, specifically, Long Short-Term Memory (LSTM) variation, which represents a main component of the LRCN classifier model discussed along with our proposed gesture classifier in the next chapter.

Chapter 4

Proposed Deep Learning Based Multiclass Hand Gesture Classifier

4.1 Introduction

Hand gesture recognition can be treated as a multiclass classification problem that maps an input video sequence to one of the three classes our model has learned: C1 = Swiping Hand Left (Park In), C2 = Swiping Hand Down (Park Out) and C3 = Doing Other Things. The experimental results presented in this chapter will serve to evaluate the performance of the proposed gesture model, and to compare it with the state of the art method applied to this thesis's use case. In this chapter, we present the machine learning methodology we followed in order to build the different components of our end-to-end multiview hand gesture recognition self-parking system. First, the deep neural network architecture and training process are presented. Then, the different experiments leading to the tuning of the hand-gesture classification network are described. Finally, the different test scenarios conducted while assessing the classifier as well as their corresponding results are reported.

4.2 Training Dataset for Dynamic Hand Gestures

Training deep models requires a big volume of quality data and labeling data can be costly and error prone. Although the network architecture can strongly influence the performance of an AI system, the amount of training data has the biggest impact on performance. Fig. 4.1 (a modified version of a famous slide from Andrew Ng.[2]) shows that the quality gap in AI products is defined by the amount of data. In order to evaluate several variations of

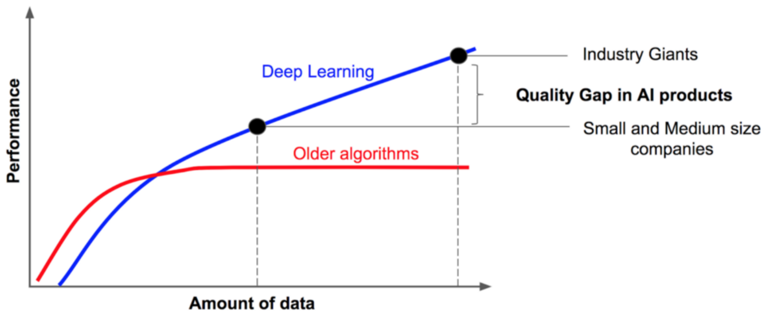


Figure 4.1: Deep learning vs classical machine learning

the gesture network model architectures and their performances, various experiments were conducted in this thesis using the 20BN-JESTER dataset.

4.2.1 20BN-JESTER Dataset

The 20BN-JESTER dataset consists of a large collection of densely-labeled video sequences taken by a static camera (webcam or laptop camera) that show humans performing pre-defined hand gestures. This dataset was collected thanks to a large number of crowd workers and made available by the German company TwentyBN[84] as free of charge for academic research. In this database, we find a total of 148 092 video sequences. The data was provided as a big archive containing directories numbered from 1 to 148 092. Each folder corresponds to one video clip (single gesture) and contains JPEG images that were extracted at 12 fps having a height of 100px and variable widths. The length of sequences differs from one sequence to another (from 27 to 48 frames per video). The dataset groups together 27 classes that represent the different hand gestures, namely: Swiping Hand Left, Swiping Hand Down, Rolling Hand Forward, Doing Other Things, No Gestures, etc... In each class, the hand gesture is performed by participants that represent a generalized distribution of different gender, age, skin color, and with different speeds. The latter, makes this dataset one of the largest data collections available to build a robust deep learning-based gesture classifiers.

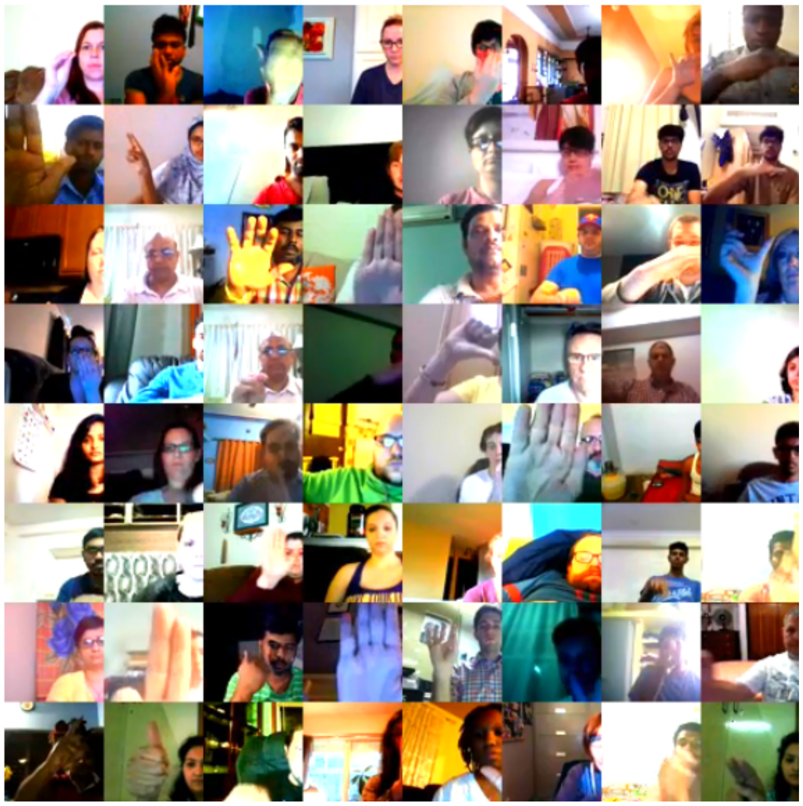


Figure 4.2: The 20BN-JESTER: a dynamic hand gesture dataset

4.2.2 Our Choice of Hand Gestures

It is then necessary to consider an appropriate subset of gestures related to our application. For this thesis, the goal is to recognize three dynamic hand gestures for parking in and parking out actions as well as other gestures (including no gesture). The two gestures that we want to recognize are: Swiping Hand Left (Park In trigger action) and Swiping Hand Down (Park Out trigger action). We considered these two gestures because they are among the most used in human-human interaction and will be perfectly adapted to a natural man-car interaction. Furthermore, among other possible gestures available from 20BN-Jester dataset, the high neural discriminability (i.e. decodability) between the two chosen gestures contributed in giving us the best model performance during our experiments. Figure 4.3, 4.4 and 4.5 show respectively a sample of each of these gestures.

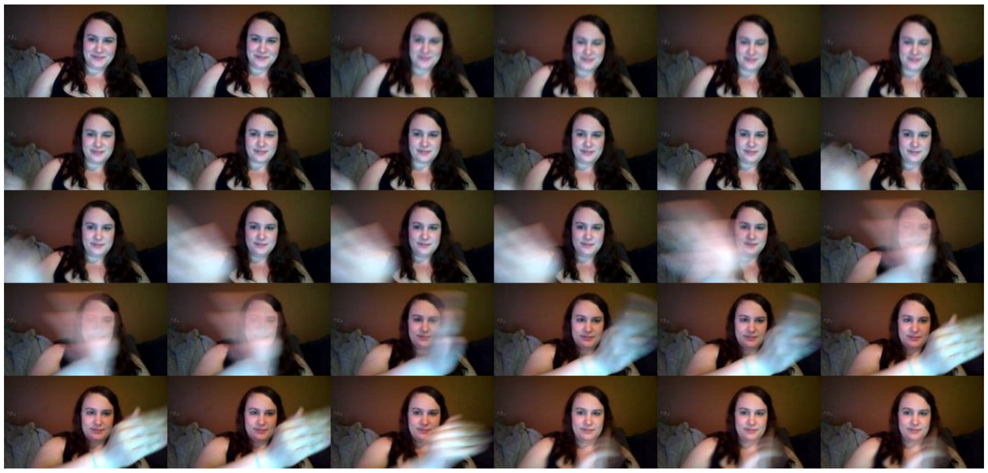


Figure 4.3: Sample "Swiping Hand Left" hand gesture from 20BN-Jester dataset

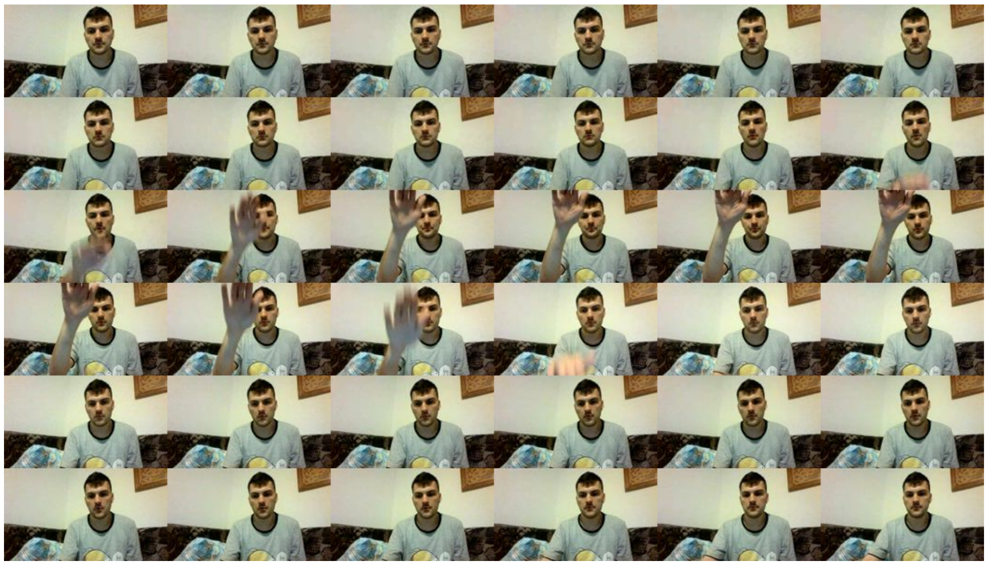


Figure 4.4: Sample "Swiping Hand Down" hand gesture from 20BN-Jester dataset

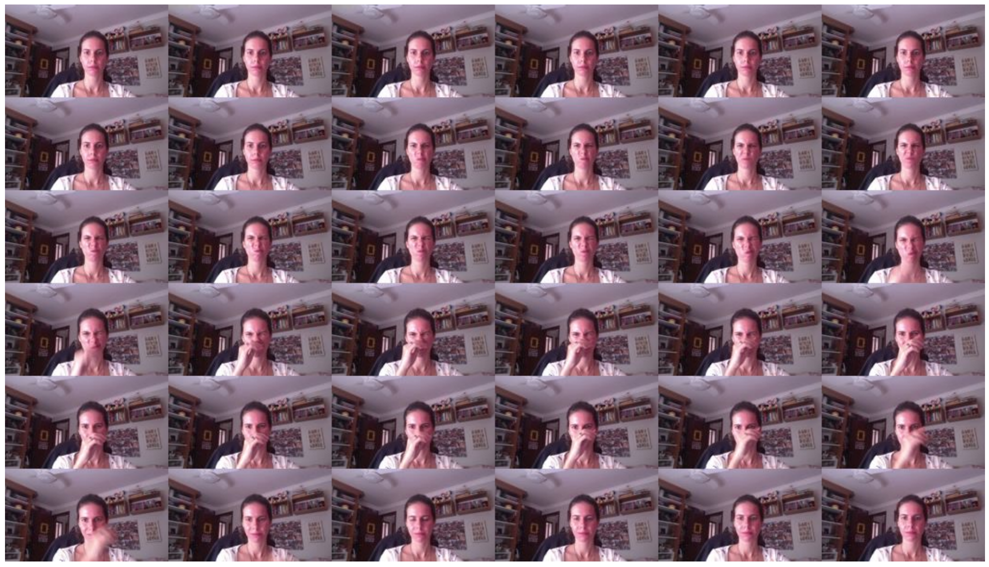


Figure 4.5: Sample from 20BN-Jester dataset showing a person performing a random hand gesture (Doing Other Things class)

4.2.3 Data Preprocessing

Due to the fact that video sequences from 20BN-JESTER dataset have different length (variable number of frames), the first step in our data preparation phase is to sub-sample every video down to 30 frames. So a 31-frames video and a 45-frames video will both be reduced to 30 frames, with the 45-frames video essentially being fast-forwarded. The decision to fix the sequence length to 30 was made after the inspection of the 20BN-JESTER dataset which is mostly composed of videos with a length that varies from 27 to 46 frames. Also, a data cleaning step was performed to limit samples to only videos having a duration greater than the sequence lengths, therefore discarding all shorter videos (*e.g.* 28 frames).

4.2.3.1 Data Splitting

For the context of our work, our model is trained to classify three gestures: Park In (Swiping Hand Left), Park Out (Swiping Hand Down) and Doing Other Things (which covers other possible gestures including no gesture). For that purpose, we used 20BN-JESTER dataset to extract a subset of data containing all videos for the aforementioned three classes. As discussed in section 4.3.2.2, we divided our dataset into 3 subsets: training (D_{train}), validation (D_{valid}) and evaluation (D_{eval}) , the latter two being generally smaller than the first. It is through the ratio (Training: 75%, Validation: 12.5%, Testing: 12.5%) that we can ensure the capacity of the model to generalize well and avoid overfitting. As illustrated in table 4.1, our training data set consists of 2601 video sequences for the Park In gesture (Swiping Hand Left), 2641 videos for the Park Out gesture (Swiping Hand Down) and 8601 videos for "Doing Other Things". The latter class has more than 3 time the number of video sequence to represent real life scenarios, as most gestures do not belong to the first two classes.

Training Data	
Gesture Class	Number of Videos
Park In (Swiping Hand Left)	2601
Park Out (Swiping Hand Down)	2641
Doing Other Things (including No Gesture)	8601

Table 4.1: Number of videos in the training dataset for different gesture classes

Validation Data	
Gesture Class	Number of Videos
Park In (Swiping Hand Left)	437
Park Out (Swiping Hand Down)	428
Doing Other Things (including No Gesture)	2438

Table 4.2: Number of videos in the validation dataset for different gesture classes

Evaluation / Testing Data	
Gesture Class	Number of Videos
Park In (Swiping Hand Left)	430
Park Out (Swiping Hand Down)	430
Doing Other Things (including No Gesture)	2437

Table 4.3: Number of videos in the evaluation dataset for different gesture classes

There are many machine learning methods in the literature that work well on temporal classification tasks as encountered in our dynamic hand gestures classifier. After a review of many of these methods and reported results, we limited our experiments to the following learning algorithms: 3D Convolutional Neural Networks and Long-term Recurrent Convolutional Networks.

4.3 Dynamic Hand Gesture Recognition with 3D CNN

Using supervised convolutional neural network (CNN) models on image recognition tasks has achieved a dramatic progress. Also, a number of extensions have been proposed to tackle the challenging video recognition problem. Based on the work done by Pavlo Molchanov and al [55] on hand gesture recognition using 3D Convolutional Neural Networks, we trained and fine tuned a variant of 3D CNN classifier which we present in this section.

4.3.1 Network Architecture

Three types of layers are usually used to form a convolutional network as detailed in the third chapter. A 2D CNN is composed of convolution layer(s), Pooling layer(s) and finally fully connected Layer(s). The fully connected layer(s) are often used as the network output. Usually, a convolution layer is followed by an activation function and then a pooling layer, this sequence can be repeated several times up to the fully connected layer to form a convolution network that is often denoted under the CONVNET notation. It is also common to use more than one fully connected layer before the output of the network. On the other hand, 3D CNN applies a third dimensional filter to the dataset and the filter moves 3-directions (x, y, z) to learn the low-level feature representations. Their output shape is a 3-dimensional volume space such as cube.

Our model will therefore consists of eight convolution layers, five max-pooling layers, two fully connected layers (FC) and finally a softmax output layer. Fig. 4.6 shows the final 3D CNN architecture of our gesture model for classifying 3 different types of dynamic hand gestures.

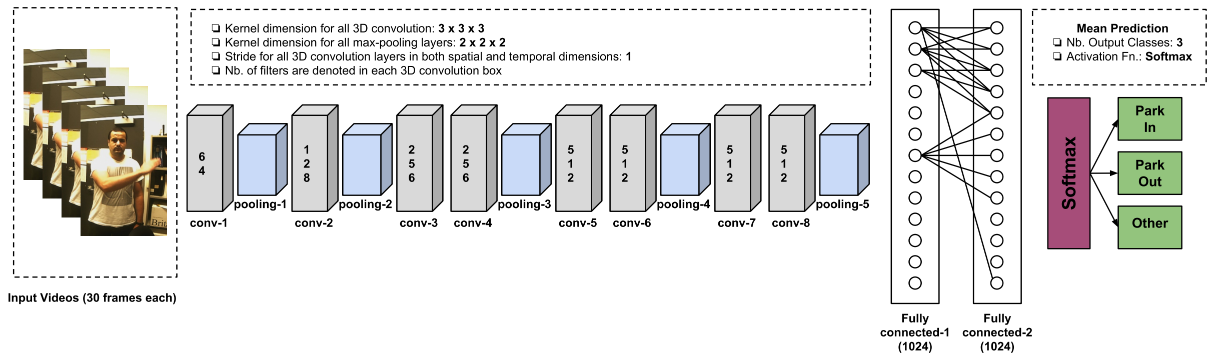

Input Videos (30 frames each)
❑ Kernel dimension for all 3D convolution: 3 x 3 x 3
❑ Kernel dimension for all max-pooling layers: 2 x 2 x 2
❑ Stride for all 3D convolution layers in both spatial and temporal dimensions: 1
❑ Nb. of filters are denoted in each 3D convolution box
64
conv-1
pooling-1
128
conv-2
pooling-2
256
conv-3
256
conv-4
pooling-3
512
conv-5
512
conv-6
pooling-4
512
conv-7
512
conv-8
pooling-5
Fully connected-1 (1024)
Fully connected-2 (1024)
Mean Prediction
❑ Nb. Output Classes: 3
❑ Activation Fn.: Softmax
Softmax
Park In
Park Out
Other

Figure 4.6: Our final model architecture: 3D ConvNet network for dynamic hand gesture recognition

To determine the optimal architecture and parameters of our gesture model (number of convolution layers, number of neurons per layer, number of max-pooling layers, etc.), many models with different configurations have been trained on the dataset presented in 4.2.1. The results obtained from those experiments have helped us determine the best model architecture that produced the highest performance for our use case.

One of the drawbacks of DNN is the difficulty to select the network hyper-parameters which makes the network tuning one of the major phases in a connectionist modeling based machine learning application. In the following sections, we highlight the multiple techniques followed to tune the 3D CNN gesture neural network.

4.3.2 Deep Neural Network Tuning

4.3.2.1 Dropout Selection

Dropout is a regularization technique for neural networks proposed by Srivastava, et al. in their 2014 paper [79] consisting of randomly dropping units along with their connections from the neural network architecture during the training phase. Hence, avoiding overfitting. Dropout is mainly applied to connected layers and requires a parameter to know the number of units to be eliminated at each iteration which is often expressed as the rate of units to be conserved. For example, a rate of 10% will eliminate 90% of connections. This technique indicates dropping out hidden and visible units of a neural network, which results in deleting them from the network along with the outgoing and incoming connections as demonstrated in Fig. 4.7. In Keras, dropout is simply implemented by randomly selecting nodes to be dropped-out with a given probability (*e.g.* 50%). For our 3D CNN model training, a dropout of 50% was used between the two dense fully connected layers after convolutional and pooling layers.

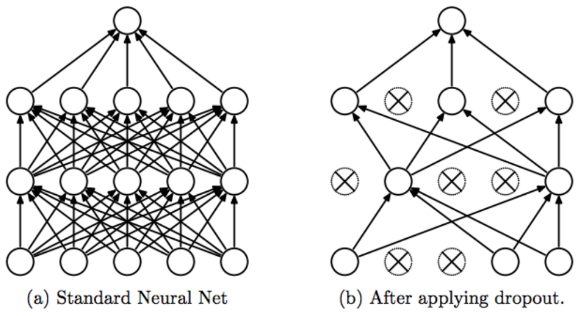


Figure 4.7: Dropout neural net model. Left: NN with 2 hidden layers. Right: A thinned NN after applying dropout to the network on the left

4.3.2.2 Early stopping

When training a learner with an iterative method, such as gradient descent, early stopping consists of a regularization technique that involves stopping the training of the neural network when the minimal validation error is achieved (or in other words validation accuracy starts to decrease). Thus, preventing the generalization performance from degrading and falling into overfitting. The basic idea is illustrated in the Fig. 4.8 below.

We denote by D our training dataset, by $\mathrm{D_{train}}$ the subset from D used for training the algorithm, by $\mathrm{D_{valid}}$ the set of examples that are not in D used for validation. The whole $\mathrm{D_{valid}}$ must also be different from the test set, which cannot be used in the training procedure since it serves to simulate the application of the model on new entries.

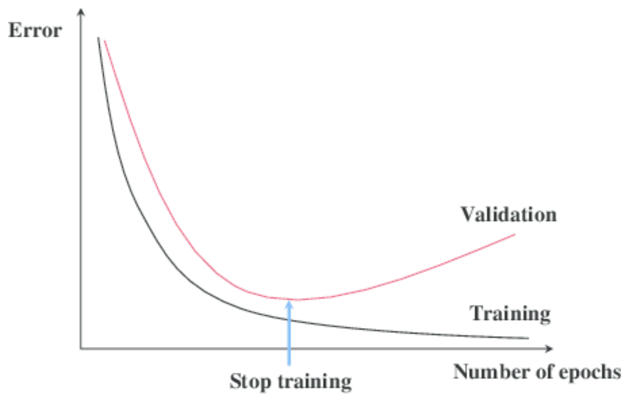


Figure 4.8: Profiles for training and cross-validation errors

We use the validation set to determine when to stop the optimization by monitoring the progress of the calculated error on D_{valid} and stopping the optimization when the error starts to increase. In our project, and after experimenting with various values, early stopping method was used with a patience set to 5, which represents the number of epochs before the validation loss stops improving. The following code listing illustrates the Keras implementation details:

```
keras.callbacks.EarlyStopping(monitor='val_loss',
                              min_delta=0,
                              patience=5,
                              verbose=0, mode='auto')
```

4.3.2.3 Optimization Parameters

In the context of our work, we used Adam (adaptive momentum estimation) optimization algorithm, an extension to stochastic gradient descent (SGD), that has been recently further adopted by researchers in deep learning applications on computer vision and natural language processing (NLP). Tow parameters are required for Adam optimizer: adaptive learning rate and decay.

Adaptive Learning Rate Learning rate is a hyper-parameter that controls the step ratio while adjusting the weights of our network with respect to the loss gradient. We denote α as the learning rate. The following formula shows the relationship:

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1) \tag{4.1}$$

If α is too small, gradient descent can be slow. If α is too large, gradient descent can overshoot and miss the global minima. It may fail to converge, or even diverge. In our algorithm, we chose the value of the learning rate as 1e−5 which gave us the best model performance after running many experiments with various values.

Learning Rate Decay One of the things that might help speeding up the learning algorithm, is to slowly reduce the learning rate overtime. This allows to adjust the learning rate using which is called learning rate decay. Common learning rate decays include time-based decay, step decay and exponential decay. The following formula illustrates how the learning rate α is updated as the number of epochs is progress:

$$\alpha = \frac{1}{1 + decayRate * epochNr}\alpha_0 \tag{4.2}$$

Whereas the learning rate decay does help speeding up training, during our experiments, we noticed that its importance in terms of the hyper-parameters to be tuned is lower than the learning rate α which has a huge impact on the model performance if well tuned. The final decay value used in our training algorithm was 1e−6.

4.3.3 Model Training

The purpose of our classification is to decide whether a video contains one of the two relevant gestures for our use case: Swiping Hand Left (Parking in) or Swiping Hand Down (Parking out). To resolve this problem, training the classifier was performed using a subset of labeled RGB images from 20BN-Jester dataset as previously detailed in section 4.2. For the implementation of the training algorithm, we used Keras, an open source neural network library written in Python and TensorFlow as backend. Table 4.4 shows the specifications of the hardware used for our implementations and experiments. The

Property	**Specification**
Processor	Intel Core i5-6600 CPU @ 3.50 GHz
Graphical Processing Unit (GPU)	NVIDIA GeForce GTX 1070/PCIe/SSE2
Memory	16 GB
Operating System	Ubuntu 16.04.5 LTS

Table 4.4: Hardware specification used the implementation, training and testing of our gesture recognition classifiers

input to the 3D CNN network is 30 frames from the training dataset of a given dynamic hand gesture reshaped to (176x100x3) size. We used the back-propagation algorithm to adjust the networks weights and ReLu as the activation function, with a batch size of 6 (experimented with higher values of batch size to speedup training but rapidly hit the memory limit). Also, a dropout of 0.5 was used between the two fully connected layers which helped avoid overfitting. We compute the validation error (aka. loss) after each epoch with an early stopping patience value set to 5, to stop the training once the validation error stops decreasing for 5 consecutive epochs. Training our classifier took ∼11 hours and went through a total of 10 epochs. Fig. 4.9 and Fig. 4.10 show the loss and accuracy curves for validation and training when the network is being trained. We based our interpretation of these results on the following definitions:

- Underfitting: Refers to a model that can neither model the training data nor generalize to new data. In this case, training and validation losses are both high and accuracy is low.
- Overfitting: An overfit model is one where the loss on the training set is low and continues to decrease, whereas the validation loss decreases to a point and then begins to increase at the same time where accuracy begins to degrade.
- Good fit: A good fit model has a good accuracy on both training and validation sets. This can be diagnosed from the loss curve where the train and validation losses decrease and stabilize around the same point.
- Unknown adjustment: The validation loss in this case continues to decrease until it reaches a low value, but at the same time that of the training continues to increase to higher levels.

Our trained 3D CNN gesture classifier is considered as a good fit model based on the obtained loss/accuracy curves. We can see that our model did not experience a blatant case of overfitting. Validation loss reached its lowest value of 0.074 at the 5th epoch where the validation accuracy was at 0.977. Whereas, training loss continued decreasing to reach a minimum of 0.039 at the 10th epoch with a training accuracy of 0.989.

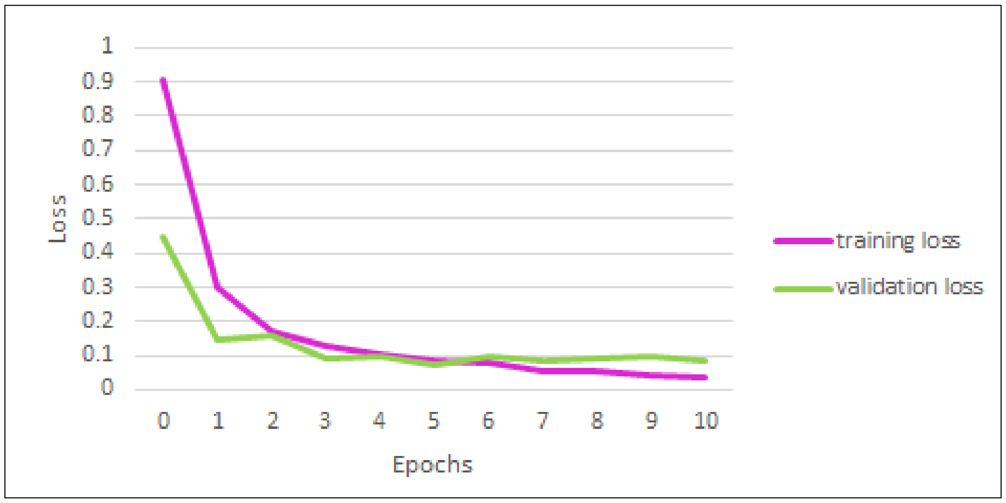


Figure 4.9: Training and validation loss v.s. training epochs (3D CNN)

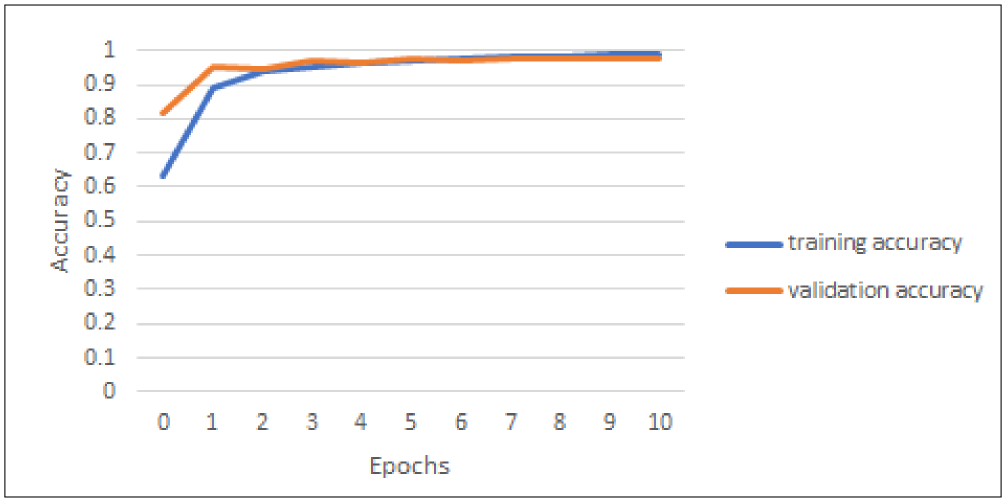


Figure 4.10: Training and validation accuracy v.s. training epochs (3D CNN)

4.4 Dynamic Hand Gesture Recognition with LRCN

This model proposed by Jeff Donahue in 2016 represents a Long-term Recurrent Convolutional Network (LRCN) which combines a deep hierarchical visual feature extractor (such as a CNN) with an LSTM model that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data, visual, linguistic, or otherwise[20]. The reported state-of-the art results in this paper on three vision problems (activity recognition, image description and video description) made Long-term Recurrent Convolutional Network one of the approaches we considered to solve our problem of dynamic hand gesture recognition. The steps of the LRCN model training are detailed in the following sections.

4.4.1 Model Architecture

CNNs have been proved powerful in image related tasks like computer vision, image classification, object detection etc. LSTMs are used in modelling tasks related to sequence-based predictions. LSTMs are widely used in NLP related tasks like machine translation, sentence classification and generation. LRCN, also known as CNN-LSTM model was specifically designed for sequence prediction problems with spatial inputs, like images or videos. As shown in Fig. 4.11, we trained an LRCN network on the same gesture dataset used for training our 3D CNN model by feeding 30 input frames representing one hand gesture to a feature extractor layer (CNN) and combine it with LSTMs to support the sequence prediction. Donahue et al.[20] reported state-of-the-art classification accuracy on human action recognition task by feeding sampled frames from the video sequences containing the human action to the LRCN. In this thesis, we adapted the LRCN model to our specefic problem of dynamic hand gesture classification and used the obtained results during our comparative analysis with the proposed 3D CNN classifier.

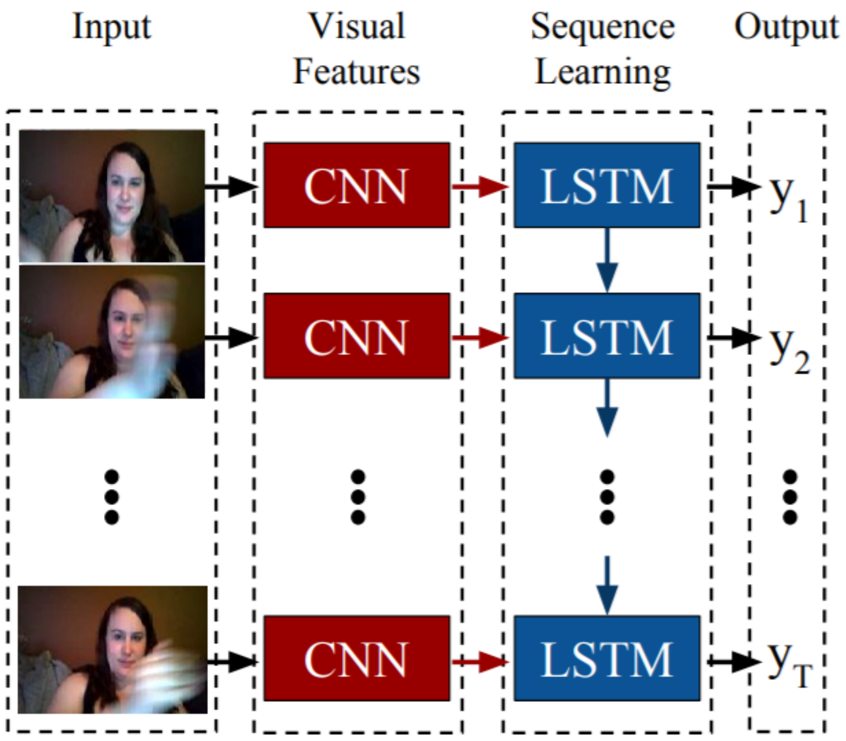


Figure 4.11: Long-term Recurrent Convolutional Network (LRCN) architecture

4.4.2 Model Training

The training of the LRCN classifier was performed using the same computer specifications (GPU, RAM, etc,) used for training the 3D CNN classifier and on the same training/-validation dataset. We used the Adam optimizer with a learning rate of 1e−5, decay of 1e−6 and applied a dropout of 0.5 before the LSTM layer for dimensionality reduction. The total duration of the training using a batch size of 6 was ∼36 hours which is more than 3 times longer than the training duration of the 3D CNN classifier due the much slower training speed of LSTMs [13]. It went through a total of 37 epochs before the model started to converge. We examined the training and validation loss curves, shown in Fig. 4.12 and Fig.4.13, when the network is being trained and we observed that the validation loss stopped decreasing after the 23rd epoch to reach a minimum of 0.248 at the 32nd epoch and then started increasing again until the early stopping mechanism triggered to stop the training 5 epochs later. In comparison with the 3D CNN classifier training, where a much lower validation loss of 0.074 was reached in much shorter amount of time. At the same time, validation accuracy reached a maximum of 0.927 at the 36th epoch compared to 0.977 for 3D CNN. Based on the LRCN training data analysis, the obtained model can be considered as a good fit model since no remarkable overfitting symptoms are observed.

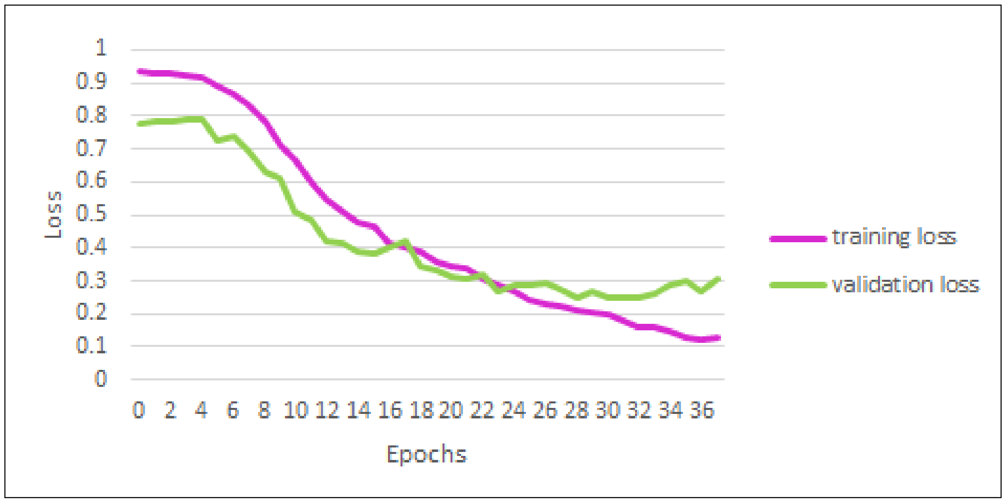


Figure 4.12: Training and validation loss v.s. training epochs (LRCN)

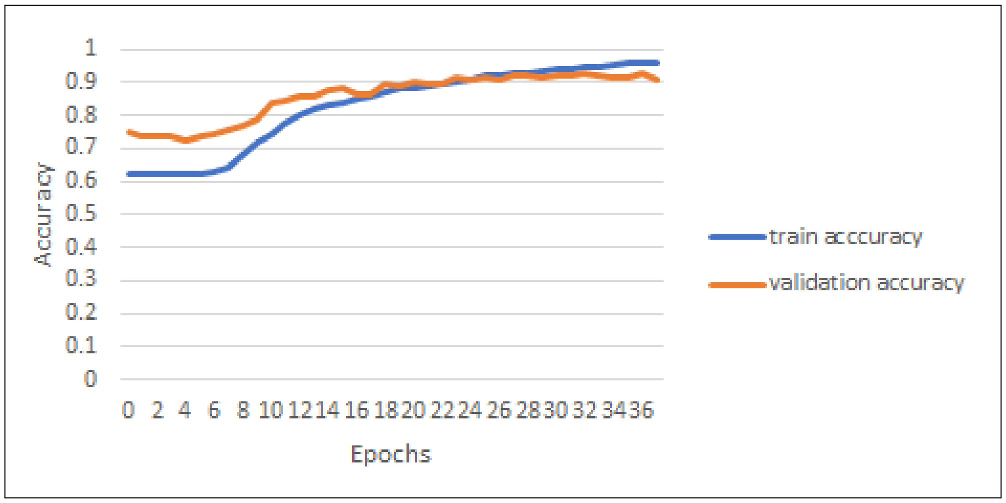


Figure 4.13: Training and validation accuracy v.s. training epochs (LRCN)

It's still difficult at this stage to draw conclusions about the model that allows the best performance on our task of dynamic hand gesture recognition. Therefore, in the following section, we will present our evaluation results of both models tested on our evaluation dataset.

4.5 Experimental Results And Discussion

This section is devoted to the presentation of the experimental results relating to the two models introduced in section 4.3 and 4.4, namely the 3D CNN model and the LRCN model. As mentioned earlier, one of the most important motivations behind the introduction of these two models is their generalisation capacity in comparison with other approaches in literature. The experiments therefore were carried out on dynamic hand gesture recognition task using our evaluation dataset presented in table 4.3. In addition, the experimental results presented in this section will also serve to evaluate the performances of the two model, and to compare the proposed 3D CNN model to the state of the art on the issue studied.

4.5.1 Evaluation Metrics

In order to evaluate the performance of the different learning algorithms, multiple evaluation metrics are proposed in literature. The empirical evaluation of algorithms is a question that occupies several researchers. Most of the measures we used in our work focus on the ability of classifiers to correctly identify the observations of each class. If we denote by TP the number of true positives, by TN the number of true negatives, by FP the number of false positives and by FN the number of false negatives, then the evaluation metrics used in our work are defined like the following:

Accuracy The accuracy represents the good classification rate of the model (*i.e.* in an image classification problem, the model accuracy is the ratio of the number of images correctly classified on the total number of testing images.). In our work, we would like to identify the ability of our developed model to distinguish between the different classes (*e.g.* Swiping Hand Left, Swiping Hand Down, Doing Other Things).This can be mathematically stated as:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.3}$$

Precision Precision is calculated as the ratio of number of correct classifications on the total number of classifications. It can be defined mathematically as:

$$Precision = \frac{TP}{TP + FP} \tag{4.4}$$

Recall It represents the ratio of number of correct classifications on the total number of positive values. Recall is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{4.5}$$

F1 Score The F1 score is the harmonic average of the precision and recall:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.6}$$

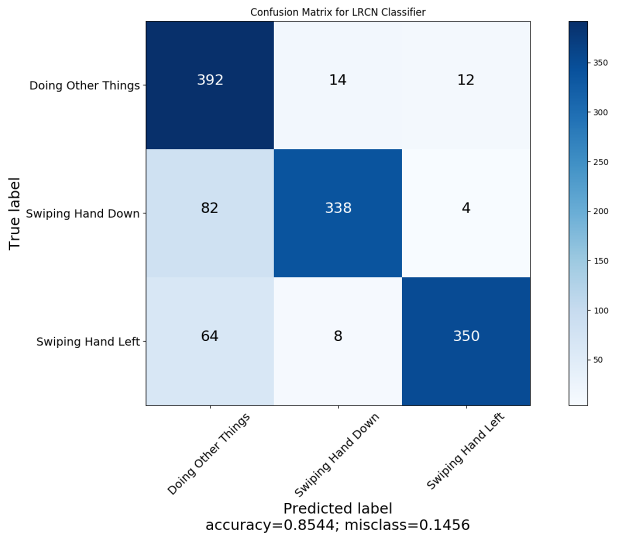
4.5.2 Comparisons and Analysis

Table 4.5 shows the performance of two different classifiers on each class. The results shows that the 3D CNN classifier is dominating LRCN on the three classes (Swiping Hand Left, Swiping Hand Down and Doing Other Things). The performance of the two classifiers is similar on the average precision, especially for the Swiping Hand Left and Swiping Hand Down classes, than that on recall and F1-measure. Furthermore, the table also shows that while the LRCN classifier achieved a high precision for Swiping Hand Left and Swiping Hand Down classes that is comparable to that of the 3D CNN, we notice on the other hand a relatively poor recall on the same two classes; which means the LRCN system classifies more samples into Doing Other Things, hence the high recall value for the latter class and poor precision. Now going back to our use case in this project: a gesture recognition self-parking system, both metrics, precision and recall, are important; we would like to achieve a high precision on the Swiping Hand Left (Park In) and Swiping Hand Down (Park Out) classes, but most importantly a high recall value to give a better user experience to the driver using the system while avoiding cases where the driver need to repeat the hand gesture many times to trigger the parking action.

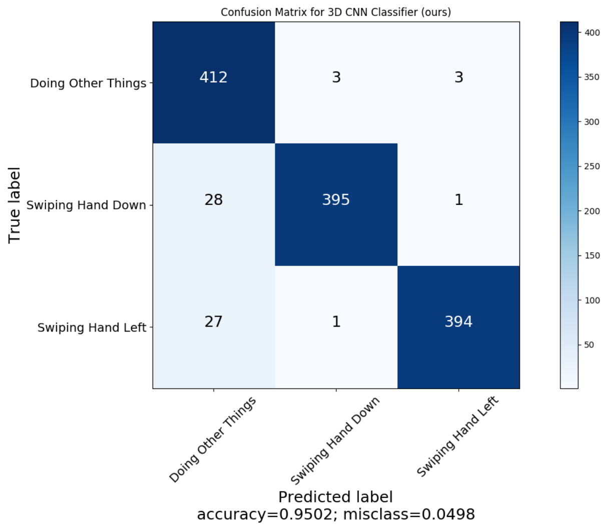
Class	**LRCN**			**3D CNN (ours)**		
	Precision	Recall	F1-Measure	Precision	Recall	F1-Measure
Swiping Hand Left (Park In)	0.96	0.83	0.89	0.99	0.93	0.96
Swiping Hand Down (Park Out)	0.94	0.80	0.86	0.99	0.93	0.96
Doing Other Things	0.73	0.94	0.82	0.88	0.99	0.93
Average	**0.88**	**0.85**	**0.86**	**0.95**	**0.95**	**0.95**

Table 4.5: Per class performance comparison between LRCN and 3D CNN

The confusion matrices on the validation set can be seen below in Fig. 4.14, the overall accuracy of the LRCN classifier is **0.8544** whereas that of our 3D CNN classifier is at **0.9502**. The confusion matrix for LRCN shows clearly that many samples of Swiping Hand Left and Swiping Hand Down are classified as Doing Other Things, hence the poor recall noticed earlier. As expected, the LRCN model performed poorer that 3D CNN. The proposed explanation is that the position of the hand in each of the 30 frames will differ from sample to sample, which leads to feeding the LSTM with different positions of the hand in the respective indexes of the 30 frames. This could confuse the LSTM which is reflected in the the number of false negatives.



(a) LRCN gesture classifier



(b) 3D CNN Gesture Classifier (ours)

Figure 4.14: Confusion Matrices for both LRCN and 3D CNN Hand Gesture Classifiers

4.5.3 Transfer Learning and Final Model Fine-tuning

Among the most common challenges faced when applying machine learning to solve a specific problem, is that training data can be slightly different from the real-world data faced by the algorithm. Hence, the performance of the end-to-end system once faced with real data may not be at the desired level. We noticed that the trained 3D CNN classifier did not perform well when tested live. Two main factors had a major impact on the performance of our system:

- Driver To Camera Distance: The closer the driver is to the multiview camera, the higher is the accuracy of the system. A camera distance within a range of [60cm, 110cm] produced the best performance. Whereas, in the real-world scenario, the car driver would have a distance of at least 2 meters from the car to trigger the auto parking.
- Height of the Multiview Camera System: We noticed also during our end-to-end testing of the system that the camera system needs to be at a certain height (slightly lower than the user) in order to achieve the best system performance and accuracy. Once we place the multiview camera at the same level or slightly higher than the user, gesture detection accuracy starts to degrade.

The previous observations can be explained by the nature of the 20BN-Jester dataset we used to train our gesture model. In fact, most of the video samples in 20BN-Jester dataset are taken using a laptop or desktop computer webcam placed at a relatively close distance (50 to 100cm) and slightly lower level from the user. Hence, the sensitivity of our end model to those factors. In order to overcome these limitations, enhance the system performance and end user experience, we fine-tuned our model using transfer learning techniques. The following section would describe the contingency steps that were taken to overcome the aforementioned challenges.

4.5.3.1 Transfer Learning: Dataset Augmentation

A dataset containing generalized gesture videos that are relevant to our use case of autonomous parking was not available. Therefore, we collected a second dataset in our lab and used data augmentation techniques to generate more data samples. The created dataset consists of a reasonable amount of videos (220 sequences) where many subjects performed the Swiping Hand Left and Swiping Hand Down gestures at variable distance from the camera system and at different setup heights. On the background of the user, we placed a green screen that enabled us to use the Chroma Keying technique (a.k.a. green screen keying, used for decades in film studios by placing human characters in otherworldly situations without them having to leave the studio) to create new videos using parking lot backgrounds, reflecting the real-world scenario of parking situations, and various other backgrounds for data augmentation purpose. Fig. 4.15 shows the process we followed to generate the new dataset using Chroma Keying.

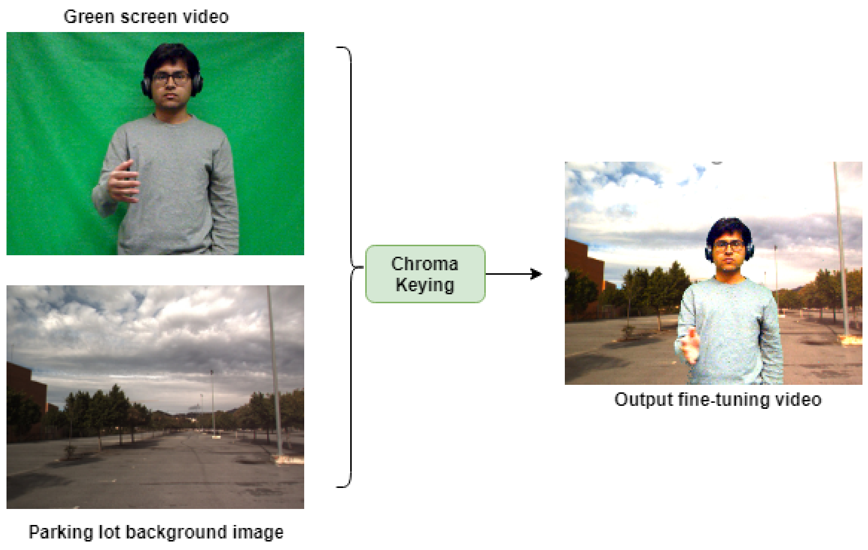


Figure 4.15: Data generation using Chroma Keying

4.5.3.2 Fine-tuning Algorithm

The early layers of the 3D CNN network already trained with the 20BN-Jester large dataset can extract generic features, so we used methods that further tunes a pre-trained model. Given the relatively small dataset (220 videos) compared to the 20BN-Jester dataset, we did only fine-tune the last layer of our 3D CNN which enhanced significantly the classification performance. The evaluation metric for live tests was empirical and based on the automaker satisfaction of the performance. After the dataset augmentation, the automaker reported a twice as good performance.

4.6 Summary

In this chapter, we have presented a dynamic hand gesture recognition approach by implementing a variant of 3D convolutional neural network which achieved a classification accuracy of 95.02%. The experiments were carried out on an evaluation subset of the 20BN-Jester data set [84]. We also compared our model and evaluated its performance against the state of the art method for visual recognition and description, Long-term Recurrent Convolutional Network (LRCN). We found that the latter is significantly slower to train and also under-performed our 3D CNN model for the dynamic hand gesture recognition task. Finally, due to the not satisfying performance of the gesture classifier once tested under real-world conditions representing the use case of this thesis, namely dynamic hand gesture recognition for car self-parking, we fine-tuned our final pre-trained model using a more representative dataset collected in-house that enhanced the final system performance significantly.

Chapter 5

An End-to-End Mutliview Gesture Recognition Operating in Real-time

5.1 Introduction

One of the main parts of our thesis is the implementation and deployment of an end-to-end system for dynamic hand gesture recognition. The prototype presented in this chapter was shared with partnering automotive manufacturer and received a very positive feedback.

In this chapter, we will present our proposed embedded multiview vision based gesture recognizer for autonomous parking system. We will cover the high level system architecture, hardware choice as well as system implementation and deployment details.

5.2 System Architecture

In this section we present the architectural design and implementation of a car vision-based self-parking system using deep learning methods to detect and classify driver gestures captured by a Multiview 360-Degree camera system. We describe the overall design of the proposed system as well as study the hardware and software frameworks used to build our solution.

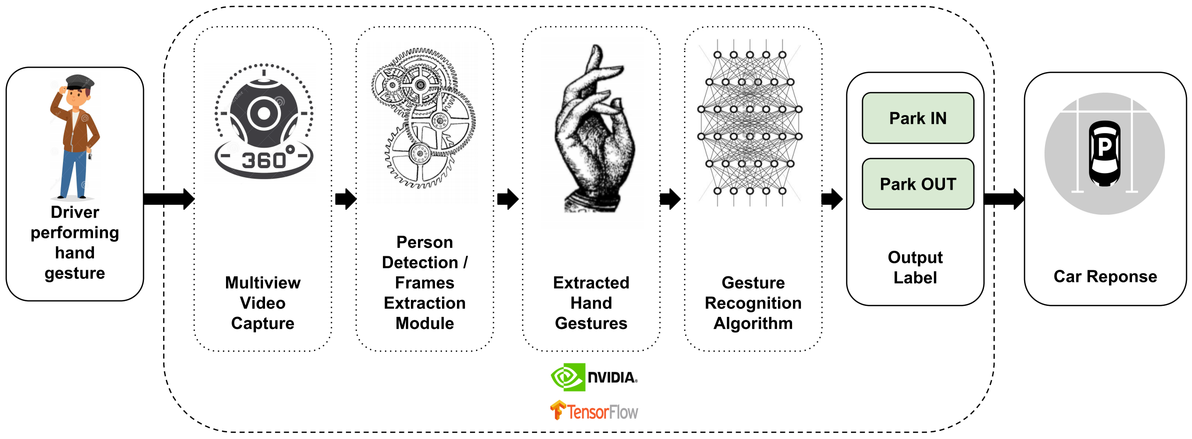


Figure 5.1: Proposed multiview hand gesture recognition system overview

5.2.1 Video Hardware Choice

For capturing 360 degree video frames, many hardware solutions exists on the market, and our choice was the HexCamera (e-CAM30_HEXCUTX2) from e-con systems[1] which consists of a multiple camera solution for NVIDIA Jeson TX1/TX2 developer kit. The setup consists of six cameras with 3.4MP each and an adaptor board to interface with the J22 connector on the Jetson. The camera can stream 720p(HD) and 1080p (Full HD) at 30fps in uncompressed YUV422 Format. The camera system is presented in Fig. 5.2.

[1] https://www.e-consystems.com/multiple-csi-cameras-for-nvidia-jetson-tx2.asp, accessed: 01/29/2019

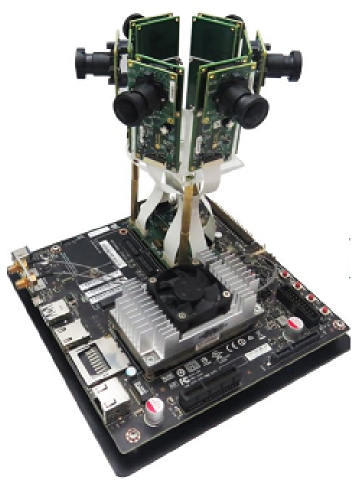


Figure 5.2: e-CAM30_HEXCUTX2 - Six synchronized full HD cameras for NVIDIA Jetson TX1/TX2

5.2.2 Person Detection and Frames Extraction Module

This module is the first core component of our end-to-end system. As mentioned in the previous section, the output of the multiview camera is a six frames stream representing 360 degree view. One of the main challenges, is to adapt the output of the multiview camera to the hand gesture recognition module presented in the previous chapter. The 3DCNN gesture recognition module was trained on video frames of size 176x100 where every video consists of one single subject performing the hand gesture. Given that the multiview camera output may contain multiple subjects in crowded environments (e.g. parking lot), the first step the person detection and frames extraction module performs, is the detection of all subjects present in the six frames video input. This module detects all the persons present in the 360 camera view feed, calculates the bounding box coordinates and finally crops over every 30 frames (required input length for the 3DCNN gesture recognition module) and saves separate image files for every subject. Once this step is complete, the resulted frames are passed to the 3DCNN network to perform the hand gesture recognition. It's important to note that the authentication of the car's owner is out of the scope of this thesis, which means any subject performing one of the two gestures (Swiping Hand Left and Swiping Hand Right) would trigger the parking operation.

The person detection module uses an underlying object detection library. In this thesis, we evaluated two object detection tools that were released recently using the convolutional neural networks: Faster R-CNN and YOLO. We chose these tools because YOLO allows to get the best results on VOC2007[2] data and VOC2012[3] and Faster R-CNN is one of the most used CNN methods so far. In the next section we will highlight our evaluation details as well as the choice of the object detection library used in our final system.

5.2.2.1 Faster R-CNN

The first classifier tested for person detection is the algorithm created by S.Ren et al [69] which relies on a detection made entirely with convolutional neural networks (Fig. 5.3):

- A first convolutional neural network takes as input any image of any size and outputs regions where the objects to be detected might be.
- the second network takes as input the regions proposed by the first network and decides whether they contain the object to be detected.

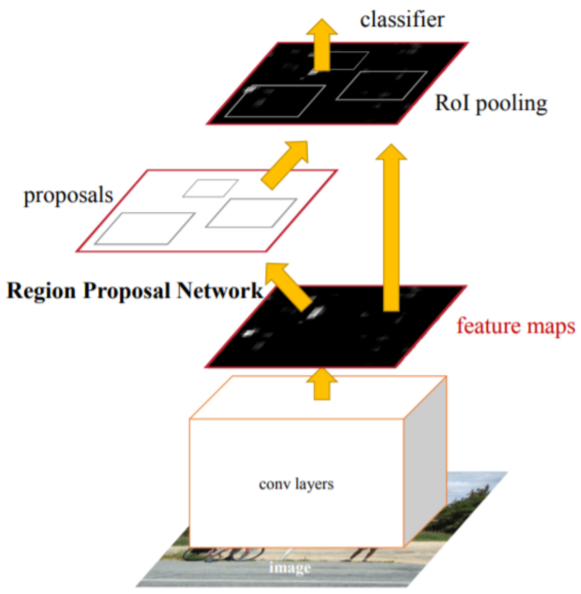


Figure 5.3: Faster R-CNN network for object detection[69]).

[2]http://host.robots.ox.ac.uk/pascal/VOC/voc2007/, last accessed: 02/20/2019
[3]http://host.robots.ox.ac.uk/pascal/VOC/voc2012/, last accessed: 02/20/2019

5.2.2.2 YOLO: You-Only-Look-Once

The second considered classifier, and finally selected, is the algorithm developed by J.Redmon et al. in 2016 [67]. Shorthand for You-Only-Look-Once,YOLO is a neural network capable of detecting what is in an image its location, in a single forward pass of the image through the network, which was a shortcoming of previous methods. It gives the bounding boxes around the detected objects, and it can detect multiple objects at a time that was invented with the purpose of being able to perform real-time inference. This algorithm is based on two steps that are applied to images of predefined size when learning (Fig. 5.4):

- Object detection using a convolution neural network (CNN).
- A bounding box on the image where the predicted class of the object if it exists (in our case a person or nothing).

The network first divides the input image into a grid (13 by 13 cells are used in this thesis). Each cell is responsible for predicting a fixed amount of bounding boxes (5 bounding boxes per cell is used in this thesis). Along with every bounding box prediction, each box is associated with a confidence score prediction and a class prediction. The confidence score is a value of how certain the network is that the predicted bounding box encloses an object of any kind.

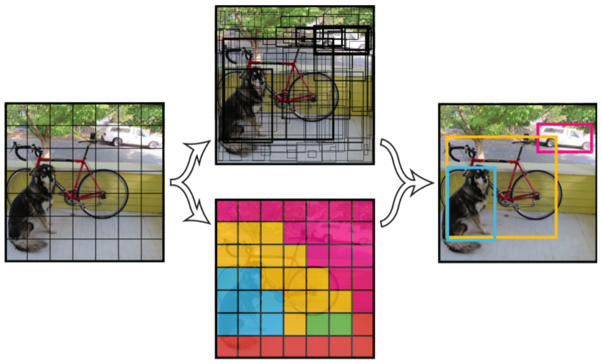


Figure 5.4: Real-time object detection with YOLO: the algorithm models detection as a regression problem. It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes and class probabilities [67]

5.2.2.3 Comparison of Both Techniques

One of the main differences between YOLO and Faster R-CNN is the computation time, YOLO allows to have a detection of 37 frames per second for an image of 445x445x3 while Faster R-CNN allows you to have only 5 frames per second. Moreover, on the VOC2012 and VOC2007 data sets, YOLO seems to give better results. For our final implementation, YOLO v3, the latest version which is extremely fast and accurate, was used in conjunction with the underlying meaty part of the network: Darknet (a framework to train neural networks, it is open source and written in C/CUDA and serves as the basis for YOLO). For the purpose of our use case, we limited the object detection in YOLO v3 to only one class: Person.

5.2.2.4 Person Detection Workflow on Multiview Frames

Our six cameras video system is streaming a live video feed in the format shown in Fig. 5.5. It consists of the concatenation of six frames covering a 360 degree view.

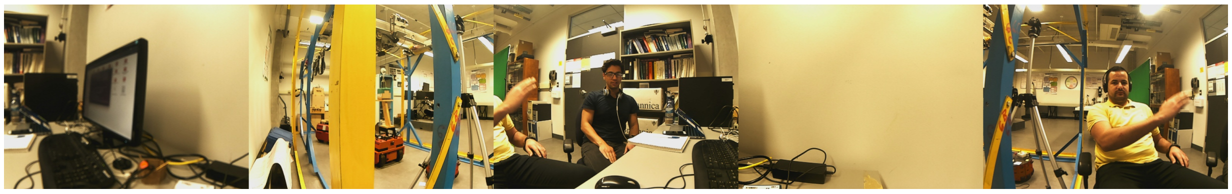


Figure 5.5: Sample output of the e-con hexcam six camera video output

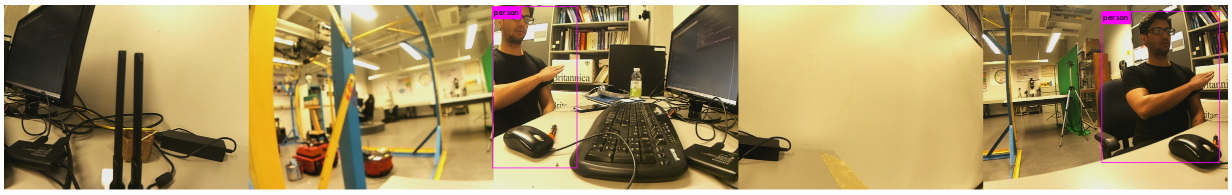


Figure 5.6: Example of YOLO v3 real-time person detection and bounding boxes calculation

The implemented algorithm continuously captures 30 frames (expected sequence length by the 3DCNN gesture recognition network) at 12FPS and passes them to the YOLO based person detector. The latter, only looks at the first frame of the 30 frames input and detects all the persons present, calculates the bounding boxes (example shown in Fig. 5.6) and finally crops all of the 30 frames based on the respective boxes coordinates. The cropped videos are then saved in memory (Numpy array) to be passed one at a time to the gesture recognition module. An example of the cropped videos is shown in Fig. 5.7

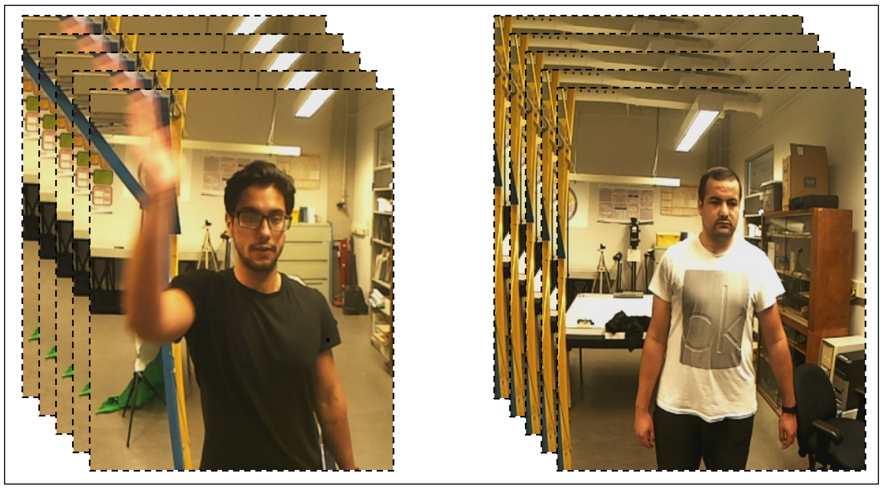


Figure 5.7: Sample cropped persons images: Left image shows a detected person performing the "Swiping Hand Left" gesture (used as Park IN trigger). Right image shows a detected person not performing any hand gesture. Both outputs consist of 30 frames each and are both passed to the 3DCNN gesture recognition network.

5.2.3 Gesture Recognition Module

The hand gesture recognition module represents the second core component of our end-to-end system after the person detection module. It encompasses mainly our 3DCNN dynamic hand gesture classifier detailed in the previous chapter. Once the persons detection and cropping step is complete and the output frames (such illustrated in Fig. 5.7) are extracted, we resize them to match the dimensions of the expected input video by the 3DCNN classifier, in our case with height x width x frames equal to 176 x 100 x 30. Then, we pass every input (consisting of 30 frames) to the hand gesture recognition network for classification. The output of this module can be one of the three classes: Swiping Hand Left (Park IN), Swiping Hand Down (Park OUT) or Doing Other Things (ignored by the system). Once one of the relevant classes (Park IN or Park OUT) is detected, the algorithm drops the rest of the input cropped videos and trigger the corresponding parking action.

5.3 Implementation

We highlight in this section the implementation details of our real-time system, challenges faced during deployment of the live inference software on the embedded platform and how we overcame them.

5.3.1 Tools and Languages

The implementation of the different modules described in section 5.2 as well as the real-time prototype makes use of several programming technologies that are illustrated in Fig. 5.8.

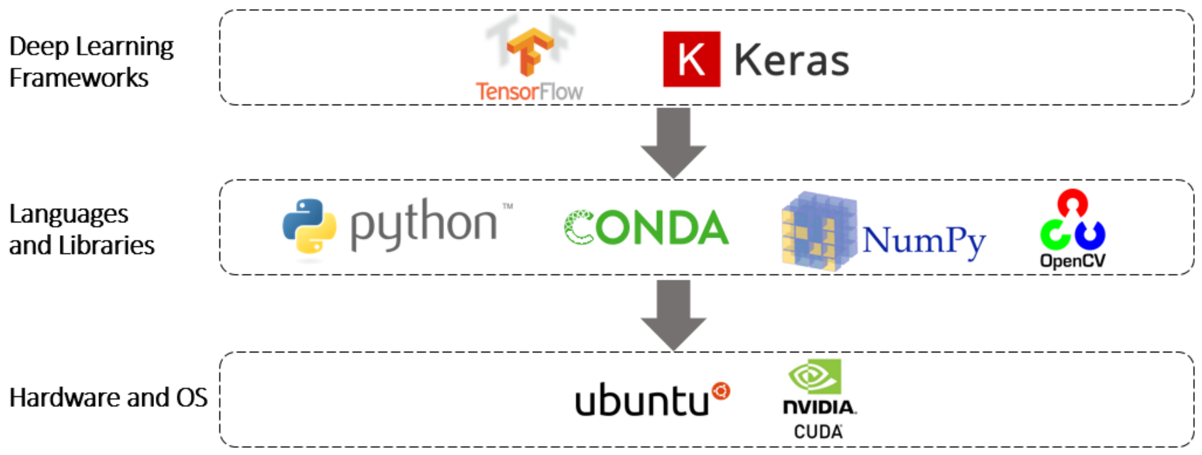


Figure 5.8: Technologies used to implement the different components of the real-time multiview gesture recognition prototype

Keras is an open source Python library (MIT license) developed by a Google engineer, François Chollet (author of "Deep Learning with Python" book [14]). Keras was developed to facilitate development and experimentation with deep neural network models. We used Keras and Python3 to implement the training and fine tuning algorithms of our gesture classifier. It allowed us to implement complex architectures like 3DCNN or LRCN in a reasonable amount of time. We used the GPU version of Tensorflow v1.6 as a backend. Several other libraries and frameworks were used for video streaming and processing(OpenCV3), for data manipulation(NumPy) and package manager and environment management system (Conda). The real-time end-to-end system runs on NVIDIA Jetson TX2 board using TensorRTv1.6, CUDA8 Toolkit and CUDNNv6.

5.3.2 Deployment on an Embedded Platform

The person detection framework was able to operate real-time together with dynamic hand gesture recognition classifier on an NVIDIA Jetson TX2 board. The live inference software was implemented to run on the Jetson platform using the GPU-accelerated version of Tensorflow framework. One of the first challenging steps was the installation of the e-con hexcam six cameras system described in 5.2.1 on the Jetson TX2 board, including setting up the drivers which were only compatible with an older version of L4T r27v1.0 (the operating system of the NVIDIA Jetson board). A lot of effort was required to run that hardware with a newer version of CUDA and CUDNN libraries required by the person detection and gesture recognition modules.

Once the six cameras system was installed and functional, the next challenge was running both the person detection module, which includes loading the Darknet model (object detection network) and YOLOv3 network configuration, and our 3DCNN hand gesture recognition model. Due to the limited available RAM on the Jetson board (8GB), we faced many issues running the end-to-end system where the GPU was running out of memory while loading the 3DCNN model (1.1GB) into memory. To overcome these issues, the following optimization techniques were applied:

- Optimize model saving: Keras offers different model saving methods. The most common method, which we used during our initial test, is `model.save()`. This method saves the architecture of the model allowing to re-create it, the weights of the model, as well as the training configuration (loss, optimizer) and the state of the optimizer. This resulted in a heavy model of 1.1GB. To significantly reduce the memory footprint of the 3DCNN model, we used `model.to_json()` to make a JSON string of the architecture and save it, and `model.save_weights()` to make a separate file containing the weights. The resulted files were considerably smaller at 379.2MB, that's about a third the size of the full `model.save()` result.
- Use Half-precision floating-point format (a.k.a FP16): This consists of setting the format to 16-bit to represent the network weights, as opposed to the standard 32-bit floating point, or FP32. This allowed us to reduce the memory by cutting the size of our tensors in half without sacrificing the accuracy for the model.

- Tune Tensorflow GPU configuration: TensorFlow maps by default almost all of the GPU memory to the process. One way to optimize memory allocation, is to limit the percentage of the overall amount of memory Tensorflow process can allocate. In our case, we set that limit to 25%, which reduced significantly the initial memory allocation requirements. In Tensorflow, this could be achieved by:

```python
    # set tensorflow gpu options
    gpu_options = tf.GPUOptions(per_process_gpu_memory_fraction=0.25)
```

5.3.3 Prototype Overview

This thesis's use case is a multiview hand gesture recognition system for autonomous car parking, therefore, we mounted the Jetson TX2 board on a tripod at a height comparable to a mid-size car roof height (∼180cm) as shown in Fig. 5.9. Mounting our system on a vehicle was out of the scope of this project. Hence, we created an animated web application that our demo application triggers via HTTP requests based on the output of the inference program.

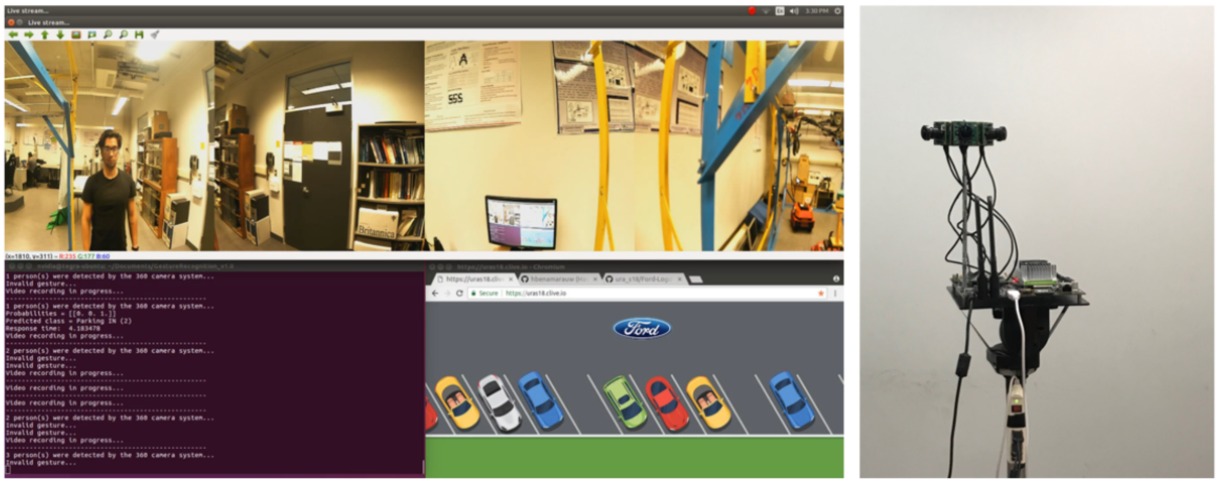

```
1 person(s) were detected by the 360 camera system...
Invalid gesture...
Video recording in progress...
--------------------------------------------------
1 person(s) were detected by the 360 camera system...
Probabilities = [[0. 0. 1.]]
Predicted class = Parking IN (2)
Response time:  4.183478
Video recording in progress...
--------------------------------------------------
2 person(s) were detected by the 360 camera system...
Invalid gesture...
Invalid gesture...
Video recording in progress...
--------------------------------------------------
Video recording in progress...
--------------------------------------------------
Video recording in progress...
--------------------------------------------------
2 person(s) were detected by the 360 camera system...
Invalid gesture...
Invalid gesture...
Video recording in progress...
--------------------------------------------------
3 person(s) were detected by the 360 camera system...
Invalid gesture...
```

Figure 5.9: A snapshot of the experimental results from a live demo

The processing time for the end-to-end inference (person detection + hand gesture classification) was on average ∼2 seconds. The main portion of that latency is coming from the person detection and cropping part. The gesture recognition took less than 1 second during our tests.

5.4 Summary

In this chapter, we have successfully implemented the proposed Multiview gesture recognition framework onto an embedded platform using Nvidia Jetson TX2 Developer Kit. We achieved a run time of 2 seconds on 16 frames/s. We presented the implementation details of the different components of our final prototype and the challenges faced, mainly due to resources limitations (e.g. RAM) on the embedded platform and how we overcame them.

Chapter 6

Conclusion and Future Work

This chapter provides a summary of this thesis, and includes a review of the proposed system and a highlight of the main results and conclusions. Future improvements of this work are given in section 6.2.

6.1 Conclusion

In this thesis, we propose a vision-based multiview dynamic hand gesture recognition system and its application to vehicle self-parking. We study existent self-parking systems and we present their user experience shortcomings, mainly their dependency to an intermediary device. In an effort to develop the next generation of vehicle self-parking feature, we partner with one of America's big three automakers to prototype a robust end-to-end system that operates in real world environment.

Our main motivation for this thesis was to eliminate the intermediate medium between the car and the driver to offer a friendly user interface for the self-park feature. To achieve the aforementioned attribute, we solely rely on a vision-based gesture recognition solution.

The most comprehensive available database to train a dynamic hand gesture recognition classifier is "20BN-Jester"[84]. For simplicity, the developed feature has two commands, represented by two hand gestures "swiping hand left" and "swiping hand down". Hence, we filter the comprehensive "20BN-Jester" dataset to only two classes with an additional "doing other things" class, which is represented by a randomly selected non-command hand gestures.

As a first step to recognizing the hand gesture, the solution requires a person detection algorithm at the front of the pipeline. We research multiple alternatives and select the best performing model, in this case, YOLO.

YOLO, paired with a six-camera based sensor offers a pre-processing module that detects and identifies all instances of "persons" in the 360 view of the vehicle. Each of the identified objects (person present within the vehicle field of view), is then processed by a 3DCNN classifier. The latter is a multi-class classification, which maps the first hand gesture (swiping hand left) to the part-in command and the second hand gesture (swiping hand down) to the park-out command. A third class (garbage model) is necessary to capture any gesture that doesn't fall in the aforementioned buckets. We achieve an accuracy of 95.02% with the selected 3DCNN, while alternatives such as LRCN, performed at a maximum accuracy of 85.44%.

The reported results were based on experiments ran in a lab environment. Once tested in a real world setting, we noticed a significant drop in accuracy, due to the varying distance and height of the user with respect to the camera. This is expected, as all of the training dataset consists of laptop/webcam collected videos which implies a limited range of distance and height of the person performing the gesture.

To overcome this limitation of our training data, we decide to leverage transfer learning and collect custom made data that would generalize our model on different backgrounds, distances and heights of the classified subject.

This end-to-end solution is developed as the vehicle for a host. Hence, multiple optimization techniques are applied to ensure the resulting model would operate in real-time on an embedded platform (NVIDIA Jetson TX2): less that 2 sec of processing for one hand gesture command, which is considered as a successful real-time implementation.

6.2 Future Work

Future work addresses some of the shortcomings of the proposed solution and how it can be expanded to cover more use cases.

In order to overcome the limitation of training data with regard to varying distances and heights, one approach is to add a pre-processing step that uses the body skeleton to locate the hand and then zoom in/out the camera or the 3D object according to how far the subject performing the hand gesture is. This would enhance the gesture classifier robustness while rendering the data augmentation step an additional enhancement.

The LRCN classifier performs poorly on the training dataset due to the different hand position in each of the 30 frames from sample to sample, which leads to feeding the LSTM with different positions of the hand in the respective indexes of the 30 frames. We suggest to normalize the full hand gesture over the 30 frames instead of normalizing the sequence (video). This would require an additional module to detect the start and end of the gesture in the video sequence.

As the vehicle for a host, a cost-effective alternative to our embedded gesture recognizer can make use of cloud deployed centralized gesture classifier. In this case, the vehicle would make an online prediction request, assuming that the car is internet-connected.

The next step towards a market-ready solution, is the deployment of our system on a vehicle. Using technologies like Bluetooth proximity and key fob smartphone app, a future optimization can reduce the person detection overhead, especially in crowded environments, by orienting the camera focus towards the car owner's direction. Hence, significantly reducing the input size of the gesture classifier.

References

[1] M Ebrahim Al-Ahdal and Md Tahir Nooritawati. Review in sign language recognition systems. In *2012 IEEE Symposium on Computers & Informatics (ISCI)*, pages 52–57. IEEE, 2012.

[2] AnrewNg. Cs229: Machine learning. `http://cs229.stanford.edu/materials/CS229-DeepLearning.pdf`, 2018.

[3] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE transactions on pattern analysis and machine intelligence*, pages 179–190, 1983.

[4] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

[5] Yoshua Bengio et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[6] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. In *Advances in neural information processing systems*, pages 153–160, 2007.

[7] Yoshua Bengio, Patrice Simard, Paolo Frasconi, et al. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[8] Oliver Brdiczka, Matthieu Langet, Jérôme Maisonnasse, and James L Crowley. Detecting human behavior models from multimodal observation in a smart home. *IEEE Transactions on automation science and engineering*, 6(4):588–597, 2009.

[9] Claude Cadoz and Marcelo M Wanderley. Gesture-music, 2000.

[10] Sylvain Calinon and Aude Billard. Stochastic gesture production and recognition model for a humanoid robot. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2769–2774. IEEE, 2004.

[11] Hyeong Soo Chang, Michael C Fu, Jiaqiao Hu, and Steven I Marcus. Google deep mind's alphago. *OR/MS Today*, 43(5):24–29, 2016.

[12] KG Manosha Chathuramali and Ranga Rodrigo. Faster human activity recognition with svm. In *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*, pages 197–203. IEEE, 2012.

[13] Xie Chen, Xunying Liu, Mark JF Gales, and Philip C Woodland. Improving the training and evaluation efficiency of recurrent neural network language models. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5401–5405. IEEE, 2015.

[14] Francois Chollet. *Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek.* MITP-Verlags GmbH & Co. KG, 2018.

[15] Dan C Ciresan, Ueli Meier, Jonathan Masci, Luca Maria Gambardella, and Jürgen Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237. Barcelona, Spain, 2011.

[16] Andrew Clark and Deshendran Moodley. A system for a hand gesture-manipulated virtual reality environment. In *Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists*, page 10. ACM, 2016.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[18] Emel Demircan, Dana Kulic, Denny Oetomo, and Mitsuhiro Hayashibe. Human movement understanding [tc spotlight]. *IEEE Robotics & Automation Magazine*, 22(3):22–24, 2015.

[19] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[20] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[21] Omid Fakourfar, Kevin Ta, Richard Tang, Scott Bateman, and Anthony Tang. Stabilized annotations for mobile remote assistance. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1548–1560. ACM, 2016.

[22] Sebastien Frizzi, Rabeb Kaabi, Moez Bouchouicha, Jean-Marc Ginoux, Eric Moreau, and Farhat Fnaiech. Convolutional neural network for video fire and smoke detection. In *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 877–882. IEEE, 2016.

[23] Kunihiko Fukushima. Neural network model for a mechanism of pattern recognition unaffected by shift in position-neocognitron. *IEICE Technical Report, A*, 62(10):658–665, 1979.

[24] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. Learning precise timing with lstm recurrent networks. *Journal of machine learning research*, 3(Aug):115–143, 2002.

[25] Scott A Green, J Geoffrey Chase, XiaoQi Chen, and Mark Billinghurst. Evaluating the augmented reality human-robot collaboration system. In *2008 15th International Conference on Mechatronics and Machine Vision in Practice*, pages 521–526. IEEE, 2008.

[26] Kirsti Grobel and Marcell Assan. Isolated sign language recognition using hidden markov models. In *Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on*, volume 1, pages 162–167. IEEE, 1997.

[27] Ye Gu, Ha Do, Yongsheng Ou, and Weihua Sheng. Human gesture recognition through a kinect sensor. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 1379–1384. IEEE, 2012.

[28] Otkrist Gupta, Dan Raviv, and Ramesh Raskar. Deep video gesture recognition using illumination invariants. *arXiv preprint arXiv:1603.06531*, 2016.

[29] Aashni Haria, Archanasri Subramanian, Nivedhitha Asokkumar, Shristi Poddar, and Jyothi S Nayak. Hand gesture recognition for human computer interaction. *Procedia Computer Science*, 115:367–374, 2017.

[30] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[31] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[32] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[33] Alan L Hodgkin and Andrew F Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of physiology*, 117(4):500–544, 1952.

[34] Jie Huang, Wengang Zhou, Houqiang Li, and Weiping Li. Sign language recognition using 3d convolutional neural networks. In *2015 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2015.

[35] Wolfgang Hürst and Casper Van Wezel. Gesture-based interaction via finger tracking for mobile augmented reality. *Multimedia Tools and Applications*, 62(1):233–258, 2013.

[36] BMW Media Information. Bmw at the consumer electronics show (ces) 2016 in las vegas. `https://www.bimmerpost.com/goodiesforyou/autoshows/ces2016/bmw-ces-2016.pdf`, 2016.

[37] Global Market Insights. Automotive gesture recognition market to exceed 13 billions by 2024. `https://www.gminsights.com/pressrelease/automotive-gesture-recognition-market`, 2019.

[38] Bongjin Jun, Inho Choi, and Daijin Kim. Local transform features and hybridization for accurate face and human detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(6):1423–1436, 2013.

[39] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[40] Ji-Hwan Kim, Nguyen Duc Thang, and Tae-Seong Kim. 3-d hand motion tracking and gesture recognition using a data glove. In *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*, pages 1013–1018. IEEE, 2009.

[41] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[42] Christof Koch, Tomaso Poggio, and Vincent Torre. Nonlinear interactions in a dendritic tree: localization, timing, and role in information processing. *Proceedings of the National Academy of Sciences*, 80(9):2799–2802, 1983.

[43] Arief Koesdwiady, Safaa M Bedawi, Chaojie Ou, and Fakhri Karray. End-to-end deep learning for driver distraction recognition. In *International Conference Image Analysis and Recognition*, pages 11–18. Springer, 2017.

[44] Ajay Kumar and David Zhang. Personal recognition using hand shape and texture. *IEEE Transactions on image processing*, 15(8):2454–2461, 2006.

[45] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. -, 2001.

[46] Ivan Laptev, Barbara Caputo, et al. Recognizing human actions: a local svm approach. In *null*, pages 32–36. IEEE, 2004.

[47] Majd Latah. Human action recognition using support vector machines and 3d convolutional neural networks. *International Journal of Advances in Intelligent Informatics*, 3(1):47–55, 2017.

[48] Ferrier Lecture. Functional architecture of macaque monkey visual cortex. *Proc. R. Soc. Lond. B*, 198:1–59, 1977.

[49] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

[50] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[51] Shuping Liu, Yu Liu, Jun Yu, and Zengfu Wang. A static hand gesture recognition algorithm based on krawtchouk moments. In *Chinese Conference on Pattern Recognition*, pages 321–330. Springer, 2014.

[52] M Ángeles Mendoza and Nicolás Pérez De La Blanca. Applying space state models in human action recognition: a comparative study. In *International Conference on Articulated Motion and Deformable Objects*, pages 53–62. Springer, 2008.

[53] ML Minsky and S Papert Perceptrons. an introduction to computational geometry cambridge ma, 1969.

[54] Sushmita Mitra and Tinku Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 37(3):311–324, 2007.

[55] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[56] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stephane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[57] Iain Murray and Ruslan R Salakhutdinov. Evaluating probabilities under high-dimensional latent variable models. In *Advances in neural information processing systems*, pages 1137–1144, 2009.

[58] Michael Nielsen, Moritz Störring, Thomas B Moeslund, and Erik Granum. A procedure for developing intuitive and ergonomic gesture interfaces for hci. In *International gesture workshop*, pages 409–420. Springer, 2003.

[59] Eshed Ohn-Bar and Mohan Manubhai Trivedi. Hand gesture recognition in real time for automotive interfaces: A multimodal vision-based approach and evaluations. *IEEE transactions on intelligent transportation systems*, 15(6):2368–2377, 2014.

[60] Kei Okada, Takashi Ogura, Atsushi Haneda, Junya Fujimoto, Fabien Gravot, and Masayuki Inaba. Humanoid motion generation system on hrp2-jsk for daily life environment. In *Mechatronics and Automation, 2005 IEEE International Conference*, volume 4, pages 1772–1777. IEEE, 2005.

[61] AR Patil and SS Subbaraman. A review on vision based hand gesture recognition approach using support vector machines. *IOSR Journal of Electronics and Communication Engineering*, pages 7–12, 2012.

[62] Ronald Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010.

[63] Alan B Poritz. Hidden markov models: A guided tour. In *Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on*, pages 7–13. IEEE, 1988.

[64] Christopher Poultney, Sumit Chopra, Yann L Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144, 2007.

[65] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[66] Siddharth S Rautaray and Anupam Agrawal. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54, 2015.

[67] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[68] Stefan Reifinger, Frank Wallhoff, Markus Ablassmeier, Tony Poitschke, and Gerhard Rigoll. Static and dynamic hand-gesture recognition for augmented reality applications. In *International Conference on Human-Computer Interaction*, pages 728–737. Springer, 2007.

[69] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[70] Yu Ren and Fengming Zhang. Hand gesture recognition based on meb-svm. In *2009 International Conference on Embedded Software and Systems*, pages 344–349. IEEE, 2009.

[71] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[72] Wolff-Michael Roth. From epistemic (ergotic) actions to scientific discourse: The bridging function of gestures. *Pragmatics & Cognition*, 11(1):141–170, 2003.

[73] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[74] Ashok Kumar Sahoo and Kiran Kumar Ravulakollu. Indian sign language recognition using skin color detection. *International Journal of Applied Engineering Research (IJAER)*, 9(20):7347–7360, 2014.

[75] Sebastien Schertenleib, Mario Gutiérrez, Frédéric Vexo, and Daniel Thalmann. Conducting a virtual orchestra. *IEEE MultiMedia*, 11(3):40–49, 2004.

[76] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. *Journal of computer and system sciences*, 50(1):132–150, 1995.

[77] Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2-3):210–220, 2006.

[78] Yale Song, David Demirdjian, and Randall Davis. Continuous body and hand gesture recognition for natural human-computer interaction. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(1):5, 2012.

[79] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[80] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on pattern analysis and machine intelligence*, 20(12):1371–1375, 1998.

[81] Gjorgji Strezoski, Dario Stojanovski, Ivica Dimitrovski, and Gjorgji Madjarov. Hand gesture recognition using deep convolutional neural networks. In *International Conference on ICT Innovations*, pages 49–58. Springer, 2016.

[82] Iikka Tapio Teivas. Video event classification using 3d convolutional neural networks. ., 2017.

[83] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[84] TwentyBN. The 20bn-jester dataset v1. `https://20bn.com/datasets/jester`, 2018.

[85] A Wabel and M Vo. A multi-modal human-computer interface: Combination of gesture and speech recognition. In *Proc. of the Int. Conf. on Human Factors in Computing Systems (CHI)*, volume 150, 1993.

[86] Liang Wang and David Suter. Recognizing human activities from silhouettes: Motion subspace and factorial discriminative graphical model. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[87] Sy Bor Wang, Ariadna Quattoni, Louis-Philippe Morency, David Demirdjian, and Trevor Darrell. Hidden conditional random fields for gesture recognition. In *null*, pages 1521–1527. IEEE, 2006.

[88] Web. How tesla and nissan's self-parking cars work. `http://fortune.com/2016/01/12/tesla-nissan-self-parking/`, 2016.

[89] Web. Bmw customer report. `https://www.autonews.com/article/20180702/RETAIL01/180709977/car-sales-on-pace-to-hit-60-year-low`, 2018.

[90] Di Wu, Lionel Pigou, Pieter-Jan Kindermans, Nam Do-Hoang Le, Ling Shao, Joni Dambre, and Jean-Marc Odobez. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(8):1583–1597, 2016.

[91] Deyou Xu. A neural network approach for hand gesture recognition in virtual reality driving training system of spg. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 519–522. IEEE, 2006.

[92] Yanan Xu and Yunhai Dai. Review of hand gesture recognition study and application. *Contemp Eng Sci*, 10(8):375–384, 2017.

[93] Shujian Yu, Robert Jenssen, and Jose C Principe. Understanding convolutional neural network training with information theory. *arXiv preprint arXiv:1804.06537*, 2018.

[94] Nico Zengeler, Thomas Kopinski, and Uwe Handmann. Hand gesture recognition in automotive human–machine interaction using depth cameras. *Sensors*, 19(1):59, 2019.

[95] Liang Zhang, Guangming Zhu, Peiyi Shen, Juan Song, Syed Afaq Shah, and Mohammed Bennamoun. Learning spatiotemporal features using 3dcnn and convolutional lstm for gesture recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3120–3128, 2017.

[96] Guangming Zhu, Liang Zhang, Peiyi Shen, and Juan Song. Multimodal gesture recognition using 3-d convolution and convolutional lstm. *IEEE Access*, 5:4517–4524, 2017.