Universidad de Alcalá

Ph.D. Program in Information and Communications Technologies

Predicting Pedestrian Crossing Intentions Using Contextual Information

Ph.D. Thesis Presented by
Javier Lorenzo Díaz

2022

Universidad de Alcalá

Ph.D. Program in Information and Communications Technologies

Predicting Pedestrian Crossing Intentions Using Contextual Information

PhD. Thesis Presented by

Javier Lorenzo Díaz

Advisors

Dr. Miguel Ángel Sotelo Vázquez

Dr. Ignacio Parra Alonso

Alcalá de Henares, 27th April 2022

"You miss 100% of the shots you don't take. - Wayne Gretzky" - Michael Scott

Agradecimientos

¡Sanseacabó! La verdad es que pensaba que no llegaría este instante pero aquí me encuentro, tarde y escribiendo este texto que ya he cambiado tropecientas veces. Aunque sea solo por hoy, voy a aplicar la técnica de Agapito y a ahorrar en sueño que la situación lo merece.

Haciendo una retrospectiva, me deja bastante asombrado lo rápido que se me han pasado estos años en el grupo. Cualquiera diría que empecé mi andadura en 2017[1]. Han sido cuatro años dignos de recordar. Venga 5, vamos a contar el año que fui cuenquito. Creo que de tanto escribir la tesis en inglés se me ha quedado lo de escribir frases cortas. Vamos a cambiar el chip a modo viejo castellano emotivo, porque la situación se lo merece.

Miguel Ángel e Ignacio, quería agradeceros enormemente que me permitiérais hacer el doctorado en el grupo. Creo que sin vosotros no lo habría conseguido sinceramente y vuestro apoyo ha sido fundamental, sobre todo ayudándome a centrar mi mente en lo realmente importante y filtrando todas las ideas que se me iban ocurriendo. El doctorado me ha brindado una preparación muy difícil de obtener de otra forma y como digo, vosotros habéis sido los responsables de ello.

Vamos a seguir una estrategia descendente y a proseguir con un agradecimiento general del pasillo-de-arriba. Carlota, David y Javi, gracias por todo. Creo que no hemos tenido tanta relación como hubiese querido, ni hemos trabajado juntos mucho, pero os deseo lo mejor. Gracias Iván por lo mismo y además por las risas que nos hemos echado juntos. Gracias Noe, que aunque estés arriba creo que tu corazón pertenece al reino de los mortales (a.k.a E202). Gracias por reírte conmigo en mis momentos de mayor locura en el laboratorio. No me olvido de tus mágicos postres que junto a los helados en verano me aportaban el azúcar justo y necesario. Gracias Nacho, que sepas que al final no voy a ser JLo23 (pero casi). Intenté aprender a jugar al tenis, pero me frustré casi antes de empezar jajaja. Por último gracias Augusto, espero quedar pronto para irnos con la bici (hincharé la rueda cuando ya deposite, ¡prometido!).

Antes de comenzar con el laboratorio de abajo, especial mención a Llama, que aunque ya no esté en el grupo, como si lo estuviera. Gracias por trabajar conmigo y ser tan buen compañero y amigo.

Ahora vamos a pasar al conocido en otras ilustres tesis como "núcleo duro" o "tutores de trinchera". Yo los llamaré simplemente los "fans del batidos": Álvaro $\times 2$, Héctor y Rubén[2]. Álvaro H., gracias por ser mi compañero de mesa durante todos estos años y por ir virtualmente a las Vegas conmigo. Hemos coincidido en finalización de la memoria y casi

[1] Entre dólares para honrar al dios LaTeX

[2] Me abstengo de motes

casi depositamos a la vez. Cuando lo hagamos vamos a celebrarlo con una salva de Nerfs. Álvaro Q., gracias por tus recitales de silbidos, te dejamos al cargo del equipo predoc, con Antoine[3], Sandra y Sergio. Un consejo para todos y en especial para el primer nombrado: evitad la procrastinación y no desesperéis. Y ahora pasamos al bueno de Héctor, he tenido la suerte de continuar trabajando contigo en la empresa y espero que siga siendo así mucho tiempo. Gracias a ti también me he sacado un doble grado de muchos temas como la energía y las finanzas. Prometo dejar de comprarme la misma ropa de R&M. Por último, Rubén. Por poco te quito el puesto de miembro más longevo de INVETT pero no cayó esa breba. Creo que te debo unos sashimis de Soria para celebrar. Gracias por toda tu ayuda, tus ideas y tu mentoring personalizado, sobre todo de todos esos temas de hardware que tanto gustan en el mundo de industriales (y bricolaje). Gracias por los buenos paseos por el Sahagún cuando eramos vecinos, la verdad es que los echo bastante de menos. Me ayudaban cuando estaba perdidu.

Gracias al resto de miembros antiguos de INVETT y a Edu y a Rober.

Andrea, te quiero, no tengo palabras para expresar todo lo que has hecho por mí. Hemos vivido una pandemia encerrados, una Filomena y aún así, no concibo separarme de ti. Sin ti no soy nada y te quiero dar las gracias especialmente por aguantarme estos últimos meses, que se que he estado bastante insufrible. Gracias por animarme y por quererme como soy.

Finalmente, quiero agradecer a Edu, Fina, a mis padres y a mi hermana por todo el apoyo que me han dado con la tesis, aunque ninguno sepa 100% de que va. Quiero agradecer enormemente a mis padres todo lo que han hecho por mi. Han dado mucho para que yo esté ahora mismo aquí y solo puedo prometerles que seguiré esforzándome para que haya valido la pena. Os quiero.

[3] Joey Joe Joe Junior Shabadoo

Resumen

El entorno urbano es uno de los escenarios más complejos para un vehículo autónomo, ya que lo comparte con otros tipos de usuarios conocidos como usuarios vulnerables de la carretera, con los peatones como mayor representante. Estos usuarios se caracterizan por su gran dinamicidad. A pesar del gran número de interacciones entre vehículos y peatones, la seguridad de estos últimos no ha aumentado al mismo ritmo que la de los ocupantes de los vehículos. Por esta razón, es necesario abordar este problema. Una posible estrategia estaría basada en conseguir que los vehículos anticipen el comportamiento de los peatones para minimizar situaciones de riesgo, especialmente presentes en el momento de cruce.

El objetivo de esta tesis doctoral es alcanzar dicha anticipación mediante el desarrollo de técnicas de predicción de la acción de cruce de peatones basadas en aprendizaje profundo.

Previo al diseño e implementación de los sistemas de predicción, se ha desarrollado un sistema de clasificación con el objetivo de discernir a los peatones involucrados en la escena vial. El sistema, basado en redes neuronales convolucionales, ha sido entrenado y validado con un conjunto de datos personalizado. Dicho conjunto se ha construido a partir de varios conjuntos existentes y aumentado mediante la inclusión de imágenes obtenidas de internet. Este paso previo a la anticipación permitiría reducir el procesamiento innecesario dentro del sistema de percepción del vehículo.

Tras este paso, se han desarrollado dos sistemas como propuesta para abordar el problema de predicción.

El primer sistema, basado en redes convolucionales y recurrentes, obtiene una predicción a corto plazo de la acción de cruce realizada un segundo en el futuro. La información de entrada al modelo está basada principalmente en imagen, que permite aportar contexto adicional del peatón. Además, el uso de otras variables relacionadas con el peatón junto con mejoras en la arquitectura, permiten mejorar considerablemente los resultados en el conjunto de datos JAAD.

El segundo sistema se basa en una arquitectura *end-to-end* basado en la combinación de redes neuronales convolucionales tridimensionales y/o el codificador de la arquitectura *Transformer*. En este modelo, a diferencia del anterior, la mayoría de las mejoras están centradas en transformaciones de los datos de entrada. Tras analizar dichas mejoras, una serie de modelos se han evaluado y comparado con otros métodos utilizando tanto el conjunto de datos JAAD como PIE. Los resultados obtenidos han conseguido liderar el estado del arte, validando la arquitectura propuesta.

Palabras clave: Aprendizaje profundo, Predicción, Redes Neuronales Recurrentes, Redes Neuronales Convolucionales, Transformer.

Abstract

The urban environment is one of the most complex scenarios for an autonomous vehicle, as it is shared with other types of users known as vulnerable road users, with pedestrians as their principal representative. These users are characterized by their great dynamicity. Despite a large number of interactions between vehicles and pedestrians, the safety of pedestrians has not increased at the same rate as that of vehicle occupants. For this reason, it is necessary to address this problem. One possible strategy would be anticipating pedestrian behavior to minimize risky situations, especially during the crossing.

The objective of this doctoral thesis is to achieve such anticipation through the development of crosswalk action prediction techniques based on deep learning.

Before the design and implementation of the prediction systems, a classification system has been developed to discern the pedestrians involved in the road scene. The system, based on convolutional neural networks, has been trained and validated with a customized dataset. This set has been built from several existing sets and augmented by including images obtained from the Internet. This pre-anticipation step would reduce unnecessary processing within the vehicle perception system.

After this step, two systems have been developed as a proposal to solve the prediction problem.

The first system is composed of convolutional and recurrent encoder networks. It obtains a short-term prediction of the crossing action performed one second in the future. The input information to the model is mainly image-based, which provides additional pedestrian context. In addition, the use of pedestrian-related variables and architectural improvements allows better results on the JAAD dataset.

The second system is an end-to-end architecture based on the combination of three-dimensional convolutional neural networks and/or the Transformer architecture encoder. In this model, most of the proposed and investigated improvements are focused on transformations of the input data. After an extensive set of individual tests, several models have been trained, evaluated, and compared with other methods using both JAAD and PIE datasets. Obtained results are among the best state-of-the-art models, validating the proposed architecture.

Keywords: Deep Learning, Prediction, Recurrent Neural Networks, Convolutional Neural networks, Transformer.

Contents

List of Figures

List of Tables

List of Acronyms

2D	two-dimensional.
3D	three-dimensional.
ACC	Adaptive Cruise Control.
AD	Autonomous Driving.
ADAS	Advanced Driver Assistance Systems.
AE	Auto Encoder.
AEB	Autonomous Emergency Braking.
AP	Average Precision.
AUC	Area under the ROC curve.
AV	Autonomous Vehicle.
B-GPDM	Balanced Gaussian Process Dynamical Model.
BCE	Binary Cross-Entropy.
BiGRU	Bidirectional Gated Recurrent Unit.
BiLSTM	Bidirectional Long Short Term Memory.
C-ITS	Cooperative Intelligent Transport Systems.
CNN	Convolutional Neural Network.
COCO	Common Objects in Context.
ConvLSTM	Convolutional long short-term memory.
CRF	Conditional Random Field.
CUDA	Compute Unified Device Architecture.
DBN	Dynamic Bayesian Network.
EKF	Extended Kalman Filter.
ELK	Emergency Lane Keeping.
EU	European Union.
Euro NCAP	European New Car Assessment Programme.
FDR	False Discovery Rate.
FLOPS	Floating Point Operations per Second.

FNR	False Negative Rate.
FOV	Field of View.
fps	frames per second.
GCN	graph convolutional neural network.
GPDM	Gaussian Process Dynamical Model.
GPS	global positioning system.
GPU	Graphics Processing Unit.
GRU	gated recurrent unit.
GTA	Grand Theft Auto.
HMM	Hidden Markov Model.
HOG	Histogram of Oriented Gradients.
IMM	Interacting Multiple Model.
IMU	inertial measurement unit.
ITS	Intelligent Transport Systems.
IV	Intelligent Vehicle.
JAAD	Joint Attention in Autonomous Driving.
KF	Kalman Filter.
LDCRF	Latent-Dynamic Conditional Random Fields.
LiDAR	Light Detection And Ranging.
LSTM	long short-term memory.
MLP	Multi-layer perceptron.
NLP	Natural Language Processing.
OBD	On-Board Diagnostics.
PCA	Principal Component Analysis.
PF	Particle Filter.
PHTM	probabilistic hierarchical trajectory matching.
PIE	Pedestrian Intention Estimation.
PPV	Positive Predictive Value.
PTW	Powered Two Wheeler.
px	pixel.
RF	Random Forest.
RNN	recurrent neural network.

ROAD	ROad event Awareness in Autonomous Driving Dataset.
ROC	receiver operating characteristic.
SDG	Sustainable development goal.
SLDS	Switching Linear Dynamical System.
STIP	Stanford-TRI Intent Prediction.
SVM	Support Vector Machines.
TITAN	Trajectory Inference using Targeted Action priors Network.
TPR	True Positive Rate.
TTC	Time To Cross.
TTE	Time To Event.
UN	United Nations.
VIENA	Virtual Environment for Action Anticipation.
VRNN	variational recurrent neural network.
VRU	Vulnerable Road User.
WHO	World Health Organization.

Chapter 1

Introduction

1.1 Context

Ever since the invention of the car, mobility has evolved worldwide, becoming a fundamental part of humanity's progress, bringing noticeable socioeconomic improvements.

However, this mobile revolution is a double-edged sword. In terms of data, according to the World Health Organization (WHO) report on road safety [Org+18], 1.35 million people died in 2016 in traffic crashes, leaving between 20 and 50 million people non-fatal injured. It is troubling and devastating that this is the leading cause of death for children and young people between 5 and 29 years of age and the eighth leading cause of death worldwide. Furthermore, the global economy is also affected by an estimated 3% loss in each country's gross domestic product. As concluded on [Wor20], this is an unacceptable price to pay for mobility.

The rate of death relative to the world's population has stabilized and declined relative to the number of motor vehicles worldwide in recent years. This means that the problem is not worsening, but also means that insufficient progress is being made.

Seeking the reduction of road deaths and injuries, WHO took action establishing the objective of halving the number by 2020 in the Sustainable development goal (SDG) 3.6 target [Org]. To fulfill its goals, five recommended pillars of activities are proposed [Wor20]:

- **Road safety management**: through the creation of legal instruments for road safety.
- **Safer roads and mobility**: by increasing safety inspections, improving existing roads, and prioritizing safety in new ones.
- **Safer vehicles**: with the investment and use of Advanced Driver Assistance Systems (ADAS) e.g., Autonomous Emergency Braking (AEB), Adaptive Cruise Control (ACC), Emergency Lane Keeping (ELK), and the creation of car assessment programmes, such as European New Car Assessment Programme (Euro NCAP).
- **Safer road users**: through regulation of speed limits, drink and drug driving penalties, mandatory use of helmets and seat-belt, recommendations for decreasing distracted driving. In summary, improve road user behavior.

- **Improving post-crash care and response**: by educating bystanders and ensuring timely care for the injured.

The European Union (EU) roads are on average the safest in the world. One of the reasons is the proposal of similar activities and initiatives as part of their **Road Safety Programme** [Eur10]. In 2010, The European Commission established a target similar to that of the WHO to halve the number of road fatalities by 2020. In the period from 2010 to 2014, there was a reduction in line with the established objective. However, this reduction suddenly began to slow down, causing the number of deaths recorded to be far above the corresponding target by 2019, exceeding it by almost 8,000 (see fig. 1.1 for a graphical detail of road death evolution from the beginning of this century).

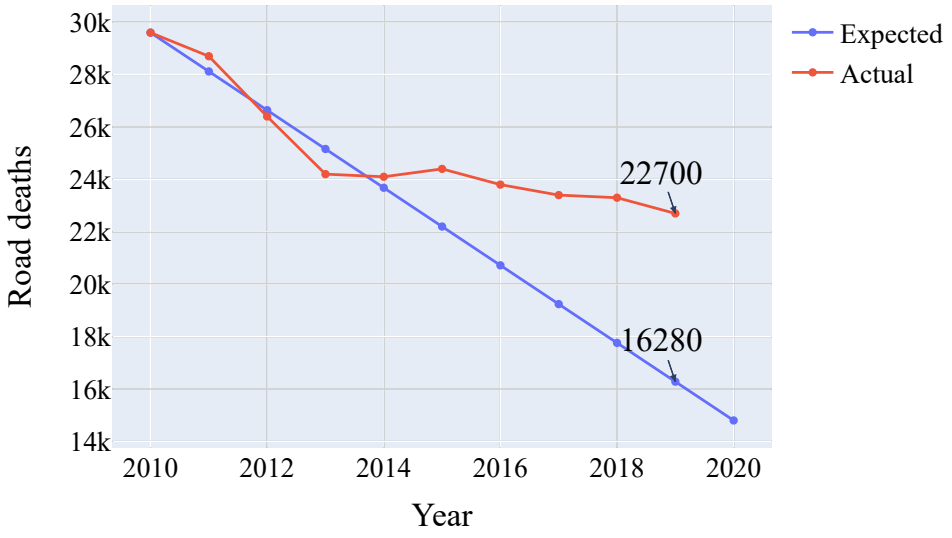


Figure 1.1: Road deaths evolution in Europe (2010–2020). The reduction accomplished in the first decade has been notorious. However, after the first third of the 2010s, the reduction has nearly stopped[1].

Among all road participants, Vulnerable Road Users (VRUs) represent more than half of road deaths worldwide. This group is composed mainly of pedestrians, cyclists, and motorcyclists, who are less protected than car occupants. Among them, the most vulnerable ones are pedestrians and cyclists. They account for 29% of all road deaths across the EU [AJ20]. One concerning fact about this group is that the reduction in road deaths is not declining as fast as those of vehicle occupants, keeping a similar proportion with a slight increase of 1% in 2018 compared to 2010. Focusing only on pedestrian mortality, it has been reduced in the EU by 19% from 2010 to 2018, while the reduction experimented by vehicle occupants is nearly 24%, excluding Powered Two Wheeler (PTW) users. The progress of this reduction, year by year, is shown in fig. 1.2.

[1]Data obtained from https://ec.europa.eu/transport/road_safety/document/download/a7cbb0ef-df21-4e75-ba91-41af24195fc3_en

Over the past few decades, there has been a global mass migration from rural to urban areas. United Nations (UN) estimated that in 2018 over 55% of the world population (≈ 4.1 billion) were living in urban areas [Uni18].

Cities and towns are home to 72% of the population of the EU. Urban roads are the scenario of 38% of road deaths in 2017. This proportion has decreased on average 2.2% from 2007 to 2015. In contrast, rural roads proportion has decreased on average 3.9%. Approximately 70% of urban road deaths are VRUs, with a majority of them being pedestrians. 99% of pedestrian deaths are caused by a collision with a motor vehicle [AJ19].

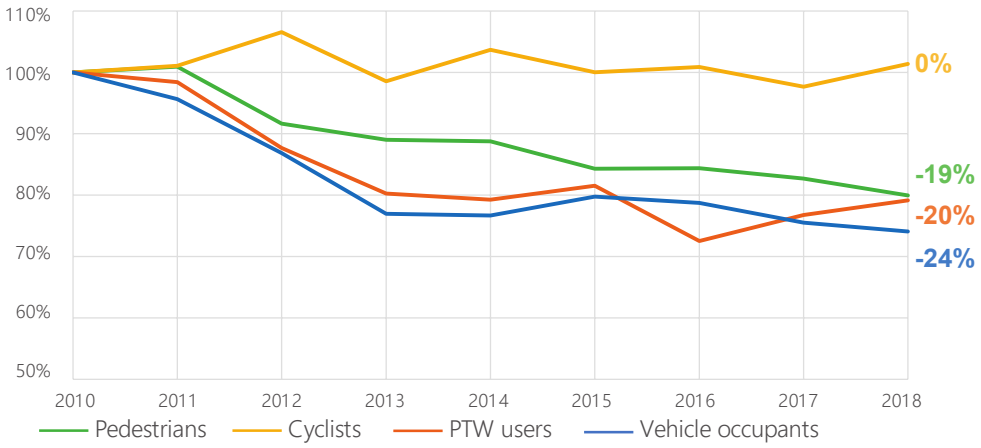


Figure 1.2: Relative proportion reduction in road mortality for the different road agents in EU (2010–2018) [AJ20].

1.2 Motivation

After reporting the high degree of urbanization worldwide and the noticeable vulnerability situation suffered by pedestrians in the city, the purpose of this work is the minimization of accidents involving pedestrians by tackling their first cause: motor vehicles, with a special focus on cars.

According to [MGLW16], human error is the cause of 93.5% of all road accidents. Most of the measures proposed by organizations such as the WHO and the EU (see section 1.1) aim to reduce human failure through mechanisms such as law enforcement, education, and safety technology development. In addition to these policies, Intelligent Transport Systems (ITS) are technologies that introduce awareness and automation to the road environment or the vehicles. In the case of vehicles, Cooperative Intelligent Transport Systems (C-ITS) focus on the communication between road agents and can help to reduce human error (e.g., pedestrian detection system in crosswalks to warn near vehicles and try to avoid collisions). However, these systems are complex and far from commercial production. Another important aspect of the work in ITS is the development of Autonomous Vehicles (AVs), which are vehicles that can perform tasks without human intervention. Nowadays, this topic is of particular interest in the industry and research community, where it has experienced significant progress over the past few decades.

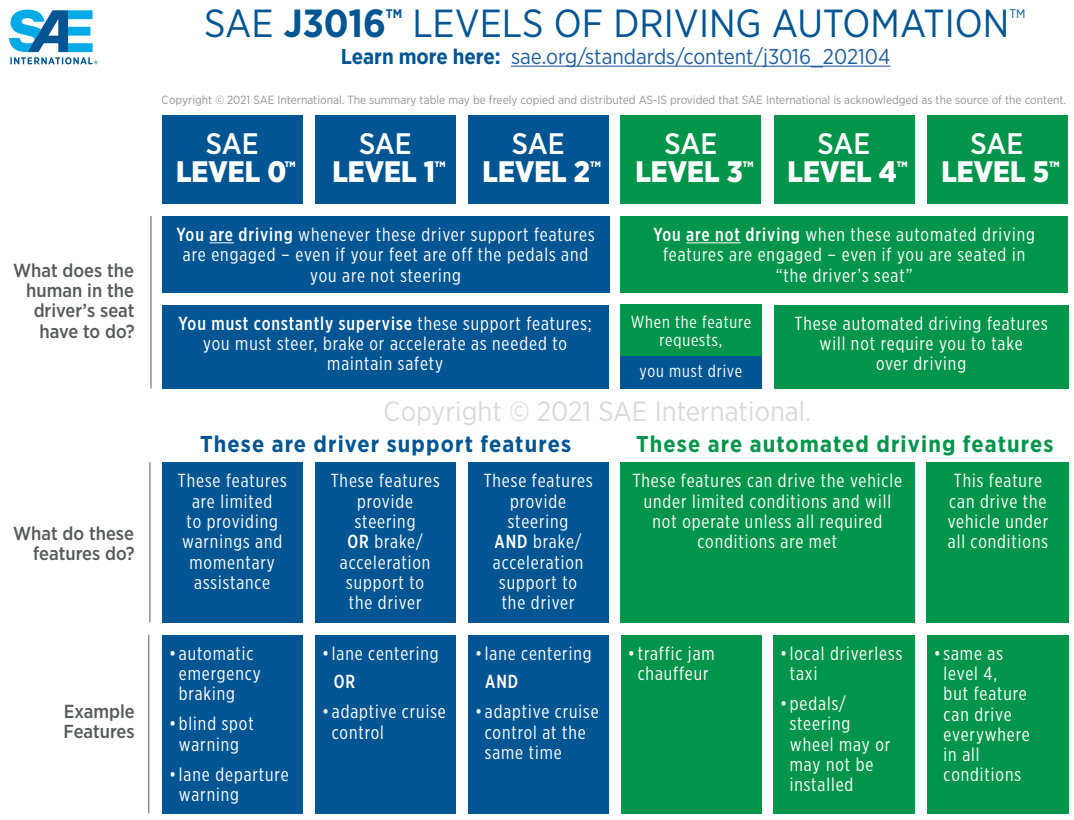


Figure 1.3: Automation level classification proposed in standard SAE J3016.

Autonomy is defined in this field as the capacity of making decisions independently of human interference [TL18]. Even though it is a simple definition, it hides a high level of complexity and it is nowadays not fully solved.

A more realistic definition of this problem is treating it as an automation classification, ranging from human-driven to fully automated (autonomous) systems. This classification was proposed in 2014 by SAE International in their standard J3016. It distinguishes among six levels of automation, as shown in fig. 1.3. On the first three levels, a human driver must supervise all the driving processes. On the last two levels, there is no need for the human to drive. The fourth level is defined as a bridge between both milestones.

Development on AV research was accelerated through the DARPA Grand Challenges US Program in 2004. Several companies have made considerable investments in the development of this technology. Volvo introduced its full autonomous test vehicle in 2017 and recently announced a deeper collaboration with Uber (see fig. 1.4a for a visual of the prototype). Waymo AVs fleet (fig. 1.4b), a subsidiary of Alphabet Inc, the parent company of Google, has completed more than three million kilometers of driving. All new Tesla cars have different levels of automation and the company claims that new models

(a) Volvo XC90 SUV with Uber's self-driving system[2].

(b) Waymo vehicles[3].

(c) Interior of a Tesla vehicle showing Autopilot capabilities[4].

Figure 1.4: Examples of automated solutions of different companies.

contain the hardware needed for level 5 automation (in fig. 1.4c, you can see a capture of a video demo showing self-driving capabilities).

AVs have several advantages compared to manual driving. In addition to the substantial improvements in safety resulting from the almost total disappearance of the human factor, the development of the autonomous car allows:

- Better driving efficiency: autonomous vehicles can be programmed to reduce "heavy-footed" driving with high acceleration and braking actions, which leads to a reduction in vehicle emissions (better fuel economy).
- Time and economic improvements at micro and macro levels: thanks to the autonomy and the level of connectivity and knowledge of the surrounding, there will be a dramatic reduction in traffic jams, saving a considerable amount of time (according to a study, they could free up as much as 50 minutes each day[5]). This reduction will also affect directly economic savings from companies and individuals, with a consequent improvement in productivity.

The current level of autonomy in commercial solutions is at a maximum level of 3. Several advances in the field throughout the last decades have been materialized in different ADAS. To a greater or lesser extent, these systems are present in most of today's vehicle models.

As explained in the previous section, concerning pedestrian safety, car-to-pedestrian impacts are among the most frequent accidents occurring in urban environments due to the high level and intensity of interactions between vehicles and pedestrians. Car manufacturers are developing pedestrian AEB systems [NCA20] (fig. 1.5) to support the driver in avoiding (by autonomous braking) or mitigating such crashes. These systems, which depend on the accuracy of the deployed pedestrian detector, can be much more effective if they can anticipate the intention or action of pedestrians to cross, as an experienced driver does.

Future vehicle systems for active pedestrian safety will require an accurate analysis of the developing traffic situation in addition to a high recognition performance [KG14].

[2] https://bit.ly/volvo-uber
[3] Source Waymo: https://waymo.com/press/
[4] https://vimeo.com/192179726
[5] Source: https://mck.co/3HlZplW

Figure 1.5: Pedestrian AEB system example case. Source: https://bit.ly/aebped_ncap

Even when a collision cannot be avoided, an earlier brake action can drastically reduce the danger of the situation. A pedestrian hit by a car traveling at $65km/h$ is four times more likely to be killed compared with a car traveling at $50km/h$ [Van21]. Quantitatively, initiating the braking action $0.16s$ earlier at a time to collision of $0.66s$ reduces the risk of serious injury from 50% to 35% with an initial vehicle speed of $50km/h$ [KG14]. The anticipation of the pedestrian crossing behavior can lead to an even earlier braking action which can help mitigate the effects of the crash. Furthermore, in cases where braking action is not possible (excessive speed), anticipation allows a maneouver to try to avoid the crash.

However, understanding pedestrian behavior is a challenging task. Instead of designing anticipation systems, AVs can be designed to behave conservatively (i.e., driving slower than the limit) and minimize complex interactions with other road users [Ras20]. However, this negatively affects the traffic flow and the passengers' experience in traveling companies such as Waymo, whose AVs can double the travel time in some situations. Without anticipation, AVs must rely entirely on the reaction time of AEB and other ADAS, being subject to malicious behaviors (e.g., performed by drunk pedestrians or pedestrians joking around).

Anticipation is one of the biggest steps on the road to autonomy and can bring together improvements at both economic and safety levels, by combining better traffic flow and improved safety [Ras20].

Advances and reduced costs in storage and computing power have allowed an incredible evolution of deep learning techniques over the last decade. Graphics Processing Unit (GPU) processors have become the defacto platform for training and serving real-time solutions for a large variety of complex vision problems, such as object classification, object detection (i.e., localizing an object and its class in multidimensional data), and different types of scene segmentation, such as semantic (pixel-level classification, in the case of the image-based case) and instance (object's detection and pixel-level classification

of its pixels). See fig. 1.6 for visual examples of each of the previously stated problems.

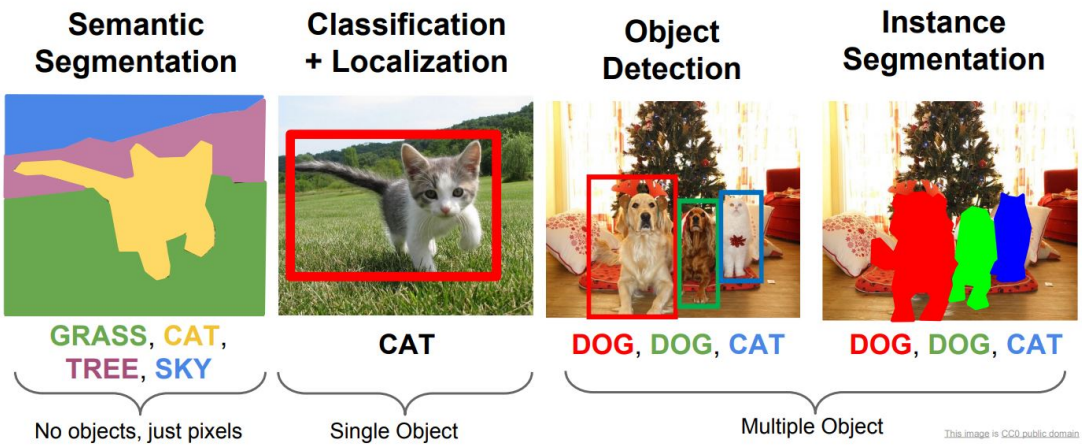


Figure 1.6: Visual examples of different computer vision tasks. Source: https://stanford.io/3hsTK2U

Anticipation or forecast tasks have been also studied inside the deep learning field, and gained popularity in recent years due to their application in several fields (e.g., finance, language modeling, and robotics). AVs' research has been focused on perception problems similar to the ones in fig. 1.6 and others with a special purpose (e.g., pedestrian detection and tracking). Their purpose is the understanding of the road environment, both road agents and infrastructure. While these systems have achieved an astonishing performance, they are limited to the spatial plane and are not aware of the temporality of data, crucial for anticipating the crossing behavior of pedestrians. To incorporate temporality into the road environment perception, there are multiple models available in the literature, such as recurrent neural networks (RNNs), 3D Convolutional Neural Networks (CNNs) and the more recent Transformer architecture [VSP+17]. Trajectory forecasting has evolved recently thanks to these temporal architectures. In the pedestrian case, it is possible to implicitly infer behavior with this method. However, a global understanding of the situation through detailed maps is needed to localize the target pedestrian in the road environment and check if the predicted future path crosses the road. This understanding is difficult to scale since it needs constantly updated maps which demands a high quantity of time, memory, and processing power. Also, it needs an expensive set of sensors, such as Light Detection And Ranging (LiDAR) and stereo cameras which must be calibrated. To avoid this constraint, an alternative for future crossing action behavior can be the use of raw image data from cheaper monocular camera sensors and other data derived, both at contextual and pedestrian level. While this alternative is easier to scale, it also includes difficulties regarding the higher dimensionality of the data used which makes it harder for the model to achieve generalization in data scarcity and imbalance scenarios. In this work, this alternative to trajectory-based systems is explored, with the development of different novel architectures and strategies to deal with data problems.

1.3 Outline

This document is organized as follows:

After the introduction of the problem in chapter 1 an analysis of previous work on pedestrian crossing behavior is conducted in chapter 2, detailing techniques and datasets used for the task, highlighting the objectives of this work.

Chapter 3, the first work carried out in the course of this thesis is presented. Before dealing with crossing anticipation, the AV needs to distinguish between the different road agents and also filter the ones which are not relevant or at potential risk. For this reason, a dataset was built for this task to prove if deep learning methods can correctly distinguish them.

Chapter 4 details the first system developed for pedestrian crossing action prediction in this Ph.D. dissertation, based on the combination of RNN and CNN modules. Different modifications were made to the architecture to improve the performance.

Chapter 5 describes the final system developed in this work. Composed of a combination of different transformer architectures and spatio-temporal CNNs, the model was trained end-to-end, focusing experimentation on data (data-centric). The system was finally evaluated in a benchmark with other previous systems, validating the transformer architecture as an alternative to RNNs.

Finally, chapter 6 contains a general discussion along with the conclusions and main contributions of the work.

Chapter 2

State of the Art

The pedestrian safety field has relied mainly on road-safety education, safer infrastructure construction, safer vehicles, and regulation improvements [Org+18]. However, pedestrians' fatality rate decreases lower than the average fatality rate [Eur18]. Nowadays, Advanced Driver Assistance Systems (ADAS) are the commercial solutions focused on improving road safety. Autonomous Emergency Braking (AEB) solutions[1] can improve pedestrian safety, but, as the name indicates, it is a system which is activated before just before a collision. According to [KG14], minimal anticipation or prediction of the crossing behavior of pedestrians can lead to a substantial lowering in their injury risk. By combining both anticipation and emergency action provided by AEB, safer autonomous driving technologies can be developed, improving the safety of pedestrians without disrupting traffic flow [Ras20]. For this reason, the focus of this work is the search for the required anticipation through the prediction of the crossing behavior of the pedestrian. To approximate the anticipation of the future behavior of pedestrians we have selected the task of video-based crossing action prediction, an specialization of the video action recognition line of research.

2.1 Pedestrian crossing action prediction

A human action refers to the process of doing something to deal with a particular problem or goal. Video action recognition aims to teach a machine to recognize actions from videos. Also known as video action classification or prediction, it has been one of the most representative tasks in **video understanding** [ZLL+20]. With the increase in computation capability and the previously-commented deep learning emergence, the last decade has witnessed great advances. The understanding of human actions helps in many applications, such as surveillance, robotics, video retrieval, and autonomous driving.

Inside this task, there are two possible categories based on the temporal point of prediction. The first group, **action anticipation** methods, observes the video data for a specific time interval and tries to infer the action performed in a future time step. The second group, usually known as **early action prediction** methods, tries to infer the current

[1] https://www.mobileye.com/blog/how-adas-and-data-can-lead-the-way-in-pedestrian-safety/

(a) Action anticipation.

(b) Early action prediction.

Figure 2.1: Types of action prediction based on the temporal point predicted. f_n, A_n represents the input frame and the output action at time $t = n$, respectively. h is the number of time units predicted in the future from N.

action performed on the video [RKT20]. A visual representation of the two strategies is shown in fig. 2.1.

In the scope of this thesis, pedestrian crossing intention is approximated through **action anticipation** systems based on different deep learning methods. **Intention** here refers to the determination to act in a certain way, influenced by an indeterminate number of factors, both internal and external. In the following chapters (chapter 4 and chapter 5), the presented methods approximate it through the prediction of future crossing action. Video action classification methods represent good base models, which are extended to future horizons as part of this work, trying to approximate this intention in a short-term time horizon.

Pedestrian crossing behavior prediction is defined in this work as a **binary classification problem**, where the model tries to classify if the target pedestrian is going to cross in a future short-term horizon ($T \leq 2s$). The input to the model is based on both features extracted from the pedestrian (e.g., bounding box coordinates, pose, gaze direction, body orientation, motion state) and context, which could be represented as relative measures such as the distance to the curb. However, its main source is image data, from where contextual features are extracted in an unsupervised learning process since the model is not trained to learn specific context features, but the ones that find most useful for the task during the optimization process.

Throughout this chapter, we present an overview of techniques and datasets in the literature focused on pedestrian crossing action prediction. In section 2.2, a detailed study of datasets for the task is presented, highlighting their limitations for the task. The methods available in the literature are discussed in section 2.3 with their categorization based on different features. In section 2.4 a recent benchmark focused on the task is detailed since it is used extensively in chapter 5. Section 2.5 contains several conclusions extracted from the previous analysis of datasets and methods. Finally, in section 2.6 the main objectives pursued in this thesis are detailed.

2.2 Datasets

The anticipation and understanding of human behavior have gained interest over the last decade. Pattern-based methods have arisen exponentially since 2018 [RPH+20], trying to estimate trajectory from past kinematic data. Additional information such as social interaction, semantic maps, and 3D information is also used to improve the forecasting [RKT20] [RYL+20]. These methods can provide an implicit prediction of the crossing action of pedestrians if road boundaries are available in the still image (e.g., by using

the semantic maps mentioned before). Furthermore, the top-view perspective adopted in most datasets, through stationary or aerial cameras, allows better annotations, lower occlusion levels, and better kinematic estimation. See fig. 2.2 for visual examples of both types of datasets.

However, in the field of Intelligent Vehicles (IVs), sensors are usually placed on or inside the ego-vehicle. For this reason, the extraction of the previous data is noisier, being affected by ego-vehicle motion, perspective occlusion, and illumination changes. This noise makes labeling more difficult and expensive. Data available from this perspective was less common than in the top-view case. However, in the last decade, the interest in IV field has enabled the emergence of new datasets, highly structured and detailed, with a wide range of sensors available. These datasets are labeled for one or more perception tasks (e.g., object detection, image segmentation, localization, mapping). The lack of temporal information in most of them makes it difficult to gather pedestrian tracks and behavioral data [RKKT19]. In this section, the focus is placed on the temporal data, detailing the available datasets for the task of pedestrian behavior understanding.

(a) ZARA 01 static scenario. Obtained from https://repo.vi-seem.eu/handle/21.15102/VISEEM-316.

(b) Stanford Drone dataset. Obtained from https://cvgl.stanford.edu/projects/uav_data/.

Figure 2.2: Visual examples of top-view scenarios. On fig. 2.2a a stationary security camera is used to record behavior of people in the street. On fig. 2.2b, an aerial camera from a drone is used to film a roundabout surroundings.

2.2.1 Daimler Pedestrian Dataset

Daimler Pedestrian Path Prediction Benchmark dataset [SG13] was, to the extent of our knowledge, the first one to tackle pedestrian path prediction from the ego-vehicle perspective. Contains 68 pedestrian sequences, 13 recorded from a stationary vehicle, and 55 from a moving vehicle at speeds from 20 to 39km/h, using actors instructed about each sequence. Four different male pedestrians are used in eight different locations. Sequences' length varies from a minimum of 2.53s to a maximum of 13.27s with an average length of 7.15s. Pedestrian occlusion level is generally low. Frames are stored as stereo pairs in

grayscale format, with a resolution of 1176×640 px with a framerate of 16 fps. Calibration data, vehicle speed, and yaw rate are also provided.

The ground truth is composed of trajectory data extracted with stereo vision, giving the pedestrian location (assuming static ground plane), without height information. Manually labeled 2D bounding boxes are provided along with detections from a Histogram of Oriented Gradients (HOG)-based linear Support Vector Machines (SVM) detector. Each sequence is labeled with a unique pedestrian motion tag: crossing, stopping, starting, and bending-in. In the first two tags, the pedestrian walks laterally towards the road, **crossing** it or **stopping** abruptly respectively. In the third case, the sequence begins with the pedestrian waiting at the curbside, ending with him **starting** to walk laterally towards the road. Finally, in the fourth case, the pedestrian walks alongside the street until he **bends in** and begins to cross. In fig. 2.3, a visual example of each motion case is provided.

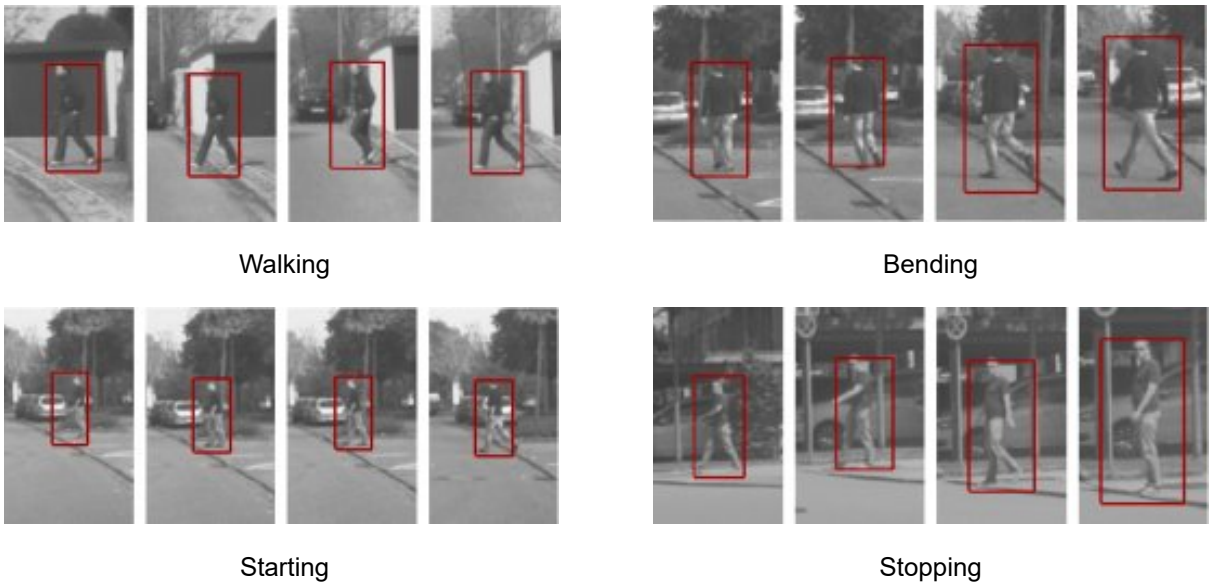


Figure 2.3: Examples of each motion type in dataset, obtained from [SG13].

An update to the annotated data was made in 2014 in [KSFG14] (ECCV'14 version). The update affected some sequences from the original dataset in addition to new ones (from which image data **is not provided**). Discrete head orientation, with 16 clockwise increasing orientation angles is provided. Event tags and Time To Event (TTE) in frames are also provided, being 0 when the last foot is set at the curb in non-crossing pedestrians and the moment just before entering the road in the case of the crossing ones. The minimum distance between the pedestrian and the vehicle and the distance of the pedestrian to the curb are also provided, along with the state of movement (standing or walking) and a group of binary flags extracted from the previous data.

This dataset was at the time a good starting point for the study of pedestrian behavior, establishing interesting variables that could be extracted from sensor data. Although in recent years it has become obsolete in size and amount of labeling data with respect to

Figure 2.4: Frames extracted from three sequences in ROad event Awareness in Autonomous Driving Dataset (ROAD) dataset. A high diversity of weather and time-of-day conditions can be observed, even with scenes where the human eye struggles to see (section 2.2.2 and section 2.2.2).

other datasets in this list, it is still one of the most complete ones, due to the combination of 2D and 3D annotations.

2.2.2 ROad event Awareness in Autonomous Driving Dataset

ROAD [SAM+21] is a naturalistic dataset, released in 2021, focused on detection and anticipation of actions performed by road agents (e.g., pedestrians, vehicles and cyclists). It consists of 22 sequences with an average length of 8 minutes, resulting in 122K annotated frames with resolution 1280×960. The sequences belonging to Oxford Robot Car Dataset [MPLN17], recorded following a route through Oxford over 100 from May 2014 to November 2015. The framerate of the original videos varies from 11 to 16 fps. To homogenize this variable before the labeling process, video data is encoded at the rate of 12 fps. Timestamps from original videos are retained to allow the synchronization with Light Detection And Ranging (LiDAR) and global positioning system (GPS) data.

Videos were selected manually in order to maximize the diversity of weather and time of day conditions. In fig. 2.4, some visual examples of three of these videos are displayed.

Annotations are provided in *road events* format, triplets composed of a moving agent, the action it performs, and the location it takes place. Each *road event* is related to a time series of manually annotated bounding boxes belonging to a road agent. In total, 560K detection bounding boxes were labeled, with a total of 1.7M unique labels (560K agent labels, 640K action labels, and 499K location labels).

Triplets can be decomposed in *agent-action* pairs (location invariant). This flexibility in labeling, allows researchers to focus on different tasks, such as *agent detection* (detect agent class), *action detection* (detect action class) or *location detection* (detect location class).

Focusing on pedestrian agent class, there is a total of 4212 unique tracks. Among

(a) Driver manoeuvre.

(b) Accidents.

(c) Traffic rules.

(d) **Pedestrian intention.**

(e) Front car intention.

Figure 2.5: Frames from sequences in the five different scenarios in VIENA^2 dataset. Obtained from https://sites.google.com/view/viena2-project/home

them, there are 239 with the action label *waiting to cross* and a total of 449 with crossing behavior, dissected into 185 crossing from the left of the ego-vehicle, 134 from the right, and 130 without initial crossing point specified. The rest of the pedestrian tracks are non-crossing. One possible way of using these pedestrians for crossing action prediction is the clusterization of different groups of pairs or triplets in two main groups: *crossing*, *not crossing*. The main problem with these statistics is that various tracks usually belong to the same pedestrian. Therefore, the number of pedestrians in each of the groups can be significantly lower than the previous numbers.

2.2.3 VIENA^2

VIENA^2[ASS+18] is a simulated driving dataset consisting of 15K videos recorded in GTA V game scenarios at 30 fps with a resolution of 1920 × 1280 px. Sequences are 5 seconds long. All recorded scenes are classified into 5 different groups (see fig. 2.5 for visual examples). 3 of them are focused on the ego-vehicle driver intention, while the other two are focused on other cars and **pedestrians**. Pedestrian group, named **Pedestrian Intentions** addresses the problem of early understanding of pedestrian intention by labeling each video with one of the following labels: pedestrian is going to cross the road; pedestrian has stopped and does not want to cross; and pedestrian is walking on the sidewalk, along the road. Since videos do not provide other labeling and focus on action anticipation/classification, an additional group of sequences without pedestrians is provided. Thanks to the simulated environment, corner cases such as accidents can be reproduced without risk of injury. Moreover, different weather situations and daylight conditions can be sampled equally. However, pedestrian crossing scenarios are limited by the in-game deterministic pedestrian model, crossing in the majority of cases in designated points such as zebra

crossings, with nearly any corner case, where pedestrians cross unexpectedly.

2.2.4 Joint Attention in Autonomous Driving

Joint Attention in Autonomous Driving (JAAD) dataset [RT18a] was one of the first big-scale public datasets with pedestrian behavioral annotations. This dataset was recorded in two vehicles, without ego-vehicle information. Cameras were mounted inside the cars in the center of the windshield below the rearview mirror. There are 346 videos in total, 286 recorded in Europe (276 at Kremenchuk, Ukraine; 6 at Hamburg, Germany, and 4 at Lviv, Ukraine) and 60 in North America (55 at North York, Ontario, Canada and 5 at New York, USA). Most of the recording was done in urban areas, with only a few in rural conditions. 3 different cameras were used in the recording process:

- **GoPro Hero+**: Camera used in videos 1 to 60. The resolution is 1920×1080 px with a FOV of 170° and a framerate of 30 frames per second (fps).
- **Highscreen Black Box Connect**: Camera used in videos 61 to 70. The resolution is 1280×720 px with a FOV of 110° and a framerate of 60 fps for 8 of them, and 30 fps for the other two.
- **Garmin GDR-35**: Camera used in videos 71 to 346. The resolution used is 1920×1080 px with a FOV of 110° and a framerate of 30 fps.

Videos are short (ranging from 5 to 15 seconds long [RT18b]), with the consequent short length of pedestrian tracks (121 frames on average).

There are 2786 annotated pedestrian tracks with 686 of them with behavioral annotation (495 cross the street and 191 not). Those are only the pedestrians that interact with or require the attention of the driver. These tracks raise a total of 378643 bounding boxes manually annotated distributed over more or less 75K frames of a total of 82032 frames, with categorical occlusion levels included (none, partial, and fully occluded). All sequences were collected in a naturalistic environment. Several pedestrian attributes are provided such as age groups and gender. With the bounding box, behavior events are labeled including action (standing or walking), reaction (speeding up, slowing down), hand gestures, gaze direction (regarding the ego-vehicle) and nodding. Each video also contains information about the time of day (most videos were recorded during the day), weather (clear, snow, rain, and cloudy), location (street, indoor, and parking lot), and the existence of a designated crossing. In fig. 2.6, an example sequence with part of the behavioral annotations is shown.

2.2.5 Pedestrian Intention Estimation

PIE dataset [RKKT19] shares creators with JAAD. This dataset provides new behavioral data, without many of the limitations of JAAD. It uses a single camera setup, inside the vehicle below the rearview mirror. Image resolution is 1920×1080 px, with FOV of 157° and a framerate of 30fps. All sequences are separated into 6 sets, with 10 minute length approximately. All of them were recorded in the urban environment of Toronto, Canada.

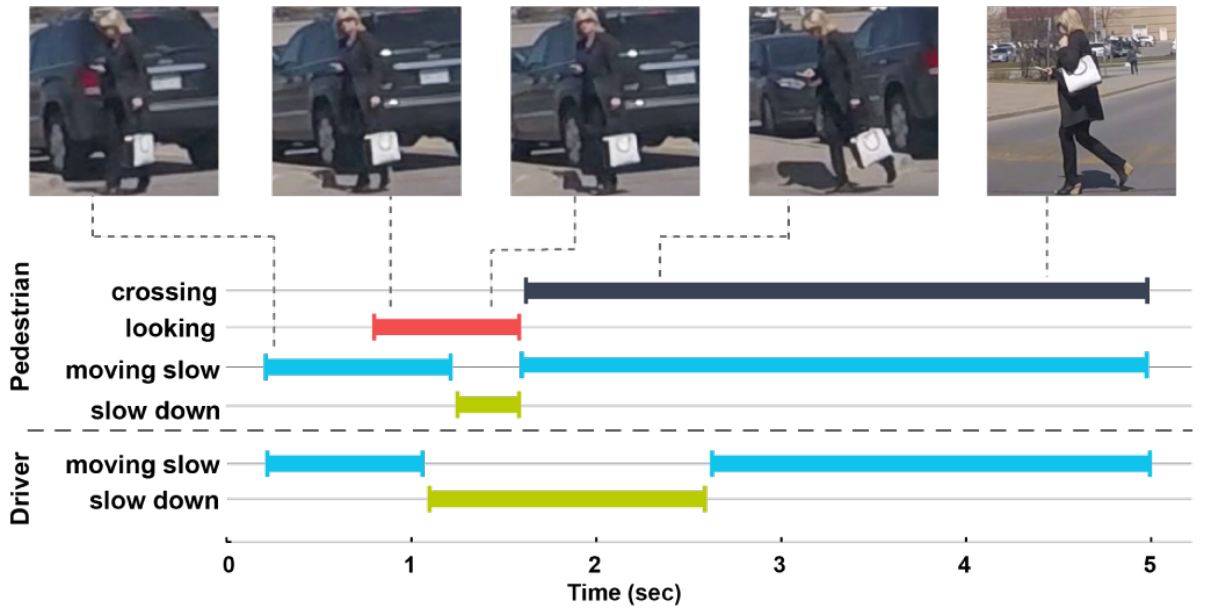


Figure 2.6: An example sequence including part of the behavioral annotation in JAAD dataset. The behavioral tags are updated at the video framerate and categorical behavioral tags are also included for the driver of the ego-vehicle. Obtained from https://github.com/ykotseruba/JAAD/blob/JAAD_2.0/behavior.png.

The environment varies in terms of foot-traffic density, street width, and weather (sunny and cloudy). All scenes were recorded during the daylight.

Only pedestrians who can potentially interact with the ego-vehicle are annotated with 2D bounding boxes, occlusion flags (not, partially, or fully occluded), and actions (walking, standing, looking, or crossing). Crossing intention confidence is also provided. This parameter is estimated from the results of an experiment carried out with humans and serves as a baseline. Road environment annotations are more detailed than their predecessor, with bounding boxes provided for traffic signs, traffic lights, and zebra crossings (see fig. 2.7). Bounding boxes for vehicles (with different types) are also provided if there is an interaction between them and the labeled pedestrians.

Ego-vehicle data is provided using an On-Board Diagnostics (OBD) sensor synchronized with the camera (speed and heading). GPS information is also provided.

2.2.6 Trajectory Inference using Targeted Action priors Network

TITAN dataset [MDC20] was recorded in central Tokyo. It consists of 700 short video clips extracted from 10 hours of data, between 10 to 20 seconds long. Videos are recorded at 60 fps but annotated at 10 Hz. The resolution of images is 1920 × 1200 px. Ego-motion data from an IMU sensor is also provided, synchronized with image data. There are a total of 8592 unique persons labeled and 5504 vehicles. In addition to bounding box with tracking information, 3 age groups are provided for person object class (child, adult, and senior). One of the main contributions of this dataset is the fine-grained action

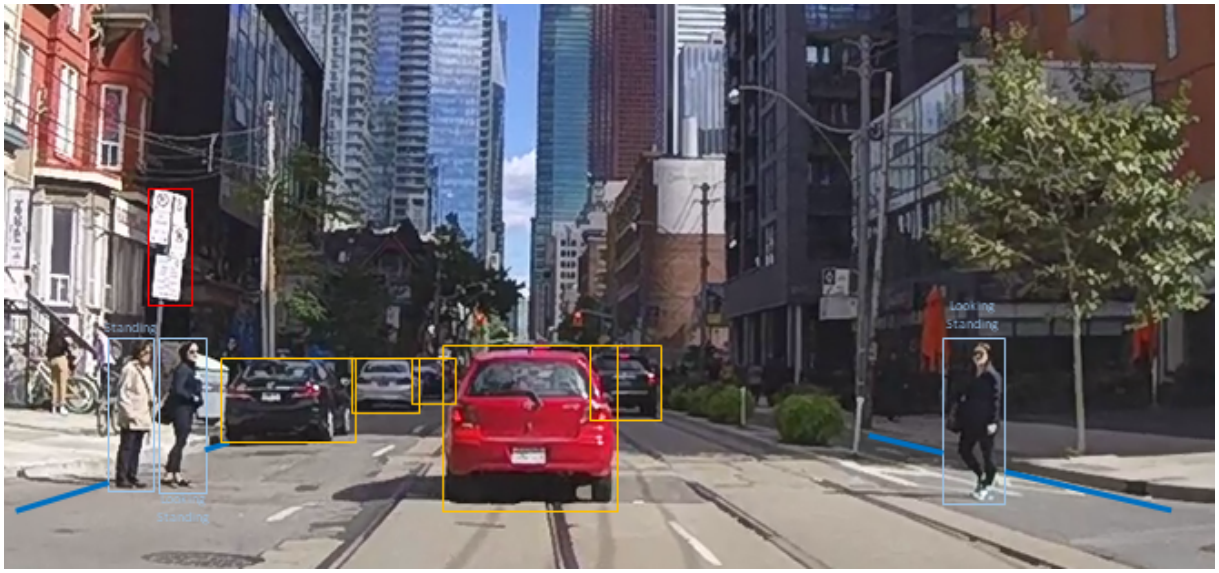


Figure 2.7: An example of objects annotated in Pedestrian Intention Estimation (PIE) dataset. As can be observed, besides the class represented with different box outline color, behavioral tags are included for pedestrians who are near the road scene or will interact with the ego-vehicle. Obtained from https://github.com/aras62/PIE/blob/master/pie_annotations.png.

annotation provided, which is, to the best of our knowledge, the most detailed one related to IV data. There are 5 action sets organized as shown in fig. 2.8. Looking for crossing-related labeling, *Simple Contextual* group provides 5 classes: waiting to cross, crossing, walking on the road, walking on the sidewalk, and jay-walking.

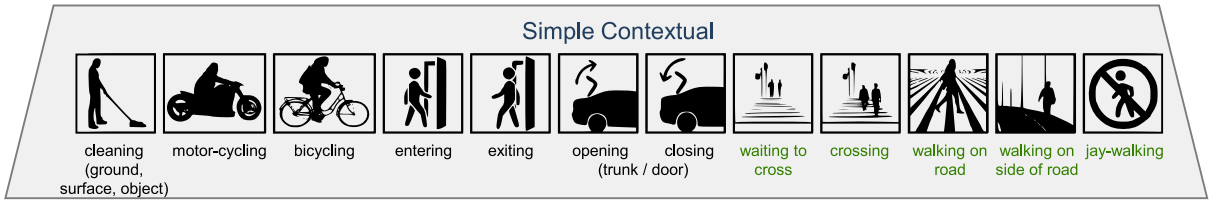


Figure 2.8: Part of the action hierarchy followed by annotations in TITAN dataset. "Simple Contextual" group are the most interesting ones for the scope of this thesis. Class names related to crossing behavior are written in green. Obtained from [MDC20].

Despite this extensive labeling, action hierarchy suffers from the typical long-tailed data disease: the majority of classes have low number of samples. In addition, the separation of crossing behavior is more focused on crossing behavior rather than balancing crossing and not crossing. 4 of the classes (all except *walking on side of road*) implies crossing in the end, and *waiting to cross* implies intention instead of action and does not have a counterpart *waiting to cross, and not crossing*. Moreover, *walking on road* is difficult to discern from *jay-walking* in terms of the beginning of the action, which can lead to an unfair difficulty added in terms of data complexity.

Furthermore, bounding boxes are of a minimum resolution of 50×10 px, making the dataset difficult to use in object detection and action detection, and avoiding difficult

cases where pedestrians and cars are small.

2.2.7 Stanford-TRI Intent Prediction

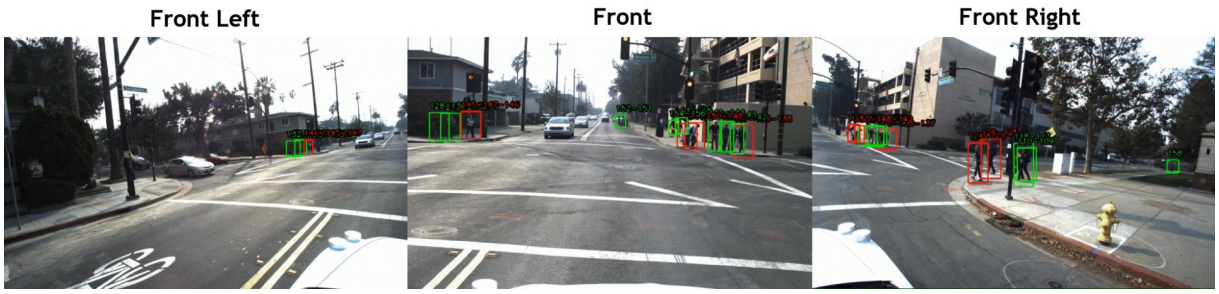


Figure 2.9: Sample of the dataset STIP for each of the three cameras used in the recording. Crossing pedestrians are visualized in green and non-crossing ones in red. Obtained from https://stip.stanford.edu/dataset.html.

STIP dataset [LAC+20] was created between Stanford University and Toyota Research Institute. Recorded in urban areas at 8 cities in the states of California and Michigan (United States), with various weather conditions present. 923.48 minutes were recorded at 20 fps (1, 108, 176 frames) with resolution of 1936×1216 and manually annotated at 2 fps, with their corresponding boolean crossing state. With 350, 000 pedestrian bounding boxes in total, the number of pedestrian tracks is higher than 25, 000, with a median length of 4 seconds. 3 cameras were used with the previous resolution, oriented left, front, and right respectively (see fig. 2.9 for a visual example). Annotations are interpolated at 20 fps using a tracker. This dataset is open to the research community under restricted conditions. While it is quite complete and diverse, the available dataset corresponds only to the hand-labeled part of sequences (2 fps). In this thesis, the input observation interval chosen has been usually 0.5 s, so it is not suitable, for comparison with other datasets.

2.2.8 Other AV-related datasets

In the world of IVs, there are other datasets that are more complete in labeling than the ones discussed in the previous list. However, all of them lack crossing action annotations and focus on other tasks such as object detection, tracking, and segmentation. Among them are Waymo Open Dataset [SKD+20], Lyft Level 5 Perception Dataset 2020 [KUH+19], Argoverse three-dimensional (3D) Tracking [CLS+19] and nuScenes [CBL+20]. These datasets can be extended to crossing action prediction tasks, leveraging rich annotations with behavioral data. This is the case of PePScenes [RYL+20], an extended version of nuScenes dataset. A subset of 719 pedestrians are labeled with behavioral tags (*crossing*, *will not cross* and *about to cross*), with 570 non-crossing and 149 crossing pedestrians. two-dimensional (2D) and 3D bounding boxes are included at a higher frequency (10 Hz instead of 2 Hz). The dataset will be made public to the research community but, at the time of writing this thesis, it is not available yet.

2.2.9 Datasets summary

As a summary, in table 2.1 you can see a detailed classification of the datasets discussed on the previous list, including the adaptation of **nuScenes** presented in section 2.2.8. The data column is focused on vision sensors.

2.3 Methods

Pedestrian behavior understanding is a challenging task inside the pipeline of IV development [RRL+18]. This task has to cope with potential problems and noise caused by other previous stages, such as pedestrian detection and tracking (e.g., occlusion, false positives, different tracking IDs for the same pedestrian). If the output of these stages is considered ideal, the uncertainty caused by the high dynamism in the pedestrian's movement makes it difficult to forecast their intention in a deterministic way, limiting them to short timescales [JCL+20].

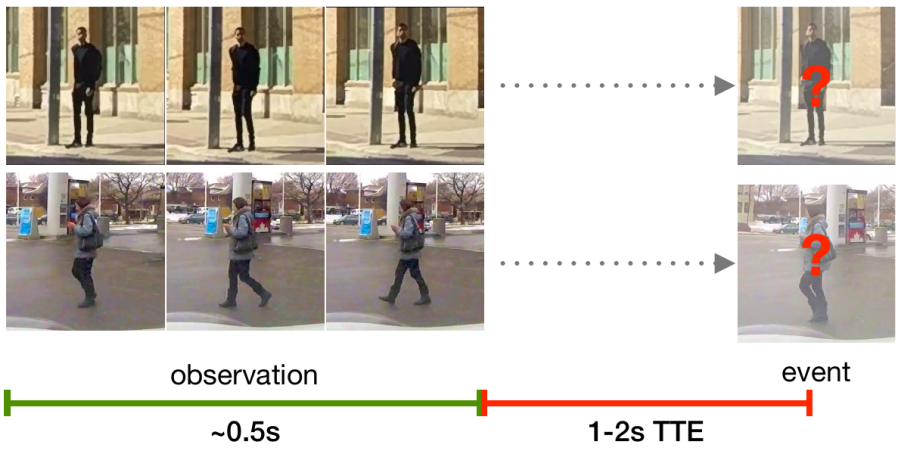


Figure 2.10: Examples for both behaviors in the crossing prediction task. The top one is a non-crossing case, and the bottom a crossing one. Obtained from https://github.com/ykotseruba/PedestrianActionBenchmark/blob/main/crossing_prediction_task.png

Pedestrian crossing behavior or action prediction can be an implicit or an explicit task. The first case is mostly related to trajectory prediction/forecasting methods. An understanding of the environment through semantic maps can lead to implicit anticipation of crossing since the future trajectory of the pedestrian can be predicted through his or her position relative to the road. In the explicit case, crossing behavior is the main task, extracted from raw sensor data or features such as pose [QPLS18]. In the case of this work, one of the main sources of information is image data. As explained in section 2.1, due to the last decade's progress in **action recognition**, the approaches developed in this work are based on methods with this task as their initial purpose. However, instead of classifying the crossing action performed in the video sequence, the model tries to anticipate the future action to be performed. This is also commonly known as **pedestrian**

Dataset	Nat ?	# ann. fr.	Tot. [s]	R/A [Hz]	# C/NC	HFoV [°]	Res.	Tr. L [s][a]	W	Loc.	Ego	cont.	3D
Daimler	✗	7.8k	0.5k	16/16	50/18	-	1176 × 640	7.15	✗	✗	✓	✗	✓
ROAD	✓	127k	10.6k	16/12	449/3763	66	1280 × 960	-	✓	✗	✓	✓	✓[b]
VIENA²	✗	-	75k	30/0[c]	426/400	-	FHD	-	✓	✓	✓	✗	✗
JAAD	✓	82k	864k	30/30	495/191	110-170	FHD	4.03	✓	✓	✗	✓	✗
PIE	✓	293k	30.3k	30/30	519/1323	157	FHD	13.36	✓	✗	✓	✓	✗
TITAN	✓	75k	36k	60/10	-	119	FHD		✗	✗	✓	✗	✗
STIP	✓	-	55.4k	20/2	25k	180 (?)	FHD	4[d]	✓	✓	✗	✗	✓
PePScenes	✓	170k	54k	12/10	149/570	360	1600 × 900	-	✓	✓	✓	✓	✓

Table 2.1: A comparison of all datasets available in the literature of pedestrian crossing action prediction task. Meaning of column headers (from left to right): "Nat." for "Naturalistic", "#" for number, "ann. fr." for "annotated frames", "Tot." for "Total length of recording", "R/A" for "record/annotation rate", "C/NC" for "Crossing/Non-Crossing cases", "HFoV" for "Horizontal Field of View", "Res." for "Resolution", "Tr. L" for "Pedestrian Track Length". The last five columns represent information present or not in the dataset: "W" (different weather conditions), "Loc." (diff. locations), "Ego." (ego-vehicle information), "cont." (context information), "3D" (3D information)

[a] Averaged.
[b] Recorded, but not annotated.
[c] One label per sequence.
[d] Approximated median.

crossing prediction [KRT21]. In fig. 2.10, the task is visually portrayed with one example for each possible behavior.

Following the taxonomy in [RDWT19] and [RPH+20], methods inside this task can be classified by their algorithmic, input, and their output nature. We will begin by discussing a distinction in terms of algorithmic nature.

2.3.1 Algorithm nature

With respect to the model nature, two groups can be distinguised: **probabilistic** and **pattern-based** methods.

In the early research stage of pedestrian behavior prediction, most of the proposed methods were based on **probabilistic** techniques. These methods are characterized by using hand-crafted features extracted from raw sensor data.

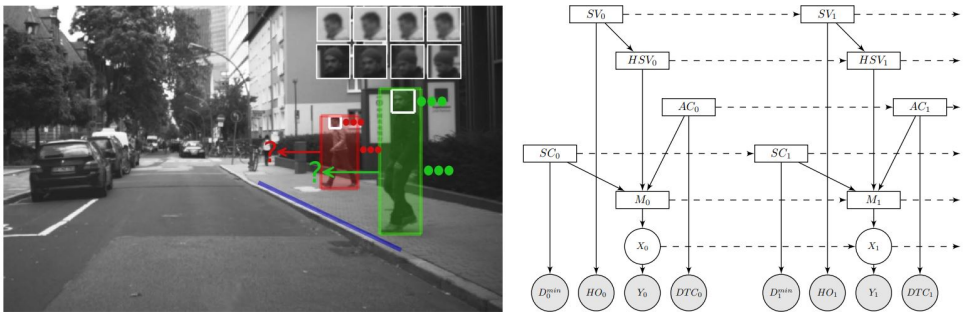


Figure 2.11: Probabilistic model based on DBN. On the left, an example frame with some of the sensor data drawn on it (e.g., curb position, pedestrian detection, head orientation, motion direction). On the right, the DBN model as a directed graph for two time steps. Grey nodes are observed, and rectangular ones are discrete variables. Obtained from [KSFG14].

In [FY11], Bayes Theorem is used to estimate pedestrian crossing action classification. While anticipation is not the focus of this work, it served as a precedent and influenced future methods. Recursive bayesian filters such as Interacting Multiple Model (IMM) and Extended Kalman Filter (EKF) are used for trajectory prediction in [SG13], where both methods are compared, with better results for IMM. While the initial purpose of the method was trajectory prediction, in [KGZ+15a], IMM method is thresholded to adapt it for classification purposes, comparing it with a machine learning method. In [KG14], two non-linear methods, Gaussian Process Dynamical Model (GPDM) and probabilistic hierarchical trajectory matching (PHTM) are compared to linear Kalman Filter (KF) and IMM. These complex non-linear methods outperform linear models in non-linear scenarios, such as the stopping scenario where the pedestrian stops instead of crossing. The obtained results are compared with the predictions of a human baseline. Humans reach an accuracy of 80% 570 ms before the action and PHTM and GPDM reach this accuracy with an anticipation of 230 ms. The recursive bayesian filter solution, based on IMM, reaches human accuracy only 90 ms before the action is performed, being the worst among all methods. Another application of GPDM is shown in [QPLS18], where a b-GPDM model, a modification whose purpose is learning 3D time-related information. The model converts pose data to a lower-dimensional space and use it to predict the

future path and pose. By combining it with a naive-Bayes classifier at the end, the system is able to predict motion action. Naive-Bayes classifier is also used [ZLXL19] with modifications to perform crossing action classification using LiDAR data. While these non-linear methods outperform simpler bayesian filters, in [SS15] the authors proposed the use of a latent-dynamic Conditional Random Field (CRF) model in combination with IMM model, to better control its transitions. This composition led to an increase in performance with respect to PHTM. In [NHD+20], a variation of CRF model is proposed for inferring pedestrian intention, outperforming in anticipation time even a recurrent pattern-based method.

Finally, the last iteration of probabilistic methods uses the Dynamic Bayesian Network (DBN) framework, which allows the use of discrete and continuous variables together and great flexibility in system design. In [KSFG14], a DBN is used to process different sources of data and choose the most suitable motion model for trajectory prediction. In [KFPG19], an extension of this model focused on cyclists, was designed. The only problem with this model is that the inputs used are dependable on the crossing situation. In fig. 2.11 an example is shown of the directed graph modeled and the dependencies between hidden variables, highly biased to a situation where the curb is present. Finally, the last method found to use this technique [HGH+16], combines environmental information (pedestrian traffic light states, pedestrian motion state) with kinematics measurements and uses a particle filter to estimate pedestrian decision and motion state. However, like the previous methods, the model relies on variables related to the environment of a zebra crossing, affecting the generalization of the system.

With the increase in data availability and computer power, **pattern-based** methods began to gain popularity. The main difference with respect to the previous group is the deterministic nature of all methods and the automatic feature extraction, unsupervised instead of calculating them in advance in a tailored way. As explained before, [KGZ+15b] compares IMM model with a pattern-based approach. It is based on a SVM classification method and outperforms the probabilistic model in terms of reaction time and accuracy. In [HJ15], another approach based on SVM is used to detect pedestrians and a naive-Bayes classifier is used to output the action performed by them. Random Forest classifier is used in [FL19] for pedestrian action classification, supported by features obtained through deep learning-based object detection and pose estimation techniques.

The neural network framework was considered early in the pattern-based group. The first work to use it, to the extent of our knowledge, is [BWKS14]. In this work, two models are designed based on context, using a simple perceptron, one of the simplest forms of neural network). [VBM+16] serves as a transition analysis between SVM models and neural networks. While both groups of methods obtain similar performance for small distances to the pedestrian, the correct classification rate for SVM model decreases faster with distance than the neural network model, showing the potential of neural networks.

Most of the recent work in pattern-based approaches relies on deep learning techniques, mainly based on neural networks, leaving shallow techniques such as SVM as a minority. Convolutional approaches such as 2D Convolutional Neural Networks (CNNs) are predominant due to their ability to extract information from image data. In [DCO17], pedestrian movement direction is predicted using different architectures (AlexNet, GoogleNet, and

ResNet10). In [RKT17], an AlexNet-based model is modified to be fully convolutional to tackle different input resolutions. For the crossing classification task, an SVM is added to the network. 2D CNN models are also used with image-shaped data, as in the case of [GPB+20] where a multi-branch network combining precomputed features extracted from 2D keypoints are used for pedestrian crossing prediction task. While these models extract useful information from high-dimensional data, they do not process temporal information.

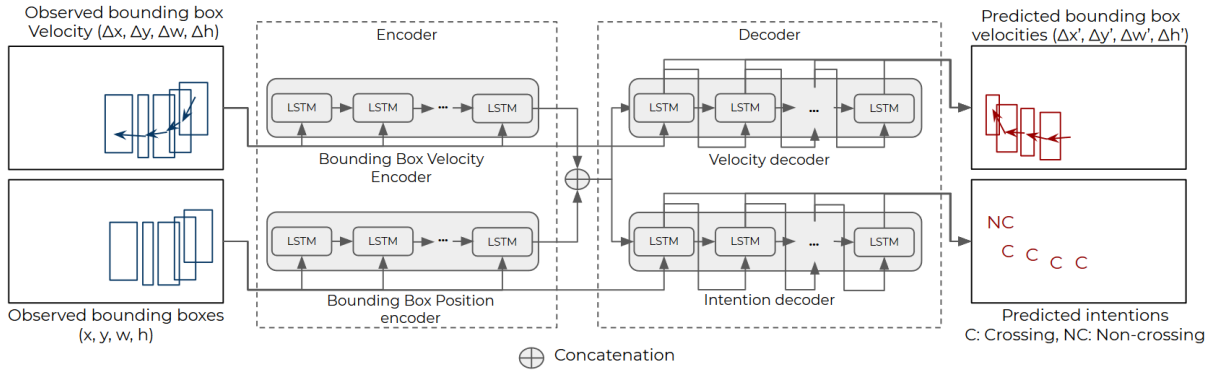


Figure 2.12: Pure RNN-based approach. Only with the use of bounding boxes coordinates and their speed, the model is able to predict both the future speed and action performed by the pedestrian. Obtained from [AA20].

Current state-of-the-art methods are based on temporal models such as 3D CNNs or recurrent neural networks (RNNs). [AA20] is a good example of pure RNN approach (see fig. 2.12). A multi-task long short-term memory (LSTM)-based encoder-decoder architecture is used to forecast future trajectory and crossing action of each pedestrian. Recurrent models are also found in combination with convolutional approaches. In [GMB+18], a 2D CNN model is used to estimate pedestrian's keypoints (pose) and an LSTM is used to anticipate action. They compare it with several methods, both probabilistics (PHTM [KG14] and Latent-dynamic CRF [SS15]) and pattern-based, including a 3D CNN approach. Another example of 2D CNN and RNN combination can be seen in [PRC+19], where they develop a method to estimate action recognition and time-to-cross the street for multiple pedestrian. A pedestrian CNN detector, based on RetinaNet is used to classify the action and, after detection, the output class and the 2D bounding box coordinates are used as input to an LSTM model. The recurrent model is used to predict the Time To Cross (TTC). Another example of this combination is deployed with the name of VRUNet in [RGB+20]. The model uses different 2D CNN networks for feature extraction, combining it with LSTM and fully-connected layers to perform a multi-task prediction. A baseline approach is also developed, mixing CNN used as feature extractors and SVMs, used to classify different features and the final crossing action. Convolutional LSTM layers are also used in [RKKT19] in combination to LSTM layers for non-image data and different attention methods. A multi-level crossing action prediction approach is proposed in [RKT20], where gated recurrent unit (GRU) cells are used to generate temporal features which are concatenated through different levels, reaching a final fully-connected layer. All input features to the GRU layers are obtained using 2D CNN networks. Another example of combining both worlds is [RRL21], where a multi-task framework is created

to benefit from different sources of information, processed by 2D CNN or LSTM models. A similar approach, with less complexity is proposed by the same author in [RYRL20]. In [LPW+20], the combination of both types of models is studied in a progressive way, including different sources of information gradually.

3D CNN methods are also present among state-of-the-art literature and allows extraction of temporal and spatial features using a single model. In [SHN19], a full real-time framework is presented including all steps in the pedestrian crossing action prediction (see fig. 2.13). In the last step of the pipeline, the crossing action is predicted using a 3D CNN based on the DenseNet 2D CNN, a well-known family of convolutional classification models. In [PBP+20], an end-to-end method based on the previous one is developed improving the previous results, by including pose information extracted with a 2D CNN model. 3D CNN models are also used for future video frames prediction. In [GV19a], they propose a 3D CNN encoder followed by 2 possible methods: convolutional LSTM or 3D CNN decoder. An ablation study is also performed using a residual encoder-decoder model. Future image predictions are used as input to a 3D CNN binary crossing action classifier. A similar approach is used in [CTBB20], where a 3D convolutional layers are mixed with convolutional LSTM layers to construct an encoder-decoder future frame prediction model, with a 3D CNN binary crossing action classifier too. However, both [CTBB20; GV19a] train the action classifier with whole future frames, being unclear on how to focus the prediction in a sequence with more than one pedestrian.

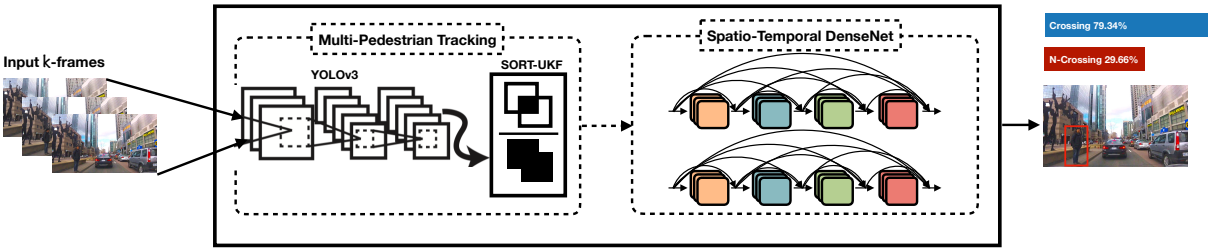


Figure 2.13: Real-time pedestrian crossing prediction framework. The pedestrian is detected with a YOLO-based object detector and tracked with SORT-UKF method. Once the track is obtained it is used as input for a 3D CNN model (DenseNet) which outputs the crossing probability of the tracked pedestrian. Obtained from [SHN19].

A combination of both types of CNN (2D and 3D) and RNN models is present in [KRT21] and in [MDC20]. In the last case, crossing action prediction is not a task in the model.

Graph neural networks are also used for the task. In [LAC+20], a scene graph is created for each pedestrian sequence using environment information. After the graph generation and processing, a GRU-based model is used to anticipate crossing action. In [CYQW19], authors follow a procedure similar to the one in [GPB+20], using a convolutional network to process normalized adjacency matrix representations based on a 14 keypoints pose graph. The model used in this work is not a graph convolutional neural network (GCN) per se, as the authors stated, but the use of the precomputed input can be seen as an approximation to graph learning. Finally, in [YMR+21], a graph-based model is proposed, modeling pedestrians' interactions with nearby road users in 3D space, using clustering

and relative importance weighting of interactions (see fig. 2.14 for an overview of the model).

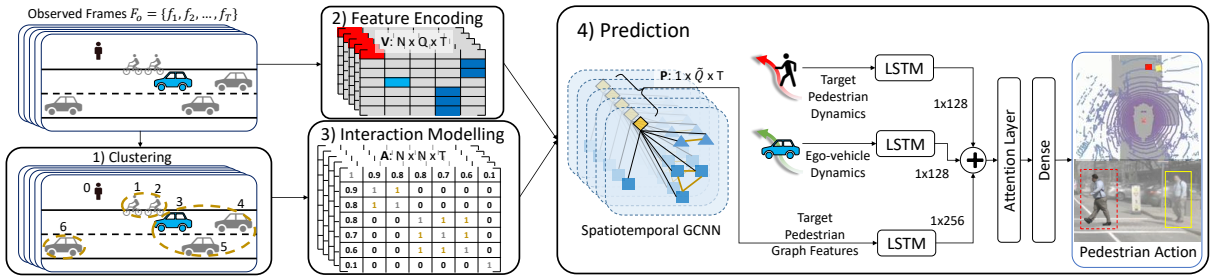


Figure 2.14: Graph-based model which uses 3D information to cluster (1) and generate the relation graph between agents (2, 3). After that, these features are used as input to a GCN followed by an LSTM which outputs a crossing probability. Obtained from [YMR+21].

Both types of methods, probabilistic and pattern-based can also be **combined**. This is the case of [RWLS17], where a recurrent mixture density network is used to provide possible future destinations for pedestrians. The input to the recurrent module (LSTM) consists of the concatenation of the output of a CNN model and a position vector. Possible predicted destinations are established as goal states of a planning system that performs motion prediction. The system is modeled as a single end-to-end neural network trained via inverse reinforcement learning. With the topology prior knowledge of the road boundaries, the crossing action can be extracted by looking at the forecasted position on the future map. However, this analysis has not been pursued in this work.

2.3.2 Output nature

Among both groups of methodologies discussed in section 2.3.1, there are various output natures. While in near all of the discussed methods the problem is reduced to a binary crossing classification with two opposite classes, there are other works, mainly inside the **probabilistic** group, where additional actions are predicted, inspired by pedestrian's motion state (e.g., *walking*, *starting to walk*, *stopping*, *bending in the road*). In [QPLS18], four classes are labeled: *walking*, *starting to walk*, *stopping*, and *standing* (not walking). In [SG13] similar classes are used, substituting *walking* with *crossing* and *standing* with *bending in*, action performed when pedestrian changes direction towards the road. In [SS15] an additional label is included to the previous group, identified as *straight*, referring to sequences where the pedestrian walks along the sidewalk. These classes were used due to their direct availability in the public data at that time, mainly Daimler Pedestrian Dataset (see section 2.2.1).

Another alternative to binary crossing classification is **binary crossing intention classification**. In the first case, an anticipation of the future crossing action is performed by the system using a fixed forecasting time horizon, as in [LPW+20]. In the second case, each pedestrian track is labeled with their final crossing action, so the time horizon is variable, lowering while the TTE is approaching. This output objective is used in [KRT21]. A variant of this output nature is presented in [RKKT19], where instead of using discrete

crossing actions labeled on PIE dataset, the authors use the normalized average response of a human experiment as a pedestrian crossing intention representation.

Time horizon is another variable of study in this classification. Two groups of methods can be extracted: short-term (up to 2.5 s) and long-term prediction (more than 2.5 s in the future). The first group is more common in current state-of-the-art methodologies. With the high variability in movement and orientation, long-term prediction is a more challenging task, only pursued in some works where goal-directed forecasting methods are implemented [KZBH12], [RK15] or [KAHS16]. Concerning the input sequence temporal length, a short period (less than one second) is the usual choice, with exceptions such as [CYQW19] and [GPB+20], where the sequence length used as input is 10 s, anticipating zero frames, converting it into a sequence classification problem.

2.3.3 Input data nature

In the case of **crossing action prediction** there are two main input feature groups, taking a pedestrian-centric view model:

- **Internal factors**: related to the pedestrian. Belonging to this group, we can highlight pose key points, the position of pedestrian, speed, state of movement, and their orientation.
- **External factors**: related to the pedestrian's surroundings, also known as their context. Here we find two subgroups. The first one is related to known context, where features are extracted manually from raw sensors. Here we can highlight semantic information, relative distances (e.g., distance to the curb, distance to the ego-vehicle), other road agents' information (position, speed, orientation), and pedestrian's attention to their surroundings (e.g., through the gaze direction). The second group is related to unsupervised context, where raw sensor data (mostly image data) is used as input directly, letting the method learn context features during training.

As previously stated, raw image data is usually the source of information. However, a distinction is made from this point, with methods relying on 2D information, and methods focusing on 3D data, extracted using stereoscopic vision [SG13; SS15; KG14; KGZ+15a; KSFG14; KFPG19; QPLS18; BWKS14; RWLS17]. In [KSFG14; KFPG19], discrete head orientation is used as input to the model. However, this variable is complemented by others such as level of criticality and relative distances. Relying solely on head orientation may not be a good idea, since pedestrians may be distracted. Attention binary level can substitute this variable [LPW+20; RKT17].

Trajectory information extracted from stereo information is present in most works that use this sensory input. Furthermore, relative distances measures are extracted from stereo data. In [BWKS14; KSFG14; SS15; KFPG19; KGZ+15a; RWLS17], the authors used distance information (e.g., to the curb, lateral), relative times, and speed information, among other variables. Moreover, pose data is extracted from stereo and used in [QPLS18].

Another possible source of 3D information is LiDAR point clouds as in [VBM+16], where the authors use raw LiDAR data as well as measurements extracted from it (e.g., pedestrian and vehicle speed, distance traveled, distance to the curb). In [ZLXL19], 3D

pedestrian data is extracted from point clouds using clustering techniques and background filtering. In [YMR+21], the authors use annotations extracted from LiDAR sensor data, hand-annotated in the dataset.

Raw image-based methods usually belong to the **pattern-based** group, especially deep learning approaches. This source of information is used directly as an input to the system [LPW+20; SHN19; RKT17; RKKT19; RKT20; RRL21; RYRL20; MDC20; CTBB20; GV19b; PRC+19; KRT21; PBP+20]. In [FY11], the posture information of pedestrians is obtained from the image. HOG features from image regions of their bounding box are obtained and processed with Principal Component Analysis (PCA) method to reduce its dimensionality. 2D pose is also estimated directly from image sensor data. It is used as a main source of information [CYQW19; GPB+20; FL19; GMB+18], or as a complimentary one [KRT21; RKT20; RRL21; RYRL20; PBP+20]. Additional information, such as binary tags for the state of movement (i.e. walking, standing) and looking state, discrete orientation, and bounding boxes coordinates are used on current state-of-the-art methods [LPW+20; RKT17; RKKT19; RKT20; RRL21; RYRL20; KRT21]. In the case of [AA20], only bounding boxes annotations are used. Semantic segmentation is also used as input in recent work [RRL21; RYRL20].

As explained at the beginning of the section, another data-related distinction is the establishment of ideal conditions for the detection and tracking parts of the system. These ideal conditions are present in the vast majority of work in the field, with some exceptions where a complete pipeline is developed [SS15; SHN19; PBP+20].

With respect to the data used, there are two datasets that are particularly popular in the literature: JAAD and Daimler datasets (see section 2.2 for a detailed description). Daimler dataset is used in earlier probabilistic work. However, JAAD is the main option in pattern-based methods.

However, there are some methods that use custom datasets for experimentation. In [FY11], two datasets are recorded, an indoor and an outdoor one, respectively, to prove the viability and adaptability of their proposal. In [QPLS18] a modified version of The Motion Capture Database [2] is used, which provides reliable 3D pose keypoints at high recording frequency (120 Hz) for different activities. The modification, called CMU-UAH, is a filtered version of sequences belonging to the classes commented previously (see section 2.3.2). In the case of [HGH+16], the dataset is recorded with three monocular cameras mounted on the ego-vehicle, while driving around a single intersection, turning left, and passing through its crosswalk several times. The dataset pedestrian samples are divided between crossing and waiting classes, with an imbalance of about 6.6 to 1, respectively. Another custom dataset is the one created for [BWKS14]. It is recorded from ego-vehicle, and twelve context variables are extracted from data: relative distances, relative times, and binary categorical orientation (heading towards the road). Two of them are only used for the zebra crossing specific model: lateral distance of pedestrian to the zebra crossing and pedestrian time to reach the zebra crossing. Pedestrians as far as 20 m are detected and tracked. Pedestrians farther than that distance are manually annotated. Crossing and not-crossing pedestrians are again unbalanced (8.9 : 1). Naturalistic recordings are mixed with scripted scenarios with actors to increase crossing samples. In [NHD+20], the

[2] http://mocap.cs.cmu.edu/

authors collect and label an in-house dataset to test their models in addition to JAAD.

In [MDC20] Trajectory Inference using Targeted Action priors Network (TITAN) dataset is created and used. This dataset is open to research but under restrictive terms of use. Similar restrictions are present in acstip, the dataset presented in [LAC+20] as an alternative to JAAD and PIE. Details about those two datasets can be found in section 2.2. In [YMR+21], due to missing 3D information in pedestrian crossing behavior datasets, they introduce PePScenes, an extended version of nuScenes (see section 2.2 for details). In [RYRL20], models proposed are trained on this new dataset in addition to JAAD and PIE.

Finally, in table 2.2 and table 2.3 show a classification summary of the main methods in **probabilistic** and **pattern-based** groups, respectively. Due to the lack of standardization in the evaluation method, most of the methods follow different evaluation schemas, even when using the same dataset. In the following section, a recent benchmark for this task will be detailed and introduced, as it is part of the evaluation procedure in the final work of this thesis.

Method	Input data: Image	Pedestrian			Context			Algorithm	Horizon [s]	Objective	Dataset
		Kin.	Or.	Cat.	P2V	P2E	E				
[KSFG14]	✗	3D	✓	✓	✓	✓	✗	DBN+SLDS	1	C-T	[KSFG14]
[KG14]	P	3D	✗	✗	✗	✗	✗	GPDM,PHTM	0.77	C-T	[SG13]
[SS15]	✗	3D	✓	✗	✗	✗	✗	LDCRF+IMM	1	C	[SG13]
[KGZ+15a]	P	✗	✗	✗	✗	✗	✗	SVM	-	C	[SG13]
[HGH+16]	✗	3D	✗	✓	✗	✓	✓	DBN+PF	-	B	O
[VBM+16]	✗	3D	✗	✗	✓	✓	✗	RNN	[1, 6][a]	B	O
[QPLS18]	✗	3D	✗	✗	✗	✗	✗	HMM+B-GPDM	1	A-T	CU
[HTDD18]	BBs	2D	✗	✗	✗	✗	✗	2DCNN+VRNN	1	B-T	[SG13]
[KFPG19]	✗	3D	✓	✓	✓	✓	✗	DBN+SLDS	1	B-T	[SG13]
[NHD+20]	✗	2D	✗	✗	✓	✓	✗	LDCRF	2,1.3[b]	C	O,[RT18b]

Table 2.2: Summary table of probabilistic methods. Abbreviations in Pedestrian and Context subcolumns (from left to right): Kin. → kinematics (e.g., 2D and/or 3D position, speed, pose data); Or. → orientation; Cat. → categorical variables; P2V → target pedestrian to ego-vehicle information (e.g., binary gaze direction, positive when looking at the ego-vehicle); P2E → Pedestrian to Environment information (e.g., distance to the curb); E → Environment (e.g., semantic information, traffic lights states). Image column values: ✗→ no image, P → pre-processed images; BBs → bounding boxes crops. Objective column values: C → crossing action prediction; B → binary crossing action prediction; A → Multiple Action Prediction; T → trajectory prediction. Dataset column values: citation → see section 2.2 for details;O → Own (not public); CU → CMU-UAH.

[a] Interval.

[b] Horizon for each dataset, respectively.

	Input data													
Method	**Image**	**Pedestrian**			**Context**			**Ego-vehicle Speed**	**Algorithm**	**Len. [s]**	**Obs. [s]**	**Hor. [s]**	**Obj.**	**Dataset**
		Kin.	**Or.**	**Cat.**	**P2V**	**P2E**	**E**							
[BWKS14]	✗	✗	✓	✗	✓	✓	✗	✗	MLP	-	0	0	B	O
[RKT17]	B	✗	✗	✓	✗	✗	✓	✗	2DCNN	0	0.5	0	B	[RT18b]
[GMB+18]	✗	P	✗	✗	✗	✗	✗	✗	RNN	0	0	1	B	O
[FL19]	✗	P	✗	✗	✗	✗	✗	✗	RF	-	0.5	0	B	[RT18b]
[SHN19]	B	✗	✗	✗	✗	✗	✗	✗	3DCNN	0	0.5	0	B	[RT18b]
[CYQW19]	✗	P	✗	✗	✗	✗	✗	✗	2DCNN	0	10	0	B	[RT18b]
[RKKT19]	B	B	✗	✗	✗	✗	✓	✓	2DCNN,RNN,ConvLSTM,attention	4.5	0.5	3	I	[RT18b; RKKT19]
[GV19b]	F	✗	✗	✗	✗	✗	✗	✗	3DCNN,ConvLSTM	-	0.5	0.5	B	[RT18b]
[CTBB20]	F	✗	✗	✗	✗	✗	✗	✗	3DCNN,ConvLSTM	-	0.5	0.5	B	[RT18b]
[YMR+21]	S	3D	✓	✗	✓	✓	✓	✗	GCN,RNN	0	0.5	-	B	[RYL+20]
[RRL21]	S	B	✗	✗	✗	✗	✓	✓	2DCNN,RNN	?	0.5	2	B-T	[RT18b; RKKT19]
[RYRL20]	B-S	B-3D	✗	✗	✗	✗	✓	✓	2DCNN,RNN	?	0.5	2	B	[RT18b; RKKT19; RYL+20]
[RKT20]	B	B-P	✗	✗	✗	✗	✓	✓	2DCNN,RNN	?	0.5	2	B	[RKKT19]
[AA20]	✗	B	✗	✗	✗	✗	✗	✗	RNN	-	0.6	0.6	B-T	[RT18b]
[RGB+20]	S	B-P	✗	✗	✗	✗	✓	✗	2DCNN,RNN	-	1	1	B	[RT18b]
[GPB+20]	✗	P	✗	✗	✗	✗	✗	✗	2DCNN	0	10	0	B	[RT18b]
[PBP+20]	B	P	✗	✗	✗	✗	✗	✗	3DCNN	0.5	0.5	0	B	[RT18b]
[LPW+20]	B	B	✓	✓	✗	✓	✗	✗	2DCNN,RNN	-	0.5	1	B	[RT18b]
[LAC+20]	B-S	B	✗	✗	✓	✓	✓	✗	2DCNN,GCN,RNN	-	4	3	B	[RT18b; LAC+20]
[KRT21]	B	B-P	✗	✗	✗	✗	✗	✓	3DCNN,RNN,attention	1	0.5	2	I	[RT18b]

Table 2.3: Summary table of pattern-based methods. Same abbreviations as in table 2.2 with the following exceptions: Len. → TTE of the last frame used in training (if 0, means that all frames before the crossing/not-crossing action are used. If empty, all frames in sequence are used); Obs. → input observation length; Hor. → maximum prediction horizon. Image column values: B → bounding boxes crops; F → full frames; S → semantic map. Inside kinematic column: B → 2D bounding boxes coordinates, P → 2D pose keypoints 3D → 3D information. In objective column: I → Crossing Intention Anticipation.

2.4 Benchmark

Due to the lack of a standard evaluation procedure and the ill-defined objective of the task of pedestrian crossing action prediction, most of the work in this field cannot be compared or analyzed in terms of overall performance. Early in 2021, [KRT21] presented a novel benchmark trying to mitigate this problem with an initial standardization.

The benchmark is based on JAAD [RT18b] and PIE [RKKT19] datasets (see section 2.2 for details) and it is publicly available[3]. JAAD dataset has two possible benchmarks. The first one is created for behavioral data only, with 686 pedestrians, 495 crossing and 191 non-crossing ones. The second one, includes all the available data, with 2100 more pedestrians for the non-crossing class. PIE has only one benchmark, with 1322 non-crossing and 512 crossing pedestrians.

JAAD sequences are separated into 177, 117, and 29 for train, test, and validation splits, discarding sequences with lower resolution, adverse weather, and night conditions. For PIE the split is the same as in [RKKT19], leaving set 3 videos for testing, sets 4 and 5 for validation and the sets $1, 2$ and 6 for training.

The input sequence length is 16 frames ($\approx$ 0.5 s). The last frame used in the observation data is between 1 and 2 seconds before the crossing event. Due to the high dynamicity of pedestrians, A longer prediction horizon is impractical for deterministic approaches. Sampled sequences overlap is 60% for PIE and 80% in JAAD.

Input data is composed of the following possible **features**: bounding boxes coordinates, ego-vehicle speed (continuous in PIE and categorical in JAAD). 2D pose information is also available, extracted using OpenPose architecture [CHS+19]. Bounding boxes are used to crop full image frames in different ways: *local box* (bounding box crop), *local context* (1.5 times wider bounding box crop) and *local surround* (*local context* without *local box* pixels, changed by grey pixels).

For measuring models' performance, different evaluation metrics are used. In binary classification two classes are used: positive (crossing) and negative (non-crossing). However, these class roles can be interchanged to calculate the same metrics for each class independently. Before describing them, four terms must be explained related to the correctness in the classification of each sample.

- True Positives (TP): samples classified as positive which are labeled as positive.
- False Positives (FP): samples classified as positive which are labeled as negative.
- True Negatives (TN): samples classified as negative which are labeled as negative.
- False Negatives (FN): samples classified as negative which are labeled as positive.

With these four terms we can measure the performance of the classifier method with the following metrics, based on the previous ones:

- **Accuracy**: Measures the correctness of the model, as a percentage of correctly classified samples between all the samples. In eq. (2.1) we can observe its definition. Being N the total number of samples evaluated, the function $1(y_i = \hat{y}_i)$ returns 1 if

[3] https://github.com/ykotseruba/PedestrianActionBenchmark

the prediction for sample i ($\hat{y}_i$) is equal to the ground truth value for that sample (y_i). However, in imbalanced problems, a high accuracy value can be misleading. For example, in a binary classification scenario, if there is a $4:1$ ratio of negative vs positive samples, if all samples are negatively predicted, the model reaches an accuracy of 80% while if 75% of positive and negative samples are correctly predicted, the accuracy is 75% while obtaining better results on average in both classes. Due to this, it is recommended to use the following metrics which represent a robust summarization of the performance of the model. These metrics (precision, recall, and F1 Score) are calculated only for the crossing class since the samples of this class are considered corner cases of utmost importance (highest risk for the pedestrian).

$$Accuracy = \frac{1}{N}\sum_{i}^{N} 1(y_i = \hat{y}_i) \tag{2.1}$$

- **Precision**: also known as positive predictive value. Its value represents the percentage of correct positive classification (true positives) among all positive classified samples. With higher precision, the system reduces the number of false positives. See eq. (2.2) for its equation. A precision of 1 indicates that 0 false positives were detected.

$$P = \frac{TP}{TP + FP} \tag{2.2}$$

- **Recall**: also known as sensitivity. Its value represents the percentage of positive samples correctly classified between all positive labeled samples in the dataset. With higher recall, the system reduces the number of false negatives. A recall of 1 indicates that the model correctly classifies all positive samples. In eq. (2.3), the equation for recall calculus is displayed.

$$R = \frac{TP}{TP + FN} \tag{2.3}$$

- **F1 Score**: calculated as the harmonic mean of precision and recall, being between 0 (worst classification) and 1 (perfect classification). Both precision and recall are weighted equally in this metric. In eq. (2.4), the metric equation is shown.

$$F1 = 2\frac{P \cdot R}{P + R} \tag{2.4}$$

- **Receiver operating characteristic (ROC) Curve**: the previous metrics are calculated with a fixed confidence threshold (in the benchmark, 0.5 is used). However, this metric is the best way to see if positive samples are classified with high confidence (near 1) and negative with low confidence (near 0). In the vertical axis, the recall (also true positive rate or sensitivity) is plotted against the false positive rate (or probability of false alarm) in the horizontal axis. If the resulting curve is the diagonal

line, the model behaves the same as a random class choice. Best performing models tend to reach the perfect classification score (point $(0, 1)$) at some point of their curve. In fig. 2.15, a graphical example of this explanation is presented with three different curves.

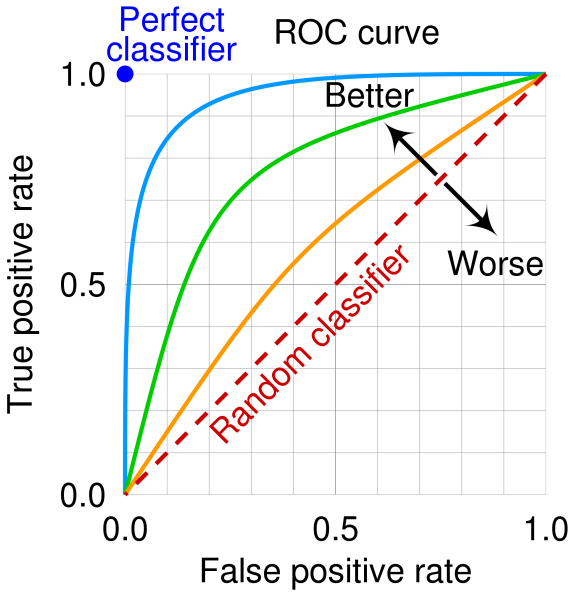


Figure 2.15: Example of ROC curves. Top-left dark blue dot represent the ideal classifier. The farther the curve moves away from this point, the worse the classifier is. Source[4].

- **Area under the ROC curve (AUC)**: it corresponds to the probability that a higher confidence is given for a randomly positive labeled sample than for a randomly negative labeled one. It is the average of all values in the ROC curve, or the integral in the case of continuous curve.

2.5 Conclusions

In this chapter, **pedestrian crossing action anticipation** task has been introduced, with a discussion of the current state-of-the-art solutions and datasets available for it. The following conclusions extracted from this extensive analysis:

- **Lack of complete datasets focused on this task**. While there are good options available with low restrictions (JAAD [RT18b] and PIE [RKKT19]) and also with company-level restrictions (Stanford-TRI Intent Prediction (STIP) [LAC+20] and TITAN [MDC20]) none of them include 3D information for a more advanced understanding of the environment. A recently adaptation of nuScenes [CBL+20] called PePScenes [RYL+20] is the most complete choice. However it is not yet available.

[4]MartinThoma, CC0, via Wikimedia Commons (https://upload.wikimedia.org/wikipedia/commons/3/36/Roc-draft-xkcd-style.svg).

- **Pattern-based methods dominate the current state of the art**. Especially models based on neural network framework. Recurrent architectures are also mixed with convolutional 2D approaches to gain insight into the temporal information.
- **Available Crossing data is imbalanced**. Pedestrian crossing action anticipation is a complex temporal task that is affected by an imbalance between classes. Crossing situations are considered edge cases that occur much less frequently. Deep learning models struggle during the learning process, since they converge to the dominant classes. Training strategies must be developed to mitigate this problem.
- **Image data is the main source of information in current state of the art**. Due to the popularity of convolutional approaches, which extract information from context or pedestrian's appearance in an unsupervised way.
- **There is a lack of standardization in evaluation**. Tasks related to pedestrian crossing behavior have evolved until the current date without a benchmark. In addition to the use of different data partitions between jobs, there are two strands within this task: crossing action prediction and crossing intention anticipation.

2.6 Objectives

After reviewing current state-of-the-art methods and exploring the data available for the crossing action prediction task in detail, we have concluded with a list of relevant aspects. Supported by this list, the objectives of this thesis are as follows:

1. To develop deep learning methods to anticipate pedestrian crossing behavior, leveraging both intrinsic and extrinsic pedestrian information.
2. To measure the influence of different input variables and determine which are more relevant in the anticipation outcome.
3. To test the developed models in real data from public datasets, highlighting possible data problems to overcome.
4. To evaluate the performance of the developed models on a specialized benchmark for pedestrian crossing action prediction.
5. To improve the developed system looking for a good ratio between lower resource consumption and better results, making it viable for real-time constraints in the future.

Chapter 3

VRU classification

Figure 3.1: Isometric view of a typical urban situation.

Today, object detection is one of the most important tasks in the field of computer vision. Its objective is the localization and classification of objects in data captured with a sensor, usually image data from a camera. Over the last decade, this line of research has experienced a large evolution thanks to the development of deep learning and especially of Convolutional Neural Networks (CNNs). Several public large-scale datasets have been released, such as Microsoft Common Objects in Context (COCO) dataset [LMB+14], with 80 different categories and 1.5 million objects, and Open Images [KDA+17], with 600 categories and near 16 million objects labeled. However, both datasets are oriented to the general case without any specialization. For this reason, there is no distinction between different types of Vulnerable Road Users (VRUs), since humans, motorcycles, and bicycles are considered different classes. VRU types distinction is important for an autonomous vehicle to understand its environment and the importance of each agent and/or object detected, especially in the urban case. As an example, an urban situation is shown in fig. 3.1. An object detector trained on one of the previously cited datasets is

deployed in the red car. To distinguish between the different types of VRUs it is possible to apply hand-made postprocessing techniques, such as joining different objects together. However, this rule is prone to error, since it cannot generalize to all situations (e.g., a motorcycle parked in the street and a human overlapping with it who is walking on the sidewalk). Another problem with the detected humans is their importance concerning the ego-vehicle road situation. For example, people sitting on a chair in a park, outside a restaurant, or just waiting for the bus are less likely to cross than standing people walking in the vicinity of the road.

Object detection is the first step in the pipeline for predicting the crossing action performed by pedestrians (see fig. 3.2). VRUs classification can help in the reduction of computations since people who do not interact with the road situation can be filtered before the tracking and other posterior steps such as action prediction.

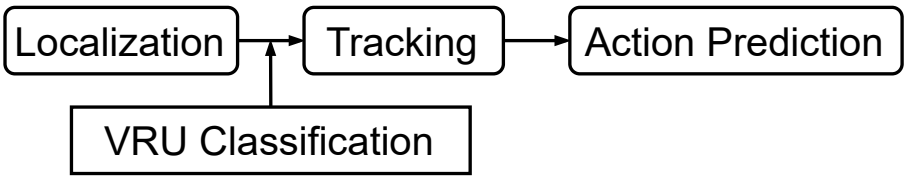


Figure 3.2: Diagram of the pipeline for pedestrian crossing action prediction.

In this chapter, a VRU classification system is proposed based on two-dimensional (2D) CNNs. Before detailing the system, the data used is presented, composed of different Autonomous Vehicle (AV) datasets. Moreover, the dataset is extended with additional external samples to fight the imbalance between classes. After presenting the data, the system is described and the results are detailed.

3.1 VRU classification dataset

The dataset was initially based on data extracted from four well-known public datasets in the AV community:

- CityPersons [ZBS17]: subset of Cityscapes dataset [COR+16] focused on person annotations. It contains bounding boxes for six types of objects: *ignore regions* (fake humans, e.g., people on posters, reflections), *pedestrians*, *riders*, *sitting people*, and *group of people*.
- KITTI Object Detection Evaluation 2012 [GLSU13]: comprises three main classes: *pedestrian*, *cyclist* and *car*. It also includes a minority of samples for *person sitting*.
- INRIA [DT05]: one of the first dataset focused on person detection. It is smaller than the rest of the used datasets. It includes only *pedestrian* class. However, in contrast with the rest of the datasets, these samples are of a bigger size, because it was not created for AV applications.
- Tsinghua-Daimler Cyclist Detection Benchmark Dataset [XFY+16]. Dataset focused mainly on cyclists. However, it includes a considerable number of pedestrian

samples. A small subset of samples belongs to *other riders*: *motorcyclist*, *tricyclist*, *wheelchair user*, and *moped rider*.

To combine all of the previous datasets, four different classes were defined: *sitting person*, *standing pedestrian*, *motorcyclist*, and *cyclist*. However, each dataset has different format concerning localization coordinates (bounding box) and classes, so we performed a class homogenization on each dataset:

- Citypersons: *pedestrian* as *standing pedestrian*, *riders* as *cyclist*, and *sitting person* as *person sitting*. *Motorcyclist* class has zero samples in the original annotations.
- KITTI: Similar to the previous one: *pedestrian* as *standing pedestrian*, *cyclist* as *cyclist* and *person sitting* as *person sitting*. Again, *motorcyclist* class has zero samples from this dataset.
- INRIA: Only *standing pedestrian* is represented by *pedestrian* class. The rest of classes have zero representation in this case.
- Tsinghua-Daimler: *cyclist* as *cyclist*, *pedestrian* as *standing pedestrian* and *motorcyclist* as a combination of *motorcyclist* and *moped rider*. *Sitting person* does not have representation in this dataset.

The final class distribution is shown in table 3.1. It is clear that the dataset is not balanced, with 55.0% of *standing pedestrian* samples, 41.0% of *cyclist* samples, 1.7% of *motorcyclist* samples, and 2.3% of *sitting person* samples.

Table 3.1: Initial class distribution in selected datasets.

	Pedestrian	**Cyclist**	**Motorcyclist**	**Person sitting**
CityPersons	19683	2189	0	1217
Kitti	4487	1627	0	222
Inria	1826	0	0	0
Tsinghua	8942	22173	1049	0
Total	**34938**	**25989**	**1049**	**1439**

To fight th imbalance problem, we included new samples extracted from the internet for both minority classes: *motorcyclist* and *sitting person*. Some visual examples of the new samples are shown in fig. 3.3. To annotate the bounding box information for these new images, we followed a semiautomatic approach using the detection results of a Mask R-CNN [HGDG18] model with an end-to-end ResNeXt-101-64x4d backbone [XGD+17], trained on COCO dataset [LMB+14].

Human labeling intervention is aimed at filtering false positives, false negatives, and correcting bounding boxes. In the case of *motorcyclist* samples, an additional step is done to join *motorcycle* and *person* classes detected by the model.

In addition to internet sampling, we extracted *motorcyclist* samples from CityPersons dataset, changing 259 wrong labeled *cyclist* samples. The new distribution of classes (see table 3.2) show a minor imbalance with a 44.3% of *person*, 32.6% of *cyclist*, 11.3% of *motorcyclist*, and 11.8% of *sitting person* samples.

Table 3.2: Final class distribution in selected datasets.

	Pedestrian	Cyclist	Motorcyclist	Person sitting
CityPersons	19683	2189	0	1217
Kitti	4487	1627	0	222
Inria	1826	0	0	0
Tsinghua	8942	22173	1049	0
Added samples	0	0	7558	7848
CityP. splitted	0	0	259	0
Total	**34938**	**25730**	**8866**	**9287**

Figure 3.3: Additional image samples obtained from internet. Motorcyclist (bottom row) and sitting person (top row).

3.2 System description

3.2.1 Problem formulation

The main task of the classifier is to learn to distinguish correctly between VRU types to reduce the workload of the steps after object detection in the pedestrian crossing action prediction pipeline. As input, the VRU image information is cropped using the ground truth bounding box information, with an additional percentage of context on both width and height. The output of the system is the VRU class from the previous dataset whose probability is higher. Since the model output logits or raw predictions for each class, the softmax/softargmax function is used to normalize these logits into a probability distribution. In fig. 3.4, a visual description of the problem is portrayed. Since the purpose of this project is the VRU classification, the detection part of the system is assumed to be ideal.

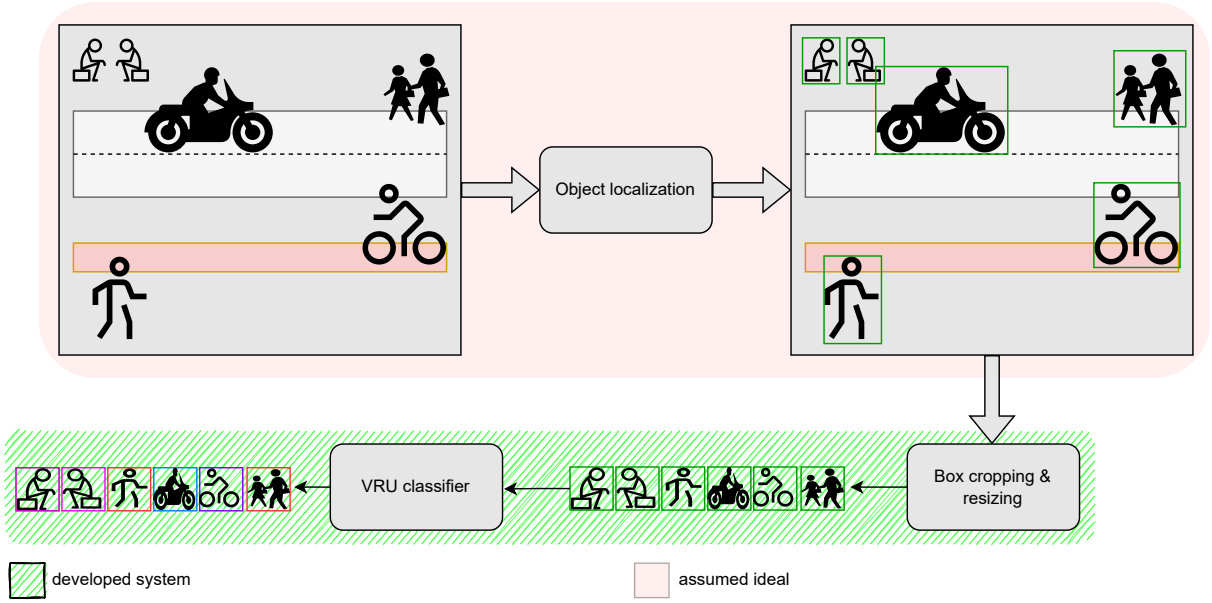


Figure 3.4: Detailed diagram of the complete pipeline of VRU detection. Only classification steps are performed in this experiments, parting from an ideal detection.

3.2.2 Model architecture

Our model is composed of a CNN backbone pre-trained on ImageNet [RDS+15] followed by a fully connected layer. The last layer of the CNN model is deleted and the rest of the backbone is connected to the new fully connected layer, adapted for the new number of classes. Different CNN architectures have been used as backbone. They are detailed in the following list:

- **SqueezeNet** [IHM+16] (213.6 fps). It is trained using an input size of $(227 \times 227 \times 3)$. This is the smallest chosen model, with $1.24M$ parameters. This architecture is fully convolutional. For this reason, instead of using a fully connected, the last convolutional layer is substituted by another convolutional layer with a 1×1 kernel and 4 filters equal to the total number of VRU classes.
- **AlexNet** [KSH12] (197.9 fps). It is also trained using an input size of $(227 \times 227 \times 3)$. It has $61M$ parameters. The last fully connected layer is substituted with another fully connected layer with an output size of 4. This substitution is identical in the following architectures. Despite its size, it has similar results on ImageNet to **SqueezeNet**.
- **GoogleNet** [SLJ+14] (183.0 fps). In this case and the rest in this list, the input size during pretraining was $(224 \times 224 \times 3)$. It has $7M$ parameters.
- **VGG-16** [SZ15] (69.9 fps). With $138M$ parameters, this is the largest model tested.
- **ResNet-50**[HZRS15] (87.8 fps). $25.6M$ parameters, however it gets better results than the VGG-16 model on the ImageNet validation set.

- **ResNet-101**[HZRS15] (61.3 fps). $44.6M$ parameters. This is the best performing model on the ImageNet validation set among the chosen ones.

3.2.3 Training strategies

Two training strategies are followed in the VRU classifier, focused on data rather than the model specifications: training on the original dataset and training on the extended dataset. The data distribution is 60%, 25%, and 15% for training, testing, and validation sets, respectively. The training set is shuffled on every epoch, to prevent the model from learning the order of the data.

3.2.4 Input data preprocessing

In the first set of training experiments, VRU cropping was done with the deformation of the images, since the input of the model is square-shaped, and most of the bounding boxes have a rectangular shape. A small ablation study was done comparing the previous cropping method with maintaining the boxes' aspect ratio in the crop. In this cropping method, bounding boxes are trimmed as a square region from the image, with the side equal to the largest between the height and width of the box. This method obtained better results in the validation set. For this reason, it was chosen as the used method in experimentation. The resulting image was normalized using ImageNet [RDS+15] mean and standard deviation.

Finally, additional context of the VRU can be included in the input image. To check if this contextual information becomes useful for the learning process, another ablation study was done adding different percentages to height and width: 0%, 5%, 10%, 20%, and 40%.

3.2.5 Oversampling of minority classes

In order to further reduce the imbalance in the extended dataset, an oversampling strategy is applied to the additional data included in the minority classes, consisting in using data augmentation on training time. Below is a list of possible augmentations from where three were selected randomly at most on each image:

- **Scaling** (fig. 3.5b): taken randomly from the interval [80%, 120%]. The same scale factor is applied to both image axis (x and y).
- **Translation** (fig. 3.5c): applied on x and y image axis, from the interval $[-20\%, 20\%]$.
- **Rotation** (fig. 3.5d): $[-25, 25]$ in degrees, with respect to the center of the image.
- **Addition** (fig. 3.5e): of a single random value from the interval $[0, 30]$ to every pixel of the image in every channel.
- **Cropping** (fig. 3.5f): pixels randomly from $[0, 20]$ interval from each side of the image. Four different values are randomly taken.
- **Horizontal Flip** (fig. 3.5g).

(a) Original image. (b) Scaling. (c) Translation. (d) Rotation.

(e) Addition. (f) Cropping. (g) Horizontal Flip. (h) Gaussian blur.

Figure 3.5: Different data augmentation preprocessing techniques applied during training. The augmentations are exaggerated for better visualization.

- **Gaussian blur** (fig. 3.5h): with randomly selected standard deviation $\sigma \in [3.0, 7.0]$. The higher the σ, the more blurred the image.

3.2.6 Training hyperparameters

Except for the batch size, the same set of hyperparameters was used for all model variants:

- *Fixed learning rate:* $\epsilon = 1 \cdot 10^{-3}$.
- *Maximum number of epochs:* 10.
- *Minibatch size*: 64 in AlexNet and SqueezeNet and 32, for the rest of the models, due to memory limitation.
- *Optimizer:* Stochastic Gradient Descent with Momentum.

3.2.7 Hardware and software requirements

All models were trained using MATLAB 2018b Deep Learning Toolbox on a machine with an Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz and an NVIDIA GeForce GTX TITAN X Graphics Processing Unit (GPU).

3.2.8 Evaluation metrics

To compare the models' performance, the confusion matrix is calculated for each experiment, with the value of Positive Predictive Value (PPV) (precision) and False Discovery Rate (FDR) as the last two rows, and the value of True Positive Rate (TPR) (recall or sensitivity) and False Negative Rate (FNR) (or miss rate) as the leftmost two columns.

Another metric used to summarize the performance of all architectures is the average F1 score over all classes.

3.3 Results

Adding context to the input image is beneficial for learning until a 10% addition, where the maximum is reached in the validation set. After that, extra context becomes useless and even detrimental to network performance. For this reason, all of the following experiments used a 10% addition of image context.

3.3.1 Initial dataset

Table 3.3: F1 Score. Imbalanced dataset.

	Pedestrian	**Cyclist**	**Motorcyclist**	**Person sitting**	**Avg.**
AlexNet	0.9489	0.9446	0.8223	**0.5869**	**0.8257**
SqueezeNet	0.9266	0.9178	0.7079	0.4973	0.7624
Googlenet	0.9404	0.9350	0.7619	0.4964	0.7834
VGG-16	**0.9530**	**0.9520**	**0.8253**	0.4361	0.7916
ResNet-50	0.9468	0.9408	0.7312	0.5434	0.7906
ResNet-101	0.9488	0.9455	0.7531	0.5358	0.7958

On table 3.3, we can see the results obtained in the case of training only with samples obtained from external datasets (initial class distribution). As expected, both majority classes, *standing pedestrian* and *cyclist*, are the best-performing ones, mainly confused between them, since a cyclist waiting on the road can be mistaken for a standing pedestrian and vice versa. However, the minority classes obtain quite different results, especially in the case of *sitting person* class. The best performing network on average is VGG-16, and in fig. 3.6, its results are detailed. Regarding the *sitting person* class, the majority of the samples are confused with *standing pedestrian*, which is its most similar class. While *motorcyclist* class has even less percentage of samples than *sitting person*, it obtains far

better results. This is probably due to the visual difference with *cyclist* class, except for light motorcycles, which can confuse the network.

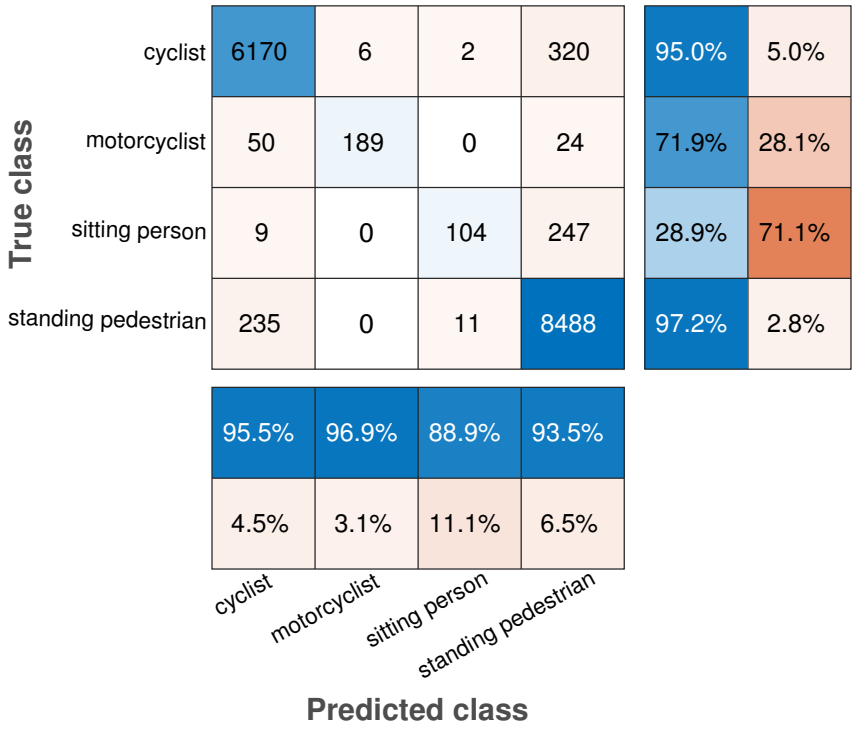


Figure 3.6: VRU classification results with VGG-16 on the initial dataset.

3.3.2 Extended dataset

After including new samples and augmenting the training data of the minority classes the results were greatly improved. On table 3.4, the F1 score of all classes is improved, specially on *sitting person* class. VGG-16 is also the best-performing model in this case. As we can observe in fig. 3.7, minority classes samples are wrongly detected only in a few cases, reaching nearly perfect scores for the *motorcyclist* class.

Table 3.4: F1 score. Equalised dataset.

	Pedestrian	**Cyclist**	**Motorcyclist**	**Person sitting**	**Avg.**
AlexNet	0.9557	0.9588	0.9763	0.9346	0.9564
SqueezeNet	0.9588	0.9635	0.9782	0.9397	0.9601
Googlenet	0.9733	0.9734	0.9864	0.9665	0.9749
VGG-16	**0.9818**	**0.9804**	**0.9909**	**0.9784**	**0.9829**
ResNet-50	0.9717	0.9720	0.9861	0.9670	0.9742
ResNet-101	0.9689	0.9699	0.9848	0.9595	0.9708

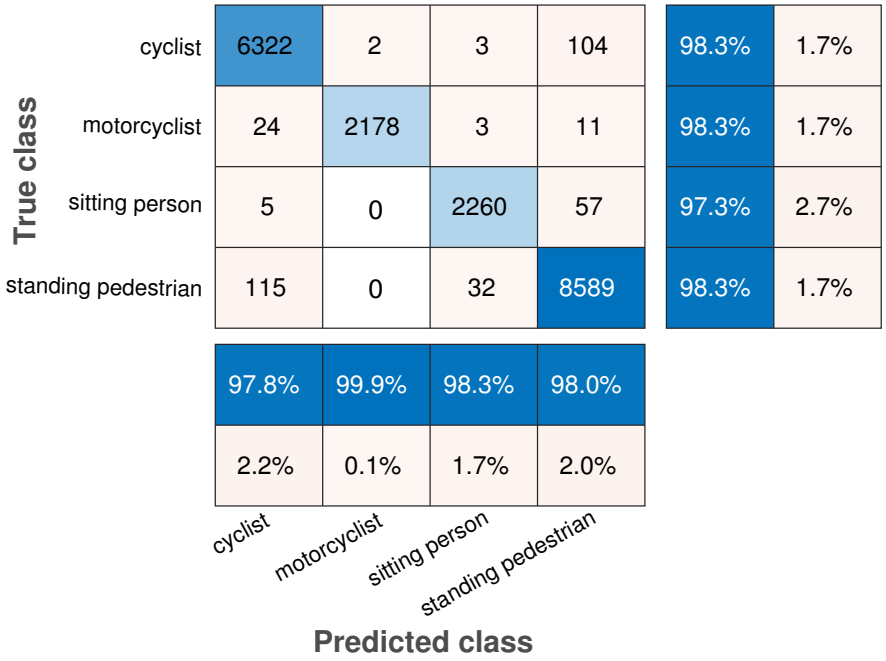


Figure 3.7: VRU classification results with VGG-16 on the extended dataset.

3.4 Conclusions

In this chapter, a VRU classification system has been proposed. The final goal of this system is the filtering of VRU agents in the urban road scenario which could help reduce the overall computation in the AV perception pipeline. Before the training experiments, a data labeling task was performed by scrapping images from the internet to fight imbalance in the chosen combination of datasets.

After this data labeling task, accelerated by human-validated automation, an initial experiment was conducted by training models based on several CNN backbones. In the first experiment, without the external labeled images, the model classified poorly both minority classes. The reason for it was the imbalance between classes. To further improve these results, the imbalance has been addressed through the inclusion of new data, data augmentation, and oversampling. Overall, the results have improved considerably, equalizing all classes. The increase in minority ones was noteworthy, achieving the same level as the majority classes, which are also slightly improved. This is verified quantitatively by the F1 score results, which are on average over 95% with a small deviation, in contrast with the previous average of under 80% with a high deviation.

By virtually over-sampling the minority classes using data augmentation techniques, the imbalance problem is reduced and the model can generalize better. Despite its good results, the images used to augment the minority classes were downloaded from several places on the internet, combining different camera points of view. This variety promotes model generalization, allowing its application in other scenarios, such as video surveillance

and vehicle-to-infrastructure communication. However, for the application pursued in this work, it might be a better option to specialize the model in the ego-vehicle point of view, since the camera is fixed in this position while the ego-vehicle is moving.

Bounding boxes annotations are obtained as the output of a state-of-the-art object detection system. By using a human validation step, detections increase their quality. Looking at the promising results, the possible improvement for the case of hand-labeling is small, and the time saved on labeling using this technique is considerable. Automatic labeling with human supervision proved to be successful. With the future improvements brought by the progress in object detection (Mask R-CNN model is nearly 4 years old), even better quality objects will be detected and human supervision will be progressively disappearing.

Data samples are geographically biased for both of the majority classes: most of the *standing pedestrian* samples are obtained in Germany, while the vast majority of cyclists belonging to the Tsinghua dataset, recorded in China. In addition, internet samples used to increase the size of minority classes are mostly stock pictures with better quality than the ones obtained in the AV environment. Nighttime samples are not present in the dataset and the weather conditions are mostly good or medium with a little number of samples with bad weather conditions. As detailed in section 2.2, several Autonomous Driving (AD) datasets have emerged in the last years. By using our semiautomatic approach, samples from these new datasets can be incorporated in the future, enriching both of them geographically and in terms of temporal and climatic diversity.

The purpose of this work is to serve as a starting point for VRU classification focused on filtering VRU agents for the AV environment. The prediction obtained with the current solution is limited to one object per input image, so a previous stage of object detection is necessary. Both stages can be joined into a single one to optimize resource usage. However, the combined VRU detection system will not yet be ready for the AV environment, since it will only serve as an initial filtering stage, being useless for cases where temporal information is crucial. In the next chapters, different temporal methods are presented for the task of pedestrian crossing action prediction.

Chapter 4

STCAP: Combining spatial and temporal neural networks for crossing action prediction

In this chapter, the first model developed in this work for pedestrian crossing action prediction is detailed. First, in section 4.1, the problem is described. After that, the dataset used is presented in section 4.2 and the model is detailed (section 4.3). After explaining the selected hyperparameters for the model (section 4.4), the hardware and software details (section 4.5), and the evaluation metrics (section 4.7), the experiments are explained in section 4.8. Finally, the results are shown in section 4.9 and a conclusion is presented in section 4.10.

4.1 Problem description

Future crossing action prediction of the target pedestrian is modeled as a binary classification where an input sequence is provided to the model and it outputs a probability of crossing in a future short-term time horizon of $1s$.

The input sequences are defined following a sliding window approach. Each target pedestrian has a track composed of a set of input sequences from different sources (e.g., image, categorical context). The combination of these features gives rise to the input macro sequence $\mathbf{X} = \{X_0, X_1, \ldots, X_{P-1}\}$, where P is a positive integer representing the length of the track for the target pedestrian. For each index t an input sequence of length $N + 1$ is obtained as a set of features $\mathbf{X}_t = \{X_{t-N}, X_{t-N+1}, \ldots, X_t\}$, where N is a positive integer representing the number of past frames included in the sequence, starting from $t - 1$. The output is defined as a one-dimensional binary label Y_{t+M} where $t + M$ is the index of the frame to be predicted. Thus, each pedestrian track is divided in $S = P - N - M$ input sequences, with $t \in \mathbb{N} : t \in [N, P - M - 1]$. To clarify the sliding window formulation, an example is shown in fig. 4.1.

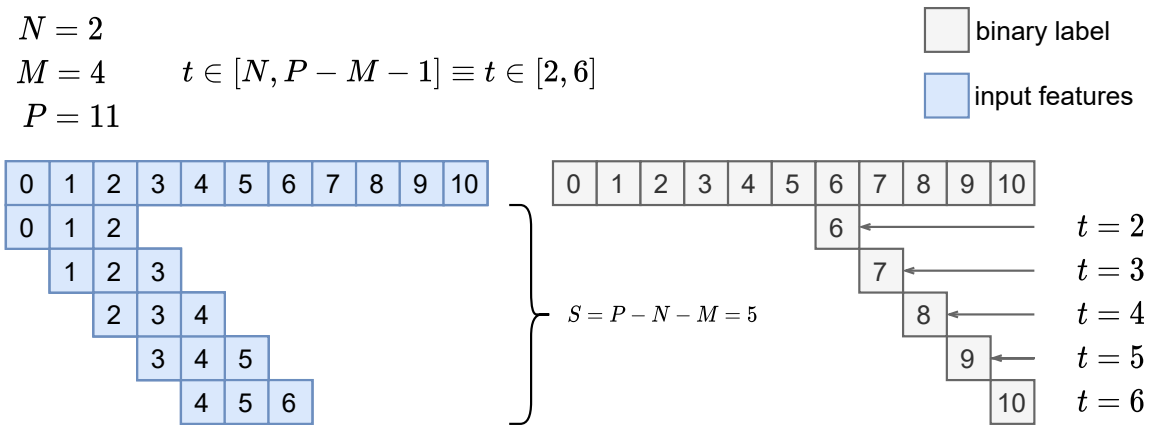


Figure 4.1: Example of the sliding window strategy used to partition the sequence.

4.2 Data

For all of the experiments included in this chapter, Joint Attention in Autonomous Driving was used. As detailed in section 2.2, Joint Attention in Autonomous Driving (JAAD) comprises 346 videos filmed in naturalistic scenarios from different cities in the world, with different light and weather conditions. The duration of the videos range from 5 to 10 s. Videos are filmed with different dash cameras, with 10 videos with a resolution of 1280×720 pixel (px) and the rest with 1920×1080 px resolution. The framerate for the videos is 30 frames per second (fps) with the exception of 8 videos at 60 fps.

Training, validation, and testing sets are obtained following the default split strategy proposed in the official repository. In these splits, videos with small resolution and low visibility (e.g., night scenes, heavy rain) are excluded. The filtered total number of videos is 323, with 177 ($\approx$ 55%)for training, 29 ($\approx$ 9%) for validation and 117 ($\approx$ 36%) for testing.

Chosen input and output sequence lengths are 0.5 and 1 s, respectively. At 30 fps and following the notation in section 4.1, the complete input sequence length is 16 frames ($N = 15$) and $M = 30$ frames in the future.

$$I_n(x, y, c) = \frac{I(x, y, c) - \mu_c}{\sigma_c}, x, y \in [0, W_s), [0, H_s) c \in [0, 2] \tag{4.1}$$

The input of the model is composed of image data and contextual categorical variables in some experiments.

From each video sequence, pedestrian tracks are extracted using their ground truth two-dimensional (2D) bounding box annotations with crossing behavior. Height and width are equalized to the maximum of both values to avoid future deformation. Bounding boxes with a height lower than 50 px were removed from the training set. Using occlusion annotations, fully occluded samples in a pedestrian track are also removed from the training set. After the extraction, each image was resized to an square shape, with height H_s and width W_s equal to 224 ($224 \times 224 \times 3$). After that, the resulting image $\mathbf{I}$ is centered and z-score normalized (eq. (4.1)) using mean $\mu = [0.485, 0.456, 0.406]$ and standard deviation

$\sigma = [0.229, 0.224, 0.225]$ of ImageNet since all Convolutional Neural Network (CNN) feature extractors are pre trained using this dataset, resulting in the normalized image I_n. Values inside μ and σ correspond to the mean and standard deviation of red, green, and blue channels (0, 1, and 2), respectively.

In order to measure the influence of additional variables, three categorical annotations are extracted from ground truth:

- **Looking / gaze direction**. Binary variable with a value of 1 if the pedestrian is looking at the ego-vehicle and 0 otherwise.
- **Orientation**. Discretized value of the orientation of the target pedestrian with respect to the ego-vehicle with four possible categories from 0 to 3: front (walking towards the vehicle, parallel to its trajectory), back (walking backward the vehicle, parallel to its trajectory), left (walking perpendicular to the vehicle's trajectory, in the left sense), and right (walking perpendicular to the vehicle's trajectory, in the right sense).
- **State of movement**. A binary variable that indicates if the pedestrian is standing (0) or moving (1).

Left and right orientation have different importance depending on the side of the road where the pedestrian is placed. For example, a pedestrian on the right sidewalk of the road walking with a left orientation (approaching the road) is more likely to cross than walking to the right (moving away from the road). For this reason, the center coordinates of the bounding box were included as an additional variable, min-max normalized to the height and width of the full image frame, to minimize the dependency on the used camera sensor.

4.3 Model general architecture

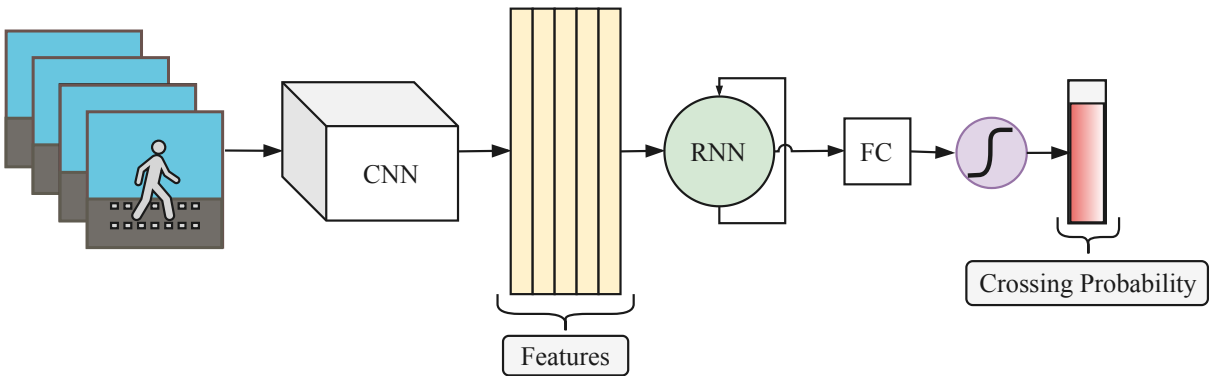


Figure 4.2: Diagram describing the proposed method.

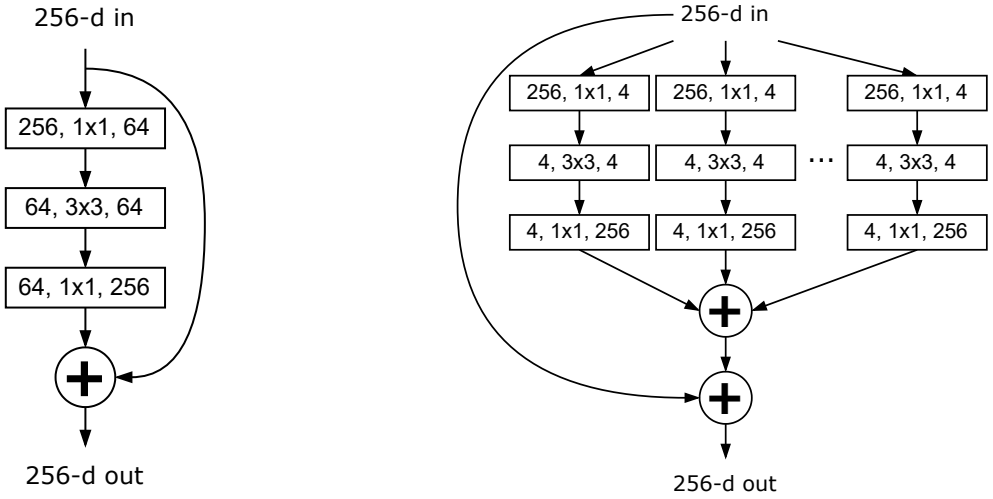
The model is divided into two main parts: a CNN image feature extractor, and a many-to-one recurrent neural network (RNN) encoder. As detailed on the high-level diagram

on fig. 4.2, the CNN output feature vector of each input sequence sample is introduced sequentially to the RNN. When all the samples of the sequence are processed, the last output of the RNN is converted to the crossing action probability using a fully connected layer with a sigmoid activation (see eq. (4.2)) at the output, which transforms it to a value between 0 and 1, representing the probability of crossing in the future time interval used for training.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{4.2}$$

4.3.1 Feature extractor

Four alternative strategies were used to extract the features from RGB sequences:



(a) ResNet basic block with skip connection strategy.

(b) ResNeXt basic block.

Figure 4.3: Basic blocks of both classification CNN families used as feature extractors. Boxes represent convolutional layers. Text inside boxes: number of input channels, size of 2D kernel, and the number of output channels. Based on figure in [XGD+16].

- CNN models from the ResNet [HZRS15] family pre-trained in ImageNet: ResNet architecture was conceived to solve the vanishing gradient problem present in deep sequential or plain networks, such as the VGG family, which makes it harder to train with the increase of networks' depth. By introducing a skip connection between convolutional layers (i.e., input activations are added to the output of the layer, see fig. 4.3a), it is possible to train deeper models without this problem. Five different variants have been used in the experiments: 18, 34, 50, 101 and 152 convolutional layers.
- CNN models from the ResNeXt [XGD+16] family pre-trained in ImageNet: this architecture improves the results of ResNet equivalent models by dividing each convolutional layer into a group of convolutional layers with less number of filters ag-

gregated at the end (see fig. 4.3b), capturing stronger representations of the input data. Two different variants were used in the experiments: 50 and 101 layers models, both with a cardinality value of 32 (number of parallel convolutional layers in each block) and an internal dimension of each block (or width) of 4 and 8 respectively.

- ResNet34 encoder of a custom convolutional autoencoder: an autoencoder is an encoder-decoder architecture trained for input reconstruction in a self-supervised manner. The input image is compressed into a context vector and with this information, the decoder has to reconstruct the input image (see fig. 4.4) Only the trained encoder is used as a feature extractor. The network was trained with a constant learning rate of 10^{-3} and using Binary Cross-Entropy (BCE) measure as the loss function.

- Convolutional encoder of a SegNet [BKC16] based autoencoder[1]: SegNet model is an encoder-decoder architecture initially designed for image segmentation (i.e., assigning to each image pixel a class). The selected model was pre-trained in Cars Dataset [KSDF13]. The encoder architecture is the same as in the VGG16 architecture, without the fully connected layers used for classification.

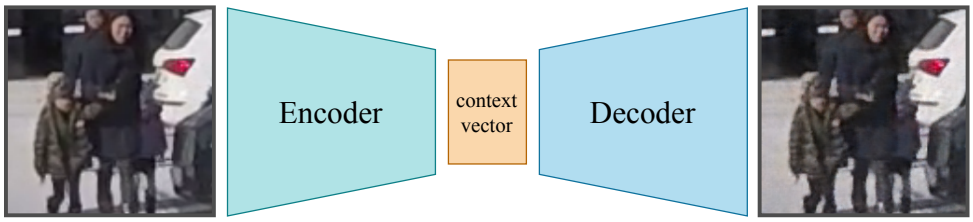


Figure 4.4: High-level convolutional autoencoder example.

We selected ResNet and ResNeXt families due to their competitive results on ImageNet ranking. Since ResNeXt is an evolution of ResNet, we also expected better results.

Autoencoders architectures were selected in order to check if the compressed representation (context vector) includes information useful for crossing action prediction.

All of the feature extractors have different output shapes. After deleting the last fully connected layer used for the classification of ImageNet (1000 classes), the last layer of ResNet and ResNeXt models is an adaptive average pooling layer with an output height and width of 1. The channel number C is maintained, and it depends on the network depth: 512 for ResNet 18 and 34 and 2048 for the rest of the models. This output of dimensions $1 \times 1 \times C$ is flattened into a one-dimensional vector of size C.

In the autoencoder cases, the output is of size $7 \times 7 \times 512$ ($C = 512$). An average pooling layer with a 7×7 kernel was used to reduce the dimensionality to a tensor of size $1 \times 1 \times C$. This output, similar to the one from the classification families, is flattened into a one-dimensional vector of size C.

This vector is obtained for each input image. After obtaining all features from the input sequence, they are introduced sequentially from X_{t-N} until X_t into the recurrent model.

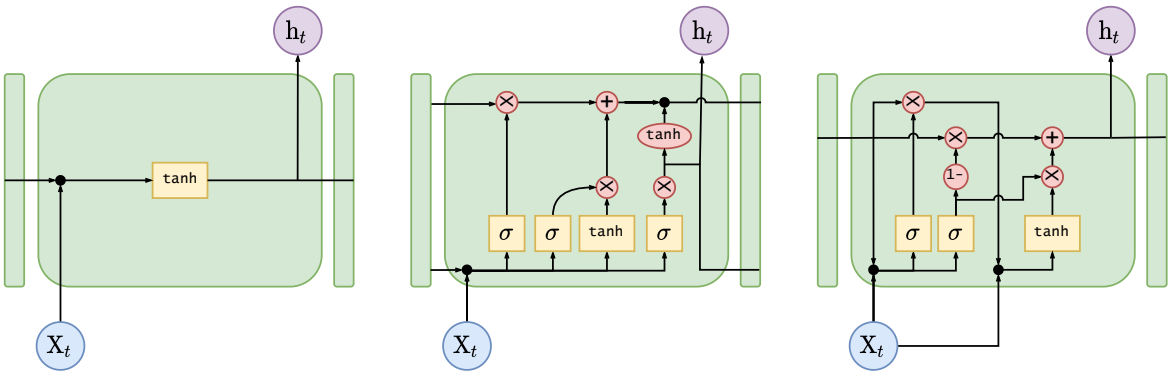
[1] https://github.com/foamliu/Autoencoder

$$D = \mathbf{min}\left(\left\lfloor \frac{N_c}{2} + 1 \right\rfloor, 50\right) \tag{4.3}$$

In addition to the convolutional feature extractor in charge of image data, categorical variables are used as input to the model. These variables are embedded in order to learn the multidimensional relationship among their possible categorical values. Each embedding layer has an output dimension D established following the heuristic proposed by Jeremy Howard[2]. In eq. (4.3), the heuristic is shown, where N_c is the number of categories for a variable.

Bounding boxes center coordinates are not processed by an embedding layer. Embedding output size following eq. (4.3), is 2, 3 and 2 for gaze direction, orientation and moving state, respectively. Including center coordinates and CNN feature extractor output dimension C, the maximum dimension for the concatenation tensor is $C+2+3+2+2 = C+9$.

4.3.2 Many-to-one RNN network



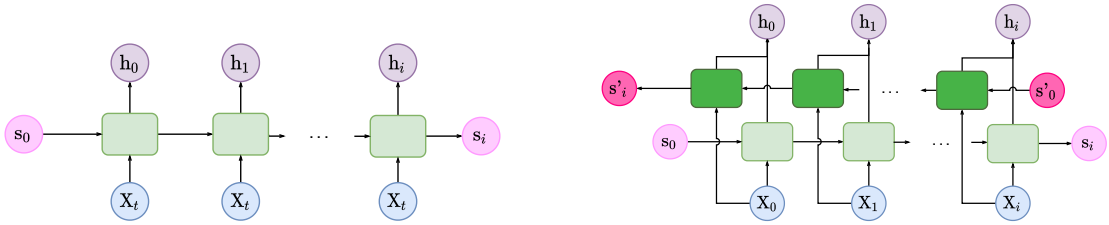
(a) Vanilla RNN. (b) long short-term memory (LSTM). (c) gated recurrent unit (GRU).

Figure 4.5: Detailed diagrams of the base RNN and two of their most popular variants. X_t is the input at time t and h_t is the output of the recurrent layer. Yellow squares are feed-forward layers with the text inside representing the activation function (σ is the sigmoid function and *tanh*, hyperbolic tangent). Red circles represent point-wise operations ($1-$ represents the operation $1-z$ where z is the input to the operator). Black dots are used for vector concatenation.

RNNs have been notably used in a multitude of problems inside Computer Vision and Natural Language Processing fields (e.g., image captioning, language translation). These networks are based on vanilla neural networks, but incorporate temporal information persistence by using loops (see fig. 4.5). However, the vanilla RNN struggles to learn from long sequences (long-term dependencies), difficulting the training task with vanishing and exploding gradient problems. To avoid this problems, two RNN variants were selected for the experiments: LSTM [HS97] and GRU [CvMG+14]. These models are more complex than vanilla RNN, including additional layers called gates which allow better control in the learning process. GRU networks are less complex than LSTM networks and obtain

[2] https://github.com/fastai/course-v3

similar results in different sequence modeling problems [CGCB14]. For this reason, we used both in the experiments in order to check if these findings are met for our problem as well.



(a) Unidirectional RNN.

(b) Bidirectional RNN.

Figure 4.6: High-level diagram of unidirectional and bidirectional RNNs.

Finally, bidirectional variants [GS05] of both LSTM and GRU models are also used in the experiments. These variants are composed of two RNN networks that process information in both directions, from the oldest sample to the most recent and vice versa (see fig. 4.6 for a detailed diagram). This allows the model to gain future knowledge for each sample. We experimented with them to quantify the improvement in the results by using this extra information.

4.4 Hyperparameter setting

After an ablation study using the grid search technique and the validation set, the final configuration used for the model was the following:

- RNN hidden dimension: 4.
- Number of stacked RNN layers: 1.
- Dropout rate applied to RNN output: 0.5.

More complex models were prone to overfitting and their simplification was needed to overcome this problem.

4.5 Hardware and software requirements

To train and test the model proposed, we used PyTorch framework [Pas+19]. All of the experiments were performed using an NVIDIA GeForce GTX TITAN X Graphics Processing Unit (GPU), in a machine used with an Intel(R) Core(TM) i5-4690K CPU @ 3.50GHz and 16 GB of RAM.

4.6 Training details

All of the sections of the models are treated differently during training. Feature extractor weights are frozen and the only trained parts are the recurrent model and the categorical

embeddings in the experiments that require it.

Adam optimizer [KB14] was chosen with a learning rate of 10^{-4}. To measure the loss, BCE was measured between the target label and the output probability obtained after a sigmoid activation. On eq. (4.4) the formula is provided. N stands for the batch size, l_n is the Binary Cross Entropy value between x_n and y_n, which represents the logit output of the network and the future crossing action label respectively for the sample n inside the batch. σ represents the sigmoid function. L loss vector is reduced using the mean operation to obtain the final loss.

$$L = \{l_1, \dots, l_N\}^\top, \quad l_n = y_n \cdot \log \sigma(x_n) + (1 - y_n) \cdot \log(1 - \sigma(x_n)) \tag{4.4}$$

In order to make deterministic experiments, a fixed random seed was established, to ensure that improvements in the results are not due to a benefitial initial training data ordering.

Finally, to avoid unnecessary computation after overfitting, a validation patience of 5 epochs was established.

4.7 Evaluation metrics

To evaluate the results, four different metrics were used: accuracy, precision (P), recall (R) (0.5 used as confidence threshold) and Average Precision (AP) score. A detailed explanation for the first three can be found in section 2.4. AP is an approximation of the Area under the ROC curve (AUC) of the precision-recall curve. It is calculated following eq. (4.5), beginning from the leftmost point in the curve (highest precision, lowest recall) ($n = 0$) until the last one (highest recall, lowest precision). Precision P_n is selected for each point as the maximum of the remaining curve $R \geq R_n$. This condition smooths the curve and allows calculating the AUC approximation using rectangular buckets between the recall points where there is a change in precision.

$$\text{AP} = \sum_{n=0} (R_{n+1} - R_n) P_{n+1}, \tag{4.5}$$

$$P_{n+1} = \max_{R:R>R_n} P \tag{4.6}$$

4.8 Experiments

Six different experiments have been done following the previous statements:

- **Image feature extraction method importance**: image feature extractors discussed in section 4.3.1 are compared to check if better performance on the ImageNet dataset is translated to better performance on the task of pedestrian crossing action prediction.
- **Image features rescaling importance**: output data from the average pooling layer at the end of the image feature extractors detailed on section 4.3.1 range from 0 to a maximum value which depends on the input and the weights of each convolutional

model. An additional rescaling step is applied to check if there is an improvement in the results. Rescaling is done by dividing the sequence of image features between the maximum value in the image.

- **Influence of additional variables**: additional categorical variables and the bounding box center were incorporated into the model in order to check their influence on the output. The model without any additional variables and with all of them is also compared, to see the overall improvement.
- **LSTM vs GRU vs bidirectionality**: check which of the RNN variants discussed in section 4.3.2 obtain the best results.
- **Final performance**: comparison between the vanilla model and the improved model. The vanilla model does not include any of the improvements included in the previous experiments. On the contrary, the improved model contains all of them.
- **Qualitative results**: to see the transition between both classes, the improved model is trained with an output of dimension 8 instead of 1, inferring the crossing probability of equispaced time steps between 0 and 1 s in the future.

4.9 Results

4.9.1 Image feature extraction method importance

Table 4.1: Different feature extraction methods results.

Method	Accuracy	Precision	Recall	Average Precision
ResNet18	62.68	62.87	98.82	69.08
ResNet34	63.32	65.71	86.75	74.33
ResNet50	65.75	69.63	80.43	75.62
ResNet101	68.95	71.29	84.49	77.16
ResNet152	62.53	64.26	90.59	75.85
ResNeXt50	**70.04**	**74.96**	78.39	79.87
ResNeXt101	69.45	74.14	78.73	**81.20**
ConvAE-ResNet	62.67	62.67	**100.00**	61.64
ConvAE-SegNet	62.67	62.67	**100.00**	68.46

Results per feature extraction method are shown in table 4.1. Pre-trained models from ResNet and ResNeXt families (trained in a supervised manner) obtain better results than Auto Encoder (AE) ones (trained in a self-supervised manner).

In supervised architectures, the increase in the complexity of the network is directly related to the increase in all performance metrics, saturating in the biggest ResNeXt model with an 81.2% of AP, only $\approx 0.3\%$ above the ResNeXt50 model. This saturation is probably due to the limited number of samples available for training. With a bigger training set, this difference should be maintained with bigger models, as happens in the

ImageNet benchmark. Another possibility is that this saturation is due to the difficulty and level of abstraction of the problem compared to static object classification.

For self-supervised architectures, both architectures perform poorly compared to the previous ones. This is shown in the recall value of 100%, which means that the model has converged in predicting that every pedestrian will cross. Although the images are reconstructed quite accurately in both convolutional AEs extractors, the output features of the encoder lack useful information for the RNN module. While being trained in a self-supervised way in JAAD [RT18b] and CARS dataset [KSDF13], both AEs perform poorly in the test set, meaning that the problem probably lies in the chosen training methodology. In ResNet and ResNeXt families, models are trained in a supervised way, letting the backbone learn meaningful high-level features focused on a limited set of classes. For this reason, these features can be translated to other tasks with some level of similarity. However, in the self-supervised training methodology, the model learns to reconstruct the input image. Since this is a regression problem, learned features are at a lower level, which resumes spatial and color information instead of class-level information. Another possible cause of it can be the inclusion of an average pooling layer after training. In the supervised case, average pooling is used during the training stage, affecting the gradients.

4.9.2 Image features rescaling importance

As shown in table 4.2, rescaling input image features contributes to an improvement of 1% in average precision with respect to not scaling them. These results show that the variation in input features slightly penalizes generalization to unseen data. However, normalization affects the recall value for the default threshold (0.5). This problem may be caused by the local normalization applied, which may hinder the network's ability to distinguish between visually similar cases, masking convolutional backbone activations.

Table 4.2: Rescaling image features results.

Normalization type	Accuracy	Precision	Recall	Average Precision
None	65.75	69.63	80.43	75.62
Rescaling	65.89	70.75	77.70	76.84

4.9.3 Influence of additional variables

The incorporation of meaningful data can act as a regulatory factor to allow greater learning generalization. Individually, looking at AP value, orientation, and looking direction are the variables with more weight followed by the state of movement. Those variables are also used by drivers when they infer the pedestrians' crossing intentions. For example, a pedestrian walking towards the road and a pedestrian at the curb looking at the driver's car are more likely to cross than a pedestrian walking parallel to the car and suddenly stopping. In the case of the bounding box center in the image, it has less weight, but it is also an important source of information to distinguish between crossing and non-crossing

cases since from the driver's perspective approximated by the camera's point of view inside the ego-vehicle, bounding boxes near the horizontal center of the image pedestrians with a higher intention to cross than those at the edges of the image. This information is probably used by the network since this input variable achieves the best recall results in the default threshold (87.91%). However, this information is only useful when driving in a straight line, because during a turning maneuver, a non-crossing pedestrian may temporarily appear in the center of the road. This relativeness caused by the image coordinate system can be mitigated with the combination of all additional variables. For the default threshold, the usage of additional variables has no gain in the recall, but the threshold can be increased further since the average precision is nearly 5% over the initial case. This will lead to a higher recall while maintaining a comparative precision value. Additional variables play an important role in the performance of the system. Their incorporation as inputs to the recurrent network in addition to image output features from the convolutional backbone improves the AP from 75.62% to a 80.00% as can be seen in table 4.3.

Table 4.3: Influence of additional variables results.

Variable	Accuracy	Precision	Recall	Average Precision
None	65.75	69.63	80.43	75.62
Looking	65.13	67.28	86.37	76.94
Orientation	67.12	69.96	83.30	77.71
Bbox center	64.78	66.59	87.91	76.15
Movement	68.76	72.71	80.30	76.75
All	68.82	74.20	77.03	80.00

4.9.4 LSTM vs GRU vs bidirectionality

Table 4.4: RNN selection results.

Method	Accuracy	Precision	Recall	Average Precision
LSTM	65.75	69.53	80.43	75.62
GRU	62.65	62.84	98.88	64.25
BiLSTM	67.33	69.20	86.28	79.07
BiGRU	67.62	76.00	70.66	80.19

The results obtained iterating over all configurations for the RNN block of the system are shown in table 4.4. It can be observed that the additional temporal information provided by bidirectionality can improve the results of both RNN-based networks. Without bidirectionality, GRU obtains worse results than LSTM. GRU recall value is near 99% which may mean a collapse of the network to the pedestrian crossing case, given the large difference in precision and accuracy. This may be the result of the limited amount of labeled data available, which is likely to cause a high instability in training highly

influenced by the hyperparameters that affect the randomness of learning. On the other hand, after adding bidirectionality, both RNNs improve the results, with Bidirectional Gated Recurrent Unit (BiGRU) performing slightly better than the Bidirectional Long Short Term Memory (BiLSTM). As the random seed is fixed for the experiments, weight initialization is the same but the complexity of LSTM in addition to the bidirectionality is probably not necessary for the binary classification problem we are facing in this case.

4.9.5 Final performance

Table 4.5: Best model for each improvement configuration.

all add. var. (section 4.9.3)	**reescaling** (section 4.9.2)	**best RNN** (section 4.9.4)	**best feat. extr.** (section 4.9.1)	**Average Precision**
-	-	-	-	75.62
✓	-	-	-	80.00
-	✓	-	-	76.84
-	-	✓	-	80.19
-	-		✓	81.20
✓	✓	✓	✓	**83.34**

All improvements have been tested separately and explained in the previous result sections. To see the combined effect of these experiments, a model was trained with all previous upgrades. The evaluation AP metric of the resultant model is shown in the last row of table 4.5, improving the base model without upgrades (first row) by nearly 8%.

Feature extractor selection plays the most important role, improving by nearly 6% the base model. Since the convolutional backbone is frozen and pre-trained on ImageNet dataset [KSH12] a direct relation is approximated between performance and model size. Bigger or more advanced models, with better results in ImageNet benchmark, obtain better results for pedestrian crossing predictions, suggesting that the frozen features used as input to the RNN-based encoder express better the high-level image features which influence the crossing behavior of the target pedestrian.

On the other side, rescaling is by far the less important upgrade, improving the base model by more or less a 1%. As explained in the section 4.9.2, the chosen technique might be masking some important features for the network to discern between positive and negative cases.

It is interesting to see that, by combining all improvements the individual improvements are not accumulated, meaning that the upgrades collide among them.

4.9.6 Qualitative results

To visually illustrate the output of the model, in fig. 4.7 two examples of prediction are shown from the test set, one per row. Both examples belong to the same target pedestrian. On the left, the image crop of pedestrian bounding boxes is shown for the first, middle,

Figure 4.7: Two examples in test set. The top one represents a non-crossing sequence and at the bottom, a crossing one. Left graphics show the output crossing probability at eight future time steps between 0 and 1 seconds (0 and 30 frames).

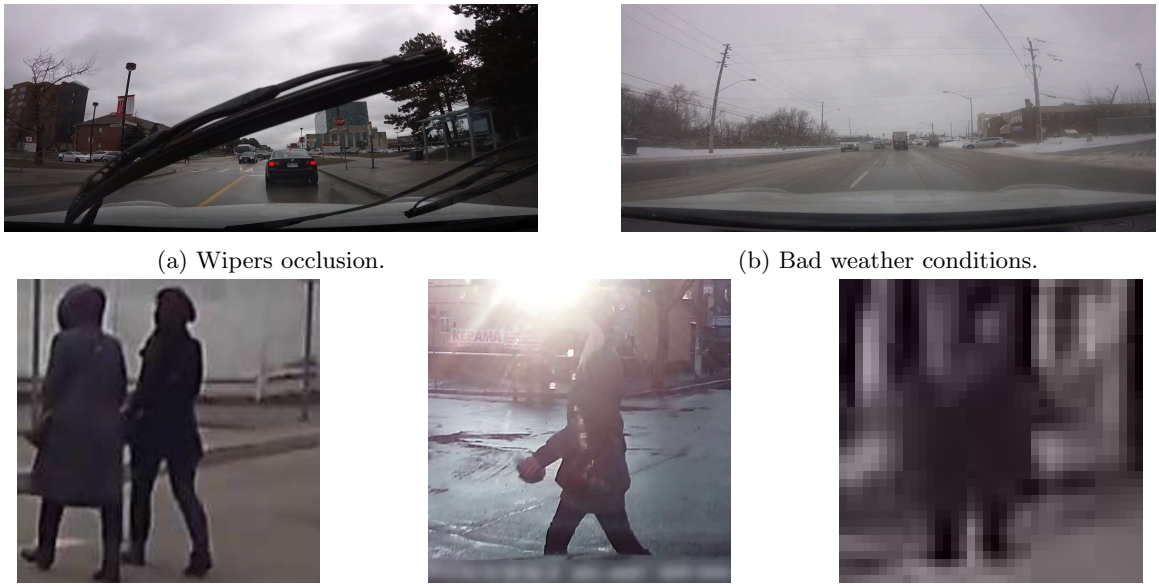
and last frames in the input sequence. To the right, the output probability is plotted as a group of eight values, equally distributed in the interval of 0 to 1 second after the input sequence time interval (videos are recorded at 30 fps). To obtain this prediction, the best model from table 4.5 was used with a change in the output dimension. This was done only for qualitative results to visualize better the evolution in the output probability since the models used in experiments approximate the intention of a pedestrian with a single probability value one second in the future.

In the top case, the pedestrian is not going to cross in one second in the future, and in the bottom one, the pedestrian is beginning to cross. As the graphs show, the probability of crossing is low in the first time step of the top graph, but this value is nearly doubled at the end of the prediction. This increasing trend might indicate a possible crossing in the future. In the bottom case, the model's output trend is also increasing, possibly indicating the pedestrian is stepping on the road.

4.9.7 Dataset limitations

JAAD dataset was, as far as we were concerned by the time of this research project, the unique large-scale dataset with pedestrian behavior annotations available publicly for the academic research community. While being crucial for understanding pedestrian behavior through image data, it has some limitations which can be taken into account for future datasets:

- **Short videos**: there are more than 300 videos, but they are between 5 and 10 seconds long, with an average pedestrian track length of 4 s, due to the manual cropping of



(a) Wipers occlusion. (b) Bad weather conditions.

(c) Non relevant pedestrians. (d) Reflections. (e) Small pedestrians.

Figure 4.8: Defiant cases from JAAD dataset.

sequences. This length is not suitable for long-term behavior prediction and does not reflect real case scenarios where the recordings are done without interruption.

- **Imbalanced annotations**: non-crossing pedestrians are underrepresented in the dataset. While there are bounding box annotations for 2786 pedestrians, behavioral annotations are provided for only 686 of them, 495 crossing, and 191 non-crossing.

- **Insufficient number of samples**: there are 2786 pedestrians annotated, but only 686 with behavior annotation. Thanks to the sliding window strategy, this number is increased considerably. However, the resulting dataset cannot be compared with public large-scale datasets used for video action classification such as Kinetics-700 [CNHZ19]. Furthermore, the sliding window affects variability, since most of the frames are shared between consecutive subsequences.

- **Camera occlusion cases**: bad weather (fig. 4.8b) and lighting conditions (fig. 4.8d) are included in the dataset. While this is a good decision since these cases are present in the real world and make the sampled data more representative, there are some situations that are challenging even for human vision. One example, due to the internal location of the camera, is the windshield occlusion in rainy shots (see fig. 4.8a).

- **Small and additional pedestrians**: there are plenty of small pedestrians in the dataset virtually impossible to detect by an Autonomous Vehicle (AV) perception pipeline. These pedestrians can be easily filtered using the labeled bounding boxes' dimensions and removed from the data. However, this filtering affects the yet scarce number of samples. For the current system studied in this chapter, each pedestrian

is treated separately and detection is considered ideal, but these additional pedestrians can affect other systems which treat all pedestrians in a frame together, with a previous detection step.

In fig. 4.8, visual examples of the previous limitations are shown, to improve the understanding of the dataset.

4.10 Conclusions

Throughout this chapter, a deep learning architecture composed of a combination of convolutional and recurrent layers has been tested for the task of pedestrian crossing action prediction. An extensive set of experiments have been performed, bringing a noticeable performance improvement with respect to the initial model.

In the feature extractor selection experiment, it is clear that supervised methods outperform self-supervised ones. This difference was probably caused by the training methodology used and, recently with works such as [HCX+21], it is proven that a self-supervised methodology can accelerate supervised training, outperforming supervised pre-training with the advantage of using unlabeled data, reducing greatly the experimentation cost.

The weights of the pre-trained visual backbone were frozen during the training of the model. Training the backbone with the rest of the model in an end-to-end manner was not explored during the experimentation for the models used in these experiments, but in the next chapter, models are trained using this methodology.

Regarding the rescaling of the output convolutional features, while average precision is higher for the rescaling case, the recall metric is more important than precision for pedestrian safety. Additional experiments must be carried out with an alternative normalization method to allow normalization at a batch or even global level, e.g., standardizing (whitening) the features by using the mean and standard deviation of the training set.

Additional variables lead to a noticeable performance increase among all improvements in experiments (75.62% $\rightarrow$ 80.00%). While this improvement is promising, all of these variables have been included in ideal conditions directly from the ground truth. Since the purpose of the proposed system is not detection, this is an accepted assumption, but detecting all of the additional variables would add complexity to the system. Moreover, the used variables are mostly image-based. Three-dimensional variables are not considered in this experiment since they are not available together with crossing action in any known public dataset, to the best of our knowledge. The monocular depth and deep optical flow can be also explored as a source of additional information to the system, providing complementary pseudo-three-dimensional information in the continuous space.

Concerning RNN architecture, bidirectionality is more important than the variant chosen. While this choice is not as important, in terms of efficiency, GRU is the best option. Regularization through dropout and weight decay has been applied to the network to improve its results and avoid overfitting. However, dealing with RNN training is difficult and the training data scarcity aggravates its already delicate learning stability. Alternative architectures such as the transformer and three-dimensional (3D) CNN can be explored. Furthermore, data augmentation and fine-tuning techniques can be applied to the convolutional backbone.

Combining all of the upgrades, results are improved by nearly an 8% of AP, a significant improvement which can be leveraged by applying some of the previously discussed additional exploring options. In addition, the final model used all of the improvements but an extensive grid search can also obtain the best combination of upgrades since they probably collide with each other.

In conclusion, despite being able to obtain a model capable of predicting crossing actions correctly in a large variety of situations, the limited data in addition to the chosen architecture have difficulted the learning process of the model.

In the following chapter, a newly created benchmark, composed of a combination of two datasets (one of them JAAD) is explored with a new proposed architecture without recurrence, based on the Transformer encoder architecture [LAI+21].

Chapter 5

CAPformer: end-to-end multi-branch system for pedestrian crossing action anticipation

On chapter 4, a model combining Convolutional Neural Network (CNN) for image feature extraction and recurrent neural network (RNN) for temporal modeling was detailed. While RNN have been the default models used in many time-based tasks, two other types of architectures have emerged in recent years that are capable of rivaling or even surpassing them in several temporal tasks: 3D CNN and the Transformer architecture [VSP+17].

3D CNNs architectures have been applied extensively to several image-based tasks such as medical 3D segmentation and classification (e.g., brain tumor segmentation, breast cancer) and video action classification or recognition (see [ZLL+20] for a detailed review). Video action recognition is a complex task that depends on modeling spatial and temporal context. The task in this thesis can be viewed as a special case of action recognition and it is worth checking the forecasting capabilities of this type of network, trying to find the perfect equilibrium between efficiency and performance. While video action recognition is usually tackled through three-dimensional (3D) CNN models, alternatives have appeared recently proposing different architectural designs focused on efficiency and/or performance improvements.

Pedestrian behavior understanding can benefit from other sources of data besides images, such as kinematics (e.g., 2D pose information) and 2D position information (i.e., bounding box coordinates). An additional source of contextual information can be found in the ego-vehicle, with the use of information obtained from and On-Board Diagnostics (OBD) sensor (e.g., speed, orientation). This temporal information is also handled usually by recurrent neural networks. However, there is another alternative to consider: transformer architecture.

In the Natural Language Processing (NLP) world, transformer architecture is the current default choice for most of its problems. Instead of memorizing data through recurrence like recurrent neural networks, all input samples are processed in parallel using an attention mechanism and learning intersample relationships at a global input level.

By taking both types of models into account, CAPformer [LAI+21] was designed for

the task of future pedestrian crossing action prediction.

5.1 Problem formulation

On chapter 4, the task pursued by the proposed model was pedestrian crossing action prediction, where an input sequence of different sources is used to predict the crossing action in a future short-term horizon of 1 s. In this case, while the task shares the same goal, it has some noteworthy differences.

As explained before, pedestrian crossing behavior anticipation is a special case of action recognition where the model, instead of predicting the current action, tries to anticipate the action in a future time horizon using a limited amount of input information. In the case of the CAPformer, the task is more complex. Instead of predicting with a future limited time horizon, the final crossing action performed by the pedestrian is predicted. This can be seen as a pseudo intention since it is the result of the final decision of the pedestrian.

As a formal definition, this problem belongs to the group of binary classification, where the crossing action is inferred at the instant when the change of action occurs (i.e., $TTE = 0s$). Fixed-length sequences of N frames are used as input features, obtained from the last part of each pedestrian track. The last sample of each sequence (index $N-1$) is obtained from the interval $[TTE_0 - t_1, TTE_0 - t_2]$, where TTE_0 is the global sample index where the final action is performed by the pedestrian. For non-crossing pedestrians, the change of action is considered as $TTE_0 = M - 3$ where M is the length of the pedestrian track.

5.2 Data

Three datasets were used for training and evaluation, as proposed in the benchmark [KRT21]. All of them are imbalanced with the non-crossing class being more represented than the crossing one, except for JAAD_{beh}:

- **Joint Attention in Autonomous Driving (JAAD)** [RT18b]: This dataset is composed of 346 short clips (only 323 are used, excluding low resolution and adverse weather or night ones) recorded in several countries using different cameras (see section 2.2 for details). Two variants of the annotations are used in the benchmark: JAAD_{beh} and JAAD_{all}. JAAD_{beh} includes only pedestrians with behavioral annotations: 495 crossing and 191 non-crossing, giving rise to 374 non-crossing and 1760 crossing samples. JAAD_{all} comprises the entire set of pedestrians in the sequences, adding 2100 non-crossing pedestrian far from the road, resulting in 6853 non-crossing and 1760 crossing sequence samples. The sequence samples from pedestrian tracks are extracted using a sliding window approach with an 80% of overlap between them.

- **Pedestrian Intention Estimation (PIE)** [RKKT19]: This dataset is composed of a continuous recording session in Toronto, Canada, spanning 6 hours during the day under favorable weather conditions. It contains 512 crossing and 1322 non-crossing

pedestrians, which leads to 3576 non-crossing and 1194 crossing samples, using an overlap of 60%.

PIE and JAAD sequences are recorded at a framerate of 30 frames per second (fps). With $t1 = 2$s and $t2 = 1$s, the resultant final interval is section 5.1 is $[TTE_0 - 60, TTE_0 - 30]$.

5.3 System description

In this section, CAPformer (Crossing Action Prediction using Transformer) is detailed. The model tries to anticipate pedestrian crossing behavior through a multi-branch temporal architecture as shown in fig. 5.1. It is composed of three parts: a kinematics encoder branch, a video encoder branch, and a fusion block. In the rest of this section, these blocks will be detailed.

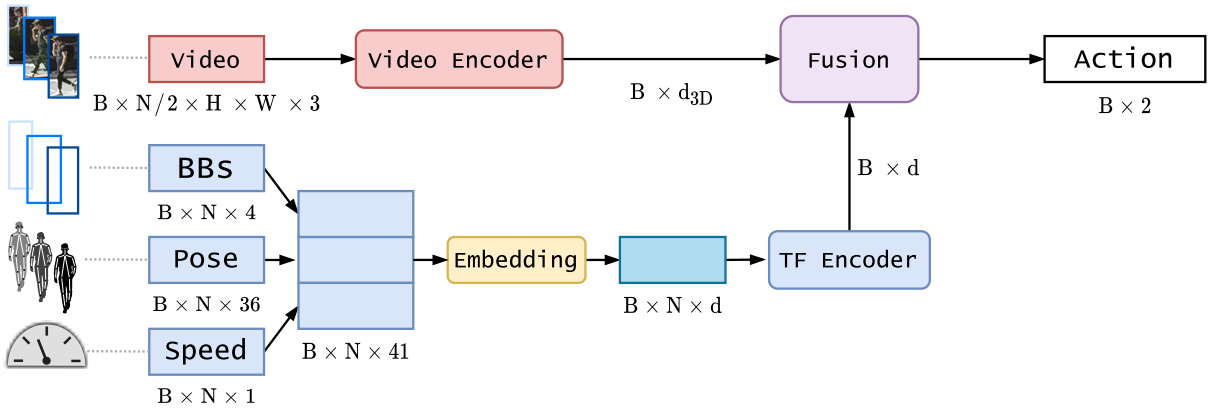


Figure 5.1: Detailed model diagram. B stands for batch size, N is the sequence length, H is image height, W is image width, BBs is bounding boxes coordinates

5.3.1 Kinematics encoding branch

This branch is in charge of encoding a heterogeneous group of features composed of 2D bounding box image coordinates, extracted directly from ground truth annotations, 2D image coordinates of pose keypoints, obtained as the output of a pose estimation algorithm [CHS+19], and ego-vehicle speed, a continuous variable obtained from the ground truth annotations extracted from an OBD sensor. In fig. 5.1, the last dimension of each kinematic feature represents the raw dimension of each sample in the sequence. In the case of bounding boxes coordinates (BBs), this number depends on the strategy used (see section 5.8.1.2). For pose information, 36 value corresponds to 18 pairs of 2D coordinates, representing the image localization of the 18 output key points of the pose estimation model.

This branch is based on different variants of the transformer encoder architecture proposed initially in [VSP+17].

5.3.1.1 Transformer Encoder

Inside a transformer encoder, there are several layers and operations involved. They will be detailed in the following paragraphs in order of complexity: scaled dot-product attention, multi-head attention, and the transformer encoder layer.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{5.1}$$

In the NLP world, transformer architecture is the current default choice. Instead of learning from data sequentially, all input samples are processed in parallel in an encoder, learning intersample relationships at the global input level. This is done through the use of **scaled dot-product attention** (eq. (5.1)). This attention mechanism maps an input set of queries (matrix $\mathbf{Q}$ of dimension $B \times N \times d_q$) and a set of key-value pairs (matrices $\mathbf{K}$ of dimension $B \times N \times d_k$ and $\mathbf{V}$ of dimension $B \times N \times d_v$, respectively) to an output. Each query is compared with all of the keys and, depending on the level of similarity obtained with the dot-product operation between each key and the query, done by the matrix multiplication between $\mathbf{Q}$ and transposed keys $\mathbf{K}^T$ (key and query dimension are equal $d_k = d_q$ and usually the model is simplified by making also value dimension equal $d_k = d_q = d_v$). This multiplication outputs the attention weights, which are multiplied with values to obtain the output. Previous to this matrix multiplication, weights are scaled by the dimension of keys and queries vectors (d_k) and passed through a softmax activation (eq. (5.2)). The scaling factor is used to avoid extremely small gradients in the softmax function, caused by large input values. Each output value represents the weighted sum of all values for a specific query, where the weight is directly proportional to the query-key similarity.

$$\text{softmax}(\mathbf{X}) = \frac{\exp(x_i)}{\sum_{j=0}^{I}\exp(x_j)}, \mathbf{X} = x_0, x_1, \ldots, x_I \tag{5.2}$$

Scaled dot-product attention mechanism is used inside transformer encoder multi-head attention layers (eq. (5.3)). These layers are composed of H attention heads. On each head i, input queries $\mathbf{Q}$, keys $\mathbf{K}$ and values $\mathbf{V}$ are forwarded through linear layers (or projections) with learnable weights $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ respectively. After getting all heads' outputs $[head_0, head_1, \ldots, head_{H-1}]$, they are concatenated and forwarded through a last linear layer with weights $\mathbf{W}_O$.

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_0, \text{head}_1, \ldots, \text{head}_{H-1})\mathbf{W}_O \tag{5.3}$$

$$\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_i^Q, \mathbf{K}\mathbf{W}_i^K, \mathbf{V}\mathbf{W}_i^V) \tag{5.4}$$

The multi-head attention mechanism allows the learning of an ensemble of possible relations between elements in the input set [Wen18].

The previous mechanism is incorporated into the transformer encoder basic layer, composed of a multi-head attention layer and a position-wise feed-forward network. This last layer consists of two linear layers with a ReLU activation between them.

After each multi-head attention and position-wise feed-forward network, a combination of skip-connection with layer normalization is applied. The skip-connection is in charge of adding the input of the layer to its output, similar to the one used in ResNet architecture [HZRS15]. Layer normalization is in charge of computing and applying the mean and standard deviation for each example in a bach independently, unlike batch normalization, which applies it across the batch dimension, introducing dependencies with the batch size hyperparameter and between training samples of each batch.

The transformer encoder is composed of L of the previous layers, and the output of each of them is the input to the next. Queries, keys and values in the encoder arrive from the same source ($\mathbf{Q} = \mathbf{K} = \mathbf{V}$). For this reason, these layers are also known as **self-attention layers**. This mechanism allows the network to look at all of the samples in the input sequence for useful information to encode the target sample.

Before forwarding features into the encoder, a linear embedding is applied to the raw input $x_0, \ldots, x_{N-1}$. This embedding consists of a feedforward layer (learnable linear transformation).

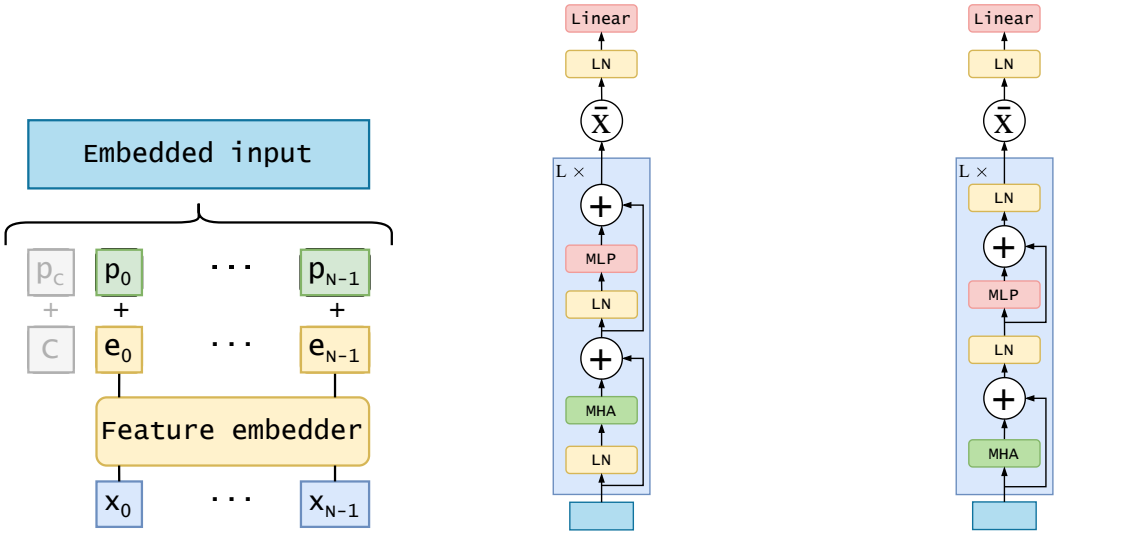
As explained before, transformer architecture processes the data in parallel. For this reason, the ordering of samples is lost and it is necessary to use an explicit stimulus. To account for the order of input data, a positional encoding vector $p_0, \ldots, p_{N-1}$ is added to each position of the previous embedded sequence $e_0, \ldots, e_{N-1}$. The calculation of positional encoding is shown in eq. (5.5). In the equation, *pos* refers to the position in the sequence, from 1 to N, and i corresponds to the position in the embedding dimension, from 0 to d. The intuition behind this sinusoidal function usage is the representation of the position in binary format but using continuous sinusoidal functions as they are more space-efficient.

$$\mathbf{P} = \begin{cases} sin(\frac{pos}{10000^{\frac{2i}{d}}}), i \textbf{ mod } 2 = 0 \\ \\ cos(\frac{pos}{10000^{\frac{2i}{d}}}), i \textbf{ mod } 2 = 1 \end{cases} \tag{5.5}$$

5.3.1.2 Transformer encoder variants

Different transformer encoder variants are used in the experiments:

- **ViT Encoder**: similar to the one proposed in [DBK+20] for image classification. It applies layer normalization (LN) on the embedded input before forwarding it into a multi-head attention layer. Its output is added to the original embedded input through a residual connection. After that, layer normalization is applied again and forwarded to a multi-layer perceptron composed of two linear layers with a Gaussian Error Linear Units activation (GELU) between them. After another residual connection, the output of the layer is used as input for the next layer. The L layer's output is summarized and the resulting vector is forwarded through layer normalization and a simple feed-forward layer. Dropout is applied through all the processes, after every feed-forward layer, except for the last one. The diagram of this architecture is shown in fig. 5.2b.



(a) Embedding. (b) ViT. (c) Vanilla [VSP+17].

Figure 5.2: Detailed diagrams of embedding procedure and transformer encoder architectures. *MHA* stands for multi-head attention, *LN* for layer normalization, *MLP* for multi-layer perceptron. Gray part of embedding diagram corresponds to the class token, which is used in ViT encoder.

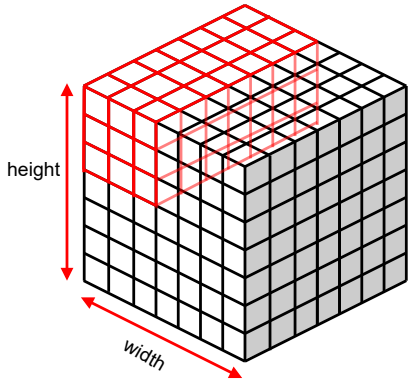
- **Vanilla Transformer Encoder**: using the original proposed encoder in [VSP+17]. The main difference with the previous one is the application of layer normalization. Instead of applying it to embedded input, it is applied to the output of the second residual connection. The diagram of this architecture is shown in fig. 5.2c. Another difference is the usage of ReLU activation instead of GELU in the multi-layer perceptron.

Regarding positional encoding, two strategies were used: ViT encoder uses a learned encoding initialized randomly, sampled from the standard normal distribution (eq. (5.6)); vanilla encoder uses a fixed positional encoding as in [VSP+17] (eq. (5.5)).

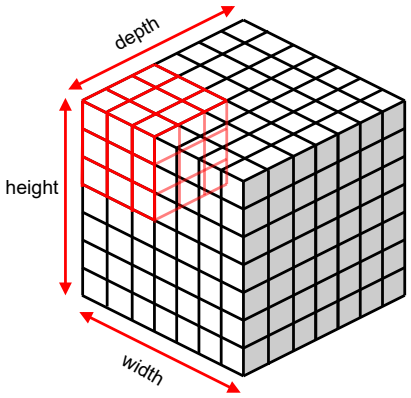
$$\mathbf{P} \sim \mathcal{N}(0, 1) \tag{5.6}$$

ViT encoder also includes a class token c, which is concatenated to the embedded output of positional encoding. This token is a learnable parameter whose purpose is to represent an aggregate sequence representation for classification tasks. Positional encoding is also applied to it (p_c). A detailed diagram of this embedding block is displayed in fig. 5.2a.

The output dimension of the transformer encoder is the same as the input one. Two different methods are applied to the sequence to summarize the temporal dimension: averaging over the temporal dimension and flattening the output logits into a one-dimensional vector.



(a) 2D kernel. The example is of size $3 \times 3 \times 7$ ($W \times H \times D_I$).



(b) 3D kernel. The example is of size $3 \times 3 \times 3$ ($W \times H \times D$).

Figure 5.3: Diagrams of 2D and 3D convolutional kernels. The 2D convolutional kernel fig. 5.3a is applied locally through the width and height, but not in the depth (number of input channels), where it is applied in a fully-connected way, obtaining as output a volume of size $W_O \times H_O \times 1$ for each 2D kernel. The 3D convolutional kernel fig. 5.3b is applied locally through the depth too, obtaining an output volume of shape $W_O \times H_O \times D_O$.

5.3.2 Video encoding branch

The video encoder is included in the model to obtain a one-dimensional embedding of the input video features of size d_{3d}. Two architectures were explored: RubiksNet [FBW+20] and TimeSformer [BWT21]. Both of them will be discussed in the following sections.

5.3.2.1 RubiksNet

3D CNN are for video data what two-dimensional (2D) CNN for image data. 3D CNNs can learn on both spatial and temporal dimensions of a video sequence, as opposed to 2D CNN which can only learn features from the spatial dimension. 3D networks achieve that by using a 3D kernel which treats the channel dimension the same way as the other two dimensions, by convolutioning through time instead of processing all input channels as a whole, as in the 2D case (see fig. 5.3).

However, 2D and 3D convolutions are expensive operations with millions of parameters. To reduce the computational and memory cost associated with these layers, efficient counterparts have been proposed in the literature. Temporal shift module (TSM) [LGH19] is one of them, which combined with 2D CNN models, obtain comparable results with other 3D CNN approaches. The shift is applied at a channel level (see fig. 5.4), between convolutional layers, mixing information between input frames from the sequence. In terms of computational efficiency and the number of parameters, TSM reached the first position on the Something-Something dataset while using a vanilla ResNet 50 backbone, getting rid of 3D convolutions. While these counterparts reduced the computational cost greatly, 2D convolutional layers continued to be used for spatial feature extraction, hence there was still room for improvement in this dimension.

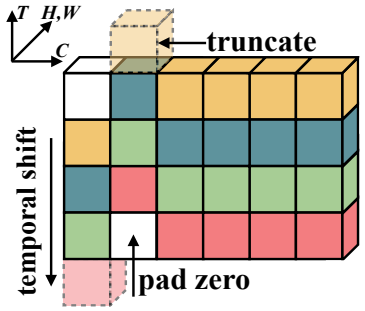


Figure 5.4: Visual diagram of the shift performed with TSM. Obtained from [LGH19].

Based on the premise that most of the spatial information is redundant between frames of a sequence, and that layers' impact on the video task (e.g., action recognition) varies significantly at different depths in the architecture, the temporal shift can be extended to the spatial domain. To make this new shift adaptable to each layer, it should be learned to make the design space of the architecture tractable.

This type of block is named *RubiksShift* and it is the elemental part of RubiksNet model [FBW+20]. This model has been chosen for its efficiency and good performance in different benchmarks since it has nearly 6 times fewer parameters and 4 times fewer Floating Point Operations per Second (FLOPS) than TSM, already considered an efficient action classification architecture.

However, RubiksNet contains custom Compute Unified Device Architecture (CUDA) blocks that perform non-deterministic operations. For this reason, its results are not reproducible even if the random seed is fixed during training. Due to this fact, the TimeSformer model is used as an alternative model, able to exhibit deterministic behavior during the experiments.

5.3.2.2 TimeSformer

During the last year, transformer architecture variants have achieved results comparable to or even better than those of CNNs in different computer vision tasks, such as classification and object detection (see [KNH+21] for a detailed overview). Among the transformer-based classification models, Vision Transformer (ViT) [DBK+20] obtained comparative results with the state-of-the-art convolutional approaches which were dominating main classification benchmarks, while spending much less time in pretraining. Without the use of convolutional layers, ViT processes non-overlapping fixed-size patches of the image, which are linearly embedded, added to a learnable positional embedding and fed to a Transformer encoder (see fig. 5.5).

Another task where transformer-based models have succeeded recently is video action recognition and one of the first models to achieve it was TimeSformer [BWT21], built on the ViT design presented in the previous paragraph. Each frame is divided into $N = HW/P^2$ image patches. Each patch is embedded through a linear layer into an embedding vector and added to a learnable positional embedding. For a single patch, spatial attention is applied with the rest of the patches in its frame. However, in the case of temporal attention, only T comparisons are done for each patch, with the patches at the same

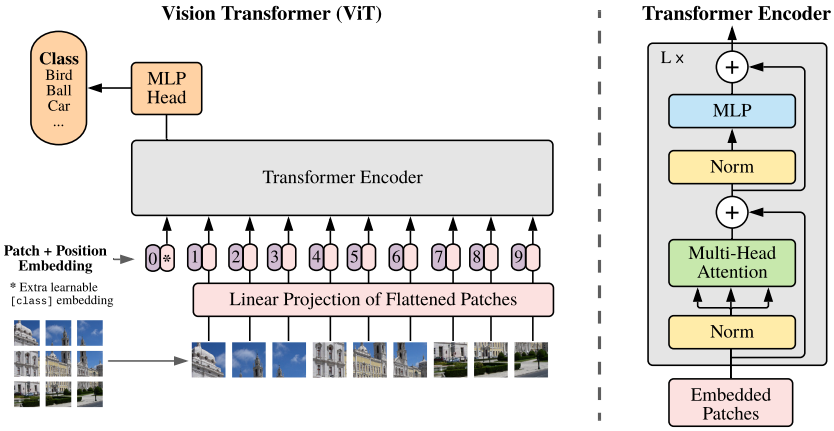


Figure 5.5: ViT architecture. Obtained from original work [DBK+20].

spatial location in other frames, where T is the number of frames. This type of attention called by the authors "Divided Space-Time Attention" was selected among the other four self-attention schemes after an ablation study (see fig. 5.6).

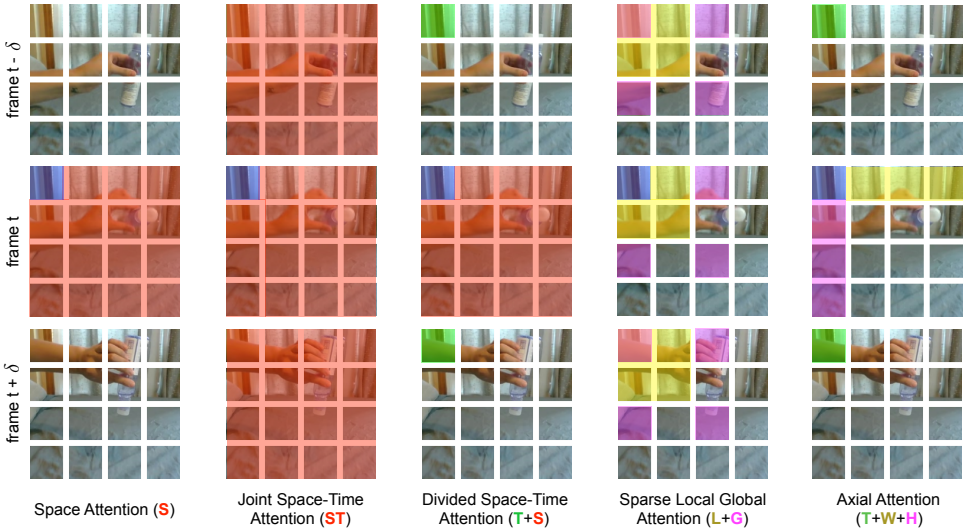


Figure 5.6: Visual diagram of divided attention and other variants. From original work [BWT21].

By the time these experiments were done, this model was the best performing one in several benchmarks such as Kinetics-400 [KCS+17] and Kinetics-600 [CNB+18]. Another advantage of the model is its faster training and inference time in comparison with 3D CNN models. While the main purpose of this model was achieving reproducibility, a fine-tuning strategy was used to train the model from a pre-trained one, helping with the lack of pedestrian crossing action data and allowing faster experimentation.

5.3.3 Feature fusion block

A feature fusion block is needed for joining the information of both branches of the model to get a final unique prediction. Two different alternatives are tested in the architecture:

- **Concatenation through fully connected**: The output of the video encoder is concatenated with the output of the kinematics encoder. This is forwarded through a multi-layer perceptron with one hidden layer, dropout regularization, and a ReLU activation. The output dimension of this layer corresponds to the number of classes.
- **Modality attention**: An attention mechanism is used to weigh the outputs of both encoders. This attention mechanism is presented in [YYD+16] and also used in PCPA model [KRT21].

5.4 Hyperparameter setting

Unless specified, all of the experiments were carried out using the same hyperparameters, to see the effects of data modification rather than model modifications. These experiments are not focused on reaching the best results but on showing the influence of different data preprocessing techniques. The default set of hyperparameters used is the following:

- PIE dataset used for training.
- Batch size of 16 samples.
- *Local box warp* used as the preprocessing for bounding boxes crops (see section 5.9.1 for details).
- TimeSformer used as backbone, pretrained on SSv2 [GKM+17] and fine-tuned. The output vector size is 1024.
- Fusion strategy: concatenation.
- Input sequence length $N = 16$.
- Learning rate with value 10^{-4} for the fusion and kinematic encoder and 10^{-5} for the video backbone.
- Input image dimension of $112 \times 112 \times 3$.
- Weight decay of 10^{-3}.

Due to the difficulty of the task given and the stochastic nature of models and training algorithms, eight different experiments have been carried out using different random seeds, displaying on result tables the mean and its standard error, to show the discrepancy between the group of eight samples and the real distribution. This additional information is useful to show the effect of the random seed and to see if the applied strategy helps in the model stabilization.

For the benchmark models, we fixed random seed to 42 for all experiments. Models in the benchmark experimental part (see section 5.8.4) used different hyperparameters to the ones used in data preprocessing experiments. They will be indicated in that part of the chapter.

5.5 Hardware and software requirements

All the experiments were done with TimeSformer as video backbone were performed on a single NVIDIA A100 GPU with 40 GB of memory. This GPU is part of an NVIDIA DGX server, with an AMD EPYC 7742 64-Core CPU @ 2.25 GHz. Experiments with RubiksNet were mostly done with an NVIDIA GeForce GTX TITAN X with 12 GB of memory and an Intel Core i5-4690K @ 3.5 GHz.

The software used for training the models was PyTorch Deep Learning Python framework [Pas+19], through PyTorch Lightning high-level library [Fal+19], a high-level wrapper that abstracts the user from repetitive tasks allowing to focus on model and data development. For experiment tracking and visualization of results, Weights & Biases [Bie20] library was used. Tensorflow 2 [ABC+16] was also used for some of the benchmarking models.

5.6 Training details

Imbalance is present in the binary classification problem of future pedestrian crossing action prediction. With a normal loss based on the Cross-Entropy metric, the model will tend to predict the majority class in most situations. To avoid it, a weighting strategy is used to simulate oversampling of the minority class. This weight is calculated as indicated in eq. (5.7), where S corresponds to the total number of samples in the dataset, C refers to the crossing class, and NC to the non-crossing class.

$$\begin{aligned} w_C &= S_{NC}/S, \\ w_{NC} &= S_C/S, \\ S &= S_C + S_{NC} \end{aligned} \tag{5.7}$$

The loss function is detailed on eq. (5.8). x refers to the logits or raw outputs of the network for a sequence, composed of two values, one for each class. c is the ground truth class for the current sequence which could be 0 or 1 (not crossing and crossing, respectively). w_c is the weight of the ground truth class $c \in \{C, NC\}$. x_c is the raw output (logit) for the ground truth class. Finally, the sum in the denominator gathers the output for all the classes. It is used to normalize the logit for the class c.

$$\text{loss}(x, c) = -w_c \log\left(\frac{\exp(x_c)}{\sum_j \exp(x_j)}\right) \tag{5.8}$$

AdamW optimizer [LH19] was used for all experiments, with a fixed learning rate, which varies depending on the experiment set. A scheduler is also included in the training

process, which reduces the learning rate by a factor of 10 if the validation loss does not improve.

To avoid wasting computing time, early stopping is included. Focused on validation loss, it also allows the early detection of possible overfitting during training. Since the validation set is small and models converge in a few epochs, a validation step is done every 25% of an epoch.

5.7 Evaluation metrics

The same metrics available in the benchmark were used, detailed in section 2.4. Accuracy does not represent a good performance estimator in imbalanced problems though, so we focused our analysis of the results on the F1 score (F1) and the area under the ROC curve (AUC). Precision (P) and Recall (R) are also included in the analysis. All of these metrics are calculated for the positive class (crossing case) using the scikit-learn library [PVG+11].

5.8 Experiments

In this section, all of the experiments carried out with the proposed architecture are detailed. The first group of experiments is an extensive ablation study focused on data preprocessing. The second one proved the generalization capability of the model with a more varied training set and the third one experimented with different combinations of video backbones and transformer encoders. The last group of experiments is aimed at validating the proposed model against other existing state-of-the-art methods in the benchmark proposed in [KRT21].

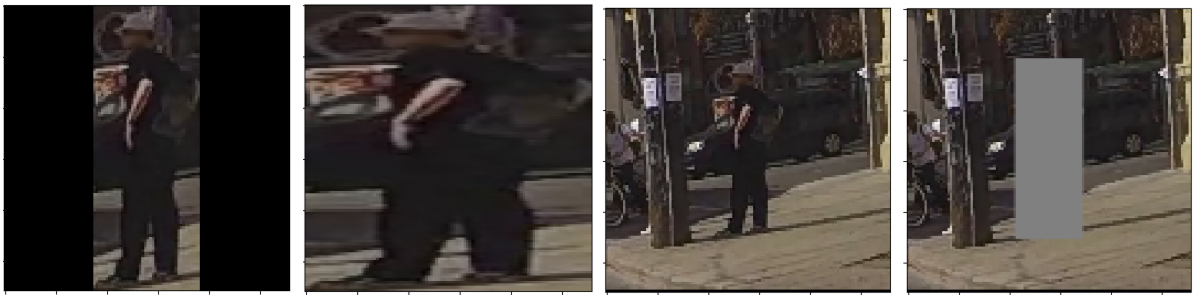
5.8.1 Data preprocessing ablation study

Despite being of utmost importance, data is rarely modified in experiments since most of the modifications focus on the model. To avoid it, several experiments concerning the input data have been proposed to study and measure the importance of each feature. The preprocessing method applied to each of them is also studied in this set of experiments. The purpose of these experiments is the exploration of possible improvements in the performance of the system previous to its modification, following a data-centric approach. Each of the following sections will explain the proposed data-centric experiments.

5.8.1.1 Bounding boxes image cropping strategies

Input image data is extracted using the ground truth bounding box coordinates of pedestrians. The image region contained inside the bounding box is cropped for each frame in the input sequence. However, the image is not used directly in the benchmark [KRT21], but it is preprocessed using one of the following strategies:

- **Local box**: the cropped region is extracted using the ground truth bounding box coordinates and, to keep the aspect ratio of the pedestrian, zero-padding regions (black pixels) are added to both sides. An example is shown in fig. 5.7a.
- **Local context**: the cropped region also uses the ground truth coordinates, but with modifications focused on including the surrounding context. The longest edge is enlarged by a specified factor. The shortest edge is enlarged to reach the same size as the longest one, resulting in a square-shaped crop. An example is shown in fig. 5.7c.
- **Local surround**: the cropped region is the same as in **local context**. In addition to it, the pedestrian bounding box coordinates are used to delete from the center of the crop the image area of the pedestrian, leaving only surrounding context information. The deleted area is filled with gray pixels. An example is shown in fig. 5.7d.



(a) Local box. (b) Local box warp. (c) Local context. (d) Local surround.

Figure 5.7: Visual example of all types of bounding boxes cropping strategies.

Besides previous strategies, a fourth strategy is included in the experiments. In this approach, called **local box warp**, pedestrian bounding box coordinates are used to crop the pedestrian region but, instead of keeping the aspect ratio, the image is resized with deformation to the squared shape required by the network. This strategy is shown in fig. 5.7b. The main objective of this strategy is to check the effect of padding regions on the overall model's performance and to check if the aspect ratio is necessary for the network to learn.

In addition to the previous cropping strategies experiments, the input size is also studied. In the benchmark's best model, part of the C3D convolutional architecture [TBF+15] is used as the video backbone. This network is trained in Sports-1M dataset [KTS+14], a video classification dataset with more than 1 million YouTube videos with 487 sport-related classes. The input size used by this model is $112 \times 112 \times 3$. While the majority of the experiments were done using this size to speed up training, a higher input size of $224 \times 224 \times 3$ has been also tested to check if it benefits our model's performance.

5.8.1.2 Bounding box coordinates preprocessing

Pedestrian bounding box coordinates extracted from ground truth annotations are also used as an input to the system. In the benchmark [KRT21], the top-left (x_{tl}, y_{tl}) and

bottom-right (x_{br}, y_{br}) coordinates are used directly, without any normalization procedure or indirectly, through its 2D speed, obtained by subtracting to each 2D coordinate the previous one in the sequence, keeping a sequence of $N-1$ elements. As an alternative, we propose normalization of these coordinates using the min-max approach (see eq. (5.9)). The minimum value for both coordinates is 0 and the maximum 1920 and 1080 for x and y coordinates, respectively, which corresponds to the frame resolution in both datasets. With this preprocessing step, high features can be reduced in the input data to avoid unstable training.

$$v = \frac{v - v_{min}}{v_{max} - v_{min}} \tag{5.9}$$

In addition to the normalization step, two transformation strategies are applied (also min-max normalized), focused on feature composition obtained from original coordinates:

- **Center and height**: center coordinates of the bounding box and its height. Width is not included to avoid redundancy, since the height can be used as a measure relative to the distance between ego-vehicle and pedestrian.
- **Center, height (position and speed)**: the change ratio (speed) of center coordinates and height are included in the input set with the previous center coordinates and height.

5.8.1.3 Pose keypoints missing data

Pose key points provided in the benchmark are obtained offline from a bottom-up convolutional pose estimation method called OpenPose [CHS+19]. The procedure followed to obtain pedestrian key points is unclear and not detailed in the paper or the code available. Bottom-up methods perform pose estimation in a single step rather than two, as is the case of top-down approaches, where humans are first detected and their pose is estimated from the detected part of the image. Bottom-up methods do not depend on the number of people in the image. However, they obtain better results with close-up human targets, and pedestrians usually began their track far away from the ego-vehicle. For this reason, most pedestrians may suffer from missing key points. Despite that, this input information is treated as ground truth data in the benchmark. For this reason, a study has to be carried out to evaluate whether this input feature is helpful or the missing key points affect negatively the performance of the system.

5.8.1.4 Ego-vehicle speed controversy

We hypothesize that ego-vehicle speed is highly correlated with pedestrian final action in PIE dataset. A reduction in speed triggers the crossing action since pedestrians tend to cross when the ego-vehicle stops. It should be noted that PIE has been recorded, like most of the available datasets, with a human driving the ego-vehicle. For this reason, ego-vehicle speed should be treated as an output of an Autonomous Vehicle (AV) and not as an input to one of its perception modules.

In the benchmark evaluation, ego-vehicle speed is included for a fair comparison with state-of-the-art models.

5.8.1.5 Input features combinations

Due to the ego-vehicle speed controversy and the predictive nature of pose key points, we performed a grid analysis of all the possible combinations of input variables.

5.8.1.6 Data augmentation applied

Under the hypothesis of the scarcity of training data in comparison with other general action datasets, we experimented with using three types of data augmentation for video sequences. If the augmentation is applied, the same change is performed in all images of a sequence:

- **Horizontal flip**: apply a random horizontal flip on the image plane.
- **Roll rotation**: apply a roll rotation of the 3D sequence, which is equivalent to applying a 2D rotation on each image in the sequence.
- **Color jittering**: apply a random change in brightness, contrast, saturation, and hue of the input sequence.

5.8.2 Learning capacity

Deep learning models learn better with more varied training data, becoming models prone to task generalization rather than overfitting. To test this property, we performed two different training schedules: one model only trained on PIE dataset, and another one trained on both PIE and JAAD. We tested both models on both datasets test sets.

5.8.3 Model architecture ablation study

5.8.3.1 Pretrained backbones

For RubiksNet backbone, we have used a variety of pretrained weights, ranging from *RubiksNet-Tiny* variant (1.9M parameters and 3.9 GFLOPs) to *RubiksNet-Large* variant (8.5M parameters and 15.8 GFLOPs). We also experimented with different pretraining datasets: Something-Something-V2 (SSv2) [GKM+17] and Kinetics-400 (K400) [CZ18] datasets. TimeSformer pretrained models are notably larger. We selected the smaller model with 121.4M parameters and 590 GFLOPs at inference. We also experimented with the previous pretraining datasets and additional ones: HowTo100M [MZA+19] (HT100M) and Kinetics-600 [CNB+18] (K600).

After several experiments with all of the above-mentioned backbones, we found that SSv2 pre-trained models outperformed the rest by a large margin. One of the possible reasons is the training schedule followed with the rest of the datasets. We also trained the backbones from scratch, but the SSv2 pre-trained model also outperforms this strategy. For this reason, we have chosen this backbone in all our experiments.

5.8.3.2 Different transformer encoders

Two different transformer encoder architectures are proposed in this work. In addition, its output is processed using the mean operation over the temporal dimension or by applying a flattening operation, resulting in a one-dimensional vector. By combining the encoder type and the output summarizing strategy, we obtained four different options. The best performing one is also compared with the model without a kinematic branch, to see if this branch improves the result.

5.8.4 Benchmark validation

After performing all of the previous experiments, we have found which preprocessing strategies improve our model's performance and also the importance of the different input features. However, for a fair comparison with PCPA, the best model in the benchmark [KRT21], we have used the same preprocessing strategy it followed. Using TimeSformer and RubiksNet backbones, we experimented with the use of modal attention or concatenation as fusion block of the output of the different branches and we have used the findings from the ablation study of the model.

To check if this performance similitude is due to the transformer encoder and not to the video backbone, we developed our best transformer encoder in Tensorflow and changed the recurrent encoders in the PCPA model with it. All non-image (kinematics) inputs are concatenated and forwarded through it after embedding them. The rest of the PCPA model remains the same, including the video 3D backbone and the fusion part. However, we changed the optimizer to AdamW and tuned the hyperparameters of our model to accelerate training. As AdamW applies weight decay, we deactivated direct regularization on layers and applied a weight decay of 10^{-4} for all experiments. We also performed all experiments only using image bounding boxes crops and coordinates, following *local box* cropping strategy and without any normalization, respectively. Our combined features transformer encoder have the following hyperparameters:

- Query, key and value size is the same $d_{q,k,v} \equiv d = 256$.
- Number of self-attention heads $n_{heads} = 8$.
- Number of transformer encoders $L = 2$.
- Multi-layer perceptron hidden layer dimension $d_{mlp} = 384$.
- Dropout rate, applied after embedding and the MLP block $p_{drop} = 0.1$.

5.9 Results

In the following section, the results of the data-centric experiments detailed in section 5.8.1 are presented and discussed. After these results, the findings are applied to the different composed architectures and evaluated in the benchmark [KRT21]. After detailing the quantitative results, random qualitative test samples are shown and commented on, highlighting the most important conclusions extracted from the visual inspection.

5.9.1 Image input nature and size

Each cropping strategy in section 5.8.1.2 has been applied alone as a training improvement for the model. The results from these image pre-processing strategies are shown in table 5.1. The fixed aspect ratio (first row) obtain considerably worst results than the rest of the strategies. Keeping the aspect ratio prevents generalization in the test set, having an Area under the ROC curve (AUC) near the considered random case ($AUC = 0.5$). This procedure obtains good results in the original benchmark work because it uses a different training strategy with a considerably lower learning rate and a bigger number of epochs. Contextual information (third and fourth rows) helps the learning process of the network, showing an increment in F1 of nearly a 17% and a 5% of increment in AUC. While both "contextual" strategies perform similarly, it is worth noticing that in the surrounding strategy, where the pedestrian information is substituted by gray pixels, the metrics standard deviation is considerably larger, meaning that context information helps, but it needs to be used in combination with pedestrian image information. Finally, regarding the pedestrian box resizing strategy with warping, it obtains the best results by a large margin, meaning that the aspect ratio in the input image is not a critical aspect and the use of all the pixels in the input image for learning (avoiding black lateral bands) is a better choice.

Table 5.1: Results obtained varying input image nature.

	F1	**P**	**R**	**AUC**
box	0.212 ± 0.078	0.309 ± 0.081	0.189 ± 0.069	0.542 ± 0.014
box warp	$\mathbf{0.454} \pm 0.040$	$\mathbf{0.532} \pm 0.015$	$\mathbf{0.417} \pm 0.055$	$\mathbf{0.636} \pm 0.019$
context	0.387 ± 0.017	0.531 ± 0.037	0.318 ± 0.025	0.599 ± 0.006
surround	0.379 ± 0.070	0.431 ± 0.072	0.350 ± 0.071	0.607 ± 0.022

Using the best performing resizing strategy from table 5.1 (local box warp, second row), two groups of experiments have been carried out to change the input image dimension. The results are shown in table 5.2. The benchmark's model video backbone uses 112×112. However, TimeSformer is pre-trained using four times bigger images of 224×224. Fine-tuning the model with an image with a lower resolution seems to affect negatively. Doubling width and height increases the level of detail in the input image, both for the pedestrian and the minimal context included in the selected cropping strategy. While F1 improves more than a 7%, AUC does not follow the same trend, keeping an identical average value. While this approach obtains better results, its main drawback is the memory consumption, which reaches $\approx 39GB$ with a batch size of 16 samples. The computational cost is also increased exponentially due to transformer input constraints. Since one motivation of this work is to obtain a lighter model than PCPA with similar results, the following data experiments and benchmark comparisons are performed using 112×112 input frames.

Table 5.2: Results obtained varying input image size.

	F1	P	R	AUC
112×112	0.454 ± 0.040	0.532 ± 0.015	0.417 ± 0.055	0.636 ± 0.019
224×224	$\mathbf{0.528} \pm 0.020$	$\mathbf{0.572} \pm 0.021$	$\mathbf{0.507} \pm 0.042$	$\mathbf{0.636} \pm 0.011$

5.9.2 Bounding box coordinates preprocessing

The results for the three preprocessing strategies applied to pedestrian bounding box coordinates are shown in table 5.3. The initial hypothesis for these experiments of reaching an improvement in the results due to the pseudo-spatial information provided by boxes coordinates is not valid. The results obtained with box coordinates are slightly better, equal, or even worse than using only image (first row). F1 score saturates with the addition of the center coordinates and the height information. This probably means that the camera coordinate system, movement, and distortion affects negatively the spatial information that can provide this source of information.

The image plane localization information provided by bounding boxes coordinates could help the network discriminate samples with similar image information (e.g., a pedestrian walking near the road but not crossing from one approaching the road with the intention of crossing). This is one possible reason for the precision improvement of more than 5% concerning the only image case.

However, this feature does not provide a clear improvement in the performance of our model through these preprocessing strategies, and incorporating speed of change (fourth row) does not help to learn.

Table 5.3: Results were obtained using different preprocessing strategies for bounding box coordinates. h refers to bounding box height; x', y', h' refers to the speed of x, y coordinates, and height, respectively; subscripts tl, br, c refer to top-left, bottom-right, and center coordinates of the bounding box.

Mode	F1	P	R	AUC
Only image	0.454 ± 0.040	0.532 ± 0.015	$\mathbf{0.417} \pm 0.055$	0.636 ± 0.019
$x_{tl}, y_{tl}, x_{br}, y_{br}$	0.425 ± 0.023	0.550 ± 0.020	0.358 ± 0.035	0.620 ± 0.011
x_c, y_c, h	$\mathbf{0.456} \pm 0.031$	$\mathbf{0.587} \pm 0.031$	0.403 ± 0.052	$\mathbf{0.639} \pm 0.013$
$x_c, y_c, h, x'_c, y'_c, h'$	0.355 ± 0.067	0.462 ± 0.078	0.300 ± 0.066	0.600 ± 0.022

5.9.3 Different combinations of input features

In the benchmark [KRT21], four different sources of information are used as input for the PCPA model. In table 5.4 we can see the results of training the model with all possible combinations of input features. While the combination of all input features proposed in the benchmark obtains good results, we can find an improvement by discarding pose key points. This feature is not hand-labeled or human validated. It is a prediction from a pose estimation model (more details in [KRT21]). The model used follows a bottom-up architecture that affects the detection of small pedestrians. This problem leads to data

with a high percentage of missing values that gather 80.9/78.5/84.4% on the JAAD dataset and 44.5/31.0/23.6% on the PIE dataset (train, test, and validation). This negative effect is partially masked by the combination with other useful features. However, when used alone, the network does not find any useful information during training and behaves as a random classifier ($AUC = 0.5$).

In the case of the bounding box coordinates, this feature does not behave as a random classifier by itself, but it is heavily affected by its combination with pose information. Combining it with speed or image information led to slight improvements, which are not significant. This observation confirms that in its present form, it does not provide any useful information.

Table 5.4: Different input combinations metrics. Abbr. I: Bounding boxes image crops, B: bounding boxes coordinates, P: pose keypoints, S: ego-vehicle speed

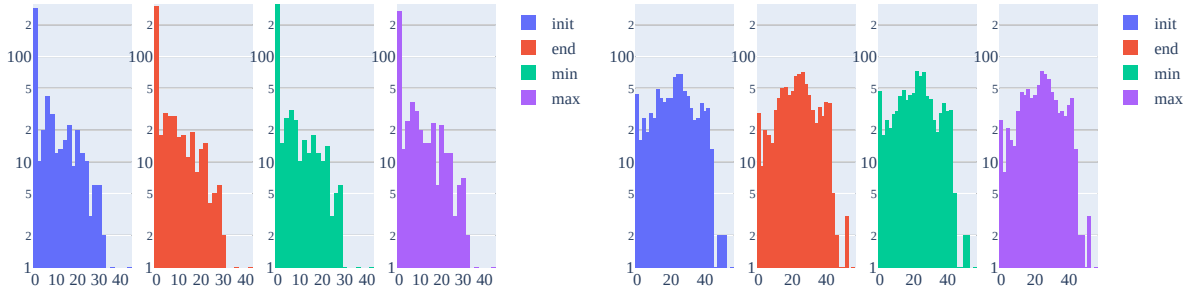
Input				F1	P	R	AUC
I	B	P	S				
	✓			0.160 ± 0.049	0.428 ± 0.115	0.108 ± 0.038	0.535 ± 0.011
	✓	✓		0.011 ± 0.007	0.204 ± 0.140	0.006 ± 0.004	0.502 ± 0.002
	✓	✓	✓	0.737 ± 0.004	0.660 ± 0.011	**0.837** ± 0.013	0.833 ± 0.003
	✓		✓	**0.746** ± 0.003	**0.698** ± 0.017	0.806 ± 0.016	0.833 ± 0.002
✓				0.454 ± 0.040	0.532 ± 0.015	0.417 ± 0.055	0.636 ± 0.019
✓	✓			0.456 ± 0.031	0.587 ± 0.031	0.403 ± 0.052	0.639 ± 0.013
✓	✓	✓		0.454 ± 0.030	0.533 ± 0.020	0.403 ± 0.039	0.633 ± 0.015
✓	✓	✓	✓	0.716 ± 0.010	0.695 ± 0.016	0.748 ± 0.032	0.808 ± 0.010
✓	✓		✓	0.726 ± 0.007	0.665 ± 0.014	0.803 ± 0.017	0.821 ± 0.006
✓		✓		0.468 ± 0.022	0.542 ± 0.013	0.420 ± 0.034	0.640 ± 0.011
✓		✓	✓	0.726 ± 0.009	0.701 ± 0.012	0.759 ± 0.028	0.815 ± 0.009
✓			✓	0.701 ± 0.013	0.685 ± 0.017	0.731 ± 0.040	0.798 ± 0.013
		✓		0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.500 ± 0.000
		✓	✓	0.734 ± 0.002	0.644 ± 0.004	0.854 ± 0.004	**0.834** ± 0.001
			✓	0.743 ± 0.003	0.680 ± 0.010	0.819 ± 0.011	0.834 ± 0.002

Image information is the second-best source of information, but also its dimension is much larger than the rest, meaning that its sampling search space is much more diverse which leads to instability in the training procedure.

Ego-vehicle speed is the feature with more weight in the results. For every combination where it is included, F1 increases by more than 25%. It is surprising that using it alone, reaches the second-best results. In the PIE dataset, the ego-vehicle speed is obtained using an OBD sensor installed in the vehicle. Only pedestrians who interact with the driver have behavioral annotations (i.e., crossing state). This is a major drawback as it is shown in the histogram in fig. 5.8a. Most of the crossing cases correspond to a low speed near zero because the ego-vehicle lowers the speed or even stops when a pedestrian crosses unless it is far away on the ego-lane, which is not usual. The need for interaction leads to a bias since the capture of a pedestrian crossing implies that the vehicle must stop to avoid risking lives. In the non-crossing scenario (fig. 5.8b), the ego-vehicle speed

has a more distributed sample, with an accumulation of cases in speeds between 10 and 30 mph. This is probably caused by the driver's confidence, who does not need to stop after noticing a pedestrian's non-crossing intention. Both figures represent the ego-vehicle speed histogram at the first sample in the benchmark time interval ($init = TTE - 60$) and in the last sample ($end = TTE - 30$), including the minimum and maximum values in that interval. The ordinate axis is on a logarithmic scale.

This differentiation between classes converts this input feature into the most valuable one, outperforming even the combination proposed in the benchmark. A model trained with this feature ends up learning the behavior of the ego-vehicle driver instead of learning to anticipate the behavior of pedestrians. For this reason, it should not be considered as input in the learning process since it will not be an input in the final deployed system, but an output of the AV.



(a) Crossing cases.

(b) Non-crossing cases.

Figure 5.8: Histogram showing the number of pedestrian tracks for each range of ego-vehicle speed in the benchmark data (TTE from 30 to 60 frames).

5.9.4 Data augmentation

We applied the different data augmentation techniques following two policies: an aggressive with 50% of augmentation probability and a milder one with 25%. Aggressive policy results are shown in table 5.5. The results for the mild policy are not shown, since there was no improvement for the non-augmented case.

Looking at the aggressive policy results, in the case of using only the video backbone (image input), rotation is the most valuable augmentation, increasing F1 by nearly a 5% and AUC a 2%. Probably, with the increase of possible positions of pedestrians the network does not overfit a single set of limited pedestrian poses. Horizontal flipping also improves the results by more than 2% in F1 probably due to the same reason proposed in the previous augmentation. Nevertheless, color jittering affects negatively the performance, meaning that color information between training samples is varied enough to avoid overfitting and the random change in color can even affect training since this change is applied differently to sequences belonging to the same pedestrian.

The model using only images benefits from most of the data augmentation proposed techniques. With the inclusion of bounding box coordinates, the performance drops. Rotation augmentation does not affect bounding box coordinates since their center of rotation

is obtained using the center of the bounding box itself. The height is also rotation invariant. There is a high drop in performance with respect to the only image model, probably caused by showing during training different rotated images with the same bounding box coordinates sequences since augmentation is applied differently at each epoch. In the case of flipping, this transformation affects the bounding box coordinates. Even with it, the performance still drops lightly compared to using only video as input. On the other hand, color jittering has the opposite effect. Results improve using both inputs and drops when only using the image.

Table 5.5: Results of different data augmentation strategies. Abbreviations: C: Color jittering, F: horizontal flip (left to right), R: 2D rotation (roll angle), I: image bounding box crop, B: bounding box coordinates.

Augm.	Input	F1	P	R	AUC
-	**I**	0.454 ± 0.040	0.532 ± 0.015	0.417 ± 0.055	0.636 ± 0.019
	I+B	0.456 ± 0.031	**0.587** ± 0.031	0.403 ± 0.052	0.639 ± 0.013
C	**I**	0.446 ± 0.032	0.572 ± 0.020	0.385 ± 0.050	0.634 ± 0.016
	I+B	0.473 ± 0.016	0.544 ± 0.011	0.422 ± 0.022	0.641 ± 0.008
F	**I**	0.475 ± 0.011	0.585 ± 0.019	0.405 ± 0.020	0.644 ± 0.005
	I+B	0.443 ± 0.029	0.557 ± 0.026	0.390 ± 0.049	0.630 ± 0.014
R	**I**	**0.504** ± 0.012	0.567 ± 0.017	**0.458** ± 0.018	**0.659** ± 0.007
	I+B	0.434 ± 0.048	0.572 ± 0.027	0.383 ± 0.063	0.631 ± 0.021
F+R+C	**I**	0.469 ± 0.015	**0.600** ± 0.015	0.390 ± 0.024	0.643 ± 0.007
	I+B	**0.513** ± 0.025	0.562 ± 0.019	**0.484** ± 0.044	**0.667** ± 0.015

Individually, all transformations behave differently, but applying all three improves the results in all combinations of input data, showing that the best possible strategy is the combination using both inputs. Due to the multiple nature of the input, it is difficult to quantify the importance of a data augmentation technique since combining inputs can convert a failed transformation into an improvement (e.g., color jittering). This problem can be caused by an increase in number of parameters since including boxes coordinates adds to the transformer encoder branch. Nevertheless, the effect of data augmentation in the complex end-to-end system is not clear enough. For this reason, this additional strategy cannot be considered a possible improvement.

5.9.5 Learning capacity

The number of unique pedestrians with behavioral annotations in PIE dataset triples the number of JAAD. However, while it includes more pedestrians and possibly a larger variety of crossing behaviors, the features learned with PIE are not rich enough to generalize to JAAD data. As it is shown in table 5.6, a model trained in PIE behaves randomly in the case of JAAD test data. The main possible cause of these results is the lack of variety in weather, light, and geographical conditions in PIE dataset in contrast to JAAD.

Table 5.6: Combined dataset training experiment. Abbr. *P*: PIE, *J*: JAAD. *Train* and *test* columns refer to the dataset used for training and testing, respectively.

Train	Test	F1	P	R	AUC
PIE+JAAD	**PIE**	0.427 ± 0.035	0.478 ± 0.023	0.419 ± 0.055	0.615 ± 0.015
	JAAD	0.566 ± 0.020	0.540 ± 0.026	0.635 ± 0.055	0.756 ± 0.018
PIE	**PIE**	0.454 ± 0.040	0.532 ± 0.015	0.416 ± 0.068	0.636 ± 0.019
	JAAD	0.207 ± 0.021	0.171 ± 0.012	0.275 ± 0.043	0.500 ± 0.013

While several improvements have been explored in dealing with data pre-processing, another strategy with a broad acceptance in the community is the increase of available training data. Deep learning architectures are fed during learning with a limited representation of reality. For this reason, the more data added to the learning set, the more accurate will become that representation and the model will improve its generalization capabilities.

Since both datasets have complementary features (pedestrian crossing behavior and weather conditions), a model trained on both datasets with the learning capability of the chosen architecture will be able to generalize to both test sets. As can be seen in table 5.6, the model trained on both datasets decreases its performance on PIE but increases a 36% in F1 in the case of JAAD. The decrease in F1 in PIE is due to the decrease in precision (recall is virtually the same). This drop can be caused by a different crossing labeling rule in both datasets and the higher difficulty of the model training. Another possible cause is the use of the same hyperparameters in both experiments.

5.9.6 Different encoder strategies

Table 5.7: Different transformer encoder configurations experiment. Abbr. *mean*: output average strategy, *flat*: output flattening strategy.

		F1	P	R	AUC
–	–	0.454 ± 0.040	0.532 ± 0.015	0.417 ± 0.055	0.636 ± 0.019
ViT [DBK+20]	**mean**	0.456 ± 0.031	$\mathbf{0.587} \pm 0.031$	0.403 ± 0.052	0.639 ± 0.013
	flat	0.415 ± 0.042	0.532 ± 0.029	0.352 ± 0.048	0.617 ± 0.020
Vanilla [VSP+17]	**mean**	$\mathbf{0.519} \pm 0.014$	0.557 ± 0.019	$\mathbf{0.500} \pm 0.034$	$\mathbf{0.669} \pm 0.008$
	flat	0.431 ± 0.039	0.571 ± 0.018	0.367 ± 0.052	0.628 ± 0.017

The results are shown in table 5.7. The mean operation over the temporal domain of the output of the kinematic transformer encoder is the best strategy in both encoder variants, outperforming the model with the video backbone only (first row). However, the flattening strategy obtains worse results than the video backbone model. This can be caused by the filtering capabilities of the mean operation. Referring to the transformer architecture variants, vanilla transformer with predefined (not learned) positional encodings performs better than ViT transformer with learned positional encoding. Learning the

position of each input sequence sample during training probably needs a larger training set and it can be challenging due to the diverse input data sources. It is important the difference in the recall, of 10% concerning the second-best case.

Since this experiment is part of a data ablation study, we are confident in using the mean as the best strategy. However, the best encoder type is not clear since both encoders have the same hyperparameters, and we cannot assure that those are equally beneficial for both architectures.

5.9.7 Benchmark

After the previous data-centric experiments which threw valuable insights about the datasets, several architectures have been evaluated in the benchmark. In table 5.8 we gathered the results obtained with these architectures.

In the case of TimeSformer-based architecture, we got slightly better results with concatenation fusion than using modality attention in PIE. However, in both JAAD variants, modality attention performed better.

As a lighter but non-deterministic alternative, RubiksNet tiny version was also evaluated as the video backbone. The obtained results were comparable with TimeSformer except in the case of JAAD_{all}, where RubiksNet outperformed TimeSformer by a large margin. This JAAD variant is heavily imbalanced, meaning that RubiksNet learning is more robust against this problem during training.

Both systems (TimeSformer and RubiksNet) were trained using 112×112 images and half of the input video sequence ($N/2 = 8$), reducing the computational and memory cost. In addition, we did not include pose information while training, which makes it unnecessary to include a pose estimation model in the prediction pipeline. Compared with PCPA, both models performed similarly.

The main problem of TimeSformer is its considerable size (a total $\approx$ 123 million parameters for the whole CAPformer). As a lighter alternative, we performed experiments with the RubiksNet tiny version, which in combination with the rest of the model, contains only 3.5 and 4 million parameters in the modality attention and concatenation variants, respectively. RubiksNet case is also lighter than the C3D backbone in PCPA, which depending on the number of input kinematic features, englobes a total amount of $\approx$ 31 million parameters. This dependence is due to the use of one recurrent encoder for each input kinematic feature.

Another advantage of the proposed architecture in this work is the reduction in the training time, from several hours to less than an hour in most cases, enabling more experimentation in the same time extent.

To see if the transformer encoder for kinematics features was the reason for these good results, we included this encoder in the PCPA model, substituting recurrent ones. We trained these models with just bounding boxes image crops and coordinates, obtaining comparable results to TimeSformer, RubiksNet, and some PCPA variants. For a fair comparison, we trained two PCPA variants with the same random seed used in our model. The first case uses the best-performing model in the benchmark and the same training procedure. We can see that there is a big difference between the final results ($F1 = 0.770$ vs $F1 = 0.735$) in PIE. It is quite noticeable the big difference obtained in the JAAD_{all}

dataset. In $JAAD_{beh}$ dataset, C3D, I3D, and MultiRNN obtain better F1 score than our method. However, looking at the AUC, they are closer to 0.5, meaning that its behavior is more random than in our case. In summary, looking at the results obtained, our method performs better than the PCPA model under the same conditions, validating our proposed encoder based on the self-attention mechanism rather than recurrent sequential ones.

Comparison of our proposed model and the best-performing models in benchmark [KRT21]. Abbr. M is modality attention; C is concatenation; T is temporal attention; I refers to bounding box image crops; B refers to bounding box coordinates; S refers to ego-vehicle speed and P refers to pose keypoints; O refers to optical flow. Rows with darker background correspond to PCPA models trained by us.

Model	Backbone	Fusion	Input	PIE		$JAAD_{beh}$		$JAAD_{all}$	
				F1	AUC	F1	AUC	F1	AUC
Ours	TimeSformer	M	I,B,S	0.761	0.844	**0.763**	0.545	0.557	0.728
		C		**0.779**	0.853	0.743	0.552	0.514	0.701
	RubiksNet	M		0.749	0.839	0.752	**0.589**	0.630	0.782
		C		0.738	0.828	0.691	0.549	0.618	0.778
	C3D	M,T	I,B	0.750	0.851	0.615	0.577	0.614	0.802
C3D		–	I	0.520	0.670	0.750	0.510	0.650	0.810
MultiRNN	GRU	–	B, S*	0.710	0.800	0.740	0.500	0.580	0.790
I3D		–	O	0.720	0.830	0.750	0.510	0.630	0.800
PCPA	C3D	C	I,B,S,P	0.730	0.830	0.630	0.480	0.580	0.800
		M		0.750	0.840	0.680	0.490	0.620	0.830
		T		0.770	**0.860**	0.710	0.480	0.620	0.790
		M,T		0.770	**0.860**	0.710	0.500	**0.680**	**0.860**
				0.735	0.834	0.630	0.484	0.530	0.779
			I,B	0.723	0.820	0.613	0.486	0.522	0.780

* We are not sure of the data used by this network. We indicated bounding boxes coordinates and ego-vehicle speed since this is the data used in the original work.

Finally, as it is shown by the results, there is a big dependency on the randomness of the data. With a different random seed to the one used in the results in table 5.8, we obtained F1 = 0.807 and AUC = 0.922 with the same RubiksNet tiny backbone in PIE. Also by training the PCPA model, we obtained without fixing the random seed a model with F1 = 0.794 and AUC = 0.875, which means that randomness considerably affects the performance of the network. This could be caused by data scarcity. Even though JAAD and PIE are datasets with high-quality annotations, the variety in them could not be enough for a model using video as input to generalize. Another possible reason could be the complexity of the task since samples from both classes have similar image features in the prediction interval proposed by the benchmark.

5.9.8 Qualitative results

After the quantitative evaluation detailed in section 5.9.7, a qualitative visual evaluation is performed over PIE dataset random test samples.

Figure 5.9: Correct predictions in different test cases obtained from RubiksNet trained model. Green and red borders represent crossing and not crossing behavior, respectively.

This visual study helps to better understand the behavior of the system by analyzing correct (fig. 5.9) and incorrect predictions (fig. 5.10). These prediction results have been obtained from one of the benchmark's RubiksNet based model. Each case is represented by three images from the input sequence: first, middle and last one. The color frame represent the class: red for not-crossing and green for crossing.

Looking at the correct cases (fig. 5.9), the network is able to predict crossing cases with an anticipation of 2 s with strong occlusion from cars (middle-right) and cyclists (bottom-right). Non-crossing cases also suffer from occlusion. In the top-left case, the network achieves again an anticipation of 2 s under a continuous occlusion from a bush.

The top-right case is a difficult case, where the pedestrian will cross in the future, but the random sampled subsequence belongs to the first part of interaction with the ego-vehicle, and the pedestrian remains in an static position the whole time. Despite this, the network is able to infer the correct behavior, possibly with the help of positional knowledge and the gaze direction of the pedestrian.

In the remaining non-crossing cases (middle-left and bottom-left), both pedestrians approach the ego-lane perpendicularly from the right. Both of them follow a similar slowing-down behavior, where the pedestrian is walking and suddenly slows down. The network probably learns these dynamics and achieves a possible generalization among similar cases, since both pedestrians are dressed differently (middle-left carries a bag) and participate in different scenes. Moreover, the bottom-left case is more challenging since

the pedestrian seems to have doubts about the future path of the ego-vehicle.

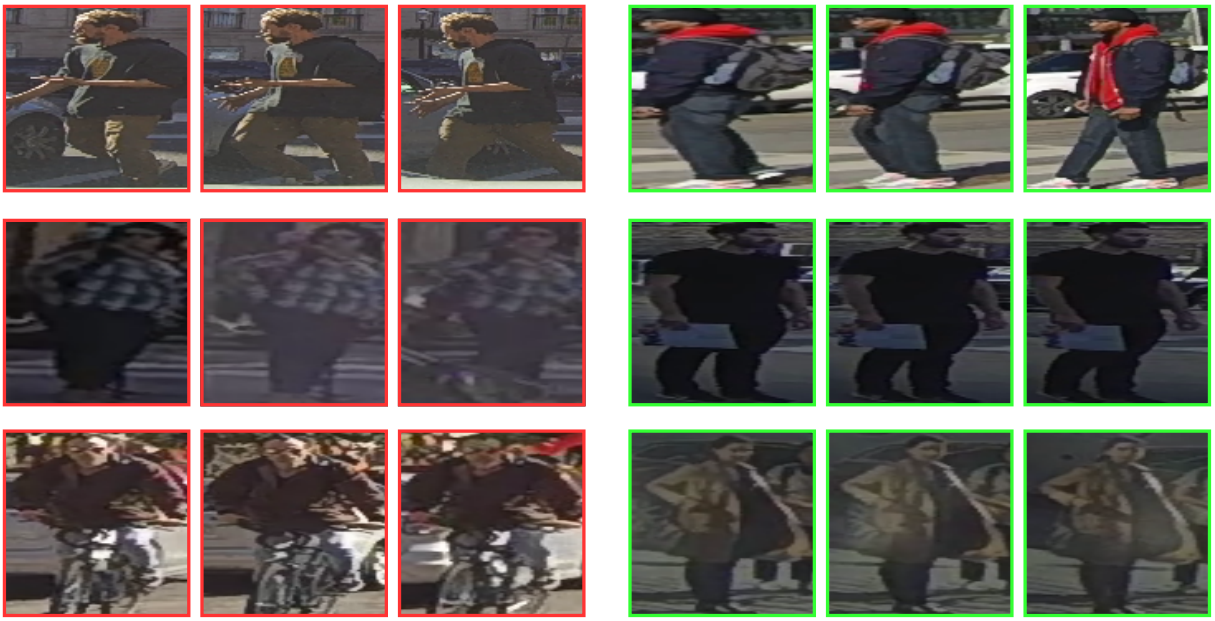


Figure 5.10: Incorrect predictions in different test cases obtained from RubiksNet trained model. Green and red borders represent crossing and not crossing behavior, respectively.

With respect to the failure cases (fig. 5.10), there is a mixture of external and internal factors affecting the predictions. In the non-crossing cases, the top and bottom examples are predicted as crossing and both "pedestrians" are on the road. In the top case, the pedestrian is jay-walking through stopped cars waiting in a red traffic light. This pedestrian is not crossing the street per se, but he is on the road so the features make the network decide in favor of a crossing case. This is a case of labeling noise, since the pedestrian is neither a crossing or a non-crossing case. An additional class or an earlier end of sequence time boundaries could be a possible solution for these cases, to ensure a fair evaluation.

The bottom example is another case of labeling noise, but in this case the problem is not about the pedestrian's behavior, but about its own classification as a road agent, since he is a cyclist. He is on the road and the network, possibly due to the contextual hints, performs a wrong classification. Another possible case is an out-of-distribution problem, since this is probably a strange case in the training set.

The middle example is probably mistaken due to the body language, which seems to be a bending into the road but it is due to the camera fisheye lens.

In the case of the crossing failure cases, the top one is a clear case of crossing. However, the network confusion may be due to the pedestrian orientation, since he is crossing in a crosswalk placed on a different road (perpendicular to the ego-lane), making also the pedestrian unaware of the ego-vehicle existance (pedestrian does not look to the ego-vehicle). The middle case is challenging due to the lack of motion information from the pedestrian, in addition to the gaze direction, not focused again in the ego-vehicle. Finally, the bottom case is similar to the top-right case in fig. 5.9, but the pedestrian quality is lower (detected at a larger distance) and the intention is not so clear, even for a human.

After analyzing correct and incorrect cases, it is quite noticeable that one of the most important factor that seems to affect the network's decision is the **pedestrians' gaze direction**. In most of the correct cases, pedestrians are looking at the vehicle through the input sequence. Among failure cases, pedestrians are not paying attention to the ego-vehicle (top row, middle-right) or are paying attention to possible vehicles in their nearest lane, since middle-left and bottom-right pedestrians are trying to cross. While gaze direction seems to be a meaningful and important feature, it is dangerous for absent-minded pedestrians and the network needs to weight the information obtained from different sources in a more equitable way.

5.10 Conclusions

Throughout this chapter, a novel architecture for pedestrian crossing action prediction has been proposed, based on a combination of 3D convolutional models and transformer encoder architectures, as an alternative to recurrent models. Thanks to the standardization of the crossing action prediction task through the benchmark released in [KRT21], the proposed model has been further validated, comparing it with the current state-of-the-art model, PCPA. Before this comparison, an extensive set of data-centric experiments has been carried out, to find relevant information about the data.

Regarding input image data, bounding box cropping and resizing strategy greatly influence the learning procedure of the system. The best-performing method, proposed in this work, applies deformation to the resized pedestrian instead of keeping the aspect ratio (default method in [KRT21]) with a 24% increase in F1 score under the same training schedule. In the case of image resizing, the proposed resizing (112×112) performed worse than using the original resizing in the benchmark (224×224), with a decrease of 7% in the F1 score. However, the second case presents a higher memory and computational consumption, which grows exponentially due to the quadratic complexity of self-attention.

Different bounding box preprocessing techniques have been tested to observe their effect on the system. Most of the preprocessing strategies negatively affected the results with respect to the only image case. Only using the center coordinates and the height of the bounding box increased the results.

Since the model input in the benchmark is composed of four different features, an extensive experiment testing all combination have been carried out. Surprisingly, the speed feature was the most important in PIE. Training a model only with it gave best results than training with all the features, since pose predictions negatively affected the results, and box coordinates achieved a slight improvement in the performance (as explained in the previous paragraph).

Data augmentation techniques were also explored for image and bounding box data. Image rotation was the best augmentation technique with an increase of a 5% in F1 score for the image-only model. However, this improvement disappeared with the composed model (image and bounding box as input) and only by combining all of the available augmentations, an improvement was observed.

The datasets used in the experiments, PIE and JAAD, shared a common task, but they present several differences. A trained model in PIE does not learn to generalize to JAAD

dataset, but training on both datasets leads to a generalization to both data, meaning that, in the future, if more data becomes available, this is a strategy to be exploited.

Different kinematic encoding strategies (only for bounding boxes) were also tested, with the mean operation over the temporal domain of transformer encoder output being the most promising. Since two transformer-based encoders were proposed, both were tested, with vanilla transformer one being the best choice.

In the benchmark, three variants of the model were evaluated: TimeSformer-based one, a heavy model reaching similar results to most of the PCPA models, RubiksNet, a lighter model with similar findings; and the same PCPA configuration, substituting recurrent encoders with transformer-based ones. Since speed cannot be used as an input variable, we perform a final comparison only with image and bounding box information.

For a fair comparison, the best-performing variant of PCPA was also trained with the same configuration. Our model obtained comparable or even better results in the three datasets. The best performing variant of PCPA was trained again using the same random initialization as our models. The results obtained were worse than the ones reported by the benchmark, meaning that the task is highly biased by random initialization.

To conclude, the combination of task complexity, data scarcity, and data imbalance makes the pedestrian crossing action prediction a challenging task that needs more quality data to deal with the high-dimensionality of video data.

Chapter 6

Conclusions and Future Work

In this chapter, the general conclusions for the whole work of this thesis are detailed. After that, a summarization of the main contributions is presented, ending with a list of possible future lines of work.

6.1 Conclusions

This thesis focuses on the task of pedestrian crossing action prediction, leveraging current commercial Advanced Driver Assistance Systems (ADAS) detection capacity and anticipating braking action to prevent accidents between pedestrians and vehicles.

- Vulnerable Road User (VRU) classification helps in the filtering of pedestrians as an initial step in the Autonomous Vehicle (AV) perception pipeline. Available open data is highly imbalanced. Therefore, labeling of new data and data augmentation focused on minority classes is a key improvement, reflected in the overall results for the proposed Convolutional Neural Network (CNN) architecture which went from 82.57% of F1 Score to a 98.29% under ideal detection circumstances.
- The combination of CNN and recurrent neural network (RNN) encoders for pedestrian crossing action prediction in the challenging Joint Attention in Autonomous Driving (JAAD) dataset obtained competitive results with a small architecture and a short training time. Applying several improvements in the architecture increased Average Precision (AP) from 75.62% to 83.34%.
- Transformer encoder proves to be a good alternative to recurrent-based models for pedestrian crossing action prediction. In the benchmark [KRT21], our model surpassed PCPA (best state-of-the-art model) by 3% in F1 in Pedestrian Intention Estimation (PIE) and by nearly 9% in JAAD challenging set. Furthermore, centering most of the experimentation on data was crucial for finding problems with speed and pose input features.
- Exploitation of additional variables (other than image) is beneficial for the learning of pedestrian crossing action prediction task. It complements image information and helps the network deal with its high dimensionality.

- The high-dimensionality of image data and the complexity of the pedestrian crossing action prediction task highlighted a data scarcity problem and the need for additional data to reach better generalization.

6.2 Main contributions

The main contributions of this thesis, focused on exploring features affecting the crossing behavior of a pedestrian, are:

- Proposal of a methodology based on different CNN architectures to differentiate between VRU types and helping in the reduction of computational cost in the overall AV pipeline.
- For the VRU classification task, a combination of several available datasets was performed with the corresponding data processing and mapping. With this data combination, a serious imbalance problem was highlighted in the resulting training set.
- Gathering a dataset from the internet for VRU classification task to deal with imbalance problem. A state-of-the-art detection method, based on Mask R-CNN architecture was used to partially automate the labeling task. The model's classification output, trained on Common Objects in Context (COCO) dataset [LMB+14], was used to get the different types of VRU with the combinations of detected objects.
- Combining data augmentation on underrepresented classes and the use of the new gathered dataset, improved the results by a margin of more than 15% on average with the initial experimentation, validating this strategy.
- A state-of-the-art model, named STCAP, based on a combination of CNN backbone for visual data and a RNN encoder have been proposed and evaluated in JAAD dataset. The model achieved competitive results for the prediction of crossing action prediction with a horizon of one second in the future.
- An extensive set of experiments to improve the previous architecture resulted in an improvement of nearly 8% in AP. The choice of a convolutional feature extractor was found to be the most important factor in the results.
- A state-of-the-art end-to-end multi-branch model named CAPformer, based on a combination of novel three-dimensional (3D) CNN backbone and transformer-based encoders to get rid of recurrence. The model was evaluated on the benchmark proposed in [KRT21], obtaining better results than the state-of-the-art best model by the time, called PCPA. The proposed model predicted crossing action with a maximum horizon of two seconds and a minimum of one.
- A data-centric set of experiments focused on PIE dataset which resulted in different meaningful findings such as the hidden importance of speed variable in the prediction.

6.3 Future Work

In the following list, a summary of the proposed future work is detailed:

- Explore graph-based architectures since they can probably model the interaction between different agents and leverage ego-vehicle and pedestrian contextual information.
- Leverage designed models to be invariant to the number of pedestrians in the scene. The model frame throughput depends on the number of objects of interest in the scene. A good starting point might be a combination of 3D CNN and object detection models.
- Deep visual analysis of available pedestrian behavior datasets and exploration of semi-automatic tools to validate labeling and reduce noise.
- Development of scenarios in AV simulators such as CARLA [DRC+17] to generate synthetic datasets with a larger variety to help the data-scarcity problem.
- Use of 3D features to avoid the dependency problem with image-based variables.
- Exploring alternatives to image-based approaches, such as optical flow or trajectory-based approaches to reduce the input data space complexity.
- Develop an end-to-end pedestrian action prediction system, from detection to final prediction, to measure the performance in the real world scenario.
- Explore confidence calibration techniques, to ensure that the crossing probability returned by the model represents a correct estimation of the real probability.

Appendices

Appendix A

Additional experimentation

In addition to the work presented in this document, additional lines of research have been explored during this thesis research work. While all of them related to pedestrians, most of them did not meet a conclusion. However, we decided to include them to encourage future researchers to take them up again if they see fit.

A.1 Bottom-up human pose keypoint estimation methods for pedestrian detection

Thanks to deep learning and CNNs, pose estimation methods have evolved during the last decade. These models take an image as input and output a prediction of the two-dimensional (2D) position of the human pose keypoints detected. There are two main groups: bottom-up and top-down approaches. In the first group, the model predicts directly the key points heatmaps (regression) without the need of detecting pedestrians. This characteristic makes their image throughput invariant to the number of people present in the input image. The second group, before estimating the heatmaps, performs a detection to gather all persons in the input image. Due to this, the model throughput is affected by the number of people present in the input image.

In the AV world, pose information can be useful for understanding pedestrian dynamics. Bottom-up methods can be considered for real-time applications if their detection performance is comparable with the top-down approaches. To measure it, we compared two top-down methods, AlphaPose [FXTL17] and keypoint Mask R-CNN [GRG+18], with the best bottom-up method available by the time of this research: OpenPose [CHS+19].

Since detected boxes coordinates are needed and are not available in bottom-up methods, a simple feed-forward neural network with one hidden layer was trained to convert detected key points into a bounding box, since using the minimum and maximum of them obtained low intersection over union with the ground truth annotations.

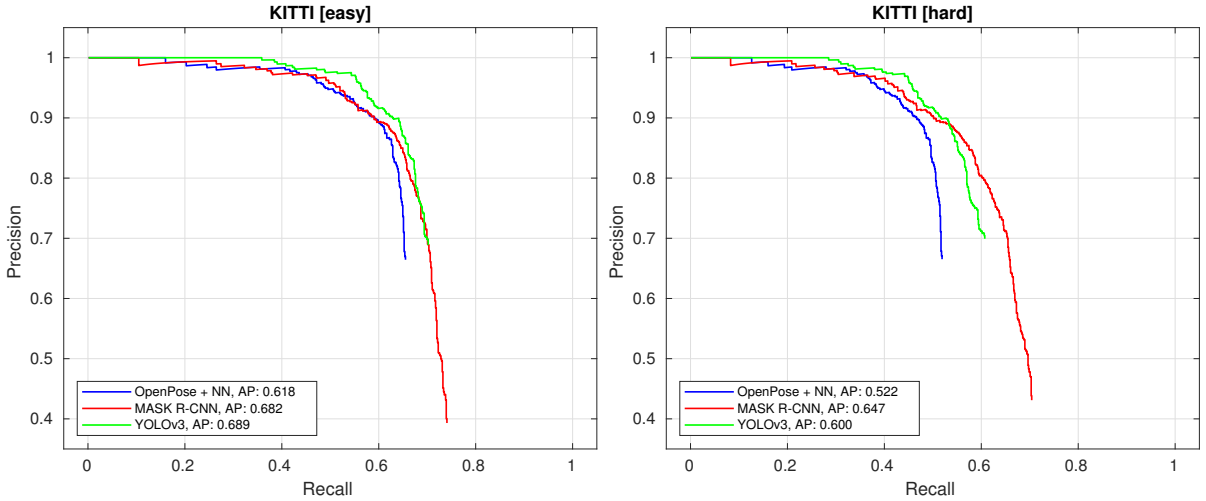
Models were tested in four different datasets: KITTI [GLSU13], INRIA [DT05], Tsinghua [XFY+16] and CityPersons [ZBS17]. Each dataset contains three different benchmarks in order of difficulty, based on the size of the predicted pedestrians.

In table table A.1 the average precision results for each set is shown. Notice that AlphaPose and Keypoint Mask R-CNN are denoted as their object detectors: YOLOv3

Table A.1: AP Detection Results.

AP	**Mask R-CNN**	**YOLOv3**	**OpenPose + NN**
CityPer.	**0.524** / **0.514** / **0.435**	0.441 / 0.357 / 0.354	0.444 / 0.359 / 0.356
KITTI	0.682 / **0.662** / **0.647**	**0.689** / 0.615 / 0.600	0.618 / 0.531 / 0.522
INRIA	0.892 / 0.892 / 0.892	0.892 / 0.892 / 0.892	**0.900** / **0.900** / **0.900**
Tsinghua	**0.799** / **0.794** / **0.711**	0.701 / 0.683 / 0.603	0.613 / 0.536 / 0.535

and Mask R-CNN, respectively. As can be observed, Mask R-CNN is clearly the best performing method, but also the heaviest one. INRIA dataset contains high-quality images with big pedestrians and that is the only reason why OpenPose outperforms the rest of the methods, showing that this dataset is no longer useful for benchmarking. OpenPose is especially affected in hard sets with small pedestrians, as can be observed in the precision-recall curves in fig. A.1a for the easy set compared to fig. A.1b.



(a) KITTI Detection Results (Easy $IoU >= 0.5$)

(b) KITTI Detection Results (Hard $IoU >= 0.5$)

A.2 Pose traces over pedestrian body for crossing action prediction

A set of experiments were done to check if the combination of 2D pose key points (obtained with OpenPose [CHS+19]) and a 2D convolutional backbone can be used to predict the crossing action of a pedestrian in the future. Since 2D CNNs lack temporal understanding, we decided to draw pose traces on the image representing the pose evolution over N previous frames. The pose trace followed a color code with brighter (higher value) pixels for more recent poses. The result of the impainting can be observed in fig. A.2 (the frame was obtained from the Daimler Pedestrian dataset [SG13]).

After generating the data, a group of models was trained, but we did not obtain good results. A possible reason is the disappearance of the key points after resizing. A possible solution that can be explored is the use of multiscale CNNs, similar to the ones used in object detection, to capture information at multiple resolutions.

Figure A.2: Pose traces examples in a sequence from Daimler dataset.

A.3 Variational recurrent network for pedestrian crossing action prediction

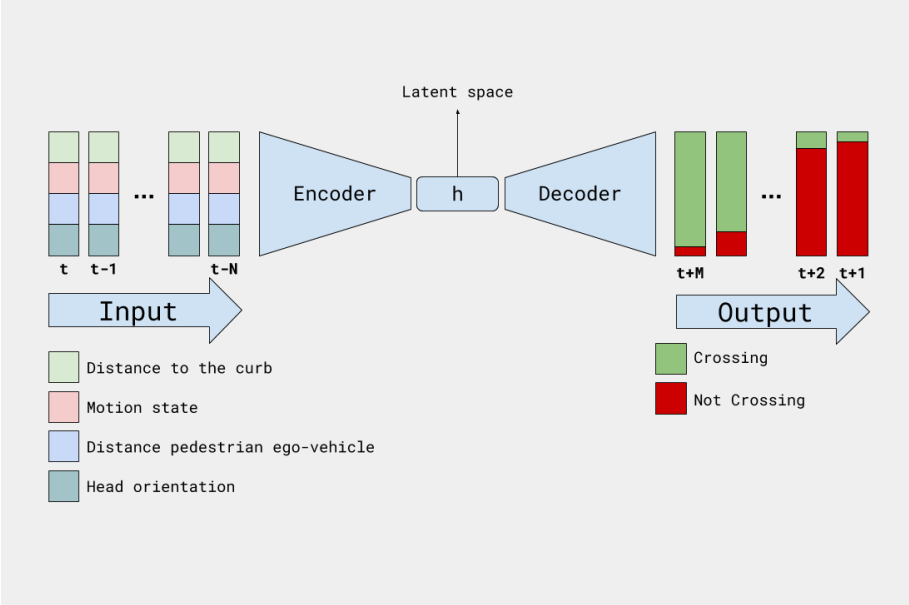


Figure A.3: Variational recurrent encoder decoder network diagram.

Before the work presented in the chapter 4 and chapter 5, a variational recurrent network was developed to predict pedestrian crossing action. On section A.3 a diagram of the model is shown. In this method, the input data is composed of distance to the curb (measured with stereoscopic cameras), motion state (categorical), head orientation (discrete), and distance between the pedestrian and the ego vehicle (also measured with stereoscopic cameras). The data is extracted from the updated Daimler dataset [KSFG14] and no image information is used. The input sequence is forwarded through the recurrent encoder and the last output of the sequence is used to estimate parameters needed to characterize the latent space (gaussian distribution). Once obtained, sampling is performed to

obtain M pieces of data which are forwarded through the recurrent decoder creating M different predictions. Thanks to this sampling, the model was able to predict the crossing probability with an uncertainty estimation, converging with the progress of the sequence.

While this solution was promising, the lack of 3D data in other large-scale datasets with behavioral annotations such as JAAD or PIE made it very difficult to continue using this approach. Another problem was related directly to the data used, which was not naturalistic, probably introducing biases compared to the real-world scenario.

A.4 Crossing action prediction through heatmap estimation

The use of a 3D CNN encoder-decoder network was proposed to estimate the future crossing action using heatmap estimation. The input to the model was composed of a sequence of image frames, belonging to the JAAD dataset. The output was composed of two heatmaps, one for crossing and the other for non-crossing cases. The selected model was a ResNet-18 adapted using 3D kernels. In fig. A.4, two examples, one for crossing (top) and the other for not crossing are shown. The heatmaps on the right represent the activation of the crossing and not-crossing respectively. While these results were promising, the two heatmaps collapsed at the end of training to similar values. Since both output matrices were independent, we did not find a solution to select one for each pixel.

Figure A.4: Heatmaps visual samples of the 3D CNN output.

Appendix B

Publications Derived from this Ph.D. Dissertation

B.1 Journal Publications

2020 **CNNs for Fine-Grained Car Model Classification**, *H. Corrales, D.F. Llorca, I. Parra, S. Vigre, A. Quintanar, J. Lorenzo, N. Hernández*, Computer Aided Systems Theory – EUROCAST 2019 (ISBN: 978-3-030-45096-0), pages 104–112.

2021 **Vehicle Maneuver Prediction on Highways Using Efficient Environment Representation and Deep Learning**, *R. Izquierdo, A. Quintanar, J. Lorenzo, I. García-Daza, I. Parra, D.F. Llorca, M.A. Sotelo*, IEEE Access, doi: 10.1109/ACCESS.2021.3106692.

2021 ***CAPformer: Pedestrian Crossing Action Prediction Using Transformer**, *J. Lorenzo, I. Parra, R. Izquierdo, A.L. Ballardini, A. Hernández, D.F. Llorca, M.A. Sotelo*, Sensors, 21(17):5694. https://doi.org/10.3390/s21175694.

2021 **Urban Intersection Classification : A Comparative Analysis**, *A.L. Ballardini, A. Hernández, S. Carrasco, J. Lorenzo, I. Parra, N. Hernández, I. García-Daza, M.A. Sotelo*, Sensors, 21(18):6269. https://doi.org/10.3390/s21186269.

B.2 Conference Publications

2016 **Drivertive Team in the Grand Cooperative Driving Challenge 2016**, *J. Alonso, I. Parra, R. Izquierdo, A. García, M. A. Sotelo, C. Fernández, R. Quintero, C. Salinas, J. Lorenzo, D. F. Llorca*, 1st Symposium SEGVAUTO-TRIES-CM, Madrid.

2016 **A Comparison of Road Curb Detection Using Curvature Values from LIDAR and Stereo Vision**, *C. Fernández, J. Lorenzo, J. Alonso, A. García, D. F. Llorca, M. A. Sotelo*, 1st Symposium SEGVAUTO-TRIES-CM, Madrid.

2016 **Two-camera based accurate vehicle speed measurement using average speed at a fixed point**, *D.F. Llorca, C. Salinas, M. Jiménez, I. Parra, A. García, R. Izquierdo, J. Lorenzo, M.A. Sotelo*, 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), pages 2533–2538.

2017 **Pedestrian intention recognition by means of a Hidden Markov Model and body language**, *R. Quintero, I. Parra, J. Lorenzo, D.F. Llorca, M.A. Sotelo*, 2017 IEEE 20th international conference on intelligent transportation systems (ITSC), pages 1–7.

2020 ***RNN-based Pedestrian Crossing Prediction using Activity and Pose-related Features**, *J. Lorenzo, I. Parra, F. Wirth, C. Stiller, D.F. Llorca, M.A. Sotelo*, 2020 IEEE Intelligent Vehicles Symposium (IV), pages 1801–1806.

2021 **Data-driven vehicle speed detection from synthetic driving simulator images**, *A. Martínez, J. Lorenzo, I. García-Daza, D.F. Llorca*, 2021 IEEE Intelligent Transportation Systems Conference (ITSC).

B.3 Open source code

Two open-source code repositories have been made available in this thesis, both related to our last publication [LAI+21] and available on GitHub. The first one is the `CAPformer` repository[1], which contains the code for the Tensorflow model developed for comparison with PCPA [KRT21]. The benchmark code is obtained from the original repository with a single change in the whole code set to ensure a fair comparison. The second repository[2] was created to develop a port of a the modality attention used as feature fusion alternative in [LAI+21].

[1] https://github.com/javierlorenzod/CAPformer
[2] https://github.com/javierlorenzod/pytorch-attention-mechanism

Bibliography

[ABC+16] M. Abadi *et al.*, 'Tensorflow: A system for large-scale machine learning,' in *Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'16, Savannah, GA, USA: USENIX Association, 2016, pp. 265–283, ISBN: 9781931971331.

[AJ20] D. Adminaité-Fodor and G. Jost, 'How safe is walking and cycling in Europe?' European Transport Safety Council, Tech. Rep., Jun. 2020.

[AJ19] D. Adminaité-Fodor and G. Jost, 'Safer Roads, Safer Cities: How to improve urban road safety in the EU,' European Transport Safety Council, Tech. Rep., Jun. 2019.

[AA20] S. Ait Bouhsain and A. Alahi, 'Pedestrian Intention Prediction: A Multi-Task Perspective,' Tech. Rep., 2020, p. 9.

[ASS+18] M. S. Aliakbarian, F. S. Saleh, M. Salzmann, B. Fernando, L. Petersson and L. Andersson, 'VIENA2: A Driving Anticipation Dataset,' *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11361 LNCS, pp. 449–466, Oct. 2018.

[BKC16] V. Badrinarayanan, A. Kendall and R. Cipolla, *Segnet: A deep convolutional encoder-decoder architecture for image segmentation*, 2016. arXiv: 1511.00561 [cs.CV].

[BWT21] G. Bertasius, H. Wang and L. Torresani, 'Is space-time attention all you need for video understanding?' In *Proceedings of the International Conference on Machine Learning (ICML)*, Jul. 2021.

[Bie20] L. Biewald, *Experiment tracking with weights and biases*, Software available from wandb.com, 2020.

[BWKS14] S. Bonnin, T. H. Weisswange, F. Kummert and J. Schmuedderich, 'Pedestrian crossing prediction using multiple context-based models,' IEEE, 2014, pp. 378–385, ISBN: 978-1-4799-6078-1. DOI: 10.1109/ITSC.2014.6957720.

[CYQW19] P. R. G. Cadena, M. Yang, Y. Qian and C. Wang, 'Pedestrian Graph: Pedestrian Crossing Prediction Based on 2D Pose Estimation and Graph Convolutional Networks,' in *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*, Institute of Electrical and Electronics Engineers Inc., Oct. 2019, pp. 2000–2005, ISBN: 9781538670248. DOI: 10.1109/ITSC.2019.8917118.

[CBL+20] H. Caesar *et al.*, *Nuscenes: A multimodal dataset for autonomous driving*, 2020. arXiv: 1903.11027 [cs.LG].

[CHS+19] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei and Y. A. Sheikh, 'Openpose: Realtime multi-person 2d pose estimation using part affinity fields,' *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[CNB+18] J. Carreira, E. Noland, A. Banki-Horvath, C. Hillier and A. Zisserman, *A short note about kinetics-600*, 2018. arXiv: 1808.01340 [cs.CV].

[CZ18] J. Carreira and A. Zisserman, *Quo vadis, action recognition? a new model and the kinetics dataset*, 2018. arXiv: 1705.07750 [cs.CV].

[CNHZ19] J. Carreira, E. Noland, C. Hillier and A. Zisserman, *A short note on the kinetics-700 human action dataset*, Oct. 2019.

[CTBB20] M. Chaabane, A. Trabelsi, N. Blanchard and R. Beveridge, 'Looking ahead: Anticipating pedestrians crossing with future frames prediction,' in *Proceedings - 2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020*, Institute of Electrical and Electronics Engineers Inc., Mar. 2020, pp. 2286–2295, ISBN: 9781728165530. DOI: 10.1109/WACV45572.2020.9093426.

[CLS+19] M.-F. Chang *et al.*, 'Argoverse: 3D Tracking and Forecasting with Rich Maps,' in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[CvMG+14] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk and Y. Bengio, 'Learning phrase representations using RNN encoder-decoder for statistical machine translation,' *CoRR*, vol. abs/1406.1078, 2014.

[CGCB14] J. Chung, Ç. Gülçehre, K. Cho and Y. Bengio, 'Empirical evaluation of gated recurrent neural networks on sequence modeling,' *CoRR*, vol. abs/1412.3555, 2014.

[COR+16] M. Cordts *et al.*, 'The Cityscapes Dataset for Semantic Urban Scene Understanding,' in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, IEEE Computer Society, Dec. 2016, pp. 3213–3223, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.350.

[DT05] N. Dalal and B. Triggs, 'Histograms of oriented gradients for human detection,' in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, ser. CVPR '05, Washington, DC, USA: IEEE Computer Society, 2005, pp. 886–893, ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.177.

[DCO17] A. Dominguez-Sanchez, M. Cazorla and S. Orts-Escolano, 'Pedestrian Movement Direction Recognition Using Convolutional Neural Networks,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3540–3548, Dec. 2017, ISSN: 15249050. DOI: 10.1109/TITS.2017.2726140.

[DRC+17] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez and V. Koltun, 'CARLA: An open urban driving simulator,' in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[DBK+20] A. Dosovitskiy *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, 2020. arXiv: 2010.11929 [cs.CV].

[Eur10] European Commission. 'Road Safety Programme 2011-2020: detailed measures,' European Commission. (2010), [Online]. Available: https://ec.europa.eu/commission/presscorner/detail/en/MEMO_10_343.

[Eur18] European Commission, 'Traffic Safety Basic Facts on Pedestrians,' *European Road Safety Observatory*, pp. 1–24, 2018.

[Fal+19] W. Falcon *et al.*, *Pytorch lightning*, 2019.

[FBW+20] L. Fan* *et al.*, 'RubiksNet: Learnable 3D-Shift for Efficient Video Action Recognition,' in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[FXTL17] H.-S. Fang, S. Xie, Y.-W. Tai and C. Lu, 'RMPE: Regional multi-person pose estimation,' in *ICCV*, 2017.

[FL19] Z. Fang and A. M. López, *Intention recognition of pedestrians and cyclists by 2d pose estimation*, 2019. arXiv: 1910.03858 [cs.CV].

[FY11] R. Furuhashi and K. Yamada, 'Estimation of street crossing intention from a pedestrian's posture on a sidewalk using multiple image frames,' in *1st Asian Conference on Pattern Recognition, ACPR 2011*, 2011, pp. 17–21, ISBN: 9781457701221. DOI: 10.1109/ACPR.2011.6166694.

[GLSU13] A. Geiger, P. Lenz, C. Stiller and R. Urtasun, 'Vision meets Robotics: The KITTI Dataset,' *International Journal of Robotics Research (IJRR)*, 2013.

[GPB+20] J. Gesnouin, S. Pechberti, G. Bresson, B. Stanciulescu and F. Moutarde, 'Predicting Intentions of Pedestrians from 2D Skeletal Pose Sequences with a Representation-Focused Multi-Branch Deep Learning Network,' *Algorithms*, vol. 13, no. 12, p. 331, Dec. 2020, ISSN: 1999-4893. DOI: 10.3390/a13120331.

[GMB+18] O. Ghori *et al.*, 'Learning to Forecast Pedestrian Intention from Pose Dynamics,' in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2018, pp. 1277–1284, ISBN: 978-1-5386-4452-2. DOI: 10.1109/IVS.2018.8500657.

[GRG+18] R. Girshick, I. Radosavovic, G. Gkioxari, P. Dollár and K. He, *Detectron*, https://github.com/facebookresearch/detectron, 2018.

[GKM+17] R. Goyal *et al.*, *The "something something" video database for learning and evaluating visual common sense*, 2017. arXiv: 1706.04261 [cs.CV].

[GS05] A. Graves and J. Schmidhuber, 'Framewise phoneme classification with bidirectional lstm networks,' in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 4, 2005, 2047–2052 vol. 4. DOI: 10.1109/IJCNN.2005.1556215.

[GV19a] P. Gujjar and R. Vaughan, 'Classifying Pedestrian Actions In Advance Using Predicted Video Of Urban Driving Scenes,' in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, May 2019, pp. 2097–2103, ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8794278.

[GV19b] P. Gujjar and R. Vaughan, 'Classifying pedestrian actions in advance using predicted video of urban driving scenes,' in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, Institute of Electrical and Electronics Engineers Inc., May 2019, pp. 2097–2103, ISBN: 9781538660263. DOI: 10.1109/ICRA.2019.8794278.

[HJ15] J. Hariyono and K. H. Jo, 'Detection of pedestrian crossing road using action classification model,' in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*, vol. 2015-Augus, Institute of Electrical and Electronics Engineers Inc., Aug. 2015, pp. 21–24, ISBN: 9781467391078. DOI: 10.1109/AIM.2015.7222502.

[HGH+16] Y. Hashimoto, Y. Gu, L.-T. Hsu, M. Iryo-Asano and S. Kamijo, 'A probabilistic model of pedestrian crossing behavior at signalized intersections for connected vehicles,' *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 164–181, 2016, ISSN: 0968-090X. DOI: 10.1016/j.trc.2016.07.011.

[HCX+21] K. He, X. Chen, S. Xie, Y. Li, P. Dollár and R. Girshick, *Masked autoencoders are scalable vision learners*, 2021. arXiv: 2111.06377 [cs.CV].

[HGDG18] K. He, G. Gkioxari, P. Dollár and R. Girshick, *Mask r-cnn*, 2018. arXiv: 1703.06870 [cs.CV].

[HZRS15] K. He, X. Zhang, S. Ren and J. Sun, 'Deep residual learning for image recognition,' *CoRR*, vol. abs/1512.03385, 2015.

[HS97] S. Hochreiter and J. Schmidhuber, 'Long short-term memory,' *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[HTDD18] M. Hoy, Z. Tu, K. Dang and J. Dauwels, 'Learning to Predict Pedestrian Intention via Variational Tracking Networks,' in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 3132–3137, ISBN: 9781728103235. DOI: 10.1109/ITSC.2018.8569641.

[IHM+16] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally and K. Keutzer, 'SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and,' Feb. 2016.

[JCL+20] A. Jain *et al.*, 'Discrete Residual Flow for Probabilistic Pedestrian Behavior Prediction,' in *Proceedings of the Conference on Robot Learning*, L. P. Kaelbling, D. Kragic and K. Sugiura, Eds., ser. Proceedings of Machine Learning Research, vol. 100, PMLR, 2020, pp. 407–419.

[KAHS16] V. Karasev, A. Ayvaci, B. Heisele and S. Soatto, 'Intent-aware long-term prediction of pedestrian motion,' in *Proceedings - IEEE International Conference on Robotics and Automation*, 2016, ISBN: 9781467380263. DOI: 10.1109/ICRA.2016.7487409.

[KTS+14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar and L. Fei-Fei, 'Large-scale video classification with convolutional neural networks,' in *CVPR*, 2014.

[KCS+17] W. Kay *et al.*, *The kinetics human action video dataset*, 2017. DOI: 10.48550/ARXIV.1705.06950.

[KG14] C. G. Keller and D. M. Gavrila, 'Will the Pedestrian Cross? A Study on Pedestrian Path Prediction,' *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 494–506, 2014, ISSN: 1524-9050. DOI: 10.1109/TITS.2013.2280766.

[KUH+19] R. Kesten *et al.*, *Lyft Level 5 Perception Dataset 2020*, \url{https://level5.lyft.com/dataset/}, 2019.

[KNH+21] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan and M. Shah, *Transformers in vision: A survey*, 2021. arXiv: 2101.01169 [cs.CV].

[KB14] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. arXiv: 1412.6980 [cs.LG].

[KZBH12] K. M. Kitani, B. D. Ziebart, J. A. Bagnell and M. Hebert, 'Activity forecasting,' in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato and C. Schmid, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 201–214, ISBN: 978-3-642-33765-9.

[KGZ+15a] S. Kohler, M. Goldhammer, K. Zindler, K. Doll and K. Dietmeyer, 'Stereo-Vision-Based Pedestrian's Intention Detection in a Moving Vehicle,' in *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, IEEE, Sep. 2015, pp. 2317–2322, ISBN: 978-1-4673-6596-3. DOI: 10.1109/ITSC.2015.374.

[KGZ+15b] S. Kohler, M. Goldhammer, K. Zindler, K. Doll and K. Dietmeyer, 'Stereo-Vision-Based Pedestrian's Intention Detection in a Moving Vehicle,' in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, 2015, ISBN: 9781467365956. DOI: 10.1109/ITSC.2015.374.

[KFPG19] J. F. Kooij, F. Flohr, E. A. Pool and D. M. Gavrila, 'Context-Based Path Prediction for Targets with Switching Dynamics,' *International Journal of Computer Vision*, vol. 127, no. 3, pp. 239–262, Mar. 2019, ISSN: 15731405. DOI: 10.1007/s11263-018-1104-4.

[KSFG14] J. F. P. Kooij, N. Schneider, F. Flohr and D. M. Gavrila, 'Context-Based Pedestrian Path Prediction,' in Springer, Cham, 2014, pp. 618–633. DOI: 10.1007/978-3-319-10599-4_40.

[KRT21] I. Kotseruba, A. Rasouli and J. K. Tsotsos, 'Benchmark for evaluating pedestrian action prediction,' in *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2021, pp. 1257–1267. DOI: 10.1109/WACV48630.2021.00130.

[KDA+17] I. Krasin *et al.*, 'Openimages: A public dataset for large-scale multi-label and multi-class image classification.,' *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[KSDF13] J. Krause, M. Stark, J. Deng and L. Fei-Fei, '3d object representations for fine-grained categorization,' in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[KSH12] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks,' *Communications of the ACM*, vol. 60, pp. 84–90, 2012.

[LGH19] J. Lin, C. Gan and S. Han, 'Tsm: Temporal shift module for efficient video understanding,' in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[LMB+14] T.-Y. Lin *et al.*, 'Microsoft coco: Common objects in context,' in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[LAC+20] B. Liu *et al.*, 'Spatiotemporal Relationship Reasoning for Pedestrian Intent Prediction,' *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3485–3492, Apr. 2020, ISSN: 23773766. DOI: 10.1109/LRA.2020.2976305.

[LPW+20] J. Lorenzo, I. Parra, F. Wirth, C. Stiller, D. F. Llorca and M. A. Sotelo, 'RNN-based Pedestrian Crossing Prediction using Activity and Pose-related Features,' in *IEEE Intelligent Vehicles Symposium, Proceedings*, Institute of Electrical and Electronics Engineers Inc., Aug. 2020, pp. 1801–1806. DOI: 10.1109/IV47402.2020.9304652.

[LAI+21] J. Lorenzo *et al.*, 'Capformer: Pedestrian crossing action prediction using transformer,' *Sensors*, vol. 21, no. 17, 2021, ISSN: 1424-8220. DOI: 10.3390/s21175694.

[LH19] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG].

[MPLN17] W. Maddern, G. Pascoe, C. Linegar and P. Newman, '1 year, 1000 km: The Oxford RobotCar dataset,' *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, Jan. 2017, ISSN: 0278-3649. DOI: 10.1177/0278364916679498.

[MDC20] S. Malla, B. Dariush and C. Choi, 'TITAN: Future Forecast using Action Priors,' *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 11 183–11 193, Mar. 2020.

[MGLW16] M. Maurer, J. C. Gerdes, B. Lenz and H. Winner, 'Autonomous driving: Technical, legal and social aspects,' 2016.

[MZA+19] A. Miech, D. Zhukov, J.-B. Alayrac, M. Tapaswi, I. Laptev and J. Sivic, 'Howto100m: Learning a text-video embedding by watching hundred million narrated video clips,' in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 2630–2640. DOI: 10.1109/ICCV.2019.00272.

[NCA20] E. NCAP, 'European New Car Assessment Programme (Euro NCAP) Test Protocol - AEB VRU systems,' Tech. Rep., 2020.

[NHD+20] S. Neogi, M. Hoy, K. Dang, H. Yu and J. Dauwels, 'Context model for pedestrian intention prediction using factored latent-dynamic conditional random fields,' *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020, ISSN: 1558-0016. DOI: 10.1109/tits.2020.2995166.

[Org+18] W. H. Organization *et al.*, 'Global status report on road safety 2018: Summary,' World Health Organization, Tech. Rep., 2018.

[Org] W. H. Organization, *SDG Indicators — SDG Indicators*.

[Pas+19] A. Paszke *et al.*, 'Pytorch: An imperative style, high-performance deep learning library,' in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035.

[PVG+11] F. Pedregosa *et al.*, 'Scikit-learn: Machine learning in Python,' *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[PBP+20] F. Piccoli *et al.*, 'FuSSI-Net: Fusion of Spatio-temporal Skeletons for Intention Prediction Network,' *arXiv*, May 2020.

[PRC+19] D. O. Pop, A. Rogozan, C. Chatelain, F. Nashashibi and A. Bensrhair, 'Multi-Task Deep Learning for Pedestrian Detection, Action Recognition and Time to Cross Prediction,' *IEEE Access*, vol. 7, pp. 149 318–149 327, 2019, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2944792.

[QPLS18] R. Quintero, I. Parra, D. F. Llorca and M. A. Sotelo, 'Pedestrian Path, Pose, and Intention Prediction Through Gaussian Process Dynamical Models and Pedestrian Activity Recognition,' *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2018, ISSN: 1524-9050. DOI: 10.1109/TITS.2018.2836305.

[RGB+20] A. Ranga *et al.*, 'VRUNet: Multi-Task Learning Model for Intent Prediction of Vulnerable Road Users,' *IS and T International Symposium on Electronic Imaging Science and Technology*, vol. 2020, no. 16, Jul. 2020. DOI: 10.2352/ISSN.2470-1173.2020.16.AVM-109.

[Ras20] A. Rasouli, 'The Role of Context in Understanding and Predicting Pedestrian Behavior in Urban Traffic Scenes,' Ph.D. dissertation, York University, Toronto, Ontario, Canada, Aug. 2020.

[RKKT19] A. Rasouli, I. Kotseruba, T. Kunic and J. K. Tsotsos, 'PIE: A Large-Scale Dataset and Models for Pedestrian Intention Estimation and Trajectory Prediction,' Tech. Rep., 2019, pp. 6262–6271.

[RKT17] A. Rasouli, I. Kotseruba and J. K. Tsotsos, 'Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior,' in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, IEEE, Oct. 2017, pp. 206–213, ISBN: 978-1-5386-1034-3. DOI: 10.1109/ICCVW.2017.33.

[RKT20] A. Rasouli, I. Kotseruba and J. K. Tsotsos, 'Pedestrian Action Anticipation using Contextual Feature Fusion in Stacked RNNs,' *arXiv*, May 2020.

[RRL21] A. Rasouli, M. Rohani and J. Luo, *Bifold and semantic reasoning for pedestrian behavior prediction*, 2021. arXiv: 2012.03298 [cs.CV].

[RT18a] A. Rasouli and J. K. Tsotsos, *Autonomous vehicles that interact with pedestrians: A survey of theory and practice*, 2018. arXiv: 1805.11773 [cs.RO].

[RT18b] A. Rasouli and J. K. Tsotsos, *Joint attention in driver-pedestrian interaction: From theory to practice*, 2018. arXiv: 1802.02522 [cs.RO].

[RYL+20] A. Rasouli, T. Yau, P. Lakner, S. Malekmohammadi, M. Rohani and J. Luo, *PePScenes: A Novel Dataset and Baseline for Pedestrian Action Prediction in 3D*, 2020.

[RYRL20] A. Rasouli, T. Yau, M. Rohani and J. Luo, *Multi-modal hybrid architecture for pedestrian action prediction*, 2020. arXiv: 2012.00514 [cs.CV].

[RK15] E. Rehder and H. Kloeden, 'Goal-Directed Pedestrian Prediction,' in *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, IEEE, Dec. 2015, pp. 139–147, ISBN: 978-1-4673-9711-7. DOI: 10.1109/ICCVW.2015.28.

[RWLS17] E. Rehder, F. Wirth, M. Lauer and C. Stiller, 'Pedestrian Prediction by Planning using Deep Neural Networks,' Jun. 2017.

[RRL+18] D. Ridel, E. Rehder, M. Lauer, C. Stiller and D. Wolf, 'A Literature Review on the Prediction of Pedestrian Behavior in Urban Scenarios,' in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2018-Novem, Institute of Electrical and Electronics Engineers Inc., Dec. 2018, pp. 3105–3112, ISBN: 9781728103235. DOI: 10.1109/ITSC.2018.8569415.

[RDWT19] D. A. Ridel, N. Deo, D. Wolf and M. M. Trivedi, *Understanding pedestrian-vehicle interactions with vehicle mounted vision: An lstm model and empirical analysis*, 2019. arXiv: 1905.05350 [cs.CV].

[RPH+20] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila and K. O. Arras, 'Human motion trajectory prediction: a survey,' *International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, Jul. 2020, ISSN: 17413176. DOI: 10.1177/0278364920917446.

[RDS+15] O. Russakovsky *et al.*, 'ImageNet Large Scale Visual Recognition Challenge,' *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, ISSN: 15731405. DOI: 10.1007/s11263-015-0816-y.

[SHN19] K. Saleh, M. Hossny and S. Nahavandi, 'Real-time Intent Prediction of Pedestrians for Autonomous Ground Vehicles via Spatio-Temporal DenseNet,' Apr. 2019.

[SG13] N. Schneider and D. M. Gavrila, 'Pedestrian Path Prediction with Recursive Bayesian Filters: A Comparative Study,' in *Pattern Recognition: 35th German Conference, GCPR 2013, Saarbrücken, Germany, September 3-6, 2013. Proceedings*, J. Weickert, M. Hein and B. Schiele, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 174–183, ISBN: 978-3-642-40602-7.

[SS15] A. T. Schulz and R. Stiefelhagen, 'A Controlled Interactive Multiple Model Filter for Combined Pedestrian Intention Recognition and Path Prediction,' in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, vol. 2015-Octob, Institute of Electrical and Electronics Engineers Inc., Oct. 2015, pp. 173–178, ISBN: 9781467365956. DOI: 10.1109/ITSC.2015.37.

[SZ15] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2015. arXiv: 1409.1556 [cs.CV].

[SAM+21] G. Singh *et al.*, *ROAD: The ROad event Awareness Dataset for Autonomous Driving*, 2021.

[SKD+20] P. Sun *et al.*, 'Scalability in perception for autonomous driving: Waymo open dataset,' in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2446–2454.

[SLJ+14] C. Szegedy *et al.*, *Going deeper with convolutions*, 2014. arXiv: 1409.4842 [cs.CV].

[TL18] A. Taeihagh and H. S. M. Lim, 'Governing autonomous vehicles: Emerging responses for safety, liability, privacy, cybersecurity, and industry risks,' *Transport Reviews*, vol. 39, no. 1, pp. 103–128, Jul. 2018, ISSN: 1464-5327. DOI: 10.1080/01441647.2018.1494640.

[TBF+15] D. Tran, L. Bourdev, R. Fergus, L. Torresani and M. Paluri, 'Learning spatiotemporal features with 3d convolutional networks,' in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489–4497. DOI: 10.1109/ICCV.2015.510.

[Uni18] United Nations, *Sustainable cities, human mobility and international migration*, English, Feb. 2018.

[Van21] W. Van den Berghe, 'European Commission Road safety thematic report – Speeding,' European Road Safety Observatory, Brussels, European Commission, Directorate General for Transport, Tech. Rep., Jan. 2021.

[VSP+17] A. Vaswani *et al.*, *Attention is all you need*, 2017. arXiv: 1706.03762 [cs.CL].

[VBM+16] B. Völz *et al.*, 'A data-driven approach for pedestrian intention estimation,' in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, Nov. 2016, pp. 2607–2612, ISBN: 978-1-5090-1889-5. DOI: 10.1109/ITSC.2016.7795975.

[Wen18] L. Weng, 'Attention? attention!' *lilianweng.github.io/lil-log*, 2018.

[Wor20] World Health Organization, *Global gathering of ministers determines road safety agenda to 2030*, 2020.

[XFY+16] Xiaofei Li *et al.*, 'A new benchmark for vision-based cyclist detection,' in *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, Jun. 2016, pp. 1028–1033, ISBN: 978-1-5090-1821-5. DOI: 10.1109/IVS.2016.7535515.

[XGD+17] S. Xie, R. Girshick, P. Dollár, Z. Tu and K. He, *Aggregated residual transformations for deep neural networks*, 2017. arXiv: 1611.05431 [cs.CV].

[XGD+16] S. Xie, R. B. Girshick, P. Dollár, Z. Tu and K. He, 'Aggregated residual transformations for deep neural networks,' *CoRR*, vol. abs/1611.05431, 2016.

[YYD+16] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola and E. Hovy, 'Hierarchical attention networks for document classification,' in *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, 2016, pp. 1480–1489.

[YMR+21] T. Yau, S. Malekmohammadi, A. Rasouli, P. Lakner, M. Rohani and J. Luo, *Graph-sim: A graph-based spatiotemporal interaction modelling for pedestrian action prediction*, 2021. arXiv: 2012.02148 [cs.CV].

[ZBS17] S. Zhang, R. Benenson and B. Schiele, *Citypersons: A diverse dataset for pedestrian detection*, 2017. arXiv: 1702.05693 [cs.CV].

[ZLXL19] J. Zhao, Y. Li, H. Xu and H. Liu, 'Probabilistic Prediction of Pedestrian Crossing Intention Using Roadside LiDAR Data,' *IEEE Access*, vol. 7, pp. 93 781–93 790, 2019, ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2927889.

[ZLL+20] Y. Zhu *et al.*, *A comprehensive study of deep video action recognition*, 2020. arXiv: 2012.06567 [cs.CV].