Politecnico di Torino

Politecnico di Torino

Master's Degree in Computer Engineering

A.Y. 2024/2025

Graduation Session July 2025

A Concept-Based Explainable AI Approach to Action Recognition in Autonomous Driving

Supervisor:
Prof. Tania CERQUITELLI
Prof. Carla Fabiana CHIASSERINI
Phd. Gabriele CIRAVEGNA
Phd. Marco PALENA

Candidate:
Antonio IORIO

Abstract

In recent years, the field of autonomous driving has garnered increasing interest. Beyond ensuring the reliable operation of autonomous vehicles (AV), increasing attention has been directed towards understanding how these vehicles make decisions and predictions. Enhancing user trust in AVs requires developing situational awareness and decision-making processes that approximate human-level performance. Incorporating these capabilities in AV not only improves safety but also facilitates more intuitive and reliable human-machine interactions. Achieving this requires AI models capable of interpreting and responding to complex road scenarios in a manner similar to an experienced human driver. The ability to accurately predict and respond to dynamic traffic conditions is critical for the success and widespread adoption of AVs.

Given the safety-critical nature of autonomous driving, interpretable predictive models are essential for building user trust. This thesis focuses on integrating methodologies that enhance the explainability of AV decision-making, making the reasoning behind actions more understandable and transparent.

The initial part of this work reviews the state of the art by analyzing various frameworks designed to collect, integrate, and fuse heterogeneous data from multiple sensors — such as cameras, GPS, and LiDAR— to support optimal decision-making and control commands. This project focuses exclusively on data from a centrally positioned, front-facing camera aligned with the vehicle's direction of travel. Accordingly, the ROAD dataset was selected as the most suitable for this case study. The dataset is particularly well-suited for analysing road scenarios and developing models that improve situational awareness and decision-making in autonomous vehicles, stems form its annotation-based structure, which utilizes Road Events to describe key elements within a scene.

Several neural network architectures are evaluated to compare the generalization capabilities of black-box models with those of explainable approaches. The main goal is to introduce a level of interpretability into the decision-making process, in order to better understand the rationale behind the actions taken by the ego-vehicle. This is achieved by applying a Concept Bottleneck Model (CBM) approach, which enables intermediate predictions of human-interpretable concepts. To identify high-level concepts within the images the 3D-RetinaNet model was employed. This model enables the extraction of concepts, i.e., key elements from the scene, similar to the way a human driver would reason when deciding on an action. Based on these extracted concepts, various neural networks were subsequently tested to predict the autonomous vehicle's decision-making behaviour, leading to a model with enhanced interpretability by grounding decisions in identifiable scene elements.

While explainability provides valuable insights into the decision-making process, it can sometimes lead to a reduction in predictive performance. Therefore, it is important to find an appropriate trade-off between model interpretability and performance.

Table of Contents

List of Tables

List of Figures

Glossary

AI

Artificial Intelligence.

AV

Autonomous Vehicle.

GPS

Global Positioning System - A satellite-based navigation system that provides precise location and time information.

LiDAR

Light Detection and Ranging - A remote sensing technology that uses laser pulses to measure distances and create 3D models of environments.

INS

Inertial Navigation System.

XAI

Explainable Artificial Intelligence.

C-XAI

Concept-based eXplainable Artificial Intelligence.

DNN

Deep Neural Network.

DRIVE

Dependable Robust Interpretable Visionary Ensemble.

DCG

Driving through the Concept Gridlock.

CBM

Concept-Bottleneck Model.

MLP

Multilayer Perceptron.

ResNet

Residual Network.

CNN

Convolutional Neural Network.

FPN

Feature-Pyramid Networks.

RE

Road Event.

Chapter 1

Introduction

In recent years, the development of autonomous driving technologies has become a focal point of research and innovation. Autonomous vehicles (AVs) promise to revolutionise transportation by enhancing safety, efficiency, and convenience. However, the widespread adoption of AVs hinges not only on their technical capabilities but also on their ability to make decisions that inspire confidence in human users.

The decision-making process of AVs is a complex interplay of perception, prediction, and action. To achieve human-like performance, AVs must develop advanced situational awareness and decision-making strategies that closely mimic those of experienced human drivers. This requires the integration of diverse data sources, such as cameras, GPS, and LiDAR, to create a comprehensive understanding of the driving environment.

Despite the availability of multimodal data, this thesis focuses on the use of data from a single, centrally positioned, front-facing camera. This design choice aligns with the goal of developing and evaluating models that can interpret driving environments in a manner analogous to human perception. By focusing on a single, forward-facing camera, the dataset simplifies the multimodal complexity of the original data while preserving the essential visual cues necessary for understanding dynamic road scenarios.

1.1 Formulation of the problem

In recent years, the development of autonomous driving systems has attracted increasing interest within the scientific and industrial communities. Most proposed solutions have focused on learning models that, starting from perceptual inputs (such as images or sensor data), are able to directly generate the action to be performed. However, little attention has often been given to understanding how such decisions

are made within the model, thus neglecting the aspect of explainability. This aspect is crucial, especially in a critical context such as autonomous driving, where a user — such as an experienced driver — might want to understand the reasons that led a neural system to choose a particular action. The lack of transparency in black-box models raises concerns about their reliability, interpretability, and acceptability in real-world scenarios.

This awareness has motivated the exploration of alternative approaches that, beyond providing accurate predictions, are also able to offer an interpretable explanation of the decision-making process. It is within this framework that the present work is positioned, aiming to integrate symbolic components within neural architectures to make explicit the key concepts underlying the decisions of the autonomous vehicle.

1.2 Contributions

The goal of this thesis is to address the problem outlined earlier — namely, designing a model capable of providing understandable explanations for how and why a certain action is chosen by an autonomous driving system.

The first step involved analysing the state of the art in order to identify existing approaches and assess their limitations in terms of explainability. Then, a most suitable dataset for the project was selected: ROAD [1]. This dataset was chosen not only because it allows working with a front-view perspective, but more importantly because of its semantic annotations — the so-called Road Events (REs) — which describe the scene context and represent symbolic concepts relevant to the decision-making process. The work proceeded in several phases. First, two models — a Multilayer Perceptron (MLP) and a Transformer — were developed with the aim of predicting the vehicle action from the conceptual representation of the scene. Then, black-box models were evaluated in order to compare their predictive performance and explore potential trade-offs between accuracy and interpretability. Finally, a Concept Bottleneck Model (CBM) was implemented to introduce a layer of explainability into the system. Specifically, 3D-RetinaNet was used to predict the concepts (REs) from the input image, followed by one of the previously developed models (MLP or Transformer) to predict the final action based on the predicted concepts. This approach enables the generation of interpretable explanations for the vehicle's decisions, enhancing the transparency of the decision-making process.

1.3 Outline of the work

The work presented in this thesis is structured into five chapters, each addressing a specific topic. Below is a brief overview of the content of each chapter:

- **Chapter 1 - Introduction**: This chapter introduces the general context of the thesis, clearly formulating the problem to be addressed and outlining the main contributions of the work. Based on this analysis, the main objective of the work is defined—that is, the specific challenge the thesis aims to overcome.
- **Chapter 2 - Related Work**: The chapter analyzes the fundamental mechanisms and advanced solutions for autonomous driving, providing the necessary context for the research problem. It also introduces the concept of explainability (XAI), with a focus on the Concept-Based XAI approach adopted to ensure model transparency and an analysis of the datased used.
- **Chapter 3 - Methodology**: The chapter presents the methodologies adopted to carry out the experiments, highlighting the choices made to address the encountered challenges and improve prediction accuracy. It includes an analysis of the neural network baselines, and an explanation of how the information was integrated to introduce a level of explainability in the decision-making process.
- **Chapter 4 - Experimental Results**: This chapter presents the results obtained by applying the methodologies described in the previous chapter. The main experiments are discussed through tables that summarize the configurations used and the corresponding evaluation metrics.
- **Chapter 5 - Conclusions And Future Works**: The final chapter summarizes the main contributions and results achieved throughout the research. It also discusses possible future directions, suggesting improvements and potential developments to advance explainability in autonomous driving based on the findings obtained.

Chapter 2

Related Works

This chapter introduces the main concepts and paradigms underlying autonomous driving systems. It begins by outlining the fundamentals of deep learning, the core technology powering modern autonomous vehicles, and discusses its role in perception and decision-making tasks. The chapter then highlights the importance of interpretability, a crucial aspect for building reliable and transparent AI systems. It presents the state-of-the-art in neural network architectures commonly used in this field. Finally, an analysis of the dataset used, including the rationale behind its selection and a discussion of its main characteristics.

2.1 Deep Learning

In recent years, deep learning has emerged as the dominant paradigm in AI, due to its ability to learn complex relationships and representations between inputs and outputs through the analysis of large-scale data [2]. Models based on deep neural networks have achieved outstanding results in fields such as image recognition, motion planning, and semantic segmentation. This success is largely attributed to their layered architecture (Figure 2.1), which enables the hierarchical extraction of features from raw input data.

A deep neural network can be modelled as a function $f : \mathbb{R}^n \to \mathbb{R}^m$, where $x \in \mathbb{R}^n$ represents the input space and $y \in \mathbb{R}^m$ the output space. Given an input vector $x \in \mathbb{R}^n$, the network produces an output vector $y = f(x)$, where $y \in \mathbb{R}^m$, through a sequence of layers, each performing a linear transformation followed by a nonlinear activation. In this context, x may represent, for example, a feature vector extracted from an image, while y could represent a set of class probabilities in classification tasks.

However, in order to achieve high performance in classification tasks, these models require extensive amounts of labelled training samples, from which they can

extract statistically useful regularities. While this enables remarkable performance in closed and well-defined domains, it also constitutes one of the major limitations of deep learning systems.

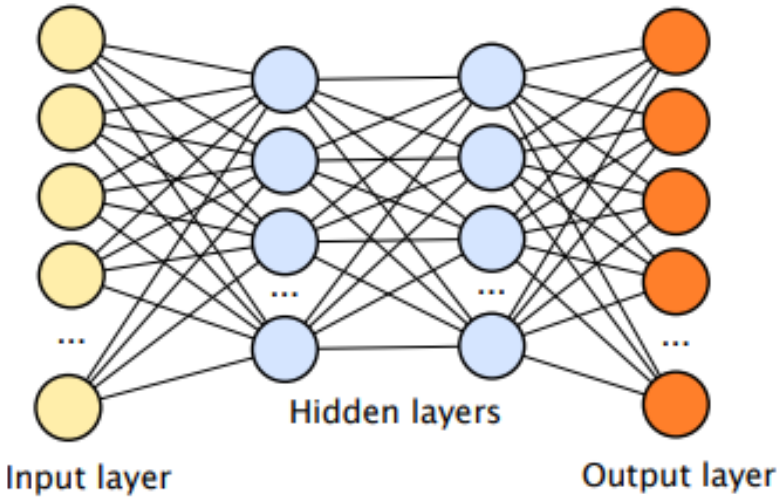


Figure 2.1: Diagram of a deep neural network. The input and output layers are shown on the left and right, respectively, with the number of output nodes matching the number of predicted classes. The two hidden layers in the center illustrate the network's internal structure, which in practice may consist of dozens, hundreds, or even thousands of layers (see [2]).

Deep learning models often exhibit a limited ability to abstract. As a result, specific approaches are needed to enhance their generalization capabilities. In the field of computer vision, a particularly relevant task is multi-view classification [3], where the same object is observed from different perspectives through images captured by multiple devices. This approach enables more accurate and detailed predictions by leveraging the complementarity of visual information. Contrary to traditional strategies that rely on data from a single viewpoint, multi-view approaches are capable of collecting and fusing data from multiple sensors. In this context, two main strategies can be adopted:

- **Centralized inference schemes**: each source node transmits the collected data (images or features) to a central controller, which performs the entire inference task (Figure 2.2).
- **Ensemble inference schemes**: each node performs a local classification on its own view and sends only the predicted label (or probability distribution) to the controller (Figure 2.3).

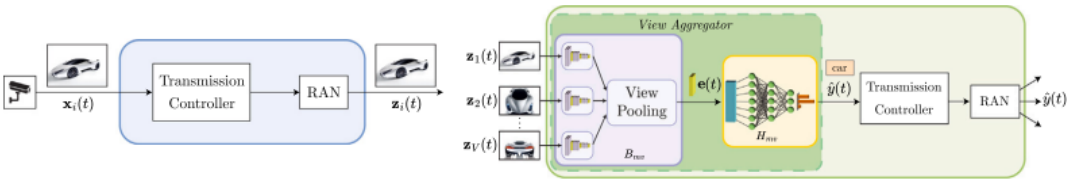


Figure 2.2: Schematic representation of the operations performed by a source node (left) and the central controller (right) in centralized inference schemes (see [3]).

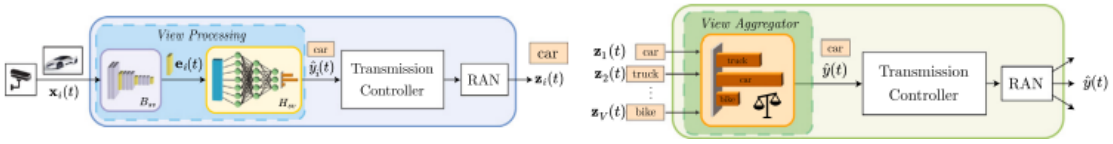


Figure 2.3: Schematic representation of the operations performed by a source node (left) and the central controller (right), in ensemble inference scheme (see [3]).

2.1.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is a specific neural network architecture, also known as a fully connected feedforward neural network [4]. It consists of a sequence of layers in which each neuron in a given layer is connected to all neurons in the subsequent layer. The architecture includes an input layer, one or more hidden layers, and an output layer. The objective of propagating data through these layers is to learn an approximating function $y = f(x, \theta)$, where x represents the input, y the desired output, and θ the set of parameters (weights and biases) that are optimized during training. Introducing non-linear activation functions after each affine transformation is essential for the network to learn complex and non-linear mappings.

Figure 2.1 illustrates the architectural structure of a MLP, highlighting the sequence of fully connected layers through which the input is progressively transformed into the final output.

2.1.2 Transformer

The **transformer** is an architecture designed to model dependencies within a sequence through a mechanism known as self-attention [5]. This mechanism allows each element in the sequence to dynamically weigh the relevance of other elements, enabling the construction of context-aware representations. For each input element, three vectors are computed through learned linear projections and are used to compute self-attention, as defined in Equation 2.1:

$$Attention(Q, K, V) = \text{softmax}\,(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

where:

- **Query (Q)**: what the element is looking for in the context, $Q \in \mathbb{R}^{n \times d_k}$,
- **Key (K)**: what each element offers to the context $K \in \mathbb{R}^{n \times d_k}$,
- **Value (V)**: the information that will be aggregated and passed on $V \in \mathbb{R}^{n \times d_k}$,
- d_k dimensionality vectors (used for scaling),
- n is the sequence length.

To enable the model to capture different types of relationships, a mechanism called **multi-head attention** is employed. It applies the attention function in parallel across multiple heads, each with its own learned linear projections and operating in a separate subspace. The outputs of all heads are then concatenated and linearly projected, as shown in Equation 2.2:

$$MultiHead(Q, K, V) = \text{Concat}(head_1, ..., head_h)W^O \tag{2.2}$$

where:

$$head_i = \text{Attention}(\text{QW}_\text{i}^\text{Q}, \text{KW}_\text{i}^\text{K}, \text{VW}_\text{i}^\text{V}) \tag{2.3}$$

where:

- W^O: is the output projection matrix applied to the concatenated attention heads to produce the final output of the multi-head attention mechanism, $W^O \in \mathbb{R}^{hd_v \times d_{model}}$
- W_i^Q: is the projection matrix that maps the input Q into the query representation for the i-th head, $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$
- W_i^K: is the projection matrix that maps the input K into the key representation for the i-th head, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$
- W_i^V: is the projection matrix that maps the input V into the value representation for the i-th head, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$

Additionally, it is common practice to set $d_k = d_v = \frac{d_{model}}{h}$ where h is the number of attention heads. This ensures that the concatenation of the outputs from all heads results in a vector of dimension d_{model}, matching the expected input size for the output projection.

However, since the attention mechanism is permutation-invariant and does not inherently encode the order of elements in a sequence, **positional encoding** is added to the input embeddings to incorporate information about the relative or absolute positions of tokens in the sequence. This allows the model to leverage the sequential structure of the data during training and inference.

A visual representation of how scaled dot-product attention and multi-head attention are computed is provided in Figure 2.4.

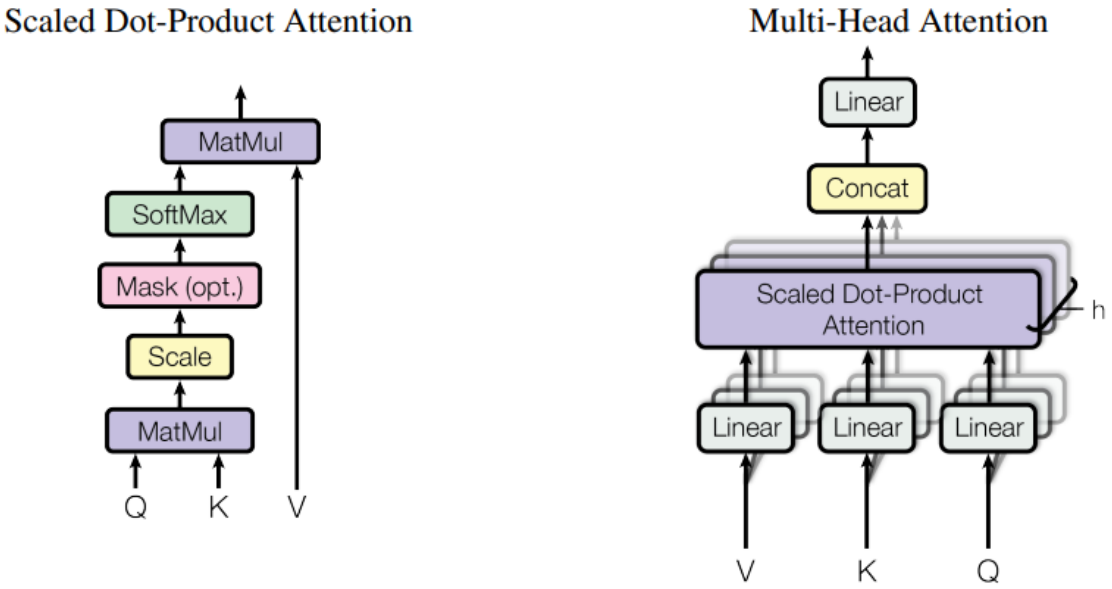


Figure 2.4: Left: Scaled Dot-Product Attention. Right: Multi-Head Attention, composed of several attention layers running in parallel (see [5]).

Figure 2.5 provides a schematic overview of the Transformer architecture, highlighting its main components and computational flow.

2.1.3 Deep Learning In Autonomous Driving

Deep learning is widely adopted in autonomous driving systems to enable vehicles to navigate safely and reactively within complex environments. Autonomous driving system typically comprises three main functional modules:

- **Perception**: a module that captures the surrounding scene to build a comprehensive description, generating raw data used to identify, classify, and track objects.
- **Decision-making**: based on the perceived information, this module determines the action the vehicle should take, selecting the most appropriate maneuver according to the surrounding environment.

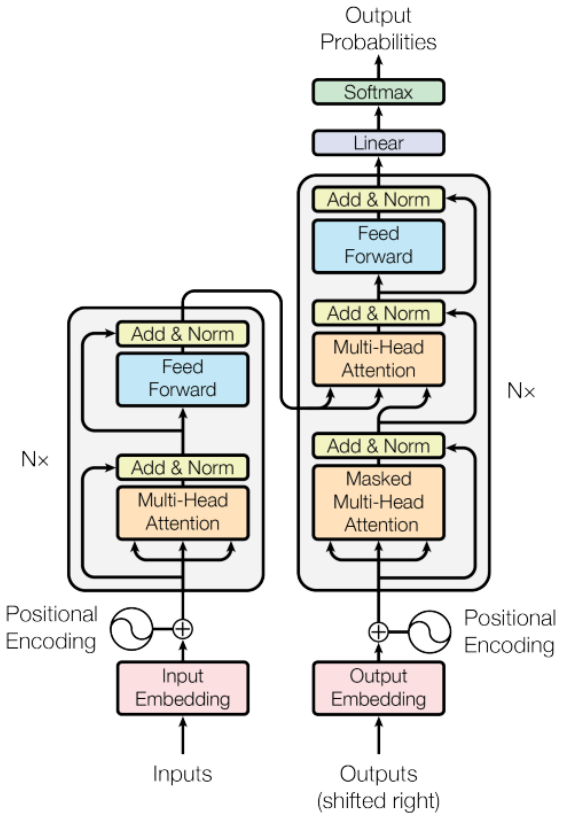


Figure 2.5: Transformer architecture (see [5]).

- **Control**: once the action to be taken is determined, this module executes the necessary commands to implement the chosen decision.

Deep learning plays a fundamental role in the decision-making module, enabling the system to learn complex patterns from perceived data and make more accurate and adaptive decisions in dynamic environments.

To accurately perceive their surroundings and make informed decisions, autonomous vehicles rely on a suite of sensors—such as cameras, LiDAR, RADAR-vehicle-to-vehicle (V2V) communication systems that provide heterogeneous and complementary information. Building on this multi-sensor approach, the study [6] investigate how autonomous vehicles make decisions in mixed-autonomy traffic, specifically in left-turn scenarios at intersections regulated by traffic lights and stop signs. Using simulated driving data and binary action labels ("go"/"no-go"), it compares the performance of an interpretable model, Time Series Forest (TSF), with two state-of-the-art classifiers, ROCKET [7] and HIVE-COTE 2.0 [8]. The results show that TSF performs competitively—particularly at stop-sign-controlled intersections—and that the inclusion of V2V data improves both predictive accuracy

and the interpretability of driver behaviour.

Vista [9] is a generalizable driving world model characterized by high-fidelity prediction and versatile controllability, designed to enhance dynamic forecasting and action evaluation in autonomous driving scenarios. Its primary objective is to overcome the limitations of previous models by generating frames at higher spatial resolution and increased frame rates (Figure 2.6), thereby preserving critical scene details more effectively. The training process is divided into two distinct phases:

1. **Learning High-Fidelity Future Prediction**: focuses on long-term future state prediction based on key dynamic priors such as position, velocity, and acceleration (Figure 2.6).

2. **Learning Versatile Action Controllability**: integrates multiple modalities of action control, ranging from high-level intentions, such as commands and goal points, to low-level maneuvers including trajectories, steering angles, and speeds. During this phase freezing the pretrained weights to learn action controls (Figure 2.6).

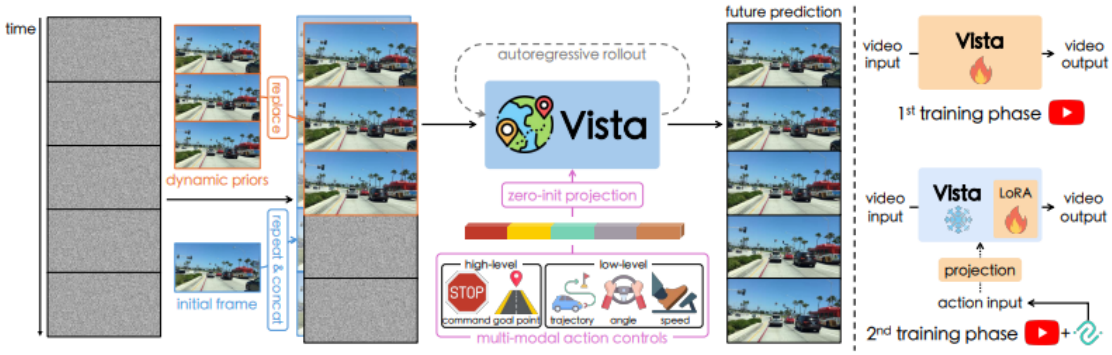


Figure 2.6: Left: Vista uses, in addition to the initial frame, additional information about future dynamics via latent replacement. Right: Two training phase (see [9]).

Leveraging a unified conditioning interface alongside novel optimization strategies, Vista demonstrates significant generalization capabilities across diverse environments and scenarios. Moreover, the model can serve as a generalizable reward function by assessing the quality of actions through the uncertainty in its own predictions, without requiring access to ground-truth data.

2.2 Explainable AI

The introduction of artificial intelligence (AI) in sensitive fields such as medicine, finance, and autonomous driving has made it necessary to increase user trust in

model predictions. To achieve this goal, it is essential that models are able to explain how predictions are generated, that is, to provide details on the factors that influenced a particular decision. This need has led to the emergence of Explainable Artificial Intelligence (XAI), where the **explanation** is the collection of features of interpretable domain that have contributed for a given example to produce a decision [10]. To foster trust and effective interaction with AI systems, explanations should be accessible to the average user, not just experts. This highlights the importance of interpretability in XAI. **Explainability** refers to the overall ability of an AI model to provide understandable and meaningful explanations for its behaviour or decisions. **Interpretability**, on the other hand, is defined as the ability to explain or to present in understandable terms to humans. There are two main approaches to achieving interpretability and explainability in AI models:

- **Transparency-based**: models are interpretable by design, where the structure itself enables users to understand how decisions are made. Simple models such as linear models, decision trees, and rules are examples of transparent models. However, transparency and predictive performance are often conflicting goals and must be balanced.

- **Post hoc**: applied to complex black-box models (e.g. deep neural networks), these methods generate explanations after training, typically without altering the model.

2.2.1 Concept-based Explainable AI

Traditional XAI techniques typically provide explanations at the feature level, focusing on low-level attributes. While effective in some scenarios, these explanations often lack semantic meaning for non-expert users. To address this limitation, recent research has introduced Concept-Based Explainable Artificial Intelligence (C-XAI) [11], which aims to generate explanations based on high-level human-interpretable attributes, referred to as **concepts**. Concept-based explanations should explain how deep neural network (DNNs) make particular decisions using concepts [12] and adhere to specific criteria such as being meaningful, coherent, and relevant to the final class [13] and also explicit and faithful [14]. The notion of a concept is inherently subjective and domain-dependent. In this work, the notion of symbolic concepts is adopted, i.e. semantic attributes recognisable by humans, such as colours, shapes, or identifiable objects.

For instance, as shown in Figure 2.7, the presence of a bird's beak is a discriminative feature useful for classification. By leveraging the class-concept relation — that is, the association between specific concepts and output classes — it becomes possible to improve the interpretability of predictions.

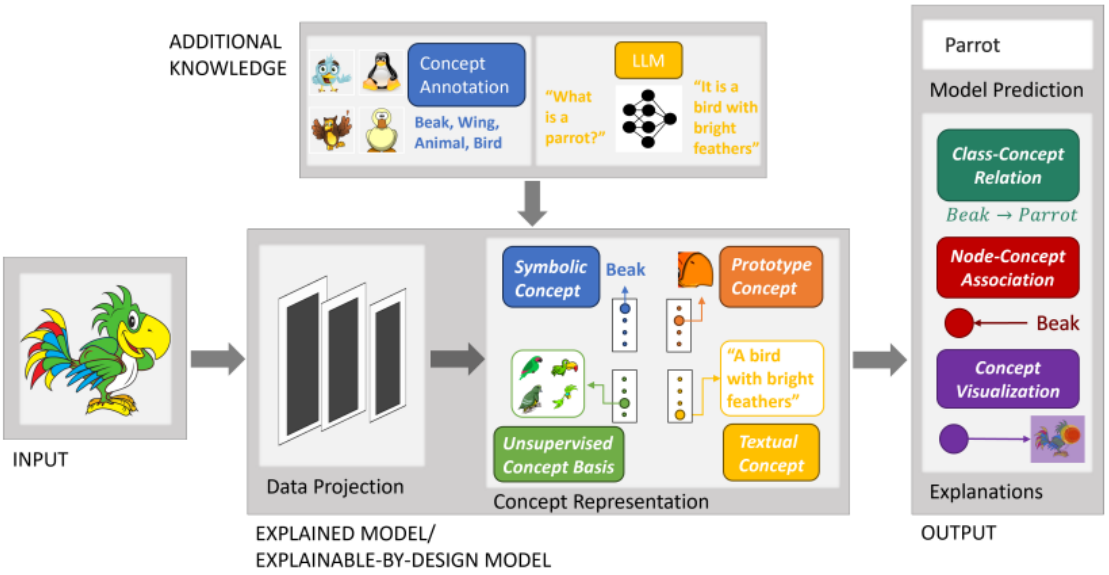


Figure 2.7: Concepts and explanations provided by C-XAI methods and models (see [11]).

2.2.2 C-XAI In Autonomous Driving

One of the main frameworks addressing C-XAI in the context of autonomous driving is DRIVE [15], which aims at improving trust and safety in autonomous driving systems. Specifically, DRIVE enhances the Driving through the Concept Gridlock (DCG) model, a post-hoc explainability framework for autonomous driving designed for deep learning models. DRIVE aims to overcome the instabilities of DCG, which are due to the sensitivity of input perturbations and parameter variations. For this reason, DRIVE builds upon four foundational properties that ensure both interpretability and robustness:

- **Consistent Interpretability (Ci)**: interpretability of the model's output should be within a bounded distance from that of its original counterpart,
- **Stable Interpretability (Si)**: interpretability must remain stable under perturbations to the input,
- **Consistent Output (Co)**: output of the model should closely match that of the original DCG,
- **Stable Output (So)**: predictive output should also exhibit stability when subjected to input perturbations.

To achieve these goals, DRIVE uses a multi-objective optimization process and techniques such as Projected Gradient Descent (PGD) to enhance robustness against perturbations while ensuring interpretable and predictive consistency.

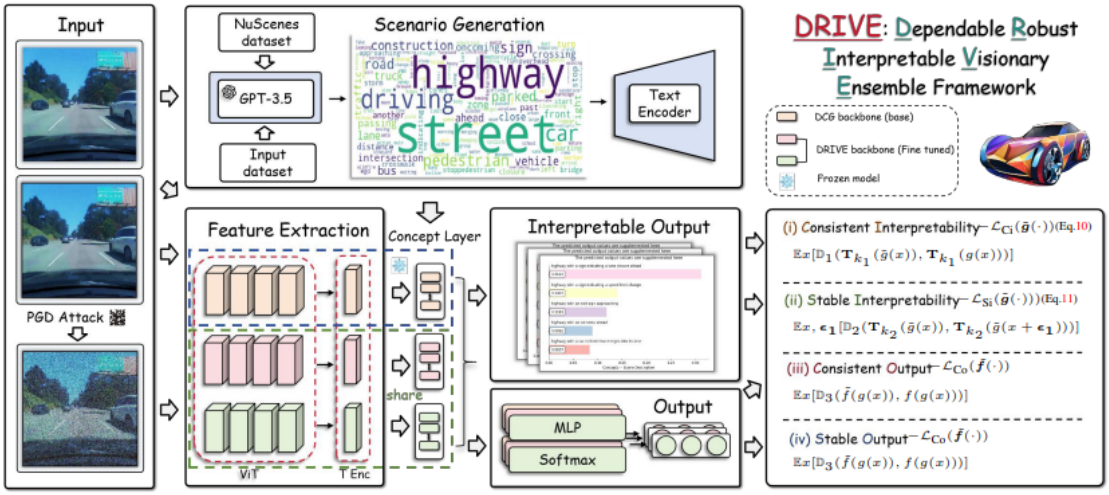


Figure 2.8: Overall pipeline of DRIVE (see [15]).

Figure 2.8 shows the DRIVE pipeline and, consequently, its main components. In particular, it is possible identify the following key elements:

- **Feature Extraction**: extracts relevant features from the input images, mapping input x into a meaningful concept space,
- **Concept Bottleneck**: driving scenarios s are constructed to describe scenes and encode contextual information. Scenarios are generated using the generative capabilities of GTP-3.5 [16] and human-created scene descriptions from the NuScenes dataset [17],
- **Temporal Encoding**: Longformer architecture [18] to capture the temporal dynamics of the input sequences,
- **Multi-Objective Optimization**: process to balance the four key attributes described above: Ci, Si, Co, and So,
- **Training**: models are trained using Root Mean Squared Error (RMSE) loss.

Experiments [15] have shown that DRIVE significantly improves the stability and reliability of explanations and predictions compared to the DCG model.

2.3 Concept Bottleneck Models

This section introduces Concept Bottleneck Models (CBMs), a class of models designed to enhance the level of explainability in machine learning systems [11]. CBMs follow a two-stage approach: in the first stage, the model predicts a set of human-interpretable concepts, $g : \mathbb{R}^n \to \mathbb{R}^k$, where $x \in \mathbb{R}^n$ represents the input space, and $c \in \mathbb{R}^k$ corresponds to an intermediate space of interpretable concepts. Which are then used in the second stage to perform the final classification, $f : \mathbb{R}^k \to \mathbb{R}^m$ then maps these concepts to the final output $y \in \mathbb{R}^m$. Thus, the overall prediction becomes $y = f(g(x))$. This structure promotes greater transparency and interpretability in the model's decision-making process.

Figure 2.9 illustrates how a CBM works. Specifically, it shows how the input—an image in this case—is mapped to a set of interpretable concepts that capture the main semantic aspects of the scene. These concepts are then used to perform the final prediction task.

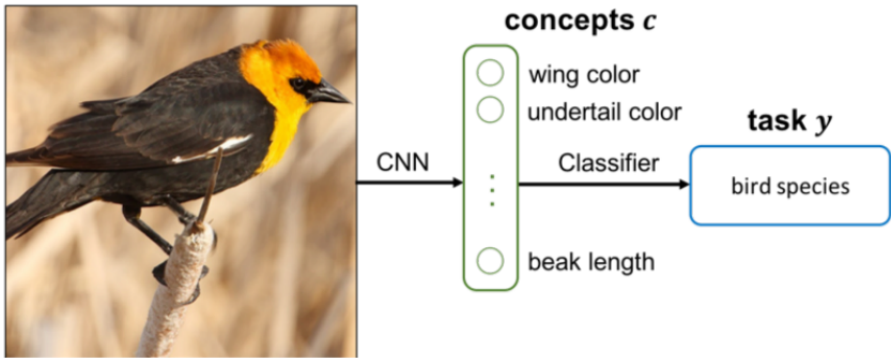


Figure 2.9: Overall pipeline of CBM (see [19]).

Specifically, two possible architectures for implementing this approach will be analyzed: a Multilayer Perceptron (MLP) and a Transformer, both employed to learn the mapping from annotated concepts to the target class.

2.4 ResNet-50

ResNet-50 is an architecture designed for image recognition tasks, belonging to the family of Residual Networks (ResNets), introduced to overcome key limitations of deep convolutional neural networks [20]. These models have proven particularly effective in addressing two well-known issues: the vanishing/exploding gradient problem and the degradation of performance as network depth increases.

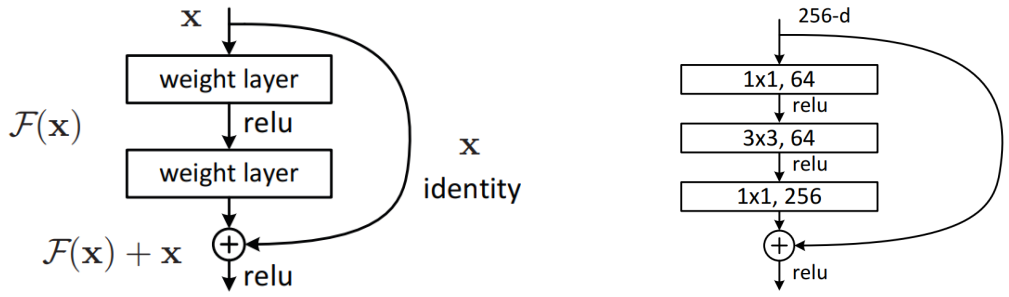
The core innovation of ResNet lies in the introduction of a residual learning framework, where network blocks are designed to learn a residual function defined

as $f(x) = h(x) - x$, rather than directly learning the target function $h(x)$.

A typical residual network block (Figure 2.10a) consist of two or more convolutional layers and a shortcut connection that bypasses those layers and adds the original input directly to the output of the block. The final output of the block is then given by $h(x) = f(x) + x$.

These connections do not introduce additional parameters or increase computational complexity, and they facilitate gradient backpropagation, thus enabling the effective training of very deep networks.

In moderately deep architectures, standard building blocks are employed, whereas in deeper networks, such as ResNet-50, a bottleneck block (Figure 2.10b) is used to optimize computational efficiency while preserving the model's representational capacity.



(a) Building block **(b)** Bottleneck building block

Figure 2.10: Comparison between two types of residual blocks: (a) highlights the relationship between the functions $H(x)$ and $F(x)$; (b) depicts the bottleneck design, adopted in deeper networks such as ResNet-50 (see [20]).

Figure 2.12a illustrates the architecture of a generic ResNet model. Compared to this general structure, ResNet-50 does not introduce architectural changes in terms of the number or organization of the bottleneck building blocks. The main differences lie instead in the internal configuration of the convolutional layers, such as kernel sizes, bottleneck structures, and channel dimensions [20].

2.5 R2Plus1D_18

The R(2+1)D is a 3D convolutional neural network architecture designed for video action recognition. It can be regarded as an extension of traditional $2D$ convolutional networks, such as the ResNet-50 model discussed in section 2.4, which applies convolutions only in the spatial domain and processes video frames independently, without capturing the temporal dynamics between them.

In contrast, R(2+1)D introduces an explicit temporal modelling component,

making it particularly suitable for action recognition tasks in video clips [21]. Unlike conventional 3D architectures that perform joint convolutions over all three dimensions $T \times W \times H$, the R(2+1)D model factorizes the $3D$ operation into a sequential application of a $2D$ spatial convolution followed by a $1D$ temporal convolution. This decomposition introduces a structural simplification while enhancing representational power.

Consequently, while in conventional $2D$ convolutional networks the input to the model typically consists of a single image — from which an output such as a classification prediction is produced — in the case of the R(2+1)D model, the objective remains unchanged, but the nature of the input differs. Specifically, the model receives as input a sequence of consecutive frames, usually 8 or 16, extracted from a video. The output, namely the classification, is associated with the last frame in the sequence, referred to as the target frame. This approach allows the model to explicitly observe the temporal dynamics leading up to the target frame, thereby improving context understanding and the quality of the final prediction. The availability of multiple preceding time steps enables the construction of a more informed and robust representation of the ongoing action.

Figure 2.11 provides a visual comparison between a standard $3D$ convolutional block and a $(2+1)D$ block, illustrating the described factorization.

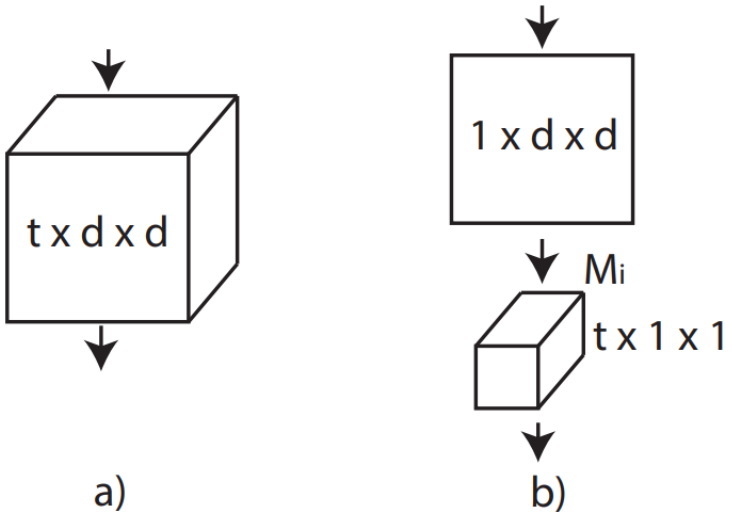
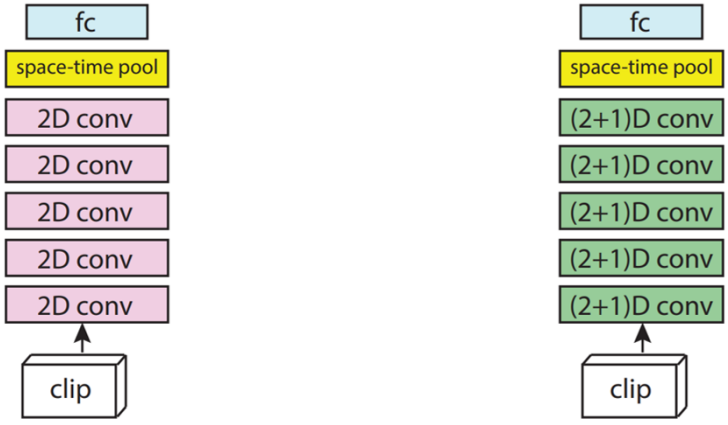


Figure 2.11: (a): Standard $3D$ convolution, where spatial and temporal dimensions are convolved jointly. (b): $R(2+1)D$ convolution, where spatio-temporal operation is factorized into a $2D$ spatial convolution followed by $1D$ temporal convolution.

A non-linear activation function (ReLU) is placed between the two convolutional steps, effectively doubling the number of non-linearities and allowing the model to

represent more complex functions without increasing the number of parameters. Additionally, the separation of spatial and temporal components facilitates optimization during training, leading to lower training and validation errors. These advantages make R(2+1)D a more effective and computationally efficient solution for video action recognition compared to both traditional $2D$ and $3D$ architectures.

Figure 2.12b illustrates a generic architecture of an R(2+1)D network. The R(2+1)D_18 model does not differ in terms of the number or organization of the residual blocks; the variation lies solely in the convolutional properties, such as kernel configuration and channel dimensions, according to the specific depth of the network.



(a) ResNet 2D Architecture **(b)** ResNet 2+1D Architecture

Figure 2.12: (a): Generic architecture of a ResNet model that processes only spatial features, using 2D convolutions applied independently to each frame. (b): Architecture of the R(2+1)D network, where spatio-temporal modeling is achieved by factorizing $3D$ convolutions into separate spatial ($2D$) and temporal ($1D$) operations.

2.6 3D-RetinaNet

3D-RetinaNet is a network inspired by 3D convolutional neural network (3D CNN) architectures [22] for video action recognition and feature-pyramid networks (FPN) [23].

FPNs are an architecture designed to address the problem of multi-scale object detection, namely the ability to detect objects of varying sizes (small, medium, large). In traditional object detection models, small objects tend to be overlooked, as deeper layers of the network gain semantic abstraction at the expense of spatial detail. An FPN is composed of three main components:

- **Bottom-up pathway**: this represents the standard flow of a CNN, where feature maps with increasing semantic richness but decreasing spatial resolution are extracted. Deeper feature maps are useful for detecting larger and more prominent objects.
- **Top-down pathway**: this consists of an upsampling process applied to deeper feature maps, restoring them to higher resolutions to recover spatial information. This enhances the network's ability to detect smaller or partially visible objects.
- **Lateral connections**: these are skip connections that merge information from the bottom-up and top-down pathways, integrating semantic content with spatial detail to generate a rich, multi-scale feature representation of the scene.

Within this structure, objects are associated with anchor boxes [24] —predefined bounding boxes with fixed sizes and aspect ratios—which serve as references for locating objects in the image. Each level of the FPN is responsible for detecting objects at a specific scale, thus enabling the model to effectively handle objects of different sizes. The anchor boxes are subsequently refined by the network to produce the final bounding box predictions.

The model's input consists of a temporal sequence of T video frames, from which it extracts a set of feature-pyramid maps [1]. For each generated anchor on these feature maps, the network predicts bounding boxes (expressed as 4 coordinates) and confidence scores for multiple classes. The architecture is flexible and allows for the use of different backbones, such as ResNet-50, Inflated 3D (I3D), and Slowfast, enabling a balance between accuracy and computational efficiency.

The model adopts a multi-label approach, predicting multiple concepts simultaneously—including agent classes, actions, and positions—and incorporates a dedicated classification head for predicting the actions performed by the autonomous vehicle itself (AV-action). A key feature of the architecture is its ability to operate in an online manner, updating predictions as new frames are acquired, without the need to reprocess the entire past sequence.

Figure 2.13 illustrates the proposed architecture for 3D-RetinaNet, which takes as input a video sequence from which hierarchical spatio-temporal multi-level features are extracted. These features are processed by a FPN that enables the detection of objects and events of varying sizes and durations. As shown in the figure, the pyramid consists of multiple levels, each including two subnetworks: the first is responsible for classifying anchor boxes and estimating the confidence of belonging to different classes, while the second performs regression of the associated bounding box coordinates. Additionally, an auxiliary branch of the architecture predicts a temporal sequence of actions performed by the AV.

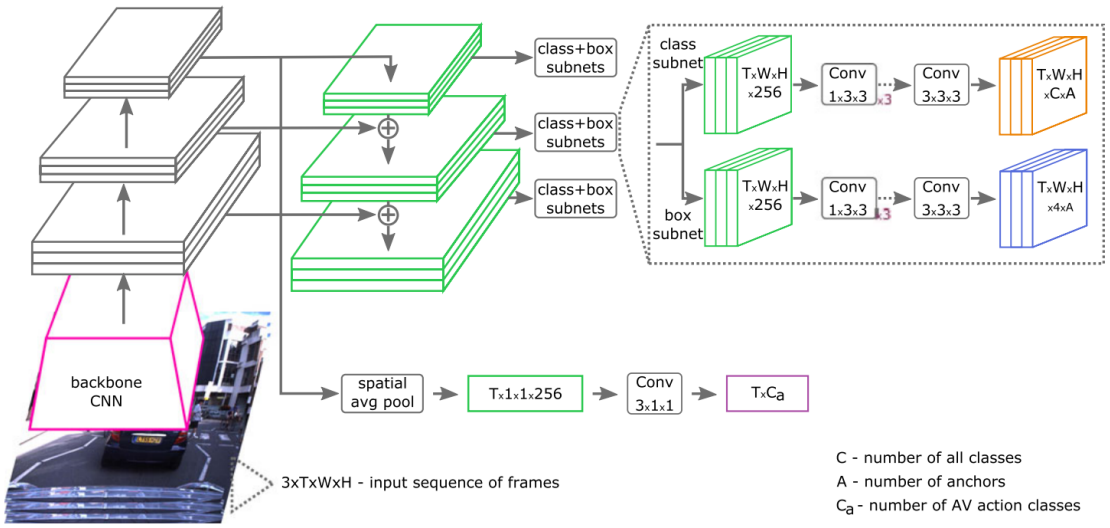


Figure 2.13: Proposed 3D-RetinaNet architecture for online video processing (see [1]).

2.7 ROAD

The ROAD dataset [1] represents a fundamental resource for studying the interactions between agents in the scene and the corresponding actions of the autonomous vehicle (AV). Thanks to its detailed annotations, the dataset captures the complex dynamics of traffic and provides a rich context for developing models capable of effectively predicting and managing critical driving situations. In particular, ROAD was specifically designed to evaluate an AV's ability to recognize and correctly interpret Road Events (REs).

It was derived from videos of Oxford RobotCar Dataset [25]. The latter consists of footage captured as a car drives through the main streets of Oxford, UK, repeatedly traversing the same routes to record a variety of conditions and situations, including heavy rain, nighttime, direct sunlight, and snow. Filming took place between May 6, 2014, and December 13, 2015, resulting in a total of 1,010.46 km of recorded street footage.

The dataset was collected using Oxford RobotCar platform, an autonomous-capable Nissan LEAF, equipped with a heterogeneous set of sensors, including multiple cameras, LiDAR, GPS, and INS. However, in the context of the ROAD dataset, only one of these sensors is used:

> Point Grey Bumblebee XB3 (BBX3-13S2C-38) trinocular stereo camera, 1280× 960 × 3, 16 Hz, 1/3″ Sony ICX445 CCD, global shutter, 3.8 mm lens, 66° HFoV, 12/24 cm baseline.

However, the dataset used to conduct the experiments, ROAD, uses only a specific subset of the original dataset. Specifically, consisting of 22 video sequences recorded between 25 June 2014 and 3 March 2015. Video sequences are split into 18 videos for training and validation of the model, and the remaining 4 videos are used for testing the trained model.

Namely, a centrally placed, front-facing camera aligned with the vehicle's direction of travel. This configuration was chosen to replicate the visual perspective of a human driver, thereby making the dataset particularly suitable for tasks involving the analysis of road scenes from a human viewpoint.

2.7.1 Road Event

Each frame in the dataset is annotated based on the concept of **Road Events (REs)**. A RE is formally defined as a triplet $E = (A_g, A_c, Loc)$, where:

- A_g is the road agent,
- A_c represents the action performed by the agent,
- Loc indicates the location of the agent relative to the field of the AV.

Multiple REs can be identified within a single frame, consequently each of them can be spatially localised using a bounding box, defined by coordinates that precisely identify the corresponding agent in the image. This allows for a clear understanding of which visual element each annotated event refers to.

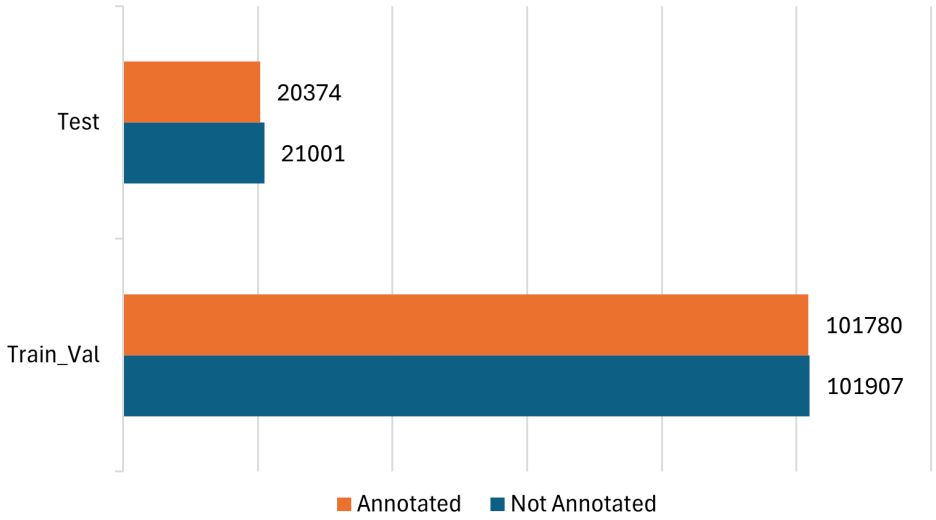


Figure 2.14: Distribution of the number of annotated and non-annotated frames in the training/validation and test datasets.

An analysis was conducted to check whether all frames were annotated, revealing that not all frames actually contain annotations. This aspect is illustrated in Figure 2.14, which compares the number of annotated and non-annotated frames, distinguishing between frames belonging to the training/validation videos and those belonging to the test videos. It can be observed that the vast majority of frames are annotated.

Road Agent

Objects or people able to perform actions which can influence the decisions made by the autonomous vehicle are termed **agents**. Agents are classified as active when they directly influence the decision-making process of the AV. Typically, these are agents that appear in the foreground of the scene, whose presence or behaviour requires a response from the system. In contrast, agents that appear in the background or do not interact meaningfully with the immediate road context are not considered particularly relevant for the final decision-making process. Table 2.1 shows and describes the possible agents annotated within the dataset.

Category	**Label**	**Description**
People	*Pedestrian*	a person including children
	Cyclist	a person is riding a push/electric bicycle
Vehicles	*Car*	a car up to the size of a multipurpose vehicle
	Small size *	***
	Medium vehicle	vehicle larger than a car, such as van
	Large vehicle	vehicle larger than a van, such as a lorry
	Bus	a single or double-decker bus or coach
	Motorbike	motorbike, dirt bike, scooter with 2/3 wheels
	Emergency vehicle	ambulance, police car, fire engine, etc
Traffic light	*Vehicle traffic light*	traffic light related to the AV lane
	Other traffic light	traffic light not related to the AV lane

Table 2.1: List of ROAD active agent classes, with description (see [1]).

Note:
* Defined in the paper, but never used in experiments
** Not defined in the paper, but used in experiments
*** Definition not provided.

Road Action

Actions represent the behaviours that agents can perform within the road scene. Each agent can be associated with one or more actions at a given point in time, depending on the context and its interaction with the surrounding environment.

Label	Description
Traffic light red	Traffic light with red light lit
Traffic light amber	Traffic light with amber light lit
Traffic light green	Traffic light with green light lit
*Traffic light black **	Traffic light with no lights lit or covered with an out-of-order bag
Moving away	Agent moving in a direction that increases the distance between Agent and AV
Moving towards	Agent moving in a direction that decreases the distance between Agent and AV
Moving	Agent moving perpendicular to the traffic flow or vehicle lane
Reversing	Agent is moving backwards
Braking	Agent is slowing down, vehicle braking lights are lit
Stopped	Agent stationary but in ready position to move
Indicating left	Agent indicating left by flashing left indicator light, or using a hand signal
Indicating right	Agent indicating right by flashing right indicator light, or using a hand signal
Hazard lights on	Hazards lights are flashing on a vehicle
Turning left	Agent is turning in left direction
Turning right	Agent is turning in right direction
Moving right	Moving lanes from the current one to the right one
Moving left	Moving lanes from the current one to the left one
Overtaking	Agent is moving around a slow-moving user, often switching lanes to overtake
Waiting to cross	Agent on a pavement, stationary, facing in the direction of the road
Crossing road from left	Agent crossing road, starting from the left and moving towards the right of AV
Crossing road from right	Agent crossing road, starting from the right pavement and moving towards the left pavement
Crossing	Agent crossing road
Pushing object	Agent pushing object, such as trolley or pushchair, wheelchair or bicycle

Table 2.2: List of ROAD action labels, with description (see [1]).

Note:
* Defined in the paper, but never used in experiments
** Not defined in the paper, but used in experiments
*** Definition not provided.

This dynamic association allows for a more realistic and detailed modelling of agent behaviour, reflecting the complexity of real-world scenarios. Furthermore, the ability to assign multiple actions to a single agent enables the capture of behavioural transitions or ambiguous situations, thereby enhancing the quality of annotations and the effectiveness of predictive models. The possible actions that can be performed are shown in Table 2.2

Location

Label	Description
In vehicle lane	Agent in same road lane as AV
In outgoing lane	Agent in road lane that should be flowing in the same direction as vehicle lane
In incoming lane	Agent in road lane that should be flowing in the opposite direction as vehicle lane
In outgoing bus lane *	Agent in the bus lane that should be flowing in the same direction as AV
In incoming bus lane *	Agent in the bus lane that should be flowing in the opposite direction as AV
In outgoing cycle lane	Agent in the cycle lane that should be flowing in the same direction as AV
In incoming cycle lane	Agent in the cycle lane that should be flowing in the opposite direction as AV
On left pavement	Pavement to the left side of AV
On right pavement	Pavement to the right side of AV
On pavement	A pavement that is perpendicular to the movement of the AV
At junction	Road linked
At crossing	A marked section of road for cross, such as zebra or pelican crossing
At bus stop	A marked bus stop area on road, or a section of pavement next to a bus stop sign
At left parking *	A marked parking area on left side of the road
At right parking *	A marked parking area on right side of the road
Parking **	***

Table 2.3: List of ROAD location labels, with description (see [1]).

Note:
* Defined in the paper, but never used in experiments
** Not defined in the paper, but used in experiments
*** Definition not provided.

The location of agents plays a crucial role, as it provides essential information for the AV to determine the appropriate action to take. However, not all agents in the scene have an associated position. Specifically, static objects such as traffic lights are not spatially localised, since they are not movable objects and do not require dynamic tracking. In contrast, for all other agents, at least one position must be defined in order to enable accurate context analysis and support the decision-making process. Possible location labels are shown in Table 2.3

An example frame is shown in Figure 2.15, where the perspective from which the dataset was generated can be observed, as well as the variety of environmental conditions captured. Additionally, the annotations are highlighted in terms of labels: Road Agent, Road Action, Location, Duplex (encoded by the agent-action combination), and Triplet (encoded by the agent-action-location combination). In the conducted experiments, the duplex was not used, whereas both the individual labels and the triplets were employed to encode the information.

(a) **(b)**

Figure 2.15: Two examples of frames sharing the same perspective but exhibiting different lighting conditions: (a) daytime image, (b) nighttime image. The Road Event annotations, as previously described, are highlighted, comparing the ground truth shown in green and the 3D-RetinaNet predictions in red.

2.7.2 AV

In each frame annotation, in addition to the information regarding the agents present in the scene and the actions they perform, the action taken by the autonomous vehicle AV is also specified. This additional annotation is essential for understanding the relationship between the behaviour of external agents and the operational decisions made by the autonomous system.

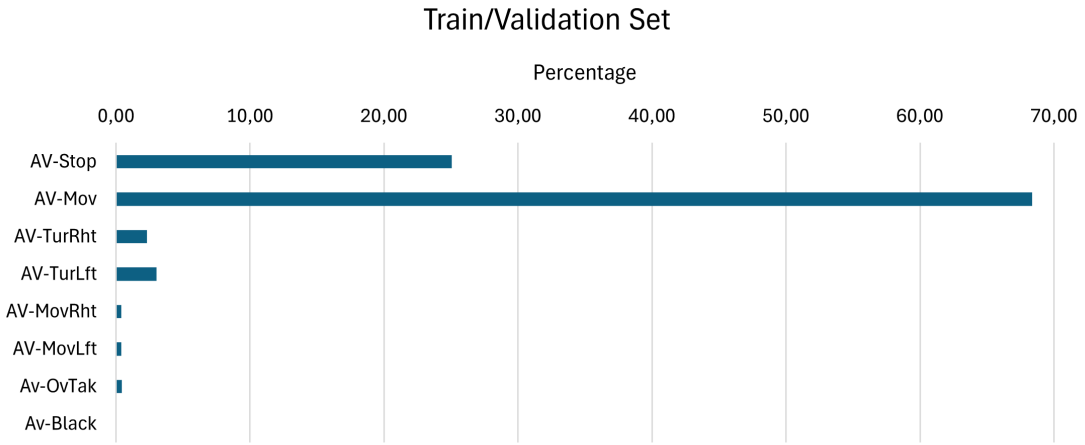
To enable this, a predefined set of possible AV actions is used, which includes maneuvers described in Table 2.4.

Label	**Description**
AV-Stop	AV not moving
AV-Move	AV on the move
AV-Turn-Right	AV turning right
AV-Turn-Left	AV turning left
AV-Overtake	AV overtaking another vehicle
AV-Move-Left	AV moving towards left
AV-Move-Right	AV moving towards right
AV-Black **	***

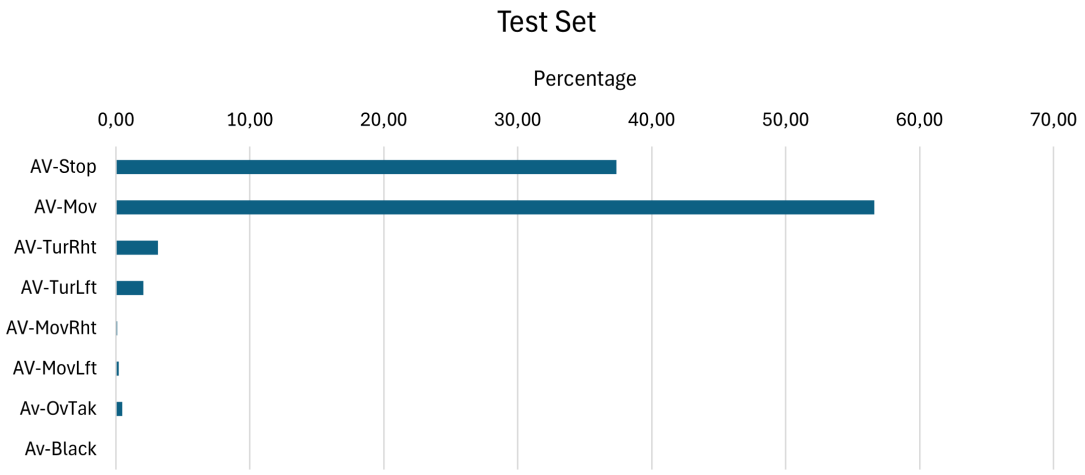
Table 2.4: AV-related action classes (see [1]).

Note:
* Defined in the paper, but never used in experiments
** Not defined in the paper, but used in experiments
*** Definition not provided.

An analysis of the distribution of the autonomous vehicle labels was conducted. The results of this analysis are presented in Figure 2.16, which shows the distributions for the two portions of the dataset. It can be observed that the training/validation set and in the test set exhibit approximately similar distributions, with both showing a peak for the AV-Mov action. However, the graph also highlights that the labels are not evenly distributed, confirming the presence of a highly imbalanced dataset.



(a) Distribution of the AV labels in training/validation set.



(b) Distribution of the AV labels in test set.

Figure 2.16: Distribution of the AV labels in training/validation set (a) and in test set (b). The graphs show the distribution of the labels as percentages.

Chapter 3

Methodology

This chapter provides a detailed description of the methodologies adopted in the development of this thesis work. Approaches employed to determine the action of the autonomous vehicle (AV) are presented, both from symbolic concepts and directly from images. Finally, the explainability layer integrated into the system is introduced, aiming to create an interpretable connection between the different stages: from visual processing to concept prediction, and ultimately to the AV's action decision.

3.1 From Concept To AV-Action

The first part of the project focused on training models capable of predicting autonomous vehicle actions starting from concepts. The main goal of this approach is to enhance the transparency and interpretability of the decision-making process, as opposed to black-box models, which predict the final action directly from the input image without explicitly revealing the factors that led to the decision. The adoption of a concept-based approach falls within the broader field of C-XAI, with the aim of developing models that are not only predictive but also interpretable and transparent.

In particular, transitioning from raw data (images) to a symbolic representation (Road Event) allows for increased traceability of the decision-making process, making it easier to understand which elements contributed to the selection of the final action. This approach thus provides a more causal explanation of the model's decisions: the predicted action is not just a numerical value, but the result of a logical combination of interpretable and observable factors within the scene. For example, the decision to brake may be explained by the presence of a pedestrian crossing the road close to the vehicle or the possibility to move forward when facing a green traffic light. However, it is important to note that a concept-based approach

also presents some challenges, such as the need for high-quality annotations and the potential loss of information compared to end-to-end models, which may exploit visual content from images more directly.

To implement this transparency-enhancing phase, a Multilayer Perceptron (MLP) and a Transformer were used. It is important to highlight that these models did not operate directly on images, but rather on structured annotations, which represent the key concepts extracted from the environment. These concepts can be interpreted as a form of symbolic representation, serving as an intermediate layer between the perceptual input (the image) and the decision output (the vehicle's action). Their purpose is to capture the most relevant elements of the scene, in a manner akin to how a human observer would perceive them (concepts). Specifically, the annotations were organized according to the Road Event schema, divided into three main categories: Road Agent, Road Action, Location. This information was encoded to be processed by indicated models, serving as input for both the training and prediction phases. The possible actions that the vehicle could perform, previously described (see subsection 2.7.2), represent the target labels of the system.

Three distinct approaches for input encoding were explored:

- **Triplet Encoding**: directly uses the annotations encoded as triplets (agent-action-location).
- **Label Encoding**: separates the information, encoding the presence of each concept (agent, action, location) individually, while maintaining the relationships between them.
- **Combined Encoding**: a hybrid version that concatenates the triplet encoding and the single-label encodings, aiming to integrate both representations.

By observing Figure 2.15a, it is possible to understand how this information is represented using different encoding methods. For example, when analysing the vehicle in front of the ego-vehicle, it can be observed that the **Triplet encoding** represents the information *Car-MoveAway-VehLane* with a single numerical value, which encapsulates the entire concept. In contrast, in the **Label Encoding**, the information is divided into three separate values, each corresponding to the agent, the action, and the position, respectively. Instead, in the **Combined Encoding**, the information results from the concatenation of the Triplet and Label encodings, thus consisting of four numerical values.

Finally, for the MLP model, a more detailed analysis was conducted through two encoding variants:

1. Binary approach, recording only the presence or absence of a given concept.
2. Quantitative approach, recording the number of occurrences of each concept within the frame, enabling the model to leverage quantitative information.

3.2 Baseline models

Two black-box models were selected as baselines: **ResNet-50** and **R2Plus1D_18**, which are neural networks that, predict the action to be taken by the ego vehicle based on a perceptual input (either an image or a sequence of images). The use of these models, along with the adopted approaches, is described in more detail in the following sections. Both models were tested to evaluate different configurations and to analyze how performance could vary depending on the adopted strategy.

3.2.1 ResNet-50

As described in section 2.4, ResNet-50 takes a single image as input and predicts, from a set of predefined actions, the one that the autonomous vehicle (AV) should perform. Based on the metrics obtained in this baseline scenario for ResNet-50, several techniques were explored with the aim of improving performance. In particular, two approaches were adopted: **Data Augmentation** and **Oversampling**.

Data Augmentation

Data augmentation is a widely used technique in the field of Computer Vision to improve the performance of machine learning models, especially when available datasets are limited in size or poorly representative [26]. The main goal of this technique is to increase the volume, quality, and diversity of training data by applying transformations that preserve the semantic meaning of the original images. In this way, the model can learn more generalizable representations, reducing the risk of overfitting and enhancing system robustness. This approach is particularly effective as it introduces synthetic variations into the training data, simulating the presence of new examples while maintaining label consistency.

In the context of this project, transformations operating at the input space level were applied, including geometric operations (e.g., rotations, translations, distortions) and photometric adjustments (e.g., changes in colour, contrast, brightness). In the experiments described in this thesis, the following transformations were employed:

- **RandomHorizontalFlip**: applies a horizontal flip to the image with a given probability (applied with $p = 0.5$). This transformation simulates mirrored scenarios, making the model more robust to the orientation of road elements.

- **RandomRotation**: applies a random rotation to the image within a specified angle range. This transformation enhances the model's tolerance to variations in object or scene orientation, simulating scenarios where the camera may

have been slightly tilted or displaced. Inside the project, maximum value is 10°

- **RandomAffine**: applies a random affine transformation that combines multiple geometric variations: rotation, translation, scaling, and shear in a single operation. Rotation is applied by selecting an angle within a predefined range, translation shifts the image by a certain percentage relative to its original dimensions, scaling adjusts the image size proportionally within a specified range, and shear alters the shape of the image through an angular distortion.

- **RandomGrayscale**: randomly converts a colour image to grayscale with a predefined probability (applied with $p = 0.1$). The main purpose of this transformation is to increase the model's robustness to colour variations, allowing it to focus more on the structural and textural features of the image rather than on colour information.

- **Resize**: is a transformation that modifies the spatial dimensions of an image by rescaling it to a specific resolution. It is applied to ensure that all input images have the same size, which is a necessary condition for most neural network architectures. Resizing may lead to changes in the original aspect ratio of the image.

- **ColorJitter**: is a transformation that randomly alters the colour properties of an image, including brightness, contrast, saturation, and hue. This variation simulates the appearance of the image under different lighting and environmental conditions, helping improve the model's robustness to non-structural visual changes.

- **GaussianBlur**: is a transformation that applies a blur to the image using a Gaussian filter, reducing high-frequency details and making the image appear visually softer. This operation simulates suboptimal visual conditions, such as slight out-of-focus effects.

- **Normalize**: is a transformation that normalizes the pixel values of an image channel by channel, by subtracting the mean and dividing by the standard deviation of each channel. This operation centers the data distribution around zero and reduces its variance.

The described data augmentation transformations were carefully configured to fulfill the purpose of increasing the dataset's variety, thereby contributing to the improvement of the model's generalization capability. In particular, they enabled the generation of diverse images, making the network more robust to visual variations not present in the original dataset. Moreover, some transformations

were designed to simulate possible distortions caused by real issues in acquisition sensors, such as changes in camera angle or loss of quality in the captured image.

The transformations were applied in two distinct modes, each characterized by a different level of intensity. The first mode, referred to as **soft**, involves the use of mild transformations with moderate parameters that introduce limited variations compared to the original image. The second mode, called **strong**, employs more aggressive transformations with more pronounced parameters, capable of generating significantly altered images. This distinction allowed for evaluating the model' s performance under progressively more challenging data augmentation conditions, simulating more complex usage scenarios.

Table 3.1 and Table 3.2 show the transformations and parameters used for the data augmentation configurations referred to as soft and strong, respectively.

Augmentation	Parameters
Resize	224×224
RandomHorizontalFlip	0.5
ColorJitter	$0.5, 0.2, 0.2, 0.1$
RandomRotation	10
RandomAffine	$degress = 0, translate = (0.05, 0.05), scale = (0.95, 1.05)$
Normalize	$mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]$

Table 3.1: List of data augmentation transformations and their parameters for the **soft** configuration.

Augmentation	Parameters
Resize	224×224
RandomHorizonalFlip	0.5
RandomGrayScale	0.1
ColorJitter	$0.5, 0.3, 0.3, 0.15$
GaussianBlur	$kernel\_size = (3, 3), sigma = (0.1, 2.0)$
RandomAffine	$degress = 15, translate = (0.1, 0.1), scale = (0.9, 1.1),$ $shear = 5$
Normalize	$mean = [0.485, 0.456, 0.406], std = [0.229, 0.224, 0.225]$

Table 3.2: List of data augmentation transformations and their parameters for the **strong** configuration.

Oversampling

Oversampling is a widely used technique to enhance the performance of machine learning models, particularly neural networks, when dealing with imbalanced datasets [27]. A dataset is considered imbalanced when the distribution of labels across classes is not uniform, meaning that some classes are significantly underrepresented compared to others. This imbalance can hinder model performance, as the learning process tends to favour majority classes, thereby limiting the ability to correctly learn the features of minority classes.

To address this issue, random oversampling was employed in this project. This technique involves randomly replicating samples from the minority classes. By doing so, the training dataset is rebalanced, mitigating the bias introduced by the original imbalance. As illustrated in Figure 2.16, some original classes occur with very low frequency; the oversampling process increases their representation, enhancing the model's capacity to generalize to underrepresented categories. However, the repeated inclusion of identical samples may result in an increased risk of overfitting, which could negatively impact generalization on unseen data.

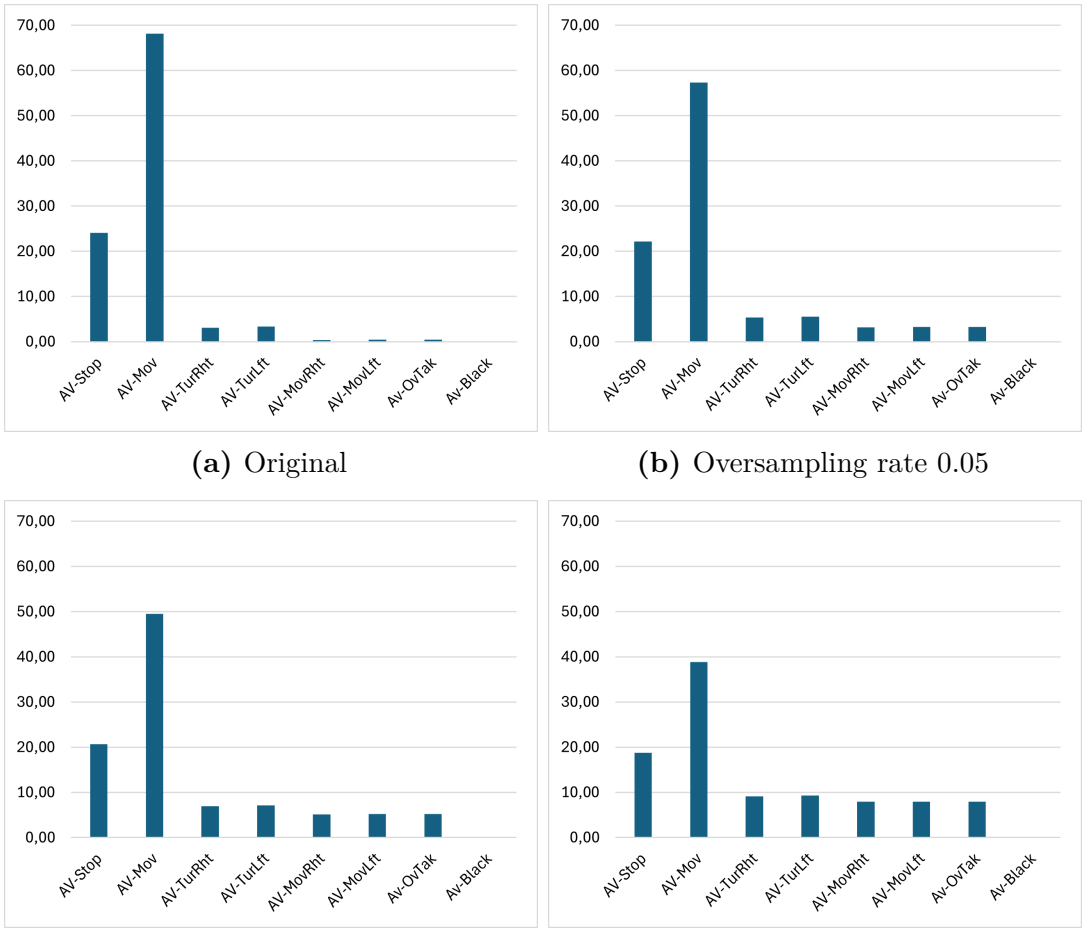
To evaluate the effectiveness of this approach, three different values of the oversampling factor were tested. This allowed for an analysis of how different degrees of artificial balancing influence the neural network's performance, and whether progressively increasing the number of replicated samples leads to significant improvements in classification accuracy.

Figure 3.1 clearly illustrates how the application of oversampling changes the class distribution within the training dataset. Specifically, it can be observed that the proportion of minority classes increases, while the majority class proportion decreases. For instance, the *AV-Move* label decreases from approximately 70% (Figure 3.1a) to around 40% (Figure 3.1d), resulting in a more balanced overall distribution.

3.2.2 R2Plus1D_18

The R(2+1)D_18 model was adopted to evaluate how a neural network behaves when the input consists of an image sequence rather than individual frames. Unlike approaches that rely on processing a single frame to make a prediction, this architecture processes an entire temporal sequence and predicts the action of the autonomous vehicle (AV) corresponding to the target frame, i.e., the last frame in the sequence. This allows the model to capture the temporal evolution of the scene and exploit the continuity between consecutive frames.

No specific experimental variations were applied to this model; rather, the main objective was to observe how the model's performance changes when temporal information is explicitly included in the learning process.



(a) Original

(b) Oversampling rate 0.05

(c) Oversampling rate 0.1

(d) Oversampling rate 0.2

Figure 3.1: Distribution of AV labels under different oversampling rates. The figure shows the original label distribution (a) and the effect of applying three different oversampling rates: 0.05 (b), 0.1(c), and 0.2(d) to the minority classes.

3.3 Concept Bottleneck Model for AV Action Prediction

In the final part of this work, the task to develop an explainable model is addressed using a **Concept Bottleneck Model** (**CBM**), previously described in section 2.3. As discussed in section 2.6, 3D-RetinaNet is a model that processes a sequence of frames to produce predictions not only related to the AV's action but also concerning symbolic concepts describing the driving scene. It also identifies symbolic concepts, referred to as Road Events (as introduced in subsection 2.7.1), along with their corresponding bounding boxes within the image.

For this reason, 3D-RetinaNet was used as the first stage of a cascade architecture, to which models such as MLP or Transformer were subsequently added. These were applied to predict the action taken by the ego-vehicle, specifically following the procedure described in section 3.1. In practice, 3D-RetinaNet is responsible for extracting a structured conceptual representation from the visual input, which is then passed to the second module in charge of action prediction. To formalize what has been described and to better align it with the functioning of the CBM, the prediction process can be represented as a sequence of operations $x \to c \to y$, which can be further divided into two distinct stages: $x \to c$ and $c \to y$. A more detailed analysis yields:

- $x \to c$: starting from an input image (x), symbolic concepts (c)—namely the Road Events—are predicted using 3D-RetinaNet, which was employed as a pretrained network on the dataset used;
- $c \to y$: based on the predicted symbolic concepts, the target action (y) is estimated using two alternative models, MLP and Transformer, both trained as part of this project.

Since the $c \to y$ prediction can be performed using two different models, it is convenient to introduce the following notation:

- **CBM-MLP**: where the $c \to y$ prediction stage is implemented using a Multilayer Perceptron (MLP). Thanks to the modularity of this approach, it is possible to select the most suitable encoding strategy, as well as to choose between a qualitative or quantitative approach, depending on the chosen MLP model.
- **CBM-Transformer**: where the $c \to y$ prediction is performed using a Transformer-based model. Similarly, in this case, the modularity allows for the selection of the preferred encoding scheme and the definition of the number of Transformer layers.

Compared to the other models explored in this work, this architecture introduces an additional level of explainability, thanks to the explicit representation of relevant scene concepts, so called Road Event. These concepts correspond to key elements that, in a real-world scenario, a human driver would consider essential in deciding which action to take. Therefore, this approach not only enables action prediction, but also provides an interpretable representation of the reasoning behind the decision.

Chapter 4

Experimental Results

This chapter presents the experimental results obtained throughout the project. It begins with a description of the hardware setup used to run the experiments, along with the main configuration parameters. Then, the evaluation metrics adopted for model assessment are introduced. The presentation of results follows a logical progression: first, models that predict the AV's action based on symbolic concepts are evaluated; next, the performance of black-box models is analysed; finally, the explainability-driven approach is discussed, involving a CBM that performs two-stage inference, first predicting the intermediate concepts and then the final action.

4.1 Parameters And Configuration

All experiments were conducted on a notebook equipped with 16 GB of RAM and an NVIDIA GeForce RTX 3070 Ti Laptop GPU with 8 GB of dedicated memory. Regarding the training parameters, most configurations were trained for 10 epochs, although in some cases the number of epochs was increased up to 30 to handle slower convergence scenarios. The batch size was set to 512 for models that did not involve image processing, while it was reduced to 32 or 16 for models working with visual data, depending on model complexity and memory requirements.

Some parameters remained constant across all experiments: the optimizer was set to Adam, the learning rate was fixed at 0.001, and the dataset was split into 80% for training and 20% for validation, as the test set was already provided separately by the dataset creators.

Additionally, for each experiment, different seeds $(0, 50, 99)$ were used when generating the training split. This choice allowed the evaluation of the model's robustness with respect to different random initializations of the data partitioning.

These values provide a general overview of the experimental configurations, while

the specific setups for each experiment will be detailed in the following sections.

4.2 Metrics

To evaluate the performance of the models, particularly their ability to accurately predict both the ego-vehicle' s action and the intermediate symbolic concepts (REs), two main evaluation metrics were adopted: **Accuracy** and **F1-score**. For clarity, these two metrics will be shown in the tables using the abbreviations Acc. (Accuracy) and F1 (F1-Score).

4.2.1 Accuracy

Accuracy is a metric that describes how the model performs across all classes. It represents the percentage of correct predictions out of the total number of samples. Accuracy is defined by the following formula:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{4.1}$$

where: **TP** is *True Positive*, **TN** is *True Negative*, **FP** is *False Positive*, and **FN** is *False Negative*

4.2.2 F1 Score

The F1-score is the harmonic mean of precision and recall, and it combines these two metrics into a single value to provide a balanced measure of classification quality. F1-score is defined by the following formula:

$$F1 = 2 \times \frac{Precision \times Recall}{Presicion + Recall} \tag{4.2}$$

where:

$$Precision = \frac{TP}{TP + FP} \quad Recall = \frac{TP}{TP + FN} \tag{4.3}$$

4.3 Multilayer Perceptron Evaluation

To evaluate the performance of the MLP model, two fundamental metrics were considered: accuracy and F1-score. These metrics were chosen because they are directly related to the model's objective—namely, predicting the action to be taken by the AV based on symbolic concepts (Road Events).

In order to ensure the reproducibility of the experiments, evaluations were carried out using different seed values. Additionally, for each input encoding strategy, a further analysis was conducted, as previously described: in the first case, only the **presence or absence** of concepts was considered, while in the second configuration, the **frequency** with which each concept appeared in the input was also taken into account. Metrics were computed on both the validation set (Table 4.1) and the test set (Table 4.2), allowing for a comparison of the model's performance on previously seen data and entirely unseen data.

Analysing the results reported in Table 4.1, it can be observed that the model exhibits good stability with respect to seed variation. The accuracy and F1-score metrics show only minor fluctuations across different experiments, suggesting that the training process is robust. Regarding the impact of the track occurrences encoding, it appears that including this information does not lead to significant changes in terms of accuracy, with an average increase that does not exceed 2%. However, the improvement is more noticeable for the F1-score, which increases by up to approximately 6%, indicating a better balance between precision and recall in predicting the correct action. Among the different encoding strategies, the **Combined** approach emerges as the best performing one. Specifically, in the configuration that includes occurrence tracking with $seed = 99$, the model achieves its highest overall performance, reaching an accuracy of 0.9264 and an F1-score of 0.7224. This suggests that combining multiple symbolic sources, enriched with frequency information, provides the model with a more comprehensive and discriminative representation that enhances its classification ability.

Approach	Track Occ.	Seed: 0		Seed: 50		Seed: 99		σ	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Triplet	✗	0.9124	0.6542	0.9096	0.6776	0.9143	0.6831	0.0024	0.0154
	✓	0.9168	0.6943	0.9143	0.6931	0.9186	0.7016	0.0022	0.0046
Label	✗	0.9060	0.6295	0.9057	0.6237	0.9080	0.6403	0.0013	0.0084
	✓	0.9209	0.6841	0.9172	0.6813	0.9191	0.6943	0.0019	0.0068
Combined	✗	0.9185	0.6719	0.9178	0.6683	0.9196	0.6750	0.0009	0.0034
	✓	0.9232	0.6970	0.9223	0.7087	**0.9264**	**0.7224**	0.0022	0.0127

Table 4.1: Accuracy and F1-score results, including the corresponding standard deviations, for the MLP model evaluated on the validation set.

Table 4.2 reports the results obtained by the MLP model on the test set. A significant drop in performance is immediately noticeable compared to the validation set, with a particularly sharp decline in F1-score values. When analysing the effect of varying the seed, the metrics remain relatively stable, indicating that the model is robust to randomness in the data split. Unlike what was observed on the validation set, the application of the track occurrences technique does not

lead to significant variations in performance: both accuracy and F1-score show only minimal differences. On the test set, the best-performing encoding is **Label** encoding, particularly in the configuration with $seed = 0$ and without tracking occurrences, where an accuracy of 0.7232 and an F1-score of 0.2394 are achieved.

Approach	Track Occ.	Seed: 0		Seed: 50		Seed: 99		σ	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Triplet	✗	0.5985	0.1727	0.5901	0.1659	0.5994	0.2030	0.0051	0.0199
	✓	0.5872	0.1668	0.6001	0.1636	0.5916	0.1668	0.0066	0.0019
Label	✗	**0.7232**	**0.2394**	0.6997	0.2262	0.7221	0.2327	0.0133	0.0066
	✓	0.7200	0.2344	0.6890	0.2237	0.6462	0.2029	0.0371	0.0160
Combined	✗	0.6694	0.2123	0.6514	0.2018	0.6563	0.2043	0.0093	0.0055
	✓	0.6676	0.2106	0.6560	0.2066	0.6473	0.2017	0.0102	0.0045

Table 4.2: Accuracy and F1-score results, including the corresponding standard deviations, for the MLP model evaluated on the test set.

4.4 Transformer Evaluation

The Transformer model was employed with the goal of predicting the autonomous vehicle's (AV) action based on a conceptual representation of the scene, namely the Road Events. The training process was carried out by providing the model with symbolic concepts as input, with the aim of generating the vehicle's corresponding action as output.

To ensure the reproducibility of the experiments and to assess the model's robustness to dataset splitting, training was conducted using different seed values. Furthermore, various levels of model complexity were explored by changing the number of Transformer layers: specifically, configurations with 1, 2, and 3 layers were tested.

It is important to emphasize that, in order to achieve better performance — as will be discussed later — the model with 3 layers was trained for 30 epochs, whereas the configurations with 1 or 2 layers were trained for only 10 epochs. As with the MLP model, performance evaluation was carried out on both the validation set (Table 4.3) and the test set (Table 4.4), to analyse the model's ability to generalize to unseen data.

Looking at the results reported in Table 4.3, which refer to performance on the validation set (expressed in terms of accuracy and F1-score), the model demonstrates a good level of stability across different seeds. The metrics vary only slightly between runs, suggesting that the model's generalization capability is not strongly affected by the specific training/validation split. When analysing the impact of the number of layers, different trends emerge depending on the type of input encoding. Specifically,

for the *Triplet* and *Combined* encodings, increasing the number of layers from 1 to 3 results in a noticeable performance degradation. Accuracy drops by approximately 4%, while the F1-score drops declines substantially—by up to 35% in the case of the Triplet encoding, where the F1-score falls from 0.6322 to 0.2740. This drop occurs despite models with 3 layers being trained for more epochs, suggesting that increased model depth does not always lead to better performance and may, in some cases, hinder the model's ability to fit even familiar data. In contrast, the **Label** encoding approach shows a consistent improvement as the number of layers increases. Accuracy rises from 0.9134 to 0.9261 instead F1-score from 0.6712 to 0.7259. This corresponds to approximately a 1% increase in accuracy and an almost 10% increase in F1-score from 1 to 3 layers. These results suggest that, for this symbolic representation, the model is able to effectively exploit the added architectural complexity to improve its discriminative capabilities. However, it is worth noting that this improvement comes at the cost of higher computational effort: models with 3 layers were trained for 30 epochs, compared to only 10 for the other configurations.

Approach	Num Layers	Seed: 0		Seed: 50		Seed: 99		σ	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Triplet	1	0.9120	0.6322	0.9100	0.6775	0.9119	0.6788	0.0011	0.0265
	2	0.9081	0.6486	0.9024	0.6169	0.9050	0.6122	0.0029	0.0198
	3	0.8669	0.2740	0.8585	0.2908	0.8716	0.2772	0.0066	0.0089
Label	1	0.8931	0.6255	0.9036	0.6265	0.9134	0.6712	0.0102	0.0261
	2	0.9159	0.6709	0.9140	0.6785	0.9197	0.6766	0.0029	0.0040
	3	0.9217	0.6883	0.9232	0.7230	**0.9261**	**0.7259**	0.0022	0.0210
Combined	1	0.9150	0.6400	0.9109	0.6671	0.9170	0.6657	0.0031	0.0153
	2	0.9022	0.4452	0.8941	0.5112	0.8980	0.4288	0.0041	0.0436
	3	0.8767	0.3093	0.8650	0.3040	0.8737	0.3043	0.0061	0.0030

Table 4.3: Accuracy and F1-score results, including the corresponding standard deviations, for the Transformer model evaluated on the validation set.

Looking at the results presented in Table 4.4, which report the metrics obtained on the test set—i.e., data never seen by the model during training and therefore more representative of a real-world scenario— it is possible to carry out an analysis similar to the one performed for the validation set.

To begin with, a good level of stability can be observed across different seeds: in most cases, the metrics vary by only a few percentage points. A notable exception is the *Label* encoding approach with 2 layers, where accuracy fluctuates by up to 4%, and the F1-score shows a more significant variation of up to around 12%, ranging from 0.2213 to 0.3442. Focusing instead on the impact of the number of layers, increasing the model depth does not result in substantial performance variations. Overall, the metrics tend to remain stable or exhibit slight improvements. Once

again, an exception can be observed: for $seed = 0$, the Label encoding approach with 2 layers achieves a notably higher F1-score compared to the other two seeds, standing out as an outlier with respect to the general trend. This configuration reaches an accuracy of 0.7072 and an F1-score of 0.3442.

Overall, even on the test set, the **Label** encoding approach confirms itself as the most effective, achieving the highest accuracy and F1-score values compared to the other two representation strategies. Conversely, the Triplet encoding consistently proves to be the least effective.

Approach	Num Layers	Seed: 0		Seed: 50		Seed: 99		σ	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Triplet	1	0.5916	0.1815	0.6088	0.1804	0.6245	0.2208	0.0165	0.0230
	2	0.5755	0.1752	0.5997	0.1710	0.6167	0.1792	0.0207	0.0041
	3	0.6441	0.1914	0.6095	0.1731	0.6478	0.1705	0.0211	0.0114
Label	1	0.6437	0.2311	0.6544	0.2151	0.6950	0.2656	0.0271	0.0258
	2	**0.7072**	**0.3442**	0.6775	0.2213	0.7160	0.2301	0.0202	0.0686
	3	0.7057	0.2565	0.7301	0.2407	0.7041	0.2344	0.0146	0.0114
Combined	1	0.6532	0.2067	0.6576	0.2132	0.6854	0.2212	0.0175	0.0073
	2	0.6493	0.2131	0.6538	0.2384	0.6933	0.2380	0.0242	0.0145
	3	0.7113	0.2220	0.6258	0.1615	0.6744	0.1854	0.0429	0.0305

Table 4.4: Accuracy and F1-score results, including the corresponding standard deviations, for the Transformer model evaluated on the test set.

4.5 ResNet-50 Baseline Evaluation

ResNet-50 is the first architecture considered in this work that uses raw images as input, without relying on symbolic annotations or explicit concepts. The goal is to predict the action that the autonomous vehicle (AV) should take, addressing the problem in an end-to-end manner and operating as a black box. This section presents the metrics and results obtained by the ResNet-50 model on both the validation and test datasets, in order to evaluate its performance in predicting vehicle actions. Initially, the results obtained using the plain network are reported, followed by those achieved after applying data augmentation techniques. Finally, the performance is shown when both oversampling and data augmentation are applied.

Table 4.5 reports the accuracy and F1-score results obtained on the validation set for the ResNet-50 model, comparing the baseline configuration with two variants employing data augmentation techniques (soft and strong). It can be observed that, for each configuration, the results vary only slightly depending on the initialization seed, indicating a certain robustness to this parameter. However, a notable exception is the Aug. Strong configuration with seed 50, where the F1-score drops from

0.9115 (seed 0) to 0.8594, marking a decrease of approximately 6%. Overall, it is evident that introducing data augmentation leads to a degradation in performance, particularly in terms of F1-score. While the accuracy remains relatively stable across the different settings, the F1-score shows a consistent downward trend, with the worst-case scenario (Aug. Strong, seed 50) showing a reduction of over 11% compared to the baseline. These results suggest that, on the validation set, the application of the considered data augmentation techniques did not yield any improvements and instead had a negative impact on classification performance.

Method	Seed: 0		Seed: 50		Seed: 99		σ	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Base	**0.9913**	**0.9769**	0.9906	0.9714	0.9903	0.9622	0.0005	0.0074
Aug. Soft	0.9773	0.9370	0.9748	0.9255	0.9773	0.9433	0.0014	0.0090
Aug. Strong	0.9715	0.9115	0.9669	0.8594	0.9731	0.9380	0.0032	0.0400

Table 4.5: Accuracy and F1-score results, including the corresponding standard deviations, for the ResNet-50 model (plain and with data augmentation) on the validation set.

Table 4.6 shows the results obtained on the test set for the same configurations previously analysed. As anticipated, the metrics show a general decrease, which is expected given that the test data were never seen during training. When analysing the variability introduced by different random seeds, the results remain mostly consistent between seed 0 and seed 99, with the exception of the Aug. Strong configuration. Seed 50, however, stands out, particularly in terms of F1-score: in the Base configuration, it leads to a significant performance drop of about 4% compared to seed 0, while in the Aug. Soft configuration it results in an increase of approximately 3.5%. Interestingly, contrary to what was observed on the validation set—where data augmentation generally led to lower performance—on the test set, the application of augmentation techniques either preserved or improved the F1-score. The best result is achieved with the **Aug. Soft** configuration and seed 50, reaching a peak F1-score of 0.3126. These findings suggest that data augmentation helped improve the model's ability to generalize to unseen data.

Building on the previous results, an oversampling technique was applied with the aim of increasing the representation of samples belonging to underrepresented classes. This approach was intended to assess whether a higher presence of such examples in the dataset could help the model learn and generalize more effectively. Three main threshold values for oversampling were considered: 0.2, 0.1, and 0.05. As these values decrease, the degree of duplication for underrepresented classes is reduced. Thresholds higher than 0.2 were avoided, as they would have led to excessive replication of the same samples, increasing the risk of overfitting. In

Method	Seed: 0		Seed: 50		Seed: 99		σ	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Base	0.6036	0.2599	0.5890	0.2189	0.5929	0.2577	0.0076	0.0231
Aug. Soft	0.6406	0.2869	**0.6668**	**0.3126**	0.6165	0.2768	0.0252	0.0185
Aug. Strong	0.6695	0.3009	0.6247	0.2681	0.6070	0.2583	0.0322	0.0223

Table 4.6: Accuracy and F1-score results, including the corresponding standard deviations, for the ResNet-50 model (plain and with data augmentation) on the test set.

addition to oversampling, the data augmentation techniques previously described were also employed, resulting in three different experimental configurations:

1. **Ovr.**: oversampling only;
2. **Ovr. + Aug. Soft**: oversampling combined with soft augmentation;
3. **Ovr. + Aug. Strong**: oversampling combined with strong augmentation.

The results of the different experimental configurations described above are presented in Table 4.7 and Table 4.8, corresponding to the validation set and the test set, respectively.

A detailed analysis of Table 4.7 (validation set), shows that performance metrics are consistently high across all configurations: accuracy ranges from 97% to 99%, while the F1-score varies between 94% and 98%. These values indicate that no configuration significantly outperforms or underperforms the others, and that all combinations of oversampling and data augmentation enable the model to achieve strong and reliable results. What stands out in particular is the comparison with the results obtained without oversampling (see Table 4.5): while accuracy remains largely unchanged, the F1-score shows an improvement of approximately 2–3%. This suggests that oversampling can help improve the balance between precision and recall, especially for underrepresented classes.

Table 4.8 reports the results obtained on the test set. Also in this case, the performance metrics appear to be generally stable, although with slightly more noticeable fluctuations compared to those observed on the validation set (Table 4.7). No significant differences emerge among the various configurations, with the exception of the Ovr. configuration with $ovr.ratio = 0.05$ and seed 50, which shows improved performance compared to the other two seeds in the same category. In contrast, the configurations that combine oversampling and data augmentation seem to contribute to greater model robustness. The best performance is observed in the configuration with **Ovr. + Aug. Strong**, Ovr. Ratio 0.1, and seed 99, which yields better results in terms of both accuracy and F1-score. A

Method	Ovr. Ratio	Seed: 0 Acc.	Seed: 0 F1	Seed: 50 Acc.	Seed: 50 F1	Seed: 99 Acc.	Seed: 99 F1	σ Acc.	σ F1
Ovr.	0.2	0.9943	0.9819	0.9924	0.9727	**0.9924**	**0.9834**	0.0011	0.0058
	0.1	0.9926	0.9807	0.9920	0.9770	0.9934	0.9793	0.0007	0.0019
	0.05	0.9927	0.9711	0.9919	0.9800	0.9930	0.9823	0.0006	0.0060
Ovr. + Aug. Soft	0.2	0.9806	0.9573	0.9788	0.9556	0.9801	0.9542	0.0009	0.0016
	0.1	0.9770	0.9437	0.9750	0.9539	0.9793	0.9612	0.0022	0.0088
	0.05	0.9768	0.9538	0.9749	0.9336	0.9784	0.9564	0.0018	0.0125
Ovr + Aug. Strong	0.2	0.9765	0.9463	0.9809	0.9561	0.9774	0.9590	0.0023	0.0067
	0.1	0.9791	0.9561	0.9749	0.9481	0.9791	0.9560	0.0024	0.0046
	0.05	0.9797	0.9534	0.9750	0.9440	0.9761	0.9535	0.0025	0.0055

Table 4.7: Accuracy and F1-score results, including the corresponding standard deviations, for the ResNet-50 model on the validation set, obtained by applying oversampling and data augmentation techniques.

comparison with the results obtained without oversampling (Table 4.6) does not reveal particularly substantial improvements on the test set. However, the use of oversampling—especially when combined with data augmentation—has helped to enhance or stabilize the model's performance overall, even if it has not led to drastic changes.

Method	Ovr. Ratio	Seed: 0 Acc.	Seed: 0 F1	Seed: 50 Acc.	Seed: 50 F1	Seed: 99 Acc.	Seed: 99 F1	σ Acc.	σ F1
Ovr.	0.2	0.5954	0.2628	0.5928	0.2713	0.5928	0.2680	0.0015	0.0043
	0.1	0.5909	0.2481	0.5943	0.2631	0.5955	0.2613	0.0024	0.0082
	0.05	0.5921	0.2621	0.6284	0.3049	0.5887	0.2508	0.0220	0.0285
Ovr. + Aug. Soft	0.2	0.6047	0.2640	0.6618	0.2987	0.6207	0.2931	0.0295	0.0187
	0.1	0.6296	0.2877	0.6461	0.2863	0.6140	0.2737	0.0161	0.0077
	0.05	0.6465	0.2895	0.6250	0.2950	0.6315	0.2764	0.0110	0.0096
Ovr + Aug. Strong	0.2	0.6253	0.2900	0.6161	0.2640	0.5920	0.2492	0.0172	0.0207
	0.1	0.6040	0.2563	0.6445	0.2931	**0.6741**	**0.3158**	0.0352	0.0300
	0.05	0.6639	0.2985	0.6167	0.2701	0.6474	0.3026	0.0240	0.0177

Table 4.8: Accuracy and F1-score results, including the corresponding standard deviations, for the ResNet-50 model on the test set, obtained by applying oversampling and data augmentation techniques.

4.6 R2Plus1D_18 Baseline Evaluation

The R2Plus1D_18 network was employed with the same goal as ResNet-50, namely to predict the action of the AV based on visual input. Unlike ResNet-50, which relies on a single image, R2Plus1D_18 processes a sequence of frames, aiming

to capture the temporal dynamics of the scene and better understand how the situation is evolving over time.

In this project, due to hardware limitations, it was not feasible to perform extensive experimentation. Therefore, a baseline configuration was adopted to compare the performance of this model with that of ResNet-50, and to assess whether the use of multiple input frames could effectively enhance the model's generalization capability. Experiments were conducted using the same random seeds adopted in previous tests, with a batch size of 16 and an input sequence of 8 frames. Specifically, in order to predict the action associated with a given target frame, the model also considered the seven preceding frames, thus capturing the evolution of the action over time. Since no additional experimental configurations were explored, the results for both the validation and test sets are reported in a single Table 4.9.

It can be observed that the metrics vary slightly with different seeds, particularly for seed 99, which shows the most significant deviation—mainly on the test set rather than on the validation set. Looking more closely, the validation set shows consistently high performance in terms of both accuracy and F1-score. On the test set, accuracy remains around 0.9378, while the F1-score reaches its highest value of 0.5360 with seed 99.

Comparing these test results with those obtained using the previous black-box model, ResNet-50 (Table 4.6 and Table 4.8), a significant improvement can be noted: accuracy increases by approximately 26%, while F1-score improves by 22%. These findings suggest that leveraging sequences of frames to predict the action at the target frame allows for a more accurate understanding of the driving context. Thus, temporal information emerges as a key factor in enhancing performance in autonomous driving tasks.

Method	Seed: 0		Seed: 50		Seed: 99		σ	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Validation set								
Base	0.9919	0.9603	0.9907	0.9700	**0.9935**	**0.9763**	0.0014	0.0081
Test set								
Base	0.9378	0.4746	0.9340	0.4635	**0.9406**	**0.5360**	0.0033	0.0391

Table 4.9: Performance of the R2Plus1D_18 model in terms of Accuracy and F1-score, including the corresponding standard deviations, reported on both the validation and test sets.

4.7 Concept Bottleneck Model Evaluation

3D-RetinaNet was the model used to introduce a level of explainability into the project, as it was employed to predict the Road Events present in the scene. These events represent key elements that an experienced driver considers when assessing the situation and deciding which action to take.

Since only pretrained models were involved at this stage, the results reported here refer exclusively to the test set. Furthermore, as previously discussed, the symbolic input (c) can be encoded in multiple ways: these encodings were also explored in this inference setup, with the output of 3D-RetinaNet being appropriately transformed. Lastly, the different random seeds used during the training of the MLP and Transformer models were also evaluated to assess their impact on final performance.

Table 4.10 presents accuracy and F1-score metrics for the two-stage inference pipeline, in which the $c \rightarrow y$ prediction is carried out using an MLP, so called CBM-MLP. Moreover, as discussed earlier (section 4.3), the use of track occurrences has not been applied in this setting, since, based on the results in Table 4.2, this configuration did not lead to any significant performance improvements.

The first observation concerns the concept prediction $x \rightarrow c$, whose results remain unchanged across different seeds. This is expected, as 3D-RetinaNet is a pretrained model and is not fine-tuned for each seed. It is also worth noting that accuracy is very high in the Triplet and Combined encodings (above 99%), whereas it drops by about 10% in the Label encoding. However, the latter shows a significant increase in F1-score (up to 35% compared to the Triplet), suggesting improved performance in classifying underrepresented classes. In this regard, Label encoding proves to be more effective for Road Event prediction.

As for the $c \rightarrow y$ prediction, the performance is generally stable across different seeds, with the only exception being the Triplet approach with seed 50, where the F1-score increases from 0.1212 to 0.1484. The best overall performance is achieved by the **Label** encoding approach, which reaches an accuracy of 0.6630 and an F1-score of 0.1938 when using seed 99. Comparing these results to those shown in Table 4.2 — where the model receives ground-truth concept labels as input — it is possible observe an average drop in F1-score of about 5%. This decline is expected, as the symbolic input in this case comes from upstream model predictions rather than ground-truth annotations. Nevertheless, the performance degradation is relatively limited, confirming the robustness of the two-stage architecture. Finally, it is worth noting that although the $x \rightarrow c$ metrics are lower for the Triplet and Combined approaches, the final prediction quality $c \rightarrow y$ does not drop significantly. The differences from the best-performing configuration remain within the 3–7% range, indicating a certain level of resilience in the MLP even when processing imperfect symbolic inputs.

Approach	Prediction	Seed: 0		Seed: 50		Seed: 99		σ	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Triplet	$x \rightarrow c$	0.9980	0.2988	0.9980	0.2988	0.9980	0.2988	0.0000	0.0000
	$c \rightarrow y$	0.5874	0.1212	0.6149	0.1484	0.5868	0.1209	0.0161	0.0158
Label	$x \rightarrow c$	0.8949	0.6454	0.8949	0.6454	0.8949	0.6454	0.0000	0.0000
	$c \rightarrow y$	0.6740	0.1935	0.6546	0.1879	**0.6630**	**0.1938**	0.0097	0.0033
Combined	$x \rightarrow c$	0.9952	0.5549	0.9952	0.5549	0.9952	0.5549	0.0000	0.0000
	$c \rightarrow y$	0.6060	0.1673	0.5994	0.1570	0.6102	0.1636	0.0054	0.0052

Table 4.10: Accuracy and F1-score results, including the corresponding standard deviations, for $x \rightarrow c$ and $c \rightarrow y$ predictions using MLP

Table 4.11 reports the same metrics presented in the previous table, with one key difference: the model used for action prediction ($c \rightarrow y$) is a Transformer, so called CBM-Transformer. As described in section 4.4, several architectural configurations of the Transformer were explored. However, the results reported in Table 4.4 show that using two layers yields better performance. Therefore, all experiments in this section were conducted using a Transformer with two layers.

Analyzing the results of the $x \rightarrow c$ prediction, the values are identical to those reported in the previous case. This behaviour was expected, as the prediction of concepts (REs) is independent of the architecture used in the subsequent stage of the model. Therefore, no variations are expected in the $x \rightarrow c$ phase, whereas differences may emerge in the $c \rightarrow y$ phase, where the architecture actually changes. The results shown in Table 4.11 confirm this observation.

Turning to the final prediction $c \rightarrow y$, the variation in seed results in negligible differences (around 2%), suggesting the model is stable across initializations. Performance also remains consistent across different encoding strategies (Triplet, Label, Combined), with F1-scores ranging from 0.1323 to 0.1698. The **Combined** encoding with $seed = 99$ yields the best results. When comparing these results with those from Table 4.4—where the symbolic input consists of ground-truth concepts—there is a performance drop, with the F1-score decreasing by up to 18% in the worst case. However, this best-performing case in Table 4.4 represents an outlier. Taking a broader view, the average drop in performance is around 6%, confirming the robustness of the two-stage architecture even when the concepts are predicted rather than provided as ground truth.

What ultimately matters is not only the intermediate predictions of the concepts, but rather the final action prediction. When comparing the metrics obtained, it is clear that the final prediction using the MLP or Transformer achieves accuracy/F1-score values of 0.6630/0.1938 and 0.6256/0.1698, respectively. This indicates that the MLP was able to generalize slightly better in this setup. When these results are compared with those of a black-box model, specifically ResNet-50, reported

Approach	Prediction	Seed: 0		Seed: 50		Seed: 99		σ	
		Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
Triplet	$x \to c$	0.9980	0.2988	0.9980	0.2988	0.9980	0.2988	0.0000	0.0000
	$c \to y$	0.5941	0.1323	0.5887	0.1255	0.5915	0.1225	0.0027	0.0050
Label	$x \to c$	0.8949	0.6454	0.8949	0.6454	0.8949	0.6454	0.0000	0.0000
	$c \to y$	0.6086	0.1518	0.5977	0.1578	0.6076	0.1525	0.0055	0.0806
Combined	$x \to c$	0.9952	0.5549	0.9952	0.5549	0.9952	0.5549	0.0000	0.0000
	$c \to y$	0.6000	0.1490	0.6253	0.1525	**0.6256**	**0.1698**	0.0147	0.0111

Table 4.11: Accuracy and F1-score results, including the corresponding standard deviations, for $x \to c$ and $c \to y$ predictions using Transformer

in Table 4.6 and Table 4.8 — where the accuracy/F1-score are 0.6668/0.3126 and 0.6741/0.3158 respectively — it becomes evident that while the accuracy remains relatively stable, the F1-score drops by approximately 12% with the MLP and 15% with the Transformer. This degradation is entirely expected: while a slight performance drop is observed, it is compensated by the ability to provide a transparent explanation for the decision made.

As such, the proposed C-XAI approach, structured as a CBM, in which the prediction follows the form $x \to c \to y$, has enabled the design of an explainable model for autonomous driving.

Chapter 5

Conclusions And Future Works

This chapter presents the final considerations on the work carried out, summarizing the thesis objective, the adopted approach, and the configurations that yielded the best results. Additionally, it discusses the main limitations encountered during the experimentation and outlines a series of possible improvements and future research directions that could be explored.

5.1 Conclusion

This thesis focuses on integrating methodologies that enhance the explainability of AV decisions, making the reasoning behind actions more understandable and transparent. The work explored how to design a model capable of providing insight into the decisions made by an autonomous vehicle. A Concept Bottleneck Model (CBM) approach was adopted, allowing the introduction of intermediate, human-interpretable concepts that support the final action prediction. This design enables a clearer understanding of the vehicle's decision-making process.

A key observation emerging from the study is the trade-off between explainability and performance. The introduction of interpretability layers inevitably led to a drop in accuracy when transitioning from a black-box approach to a C-XAI-based explainable system. Specifically, a performance decrease of about 12–15% was observed in the explainable models. This comparison was conducted using ResNet-50 as a baseline model, which — similarly to the proposed CBM — predicts the vehicle's action based on a single input frame. This comparison was more fair than using R2Plus1D_18, which exploits temporal sequences and thus benefits from a broader information context compared to a single observation. The CBM implemented in this thesis uses multiple frames to predict the concepts associated

with a target frame, but the final action prediction is based solely on the concepts corresponding to the target frame. For this reason, a direct comparison with models that incorporate temporal dynamics in both perception and decision-making — such as R2Plus1D_18 — would not be meaningful. This insight paves the way for future improvements. A natural extension would involve modifying the CBM architecture so that the $c \rightarrow y$ stage considers temporal dependencies — in other words, the evolution of concepts over time. To support this, the $x \rightarrow c$ stage would need to be extended to predict concepts not only for the target frame but also for preceding frames. The $c \rightarrow y$ model could then exploit this sequence of conceptual representations to produce a more context-aware prediction of the action. Such an extension could reduce the performance gap with temporal black-box models while preserving the transparency of the decision-making process.

Table 5.1, 5.2, 5.3 present a summary of the results obtained on the test set. Specifically, all tables report the Accuracy and F1-score metrics averaged across different seeds, referring to the best-performing configuration for each model. Table 5.1 shows the results for the MLP and Transformer models, focusing on the $c \rightarrow y$ prediction. Table 5.2 reports the results of the black-box models, which perform the $x \rightarrow y$ prediction. Finally, Table 5.3 presents the outcomes of the CBM model designed to introduce explainability, which uses 3D-RetinaNet in the $x \rightarrow c$ phase and then either an MLP or a Transformer in the $c \rightarrow y$ phase.

$c \rightarrow y$

Model	**Encoding**	**Accuracy**	**F1-Score**
MLP	Label	0.7150	0.2328
Transformer	Label	0.7002	0.2652

Table 5.1: Average Accuracy and F1-score across seeds for the best configurations of the MLP and Transformer models.

$x \rightarrow y$

Model	**Method**	**Accuracy**	**F1-Score**
ResNet-50	Aug. Soft	0.6413	0.2921
	Ovr. + Aug. Strong	0.6409	0.2884
R2Plus1D_18	Base	0.9375	0.4914

Table 5.2: Average Accuracy and F1-score across seeds for the best configurations of the ResNet-50 and R2Plus1D_18 models.

$x \rightarrow c \rightarrow y$

Encoding	Prediction	Accuracy	F1-Score
3D-RetinaNet + MLP			
Label	$x \rightarrow c$	0.8949	0.6454
	$c \rightarrow y$	0.6639	0.1917
3D-RetinaNet + Transformer			
Combined	$x \rightarrow c$	0.9952	0.5549
	$c \rightarrow y$	0.6170	0.1571

Table 5.3: Average Accuracy and F1-score across seeds for the best configurations of the CBM.

5.2 Future Works

Based on the observations outlined in the previous section and the design choices made during the project development, several strengths and possible improvements can be identified to achieve a more precise and effective explainable model. The main areas for improvement are described below:

- **CBM in temporal domain**: extend the CBM model to operate in the temporal domain, by leveraging the sequence of input frames not only to predict the concepts for the target frame, but to generate a temporal sequence of concepts. This would allow the models responsible for the $c \rightarrow y$ phase to exploit the temporal evolution of the concepts to predict the vehicle's action, enabling a deeper understanding of the dynamic context of the scene.

- **Dataset Balancing**: One of the main limitations lies in the imbalance of the dataset used, particularly regarding the distribution of driving actions. A more balanced dataset could enhance generalization capabilities and offer a more accurate understanding of which concepts are most critical for specific actions.

- **Semantic Enrichment**: Future datasets could benefit from richer semantic annotations, such as more detailed descriptions of agents, actions, and spatial positions. These additions would improve the clarity and precision of concept-based reasoning.

- **Multi-perspective Data**: Although this study focused on a single, front-facing camera to emulate human vision, incorporating additional viewpoints

(e.g., lateral views) or other sensors could offer a more complete environmental understanding. Nonetheless, maintaining concept annotations across all modalities remains essential to preserve interpretability.

- **Improved Concept Prediction**: While the accuracy in concept prediction was generally high, the F1-score indicates that there is room for improvement, particularly in handling class imbalance. Exploring more advanced detection architectures or data augmentation strategies could enhance overall robustness.

This work represents a first step toward the application of C-XAI in the context of autonomous driving. Although the results are promising, further research is needed to refine the models, improve data quality, and optimize the trade-off between performance and transparency. Most importantly, achieving better generalization on unseen data is essential, since it is impossible for a model to have encountered every possible road scenario worldwide. Consequently, generalization is a crucial aspect because models can only be trained on a subset of contexts. By addressing this, it will be possible to advance toward more reliable and interpretable autonomous systems—qualities that represent fundamental strengths in this application domain.

Bibliography

[1] Gurkirt Singh et al. «ROAD: The Road event Awareness Dataset for autonomous Driving». In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 01 (Feb. 5555), pp. 1–1. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2022.3150906 (cit. on pp. 2, 18, 19, 21–23, 25).

[2] Gary Marcus. *Deep Learning: A Critical Appraisal*. 2018. arXiv: 1801.00631 [cs.AI]. URL: https://arxiv.org/abs/1801.00631 (cit. on pp. 4, 5).

[3] Marco Palena, Tania Cerquitelli, and Carla Fabiana Chiasserini. «Edge-device collaborative computing for multi-view classification». In: *Computer Networks* 254 (2024), p. 110823. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2024.110823. URL: https://www.sciencedirect.com/science/article/pii/S1389128624006558 (cit. on pp. 5, 6).

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016 (cit. on p. 6).

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (cit. on pp. 6, 8, 9).

[6] Erika Ziraldo, Megan Emily Govers, and Michele Oliver. «Enhancing Autonomous Vehicle Decision-Making at Intersections in Mixed-Autonomy Traffic: A Comparative Study Using an Explainable Classifier». In: *Sensors* 24.12 (2024). ISSN: 1424-8220. DOI: 10.3390/s24123859. URL: https://www.mdpi.com/1424-8220/24/12/3859 (cit. on p. 9).

[7] Angus Dempster, François Petitjean, and Geoffrey I. Webb. «ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels». In: *Data Mining and Knowledge Discovery* 34.5 (July 2020), pp. 1454–1495. ISSN: 1573-756X. DOI: 10.1007/s10618-020-00701-z. URL: http://dx.doi.org/10.1007/s10618-020-00701-z (cit. on p. 9).

[8] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. «HIVE-COTE 2.0: a new meta ensemble for time series classification». In: *Machine Learning* 110.11–12 (Sept. 2021), pp. 3211–3243. ISSN: 1573-0565. DOI: 10.1007/s10994-021-06057-9. URL: http://dx.doi.org/10.1007/s10994-021-06057-9 (cit. on p. 9).

[9] Shenyuan Gao, Jiazhi Yang, Li Chen, Kashyap Chitta, Yihang Qiu, Andreas Geiger, Jun Zhang, and Hongyang Li. *Vista: A Generalizable Driving World Model with High Fidelity and Versatile Controllability*. 2024. arXiv: 2405.17398 [cs.CV]. URL: https://arxiv.org/abs/2405.17398 (cit. on p. 10).

[10] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. «Explainable artificial intelligence: A survey». In: *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*. 2018, pp. 0210–0215. DOI: 10.23919/MIPRO.2018.8400040 (cit. on p. 11).

[11] Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. *Concept-based Explainable Artificial Intelligence: A Survey*. 2023. arXiv: 2312.12936 [cs.AI]. URL: https://arxiv.org/abs/2312.12936 (cit. on pp. 11, 12, 14).

[12] Chih-Kuan Yeh, Been Kim, Sercan O. Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. *On Completeness-aware Concept-Based Explanations in Deep Neural Networks*. 2022. arXiv: 1910.07969 [cs.LG]. URL: https://arxiv.org/abs/1910.07969 (cit. on p. 11).

[13] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. «Towards Automatic Concept-based Explanations». In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf (cit. on p. 11).

[14] David Alvarez-Melis and Tommi S. Jaakkola. *Towards Robust Interpretability with Self-Explaining Neural Networks*. 2018. arXiv: 1806.07538 [cs.LG]. URL: https://arxiv.org/abs/1806.07538 (cit. on p. 11).

[15] Songning Lai et al. *DRIVE: Dependable Robust Interpretable Visionary Ensemble Framework in Autonomous Driving*. 2024. arXiv: 2409.10330 [cs.RO]. URL: https://arxiv.org/abs/2409.10330 (cit. on pp. 12, 13).

[16] Long Ouyang et al. *Training language models to follow instructions with human feedback*. 2022. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155 (cit. on p. 13).

[17] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2020. arXiv: 1903.11027 [cs.LG]. URL: https://arxiv.org/abs/1903.11027 (cit. on p. 13).

[18] Iz Beltagy, Matthew E. Peters, and Arman Cohan. *Longformer: The Long-Document Transformer*. 2020. arXiv: 2004.05150 [cs.CL]. URL: https://arxiv.org/abs/2004.05150 (cit. on p. 13).

[19] Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. *Concept Bottleneck Models*. 2020. arXiv: 2007.04612 [cs.LG]. URL: https://arxiv.org/abs/2007.04612 (cit. on p. 14).

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV]. URL: https://arxiv.org/abs/1512.03385 (cit. on pp. 14, 15).

[21] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. *A Closer Look at Spatiotemporal Convolutions for Action Recognition*. 2018. arXiv: 1711.11248 [cs.CV]. URL: https://arxiv.org/abs/1711.11248 (cit. on p. 16).

[22] Joao Carreira and Andrew Zisserman. *Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset*. 2018. arXiv: 1705.07750 [cs.CV]. URL: https://arxiv.org/abs/1705.07750 (cit. on p. 17).

[23] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV]. URL: https://arxiv.org/abs/1612.03144 (cit. on p. 17).

[24] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497 (cit. on p. 18).

[25] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. «1 Year, 1000km: The Oxford RobotCar Dataset». In: *The International Journal of Robotics Research (IJRR)* 36.1 (2017), pp. 3–15. DOI: 10.1177/0278364916679498. eprint: http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+html. URL: http://dx.doi.org/10.1177/0278364916679498 (cit. on p. 19).

[26] Alhassan Mumuni and Fuseini Mumuni. «Data augmentation: A comprehensive survey of modern approaches». In: *Array* 16 (2022), p. 100258. ISSN: 2590-0056. DOI: https://doi.org/10.1016/j.array.2022.100258. URL: https://www.sciencedirect.com/science/article/pii/S2590005622000911 (cit. on p. 29).

[27] Haibo He and Edwardo A. Garcia. «Learning from Imbalanced Data». In: *IEEE Transactions on Knowledge and Data Engineering* 21.9 (2009), pp. 1263–1284. DOI: 10.1109/TKDE.2008.239 (cit. on p. 32).