Stockholms universitet

Uncertainty Quantification and Transfer Learning in Transformer based Model: A Case Study of Autonomous Driving

Avinash Singh

Masteruppsats i matematisk statistik
Master Thesis in Mathematical Statistics

Masteruppsats 2026:8
Matematisk statistik
Juni 2026

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Stockholm University

Mathematical Statistics
Stockholm University
Master Thesis **2026:8**
http://www.math.su.se

Uncertainty Quantification and Transfer Learning in Transformer based Model: A Case Study of Autonomous Driving

Avinash Singh*

June 2026

Abstract

This thesis investigates uncertainty quantification and transfer learning for a transformer-based autonomous driving model. The task is formulated as supervised trajectory prediction, mapping multi-camera inputs to future trajectories. Directly transferring the performance of the model from the source domain (urban passenger cars) to the target domain (highway heavy trucks) is difficult due to significant differences in camera setups, vehicle dynamics, environments, etc.

A two-stage fine-tuning procedure is applied to address these domain gaps. This two staged procedure revealed that fine-tuning the latent representation while freezing downstream components substantially improves target-domain object detection by realigning pre-trained knowledge. End-to-end fine-tuning further improves trajectory accuracy by adapting predictions to target-domain motion patterns while maintaining plausible prediction behaviour.

We also study uncertainty quantification using Conformalized Quantile Regression. Raw Quantile Regression lacks reliable calibration guarantees, and therefore, Conformal calibration applied on top of Quantile regression achieves the desired empirical coverage across all prediction horizons and axes. The resulting conformalized intervals behave intuitively, widening over time and in heteroscedastic scenes.

The results show that Transfer learning effectively adapts the pre-trained model to the heavy-truck domain, proving superior to learning from scratch despite some remaining gaps. Furthermore, conformal calibration provides vital reliability information, establishing a strong foundation for uncertainty-aware trajectory prediction.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: avinjsingh@gmail.com. Supervisor: Chun-Biu Li.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my academic supervisor, **Chun-Biu Li**. His invaluable feedback and unwavering dedication ensured this research achieved the necessary rigor and depth. It is bittersweet to conclude our weekly discussions, which have profoundly shaped my learning and academic outlook.

I would also like to thank my manager **Ben Huber** for allowing me the opportunity to pursue this thesis at Traton AB. I would also like to thank my industrial supervisors **Yunus Emre Sahin** and **Cristina Cipriani** for their valuable time, guidance, insights and encouragement in this thesis. This thesis wouldn't have been possible without them.

Lastly, I would like to thank my mother **Rita**, my father **Kulbir**, my fiancée **Nitigya** and my friends and colleagues for their unwaivering love and support.

Contents

1 Introduction

A driver does more than just follow the road ahead while driving. The driver **observes** the surroundings, **anticipates** how nearby vehicles may move, **adapts** to the vehicle being driven, and often senses when a situation is **uncertain**. For autonomous driving models, this creates two related problems. First, the model must predict a future trajectory from high-dimensional sensor input. Second, the model should not treat every prediction as equally reliable, especially when the scene is ambiguous, unfamiliar, or different from the data on which the model was originally trained.

Modern autonomous driving models increasingly use deep neural networks to solve this problem in a data-driven way. Instead of manually designing each rule for observation, anticipation and prediction, the model learns from recorded driving examples. In this thesis, the driving task is formulated as a supervised learning problem. The input consists of multi-camera image sequences and route information, while the output is the future trajectory for the autonomous vehicle (ego vehicle). This formulation is practical because it uses offline driving logs and avoids unsafe trial-and-error learning in real traffic. At the same time, it remains a difficult statistical learning problem because the input is high-dimensional, temporally dependent, and strongly shaped by the data collection platform.

Transformer-based architectures are well suited to this type of problem because attention allows information to be exchanged across spatial locations, camera views, and time steps. The model to be studied, after a comprehensive literature review, is Unified Autonomous Driving (UniAD) (Hu et al., 2023), an end-to-end (Chen et al., 2024) autonomous driving architecture that performs all the auxiliary tasks such as observation, anticipation, and prediction (planning) through a shared latent representation. This is attractive because the final trajectory loss can, in principle, influence earlier scene representations through backpropagation, rather than treating observation and prediction as completely separate tasks.

However, a model trained in one driving domain cannot necessarily be applied directly to another. In this thesis, the source domain is nuScenes (Caesar et al., 2020), a passenger-car autonomous driving dataset, while the target domain is MAN TruckScenes (Fent et al., 2024), a heavy-truck driving dataset. Both of these datasets contain images, driving trajectories, and annotated elements from driving sequences or scenes that are 20 seconds long. The datasets have hundreds of such scenes with varying locations, time of the day, weather, etc. These two domains differ in several important ways, such as the number and placement of cameras, camera intrinsics and extrinsics, driving speed, vehicle type, scene distribution, and available spatial priors. A model trained on nuScenes has learned useful general driving knowledge, but that knowledge is tied to the source-domain sensor geometry and motion patterns. Direct untrained (zero-shot) application to

TruckScenes is therefore expected to suffer from domain shift.

Motivated by this, the first main goal of the thesis is to study transfer learning from nuScenes to TruckScenes. The transfer-learning procedure is divided into two stages. The latent representation is fine-tuned in the first stage while the downstream trajectory prediction modules are kept fixed. The purpose of this exercise is to adapt the model's internal scene representation to the truck-domain camera geometry and visual statistics. The full UniAD model is fine-tuned end-to-end after that, allowing both the representation and the trajectory prediction components to adapt to the target-domain motion patterns.

Uncertainty quantification forms the second main goal of this thesis. A standard trajectory prediction model outputs a single future path. Such point predictions may be useful, but they do not communicate whether the model is confident or uncertain. This is especially limiting in autonomous driving, where the same observed scene may allow several plausible futures. This thesis uses Conformalized Quantile Regression (CQR) (Romano et al., 2019) to construct calibrated prediction intervals around the trajectory outputs. The aim here is to provide formal bounds on prediction uncertainty under the assumptions of conformal prediction.

The broader goal of this thesis is, therefore, to investigate two practical questions that arise when applying modern transformer-based driving models to a new domain, i.e., how well knowledge can be transferred from a large source-domain model to a smaller target-domain truck dataset, and how final trajectory predictions can be equipped with calibrated uncertainty information? For want of time, we chose to investigate uncertainty quantification on UniAD trained on nuScenes only due to the equivalence between both in terms of implementation. This thesis investigates the following questions:

1. How large is the source-to-target domain gap when a UniAD model trained on nuScenes is evaluated on TruckScenes?

2. Can a two-stage transfer-learning procedure improve the model's target-domain perception and trajectory prediction performance?

3. Does fine-tuning adapt the predicted trajectories to the speed and motion patterns of the TruckScenes domain?

4. Can Conformalized Quantile Regression provide calibrated prediction intervals for the future trajectory, and do these intervals reflect higher uncertainty in more difficult scenes?

Through these questions, the thesis studies both adaptation and reliability.

1.1 Thesis Structure

The rest of the thesis is organized as follows. Section 2 presents the theory and methodology. It first formulates autonomous driving as a supervised learning problem, then introduces latent representations, imitation-based trajectory learning, transformer attention, and the UniAD architecture. The section then describes the source and target datasets, the domain gap between nuScenes and TruckScenes, and the two-stage transfer-learning curriculum. Finally, it introduces uncertainty quantification through Conformal Prediction, Quantile Regression, and Conformalized Quantile Regression.

Section 3 presents the experimental results. Section 3.1 evaluates latent representation fine-tuning and object detection performance. Section 3.2 evaluates end-to-end UniAD fine-tuning for motion prediction and planning using loss curves, L2 trajectory error, and qualitative trajectory examples. Section 3.3 presents the CQR results, including empirical coverage, interval widths, and qualitative examples of low- and high-uncertainty scenes.

Section 4 discusses the results in relation to the aims of the thesis, with emphasis on transfer learning, uncertainty calibration, remaining limitations, and possible future work. Section 5 summarizes the main conclusions.

2 Theory and Methodology

2.1 Supervised Learning

Autonomous driving (AD) can be viewed as a sequential decision-making problem i.e., at each moment, the vehicle observes its surroundings and must predict future motion. This makes reinforcement learning a natural fit for this problem, since an agent learns behaviour through trial and error by receiving rewards or penalties from the environment. Unrestricted exploration is not practical in real driving, and unsafe actions cannot be tested freely in the real world; mistakes may have physical and life threatening consequences. Even realistic simulators differ from real roads in appearance, sensor noise, traffic behaviour, and rare situations (Chen et al., 2024). It is more practical to begin from recorded driving data rather than from unrestricted trial-and-error interaction.

Supervised learning provides this formulation. Instead of asking the model to discover good driving behaviour by experimentation, we train it from examples where an observed driving situation is paired with a trajectory followed by a human driver. This trajectory is treated as the label, and the task becomes learning a mapping from observations to future motion, imitating the trajectory taken by a human expert. This formulation is termed as **Imitation Learning or Behaviour Cloning** (Chen et al., 2024; Codevilla et al., 2018). The intuition is similar to

learning from worked solutions in a mathematics textbook. The learner does not rediscover every method independently, but learns patterns from many solved examples.

Driving data differs from many textbook supervised learning examples because they are correlated. Consecutive frames from the same scene are not independent because the road layout, surrounding vehicles, weather conditions, and vehicle motion usually change only gradually. Thus, nearby frames contain similar information, like several slightly different photographs of the same event. This correlation must be considered when formulating the statistical learning problem and when separating data for training and validation.

2.1.1 The Classical Framework

Supervised learning provides a statistical framework for learning a prediction rule from examples. Each training example consists of an input and a corresponding output, and the goal is to predict the output for new, unseen inputs. If $\mathcal{X}$ denotes the input space and $\mathcal{Y}$ denotes the output space, a predictor is a function

$$f : \mathcal{X} \to \mathcal{Y}. \tag{1}$$

Here, $x \in \mathcal{X}$ is an input, $y \in \mathcal{Y}$ is the corresponding label or response, and $f(x)$ is the prediction (Hastie et al., 2009).

In the classical statistical formulation, the observed input–output pairs are treated as realizations of random variables:

$$(X_i, Y_i) \overset{\text{iid}}{\sim} P_{XY}, \qquad \mathcal{D}_N = \{(x_i, y_i)\}_{i=1}^{N}. \tag{2}$$

Here, X_i and Y_i are the random input and response for observation i, P_{XY} is their joint distribution, $\mathcal{D}_N$ is the training dataset, and N is the number of training examples. The notation 'iid' means independent and identically distributed (Hastie et al., 2009). This assumption is mathematically convenient, but it is an idealization for sequential driving data, where nearby frames from the same scene are temporally correlated.

To choose a good predictor, we use a loss function $L(y, \hat{y})$, which measures the penalty for predicting $\hat{y}$ when the true response is y. The ideal objective is the expected risk, while the quantity available in practice is the empirical risk:

$$R(f) = \mathbb{E}_{(X,Y)\sim P_{XY}}\left[L\big(Y, f(X)\big)\right], \qquad \widehat{R}_N(f) = \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, f(x_i)\big). \tag{3}$$

Here, $R(f)$ is the average prediction error on the population distribution, and $\widehat{R}_N(f)$ is its sample-based approximation. Since P_{XY} is unknown, empirical risk

minimization replaces the inaccessible population average by the observed sample average (Hastie et al., 2009).

The predictor is usually chosen from a parameterized family $\{f_\theta : \theta \in \Theta\}$, where θ denotes trainable parameters and Θ is the parameter space. Training searches for

$$\widehat{\theta} \in \arg\min_{\theta\in\Theta} \frac{1}{N}\sum_{i=1}^{N} L\big(y_i, f_\theta(x_i)\big), \qquad \widehat{f}_N(x) = f_{\widehat{\theta}}(x). \tag{4}$$

Here, $\widehat{\theta}$ is the fitted parameter vector and $\widehat{f}_N$ is the learned predictor (Hastie et al., 2009). For regression problems, one often writes

$$Y = f^\star(X) + \varepsilon, \qquad \mathbb{E}[\varepsilon \mid X] = 0, \tag{5}$$

where $f^\star$ is the unknown target function and ε represents the part of Y not explained by X, such as measurement noise or unobserved factors (Hastie et al., 2009).

A common regression loss is the squared-error loss $L(y, \hat{y}) = \|y-\hat{y}\|_2^2$, where $\|\cdot\|_2$ denotes the Euclidean norm. Under this loss, the population-optimal prediction is the conditional mean:

$$f^\star(x) = \arg\min_{a\in\mathcal{Y}} \mathbb{E}\big[\|Y-a\|_2^2 \mid X = x\big] = \mathbb{E}[Y \mid X = x]. \tag{6}$$

Here, a is a candidate prediction value (Hastie et al., 2009). This result is important later because standard supervised regression produces a point prediction, often interpretable as an average future response given the input.

2.1.2 Formulating the Driving Task

We now specialize the supervised learning notation to the driving task. The aim is to construct training pairs (X_i, Y_i), where X_i contains the input information available to the model at decision time and Y_i is the future trajectory to be predicted. The vehicle whose future motion is predicted is called the **ego vehicle**. Other moving objects, such as cars, pedestrians, and cyclists, are called **surrounding actors**. These terms only define the prediction problem; statistically, the task remains an input–output **regression problem**.

For training example i, let t_i denote the decision time. The visual input consists of four synchronized camera views observed over the current and three previous time steps:

$$I_{t_i-3:t_i} = (I_{t_i-3}, I_{t_i-2}, I_{t_i-1}, I_{t_i}), \tag{7}$$

where $I_t = \left(I_t^{(1)}, I_t^{(2)}, I_t^{(3)}, I_t^{(4)}\right)$ denotes the four camera images at time t, and $I_t^{(m)} \in \mathbb{R}^{h\times w\times c}$ is the image from camera m. Here, h and w are image height and

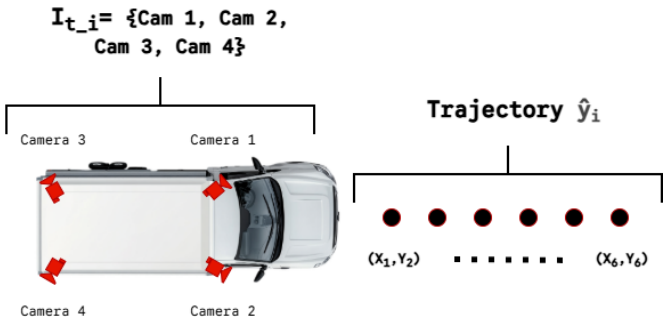

Figure 1: Illustration of the supervised input–output pair. At decision time t_i, the predictor consists of synchronized images from four cameras over the current and three previous time steps, providing temporal context. The response is the future trajectory of ego vehicle, represented by six waypoints in Cartesian coordinates (with ego vehicle at the centre (0,0)) over the next three seconds.

width, c is the number of colour channels, and $m \in \{1, 2, 3, 4\}$ are indexes of the cameras.

Several time steps are used because motion cannot be reliably inferred from one snapshot (Hu et al., 2023). A single photograph of a cyclist may not reveal whether the cyclist is stopped or moving quickly. Similarly, $I_{t_i-3:t_i}$ helps the model infer velocity and acceleration of the ego vehicle and surrounding actors, and it reduces ambiguity when an object is temporarily hidden or only partly (Hu et al., 2023).

Visual information alone may still be insufficient because the same scene can allow different valid futures. For example, at an intersection, the vehicle could turn left, turn right, or continue straight depending on the intended route. We therefore include a high-level route command (Hu et al., 2023)

$$C_i \in \{0, 1\}^3, \tag{8}$$

where C_i is a one-hot vector encoding the intended manoeuvre, such as left, straight, or right. The full supervised input is

$$X_i = (I_{t_i-3:t_i}, C_i) \in \mathbb{R}^d, \qquad d = h \cdot w \cdot c \cdot 4 \cdot 4 + 3. \tag{9}$$

Here, d is the dimension of the flattened input representation: four cameras, four time steps, $h \times w$ pixels, c colour channels, and three command entries. In this thesis, $h = 943$, $w = 1980$, and $c = 3$, i.e. pictures captured by the cameras from target domain have a size of 1980×980 with 3 RGB channels (Fent et al., 2024), which gives total dimensions $(d) = 89{,}622{,}723$.

The response variable is the future trajectory of the ego vehicle over a fixed prediction horizon containing waypoints. Waypoints are the cartesian coordinate 2-D points which represents one timestep of the future trajectory for the ego vehicle, as shown in figure 1 . We represent it by H future waypoints:

$$Y_i = (P_{i,1}, \ldots, P_{i,H}), \qquad P_{i,\tau} = \left(P_{i,\tau}^x, P_{i,\tau}^y\right)^\top \in \mathbb{R}^2. \tag{10}$$

Here, $P_{i,\tau}$ is the τ-th future waypoint for training example i, while $P_{i,\tau}^x$ and $P_{i,\tau}^y$ are its Cartesian coordinates in the vehicle-centred coordinate frame. In this thesis, $H = 6$, corresponding to six future waypoints over three seconds, so $Y_i \in \mathbb{R}^{12}$.

The supervised driving task can therefore be written as

$$\widehat{Y}_i = f_\theta(X_i), \tag{11}$$

where f_θ is the learned prediction model with parameters θ, X_i is the observed driving context, and $\widehat{Y}_i$ is the predicted future trajectory. The labels Y_i are obtained from recorded human driving demonstrations. Thus, the model is not discovering driving behaviour by trial and error; it is learning a statistical mapping from recorded situations to the trajectories that were actually followed.

2.1.3 Bridging the Dimensionality Gap: Latent Representation

The supervised formulation creates a large dimensionality gap: X_i contains tens of millions of pixel values, while Y_i contains only twelve trajectory coordinates. Learning a direct function from all raw pixels to these coordinates is statistically inefficient, because many pixel-level details, such as sky texture, lighting variation, and background colour patterns, are not directly relevant for predicting the future path. A useful model should therefore compress the image history into a representation that keeps motion-relevant information and discards noise and irrelevant information.

This intermediate representation is called a latent representation. It is "latent" because it is learned internally by the model rather than directly observed in the dataset. A simple analogy is the difference between a street photograph and a navigation sketch. A photograph contains many visual details, but a sketch that marks road layout, nearby objects, and approximate directions of motion is often more useful for deciding where to go next. Similarly, the latent representation should be a compact summary of the scene rather than a copy of the input images.

Let Φ_ϕ denote the learned encoder that maps the visual history into such an internal representation (Li et al., 2024). For training example i,

$$Z_i = \Phi_\phi(I_{t_i-3:t_i}) \in \mathbb{R}^{H_Z \times W_Z \times q}. \tag{12}$$

Here, $I_{t_i-3:t_i}$ is the multi-camera image history, Φ_ϕ is the encoder with trainable parameters ϕ, and Z_i is the learned latent representation. The quantities H_Z and W_Z denote the spatial size of the representation, while q denotes the number of learned feature channels at each spatial location. The important point is that Z_i is much smaller and more structured than the raw input and also stores extra information like speed, heading of surrounding actors etc. in learned feature channels(q) :

$$H_Z \cdot W_Z \cdot q \ll d, \tag{13}$$

where d is the raw input dimension (Li et al., 2024) defined in Equation (9).

The prediction model can then be viewed as a composition of an encoder and a trajectory module:

$$\widehat{Y}_i = g_\theta(Z_i, C_i) = g_\theta(\Phi_\phi(I_{t_i-3:t_i}), C_i)\,. \tag{14}$$

In Equation (14), g_θ is the trajectory prediction module with trainable parameters θ, C_i is the route command, and $\widehat{Y}_i$ is the predicted future trajectory. Thus, the model does not predict the trajectory directly from raw pixels. Instead, it first builds a task-specific scene summary Z_i, and the trajectory module predicts from this summary. A high-level view of this model has been shown in figure 2.

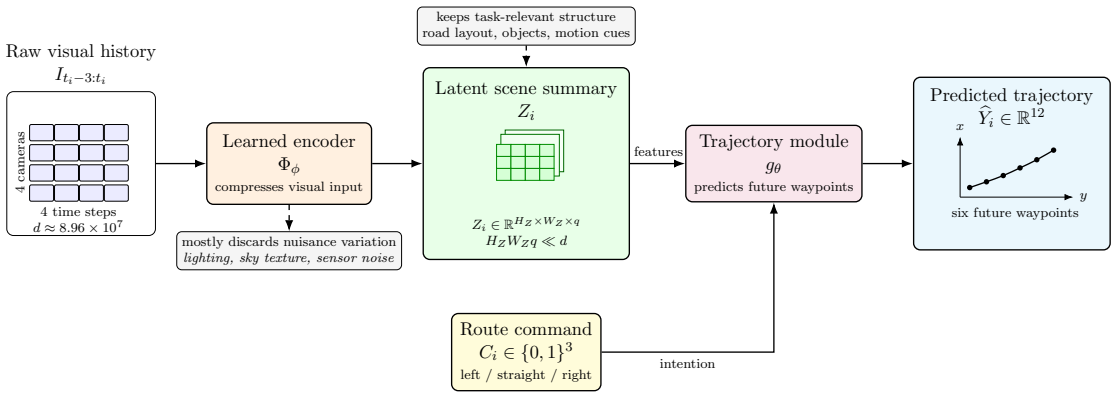

Figure 2: High-level view of the supervised driving predictor. The raw multi-camera image history is first compressed by an encoder into an internal scene summary Z_i. This representation is lower-dimensional than the original pixels and is intended to retain road layout, relevant objects, and motion cues while discarding visual details that are not useful for predicting the future trajectory. The trajectory module then combines Z_i with the route command C_i to produce the predicted trajectory $\widehat{Y}_i$.

This latent-space formulation is useful for three reasons. First, it reduces the effective dimension of the problem, making learning more data-efficient and less

sensitive to irrelevant visual noise. Second, it combines information from multiple cameras and time steps into one common internal representation, so the model has context of the scene as a whole rather than as separate images. Third, it gives the downstream predictor features closer to the quantities needed for trajectory prediction, such as spatial layout, object positions, and motion cues (Hu et al., 2023). In this sense, the latent representation acts like a learned set of notes about the scene. It is not the full raw observation, but it contains the information needed to predict the future path. In the architecture studied later, this latent representation is also reintroduced at several stages, acting like a residual source of scene information so that specialized sub-tasks like object detection, motion prediction do not lose access to the underlying driving context (Hu et al., 2023).

2.1.4 The Optimization Objective

After defining the input X_i, latent representation Z_i, and trajectory label Y_i, we specify the loss function used for training. The loss determines what the optimizer rewards or penalizes. In trajectory prediction, this is not only a numerical detail. A trajectory should be close to the demonstrated trajectory, but it should also rely on a meaningful scene representation and avoid geometrically unsafe predictions. We define geometrically unsafe predictions as paths that violate external semantic constraints (e.g., lane boundaries and obstacles) or internal kinematic constraints (e.g., steering limits and physical feasibility).

For training example i, write the ground-truth and predicted trajectories as

$$Y_i = (P_{i,1}, \ldots, P_{i,H}), \qquad \widehat{Y}_i = (\widehat{P}_{i,1}, \ldots, \widehat{P}_{i,H}), \qquad P_{i,\tau}, \widehat{P}_{i,\tau} \in \mathbb{R}^2. \tag{15}$$

Here, H is the number of future waypoints, $\tau \in \{1, \ldots, H\}$ indexes the future prediction step, $P_{i,\tau}$ is the recorded future waypoint, and $\widehat{P}_{i,\tau}$ is the corresponding predicted waypoint. In this thesis, $H = 6$.

The most direct loss is the **Imitation loss** (Ross et al., 2011),

$$\mathcal{L}_{\text{imit}}(Y_i, \widehat{Y}_i) = \sum_{\tau=1}^{H} \left\| P_{i,\tau} - \widehat{P}_{i,\tau} \right\|_2^2. \tag{16}$$

The norm $\|\cdot\|_2$ is the Euclidean norm. Equation (16) penalizes the squared distance between each demonstrated waypoint and the corresponding predicted waypoint. The intuition is like tracing a curve drawn by a teacher: the closer the predicted curve is to the demonstrated curve, the smaller the penalty.

For one waypoint, the squared loss gives the gradient

$$\nabla_{\widehat{P}_{i,\tau}} \left\| P_{i,\tau} - \widehat{P}_{i,\tau} \right\|_2^2 = 2(\widehat{P}_{i,\tau} - P_{i,\tau}). \tag{17}$$

Here, $\nabla_{\widehat{P}_{i,\tau}}$ denotes the gradient with respect to the predicted waypoint. Equation (17) shows that gradient descent pushes the prediction toward the recorded waypoint.

However, imitation alone is passive. It teaches the model to reproduce recorded behaviour, but it does not explicitly teach why a trajectory is safe. The squared distance is geometrically blind, a one-metre error into empty space and a one-metre error toward a nearby object receive the same imitation penalty. The training objective therefore combines imitation with additional structure.

The model can be written as a composition, (as defined in Equation 14),

$$\widehat{Y}_i = g_\theta(Z_i, C_i) = g_\theta(\Phi_\phi(I_{t_i-3:t_i}), C_i)$$

Where $\hat{Y}_i$ is the model prediction for the ith example.

Representation Loss The representation loss (Hu et al., 2023) ensures that Z_i remains a meaningful scene summary:

$$\mathcal{L}_{\text{rep}}(\phi) = \frac{1}{N}\sum_{i=1}^{N} \ell_{\text{rep}}(Z_i, T_i), \qquad Z_i = \Phi_\phi(I_{t_i-3:t_i}). \tag{18}$$

Here, N is the number of training examples, ℓ_{rep} is the sample-level representation loss, and T_i denotes auxiliary targets used to supervise the internal scene representation. These targets may encode relevant object locations, motion cues, or spatial structure. The purpose is that the latent representation should describe the scene, not merely act as an arbitrary numerical shortcut for the final trajectory.

This point is important because Y_i is low-dimensional. If the model is optimized only for final trajectory error, it may find shortcuts that work on the training set but fail on new scenes. In the extreme case, the representation may collapse, meaning that different scenes are mapped to nearly indistinguishable internal summaries:

$$\Phi_\phi(I_a) \approx \Phi_\phi(I_b) \qquad \text{even when } I_a \text{ and } I_b \text{ describe different scenes.}$$

Here, I_a and I_b denote two different visual inputs. If this happens, the downstream trajectory module may still reduce training loss by exploiting dataset-specific correlations, but it no longer has a reliable view of the scene. The result can be good training loss and poor validation performance (Chen et al., 2024).

A simple analogy is solving mathematics problems by memorizing final answers without understanding the method. On familiar homework problems, the answers may look correct; on a new exam question, the student fails because the intermediate reasoning was never learned. Similarly, the representation loss grades

the model's intermediate reasoning by forcing the latent space to preserve scene information needed for prediction.

For this reason, the representation objective is useful in two stages. First, the representation can be trained separately:

$$\widehat{\phi}_{\text{rep}} \in \arg\min_{\phi} \mathcal{L}_{\text{rep}}(\phi). \tag{19}$$

This builds a sound latent representation before the trajectory prediction relies on it. Second, the representation loss is present in the full objective loss as well, as shown in Equation 23, so that the latent space does not collapse when the model later focuses on minimizing final trajectory loss (Hu et al., 2023).

Trajectory Prediction (Planning) Loss The Trajectory Prediction (planning) loss (Hu et al., 2023) combines imitation with a collision-aware term:

$$\mathcal{L}_{\text{plan}}(\theta, \phi) = \frac{1}{N} \sum_{i=1}^{N} \left[\lambda_{\text{imit}} \mathcal{L}_{\text{imit}}(Y_i, \widehat{Y}_i) + \lambda_{\text{coll}} \mathcal{L}_{\text{coll}}(\widehat{Y}_i) \right]. \tag{20}$$

Here, $\lambda_{\text{imit}} \geq 0$ and $\lambda_{\text{coll}} \geq 0$ control the relative weight of imitation and collision avoidance. The prediction $\widehat{Y}_i$ depends on both θ and ϕ through Equation (14). This Loss combines the imitation loss from (16) with a collision based regularization, making it clear that the objective of training is not to simply copy the human driver but also keep safety (collision avoidance) as a constraint.

The collision-aware term is based on geometric overlap. For two planar regions A and B, define the **Intersection-over-Union** (Hu et al., 2023) score as

$$\text{IoU}(A, B) = \frac{\text{area}(A \cap B)}{\text{area}(A \cup B)}. \tag{21}$$

Here, $A \cap B$ is the overlapping region and $A \cup B$ is the union of the two regions. If the regions do not overlap, the IoU is zero; if they overlap strongly, the IoU is large (Figure 3).

For trajectory prediction, the **Collision loss** (Hu et al., 2023) is written as

$$\mathcal{L}_{\text{coll}}(\widehat{Y}_i) = \sum_{\tau=1}^{H} \sum_{j=1}^{M_{i,\tau}} \sum_{r=1}^{R} \omega_r \ \text{IoU} \left(\text{box}(\widehat{P}_{i,\tau}; w_{\text{ego}}, \ell_{\text{ego}}, \delta_r), B_{i,j,\tau} \right). \tag{22}$$

Here, $M_{i,\tau}$ is the number of relevant surrounding objects at future step τ, $B_{i,j,\tau}$ is the two-dimensional footprint of object j, and $\text{box}(\widehat{P}_{i,\tau}; w_{\text{ego}}, \ell_{\text{ego}}, \delta_r)$ is the predicted vehicle footprint (area occupied by the vehicle at a specific moment in time) centered at $\widehat{P}_{i,\tau}$. The constants w_{ego} and ℓ_{ego} are the width and length of the ego

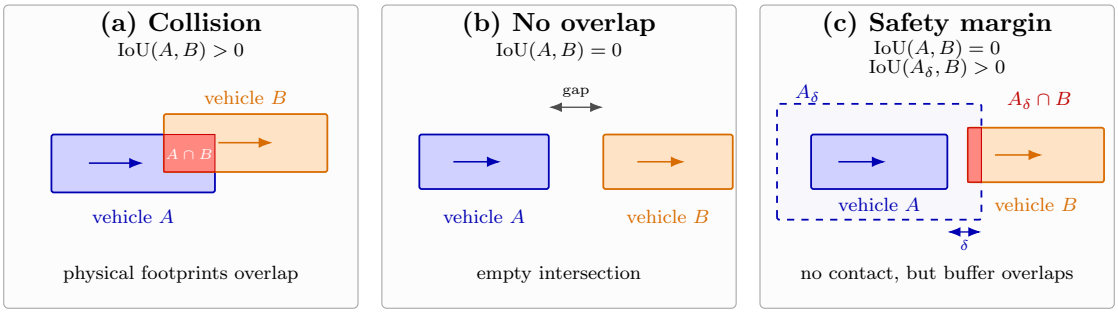

Figure 3: Top-down illustration of Intersection-over-Union (IoU) for two vehicle footprints. In (a), the physical rectangles overlap, so the IoU is positive and the situation corresponds to a collision. In (b), the footprints are disjoint, so the IoU is zero. In (c), the physical footprints are still disjoint, but inflating vehicle A by a safety margin δ gives an enlarged footprint A_δ that overlaps vehicle B, producing a positive margin-based IoU.

vehicle. The quantity $\delta_r \geq 0$ is an additional safety margin, $r \in \{1, \ldots, R\}$ indexes the margin level, and $\omega_r \geq 0$ is the weight for that margin (Hu et al., 2023).

The margin δ_r is necessary because expert demonstrations usually do not contain actual collisions. If the loss only penalized physical overlap, then almost all training examples would have zero collision loss, giving little information about near-misses. Inflating the vehicle footprint creates a soft buffer around the predicted path. This is similar to leaving personal space when walking through a crowd. One does not wait until bodies touch before changing direction. The margin makes near-collisions visible to the loss before they become hard collisions (Hu et al., 2023). Figure 3 provides a visualization of IoU with and without safety margins.

The **full objective**(Hu et al., 2023) is

$$\mathcal{L}_{\text{total}}(\theta, \phi) = \lambda_{\text{plan}} \mathcal{L}_{\text{plan}}(\theta, \phi) + \lambda_{\text{rep}} \mathcal{L}_{\text{rep}}(\phi), \tag{23}$$

where $\lambda_{\text{plan}} \geq 0$ and $\lambda_{\text{rep}} \geq 0$ control the relative importance of the trajectory objective and the representation objective. Training corresponds to solving

$$(\widehat{\theta}, \widehat{\phi}) \in \arg\min_{\theta, \phi} \mathcal{L}_{\text{total}}(\theta, \phi). \tag{24}$$

Here, $\widehat{\theta}$ and $\widehat{\phi}$ denote the fitted trajectory and representation parameters after optimization.

Overall, the objective balances three goals. The imitation term pulls the predicted trajectory toward the demonstrated trajectory. The collision term discourages predictions that overlap with, or pass too close to, surrounding objects. The

representation term keeps the latent space informative and prevents the model from achieving low training loss through a collapsed or shortcut-based representation. This does not provide a formal safety guarantee, but it makes the supervised objective more aligned with the physical requirements of trajectory prediction than imitation loss alone.

2.1.5 Advantages of the Supervised Learning Formulation

Formulating the driving task as supervised learning, or behavioral cloning, has several practical advantages. The available data already consist of input–output pairs (X_i, Y_i), where X_i is the observed driving situation and Y_i is the future trajectory followed in the recording (Chen et al., 2024). The training problem can therefore be written as

$$\widehat{\theta} \in \arg\min_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} L\big(Y_i, f_\theta(X_i)\big). \tag{25}$$

Here, f_θ is the prediction model with parameters θ, Θ is the parameter space, L is the loss function, N is the number of training examples, and $\widehat{\theta}$ denotes the fitted parameters. This objective is useful because it turns a difficult sequential decision problem into a concrete empirical risk minimization problem.

Reinforcement Learning (RL) represents the formal mathematical standard for this task because it treats driving as a Markov Decision Process, where an agent learns to maximize a long-term reward through continuous environmental feedback and causal reasoning. While RL is theoretically superior for its ability to discover super-human policies and account for the consequences of its own actions, it is notoriously difficult to deploy due to the 'reality gap' of simulators and the lack of a perfect, hand-engineered reward function (Chen et al., 2024).

Avoiding reward-function design. Driving combines several objectives: making progress, staying comfortable, obeying rules, and avoiding collisions. In a reinforcement-learning formulation, these goals must be compressed into a single reward signal, which is difficult to formulate and can encourage undesirable behaviour if the reward is misspecified. Supervised learning avoids this step by using the demonstrated trajectory Y_i as the target. The assumption is not that the human driver is mathematically optimal, but that the demonstration provides a useful example of behaviour that balances many practical constraints (Chen et al., 2024).

Using offline data efficiently. Supervised learning can train directly from recorded driving logs. The model does not need to explore dangerous actions in the real world, and it does not rely on a simulator as the main source of experience. In simple terms, supervised learning is like learning from worked examples: the model repeatedly compares its prediction with the recorded trajectory and adjusts its parameters accordingly. While Offline and Off-Policy RL theoretically allow for policy optimization from recorded logs, they suffer from overestimation bias and instability caused by bootstrapping on out-of-distribution actions. Supervised learning bypasses these issues by treating the task as a direct mapping problem, providing a more stable and predictable optimization landscape for complex architectures like UniAD. Consequently, we trade the potential for super-human optimization for the reliability and safety of expert-aligned behavior (Chen et al., 2024).

Encouraging predictable behaviour. The loss in Equation (25) penalizes deviations from demonstrated trajectories. Therefore, in familiar situations, the model is encouraged to stay close to behaviors observed in the dataset. This gives the model a useful human-driving prior. It means that the model is biased toward the support of the training data. (Chen et al., 2024).

Overall, the supervised formulation is suitable because it is data-driven, avoids unsafe exploration, removes the need for a hand-designed reward function, and provides a stable optimization objective. These advantages make it a natural starting point for the trajectory prediction problem considered in this thesis.

2.1.6 Limitations of the Supervised Paradigm

Although supervised learning provides a tractable formulation, it remains an approximation to the full driving problem. Driving is sequential and closed-loop, whereas supervised learning treats the task mainly as prediction from recorded examples. This creates two important limitations.

Compounding errors and distribution shift. During training, the model observes inputs generated by human driving behaviour. During deployment, the model's own predictions can influence the future situations it encounters. Thus, the training and deployment input distributions may differ:

$$P_{\text{train}}(X) \neq P_{\text{deploy}}(X). \tag{26}$$

Here, $P_{\text{train}}(X)$ denotes the distribution of inputs in the recorded training data, and $P_{\text{deploy}}(X)$ denotes the distribution of inputs encountered when the learned

model is used. A small prediction error can move the system into a state that is less common in the training data. The next prediction is then made from a less familiar situation, so errors can accumulate over time. A simple analogy is walking with a compass that is slightly misaligned: the first step is almost correct, but after many steps the final position can be far from the intended path (Chen et al., 2024).

This issue is especially relevant for driving data because nearby frames from the same scene are temporally correlated. Consecutive frames often contain nearly the same road layout, nearby vehicles, and motion pattern. Therefore, a large number of frames does not necessarily mean the same number of independent examples. This is one reason why evaluation should avoid mixing near-duplicate frames from the same scene across training and test sets.

The insufficiency of point predictions. A standard supervised regression model outputs a single future trajectory,

$$\widehat{Y}_i = \widehat{f}(X_i), \tag{27}$$

where X_i is the input, $\widehat{f}$ is the fitted prediction function, and $\widehat{Y}_i$ is the predicted trajectory. Under squared-error loss, this point prediction estimates the conditional mean (Hastie et al., 2009),

$$f^\star(x) = \mathbb{E}[Y \mid X = x]. \tag{28}$$

Here, $f^\star(x)$ is the population-optimal squared-error predictor at input x, and $\mathbb{E}[Y \mid X = x]$ is the expected future trajectory conditional on that input.

This is reasonable when the future is concentrated around one likely outcome. It is less suitable when several distinct futures are plausible. For example, if two different manoeuvres are possible, an average trajectory may fall between them and may not correspond to a realistic path (Chen et al., 2024). More importantly, the point prediction in Equation (27) does not say whether the model is confident or uncertain. A routine scene and an ambiguous scene may both produce one trajectory, even though the second prediction should be treated with more caution.

These limitations motivate the later uncertainty quantification part of the thesis. Instead of relying only on the point prediction $\widehat{Y}_i$, Conformalized Quantile Regression is used to construct calibrated prediction intervals. The goal is not to remove uncertainty, but to make it visible and statistically controlled under the assumptions stated later.

2.2 Transformer

The previous section formulated trajectory prediction as a supervised learning problem. The remaining question is what type of function class is suitable for the

high-dimensional input $I_{t_i-3:t_i}$, which contains several camera views over multiple time steps. A useful model must process local visual patterns, but also compare information across cameras, spatial locations, and time (Li et al., 2024). For example, a small vehicle braking ahead may be more relevant for the future trajectory than a visually larger but irrelevant background region.

Classical convolutional and recurrent architectures impose useful but restrictive structures. Convolutions process information mainly through local receptive fields, while recurrent models summarize the past through a fixed-dimensional hidden state (Zhang et al., 2023). Both mechanisms can create bottlenecks when the prediction depends on interactions between distant parts of the scene (Vaswani et al., 2017). A simple analogy is looking at a traffic scene: a human driver does not inspect every pixel equally, but focuses on relevant evidence such as nearby vehicles, road boundaries, and recent motion.

This motivates the use of the **Transformer** (Vaswani et al., 2017) architecture . The Transformer relies on **Attention** (Vaswani et al., 2017), which allows the model to form data-dependent weighted averages of other representations. Intuitively, Attention lets the model decide which parts of the observed context should influence a particular prediction. Thus, instead of compressing all spatial and temporal information through one narrow bottleneck, the model can selectively retrieve information from the full observed context window.

2.2.1 The UniAD Paradigm: End-to-End Differentiability

Before comparing modular and end-to-end architectures (Chen et al., 2024), we clarify the three functional blocks shown in Figure 4. In this context, *perception*, *prediction*, and *planning* are engineering names for three statistical subproblems. *Perception* estimates the current scene from sensor observations: which relevant objects and road structures are present, and approximately where they are. *Prediction* forecasts how surrounding moving objects, such as vehicles or pedestrians, may move in the near future. *Planning* chooses the ego vehicle's own future trajectory, using the current scene estimate, predicted surrounding actor motion, and route command. Thus, perception asks "What is around the ego vehicle now?", prediction asks "How might the surroundings move next?", and planning asks "Where should the ego vehicle go?" A simple analogy is walking through a crowded train station: first one observes people and obstacles, then anticipates how nearby people may move, and finally chooses one's own path (Chen et al., 2024; Hu et al., 2023).

While Transformers provide the mechanism for processing complex sequence data, their deployment inside the driving stack is equally important. Classical systems often use a sequence of separate modules: one model estimates the current scene, a second predicts the motion of surrounding actors, and a third produces

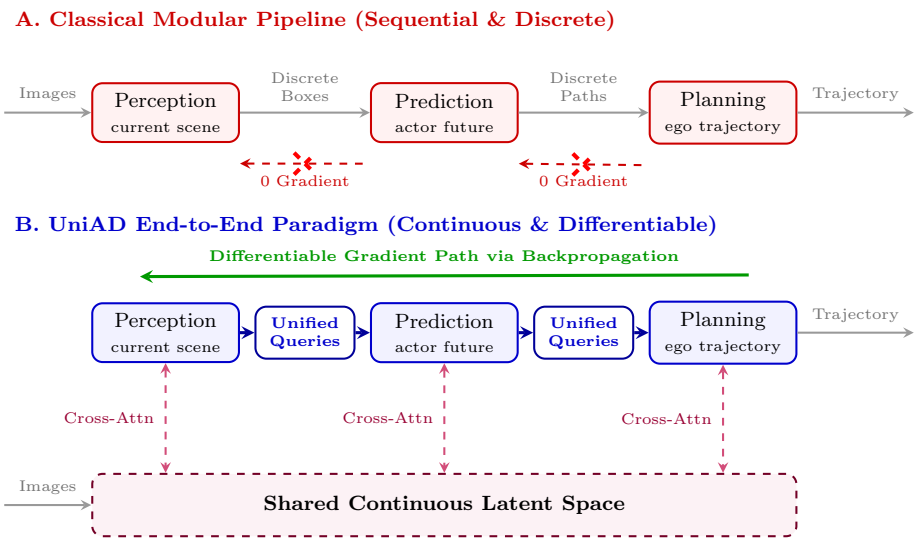

Figure 4: Compact comparison of optimization paradigms. **A. Modular approach:** perception, prediction, and planning communicate through discrete intermediate outputs, which limits gradient flow between modules. **B. UniAD-style end-to-end approach:** continuous latent features and query-based attention preserve a differentiable path from the final trajectory loss to earlier scene representations.

the ego trajectory. The limitation is that each component is optimized using a local proxy objective, such as bounding-box error, even though the final goal is accurate and safe trajectory prediction. Good performance on each intermediate task therefore does not automatically imply good performance of the end task which is trajectory prediction for the ego vehicle (Chen et al., 2024).

A second limitation is optimization. If modules communicate through discrete or non-differentiable outputs, the final trajectory loss cannot easily update earlier perception representations. For instance, if the final trajectory is poor because an object was poorly represented upstream, the planning loss may not provide a useful gradient signal to correct that earlier representation. Also, since each module compresses the full representation into its outputs, there is information loss. This information, if discarded, may be useful for the end task but not for the early sub-tasks. Since the model is sequential, this information once discarded cannot be retrieved by the end motion planner, leading to sub-optimal behavior (Chen et al., 2024).

The architecture used in this thesis, **Unified Autonomous Driving (UniAD)** (Hu et al., 2023), addresses this by framing the driving pipeline as an end-to-end differentiable Transformer based neural network. Rather than passing only dis-

crete coordinates between components, UniAD represents relevant scene information through continuous latent vectors. At an abstract level,

$$Z_i = \Phi_\phi(I_{t_i-3:t_i}), \qquad \widehat{Y}_i = g_\theta(Z_i, C_i). \tag{29}$$

Here, $I_{t_i-3:t_i}$ is the multi-camera image history for training example i, Φ_ϕ is the representation encoder with parameters ϕ, Z_i is the learned latent representation, C_i is the route command, g_θ is the trajectory prediction module with parameters θ, and $\widehat{Y}_i$ is the predicted future ego trajectory.

Because attention operations are differentiable (Vaswani et al., 2017), the final trajectory loss can influence earlier representations through backpropagation. This does not guarantee safe driving, but it makes joint optimization of representation and prediction possible. The following subsections define the attention mechanism and explain why it is useful for building continuous latent representations from spatial-temporal data.

2.2.2 The Bottleneck of Recursive Estimation

In trajectory prediction, the observed history can be viewed as a multivariate stochastic process. Let

$$\mathbf{X}_{1:T} = (\mathbf{x}_1, \ldots, \mathbf{x}_T) \in \mathbb{R}^{T \times d_{\text{in}}}, \tag{30}$$

where $\mathbf{x}_t \in \mathbb{R}^{d_{\text{in}}}$ denotes the observation or feature vector at time t, T is the number of observed time steps, and d_{in} is the input dimension. The available information up to time t is represented by

$$\mathcal{F}_t = \sigma(\mathbf{x}_1, \ldots, \mathbf{x}_t). \tag{31}$$

Here, $\sigma(\cdot)$ denotes the sigma-algebra generated by the observations up to time t. The prediction problem is to infer a future quantity $\mathbf{Y}$ from this history.

Recurrent Neural Networks (RNNs), including LSTMs and GRUs, process a sequence through a continuous hidden state $\mathbf{h}_t \in \mathbb{R}^{d_h}$, as shown in Figure 5:

$$\mathbf{h}_t = \varphi(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}). \tag{32}$$

Here, φ is a nonlinear activation such as tanh, $\mathbf{W}_{hx} \in \mathbb{R}^{d_h \times d_{\text{in}}}$ maps the current input into the hidden state, $\mathbf{W}_{hh} \in \mathbb{R}^{d_h \times d_h}$ maps the previous hidden state into the current hidden state, and $\mathbf{b} \in \mathbb{R}^{d_h}$ is a bias vector.

The hidden state $\mathbf{h}_t$ functions as a recursive summary statistic intended to encode the observed history $\mathbf{x}_{1:t}$. This compression mechanism can be formalized via the Information Bottleneck principle (Tishby et al., 2000), wherein the

network implicitly optimizes a trade-off: minimizing the representation complexity $I(\mathbf{x}_{1:t}; \mathbf{h}_t)$ while maximizing the predictive information $I(\mathbf{h}_t; \mathbf{Y})$ regarding the future trajectory. However, the standard RNN architecture imposes a strict structural constraint by confining $\mathbf{h}_t$ to a fixed-dimensional Euclidean space $\mathbb{R}^{d_h}$, inherently bounding its channel capacity. Consequently, this induces a severe representational bottleneck (Bahdanau et al., 2014). By the Data Processing Inequality, applied along the temporal Markov chain $\mathbf{x}_1 \to \mathbf{h}_1 \to \cdots \to \mathbf{h}_t$, the mutual information between early inputs and the current state, $I(\mathbf{x}_k; \mathbf{h}_t)$ for $k \ll t$, is subject to strict decay, rendering the retention of long-term dependencies mathematically prohibitive.

The optimization issue appears in the chain rule. For $k < t$,

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_k} = \left(\prod_{s=k+1}^{t} \frac{\partial \mathbf{h}_s}{\partial \mathbf{h}_{s-1}} \right) \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k}. \tag{33}$$

The product contains one Jacobian factor for every intermediate time step. If these Jacobians contract on average, the gradient signal from $\mathbf{x}_k$ to $\mathbf{h}_t$ can become small. For example, if

$$\left\| \frac{\partial \mathbf{h}_s}{\partial \mathbf{h}_{s-1}} \right\| \leq c < 1 \qquad \text{for } s = k+1, \ldots, t, \tag{34}$$

then

$$\left\| \frac{\partial \mathbf{h}_t}{\partial \mathbf{x}_k} \right\| \leq c^{t-k} \left\| \frac{\partial \mathbf{h}_k}{\partial \mathbf{x}_k} \right\|. \tag{35}$$

Here, $\|\cdot\|$ denotes a matrix or vector norm, and c is a contraction constant. Equation (35) illustrates the **vanishing-gradient problem** (Pascanu et al., 2013): distant observations may have a weak effect on the training signal. Gated models reduce this problem, but the path between distant observations still passes through many sequential updates.

2.2.3 The Transformer Paradigm: Global Estimation

Transformer (Vaswani et al., 2017) reduces this recursive bottleneck by using a different computational structure. It defines a parameterized sequence-to-sequence mapping

$$f_\theta : \mathbb{R}^{N \times d_{\text{model}}} \to \mathbb{R}^{N \times d_{\text{model}}}, \tag{36}$$

where N is the number of tokens and d_{model} is the internal feature dimension. The tokens may represent time steps, image patches, spatial cells in a latent representation, or learned query vectors.

Instead of compressing the entire history into a single hidden state, the Transformer keeps a separate representation for each token inside the context window.

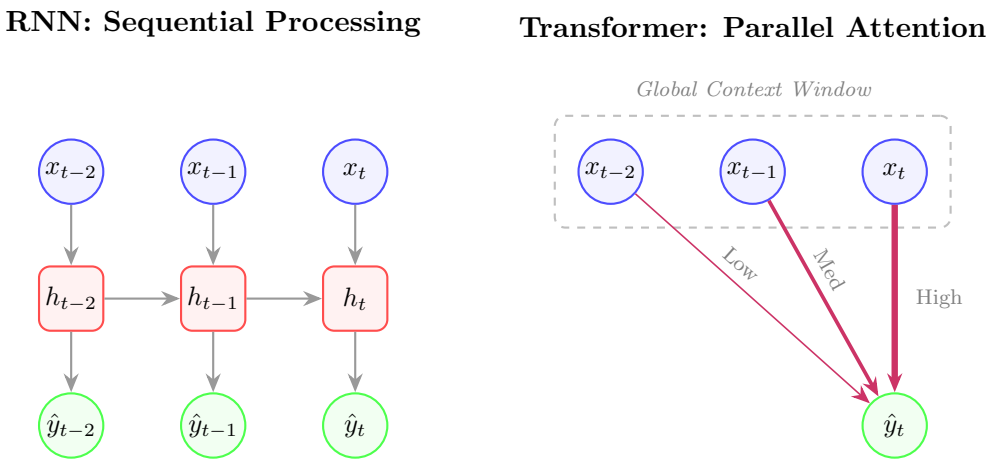

Figure 5: Comparison of historical data utilization. **Left:** A Recurrent Neural Network (RNN) processes historical data sequentially, compressing past observations into a temporal chain of hidden states (h_t). **Right:** A Transformer bypasses the sequential bottleneck by utilizing an attention mechanism, querying all historical tokens (x) directly and concurrently to construct the output ($\hat{y}_t$). The varying thickness of the purple arrows visually denotes the dynamic attention weights assigned to different historical observations.

Attention lets these tokens interact , as shown in Figure 5. Thus, the computational path between two tokens is reduced from order $\mathcal{O}(T)$ in a recurrent chain to order $\mathcal{O}(1)$ within one attention layer (Vaswani et al., 2017). This does not mean information is preserved perfectly, but it gives distant tokens a direct route to influence one another.

Statistical and Computational Trade-offs This structure changes both estimation and computation. Since token representations can be updated in parallel, the model does not need a long chain of hidden states before a distant observation can affect the current representation. During gradient-based optimization, information from different positions in the context can therefore contribute more directly to the loss gradients.

Since there are no free lunches, The cost is that self-attention compares every token with every other token. The core operation forms the pairwise affinity matrix $\mathbf{QK}^\top$, and the time and memory complexity scale as $\mathcal{O}(N^2 d_{\text{model}})$ (Vaswani et al., 2017).

Here, N^2 comes from all pairwise token comparisons, and d_{model} is the feature dimension. In trajectory prediction, N can be large because the latent representation may contain many spatial cells across several time steps. Thus, Transformers

provide flexible global interaction, but the context size must be managed carefully.

2.2.4 Scaled Dot-Product Attention

The mechanism that enables this direct information routing is **Attention** (Vaswani et al., 2017). For a statistical audience, a useful comparison is the **Nadaraya–Watson estimator** (Nadaraya, 1964; Zhang et al., 2023). Given observations $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_N, \mathbf{y}_N)\}$, it estimates the response at a target point $\mathbf{x}$ by a locally weighted average:

$$\hat{\mathbf{y}}(\mathbf{x}) = \sum_{i=1}^{N} \frac{K_h(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{N} K_h(\mathbf{x}, \mathbf{x}_j)} \mathbf{y}_i. \tag{37}$$

Here, $K_h(\mathbf{x}, \mathbf{x}_i)$ is a kernel measuring normalized similarity between $\mathbf{x}$ and $\mathbf{x}_i$, and h is a bandwidth controlling locality. Scaled dot-product attention has the same broad weighted-average structure, but the similarity rule is learned through projections rather than fixed in the original input space. More precisely, the analogue of the Nadaraya–Watson kernel $K_h(\mathbf{x}, \mathbf{x}_i)$ is the **non-negative learned attention kernel**

$$\kappa_{\text{att}}(\mathbf{q}_i, \mathbf{k}_j) = \exp\left(\frac{\mathbf{q}_i^\top \mathbf{k}_j}{\sqrt{d_k}}\right). \tag{38}$$

Here, $\mathbf{q}_i$ is the i-th query vector, $\mathbf{k}_j$ is the j-th key vector, and d_k is the query-key dimension. The raw dot product $\mathbf{q}_i^\top \mathbf{k}_j$ is therefore the pre-softmax similarity score, while κ_{att} is the kernel-like quantity that is normalized into attention weights.

Let $\mathbf{H}_{\text{target}} \in \mathbb{R}^{M \times d_{\text{model}}}$ denote the target representations to be updated, and let $\mathbf{H}_{\text{source}} \in \mathbb{R}^{N \times d_{\text{model}}}$ denote the source representations from which information is extracted. Attention first forms

$$\mathbf{Q} = \mathbf{H}_{\text{target}} \mathbf{W}_Q, \qquad \mathbf{K} = \mathbf{H}_{\text{source}} \mathbf{W}_K, \qquad \mathbf{V} = \mathbf{H}_{\text{source}} \mathbf{W}_V. \tag{39}$$

Here, $\mathbf{W}_Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\mathbf{W}_K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $\mathbf{W}_V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ are **learned projection matrices**. The matrix $\mathbf{Q} \in \mathbb{R}^{M \times d_k}$ contains queries, $\mathbf{K} \in \mathbb{R}^{N \times d_k}$ contains keys, and $\mathbf{V} \in \mathbb{R}^{N \times d_v}$ contains values. Intuitively, queries ask for information, keys are addresses used to identify relevant source tokens, and values are the contents retrieved. A useful analogy is searching in a library: the query is the question, the keys are catalogue entries, and the values are the material retrieved.

The pairwise score matrix is

$$\mathbf{S} = \mathbf{Q}\mathbf{K}^\top. \tag{40}$$

Because $\mathbf{q}_i$ and $\mathbf{k}_j$ are projected into the same latent space, their inner product measures alignment between query i and key j. A high positive score means that

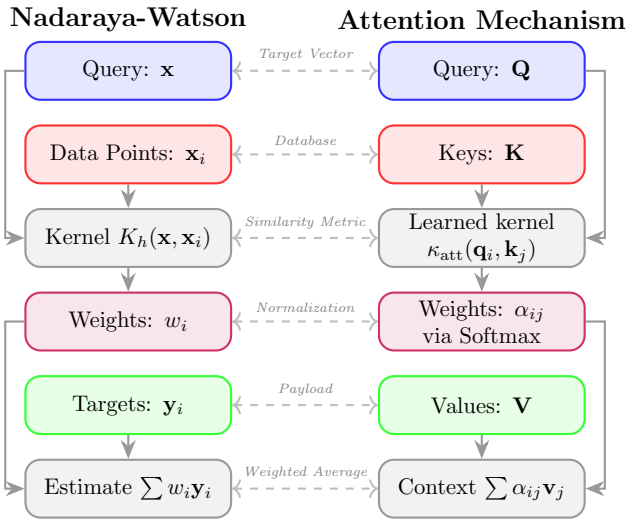

Figure 6: Analogy between Nadaraya–Watson kernel regression and scaled dot-product attention. Both methods compute a weighted average of values. Kernel regression uses a fixed similarity rule $K_h(\mathbf{x}, \mathbf{x}_i)$, while attention learns query and key projections that define the kernel-like similarity $\kappa_{\text{att}}(\mathbf{q}_i, \mathbf{k}_j)$(refer to Equation 38), which is then normalized by the softmax to produce weights α_{ij}.

the corresponding value vector should receive a larger weight. After scaling and exponentiation, this score gives the learned attention kernel in Equation (38).

The scores are scaled by $\sqrt{d_k}$. If the components of $\mathbf{q}_i$ and $\mathbf{k}_j$ are independent with zero mean and unit variance, then

$$\text{Var}\left(\sum_{m=1}^{d_k} q_{im} k_{jm}\right) = d_k. \tag{41}$$

Here, q_{im} and k_{jm} are the m-th components of $\mathbf{q}_i$ and $\mathbf{k}_j$. Thus, without scaling, the dot-product variance grows with d_k, which can push the softmax into regions with small gradients. Dividing by $\sqrt{d_k}$ stabilizes the pre-softmax variance.

The row-wise softmax gives the attention matrix (Vaswani et al., 2017)

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right). \tag{42}$$

Each row of $\mathbf{A}$ contains non-negative weights summing to one. Equivalently, the element α_{ij} can be written as

$$\alpha_{ij} = \frac{\kappa_{\text{att}}(\mathbf{q}_i, \mathbf{k}_j)}{\sum_{\ell=1}^{N} \kappa_{\text{att}}(\mathbf{q}_i, \mathbf{k}_\ell)} = \frac{\exp(\mathbf{q}_i^\top \mathbf{k}_j / \sqrt{d_k})}{\sum_{\ell=1}^{N} \exp(\mathbf{q}_i^\top \mathbf{k}_\ell / \sqrt{d_k})}. \tag{43}$$

Here, α_{ij} is the weight assigned to value vector $\mathbf{v}_j$ when updating query $\mathbf{q}_i$. These weights resemble probabilities mathematically, but they should not be interpreted as calibrated probabilistic beliefs.

For a single query $\mathbf{q}_i$, the output is

$$\mathbf{z}_i = \sum_{j=1}^{N} \alpha_{ij} \mathbf{v}_j. \tag{44}$$

Here, $\mathbf{v}_j$ is the j-th row of $\mathbf{V}$, and $\mathbf{z}_i \in \mathbb{R}^{d_v}$ is the updated representation for query i. Equation (44) captures the main intuition: attention replaces a fixed summary of the input by a learned weighted average of relevant value vectors. Because $\mathbf{z}_i$ is formed directly from all attended values, gradients can reach distant or spatially separated tokens through a shorter computational path than in Equation (33).

2.2.5 Self-Attention vs. Cross-Attention

The same scaled dot-product attention operation can be used in two ways, depending on where the Query ($\mathbf{Q}$), Key ($\mathbf{K}$), and Value ($\mathbf{V}$) matrices come from.

Self-Attention (Vaswani et al., 2017): Modeling Internal Dependencies In self-attention, all three matrices are derived from the same representation:

$$\mathbf{Z}_{\text{self}} = \text{Attention}(\mathbf{H}\mathbf{W}_Q, \mathbf{H}\mathbf{W}_K, \mathbf{H}\mathbf{W}_V). \tag{45}$$

Here, $\mathbf{H} \in \mathbb{R}^{N \times d_{\text{model}}}$ is the input representation, and $\mathbf{Z}_{\text{self}}$ is the updated representation. Self-attention computes pairwise similarity weights within a single set of representations. It should not be interpreted as a covariance matrix; rather, it learns which elements of the same sequence should exchange information. For example, in our case study, two nearby actors in a scene can influence each other's representations if their interaction matters for prediction.

Cross-Attention (Vaswani et al., 2017): Conditional Feature Extraction In cross-attention, the queries come from a target representation, while keys and values come from an external source representation:

$$\mathbf{Z}_{\text{target}} = \text{Attention}(\mathbf{H}_{\text{target}}\mathbf{W}_Q, \mathbf{H}_{\text{source}}\mathbf{W}_K, \mathbf{H}_{\text{source}}\mathbf{W}_V). \tag{46}$$

Here, $\mathbf{H}_{\text{target}}$ is the representation being updated, $\mathbf{H}_{\text{source}}$ is the representation being read from, and $\mathbf{Z}_{\text{target}}$ is the updated target representation. Cross-attention therefore acts as targeted feature extraction: the target asks for information, and the source provides the evidence.

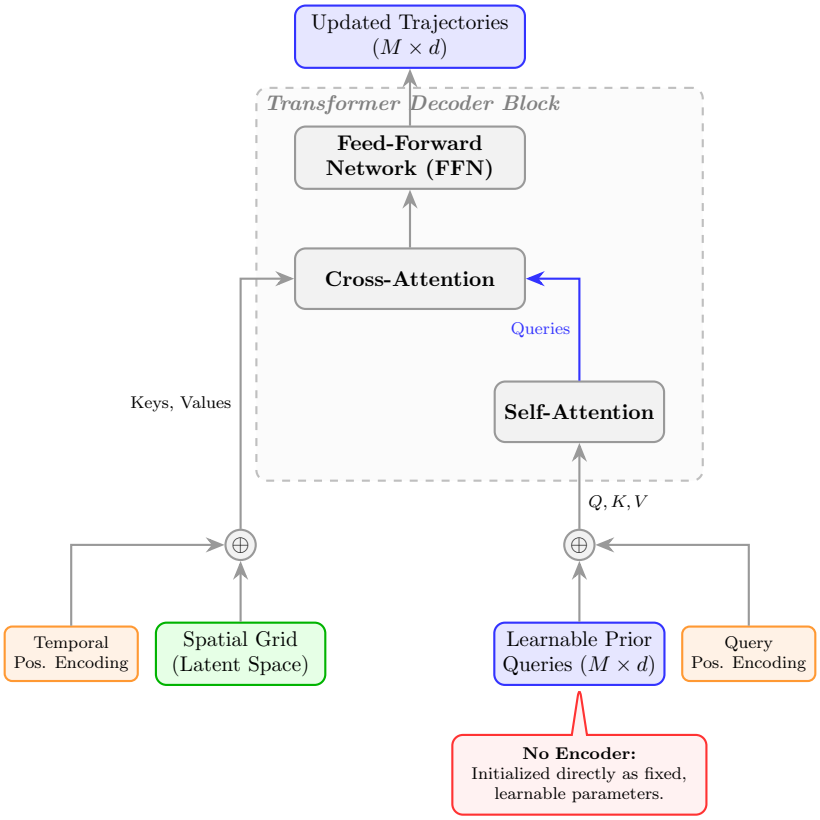

Figure 7: High-level schematic of the query-based decoder used at the planning stage. A fixed number M of learned query vectors is updated through self-attention and then uses cross-attention to extract relevant information from the larger latent representation. Positional encodings ($\oplus$) are added to preserve temporal, spatial, or query identity. This avoids full self-attention over all N latent tokens while still allowing the trajectory module to retrieve task-relevant evidence.

The Directionality of Cross-Attention The asymmetry of cross-attention is important. The representations mapped to **Q** are the ones that ask for information and are updated. The representations mapped to **K** and **V** act as the source of information. The library analogy is again useful: the target representation asks the question, the source keys determine which parts of the source are relevant, and the source values provide the content retrieved. Reversing the roles would change which representation is updated. Since we want to update target representation using context from source, therefore the target representation to be updated is always introduced as **Q** rather than as **K,V** and source context is always intoduced via **K,V** (Vaswani et al., 2017).

Application Example: The UniAD Architecture In UniAD (Vaswani et al., 2017), the architecture departs from discrete, hand-crafted intermediate representations by utilizing continuous Unified Queries, as shown in Figure 4. Formally, these queries are instantiated as a set of learnable embedding vectors, $\mathbf{Q} \in \mathbb{R}^{N \times d_{\text{model}}}$. Rather than being derived purely from the current observation, they are initialized as free parameters and iteratively optimized across the entire training dataset via stochastic gradient descent. Consequently, they encapsulate a dataset-level inductive bias, acting as structural "slots" that learn to represent potential traffic entities, map elements, and valid trajectory modes. Before incorporating external observations, self-attention mechanisms allow these queries to interact with one another, resolving spatial and semantic conflicts to ensure internal consistency. Subsequently, cross-attention layers enable the queries to dynamically extract relevant evidence—such as road geometry or motion cues—from the shared dense latent representation.

By routing information through this continuous, query-based interface rather than through discrete bounding boxes or fixed semantic classes, UniAD establishes an end-to-end differentiable computational graph. This allows the gradient from the final trajectory loss to propagate backward, explicitly shaping the intermediate perception and prediction features. While this does not theoretically guarantee recovery from severe intermediate errors, it couples the optimization objectives, enabling joint representation learning in a way that is structurally impossible for a strictly modular pipeline.

2.2.6 Temporal Structure

Attention by itself has no inherent notion of temporal order. If the same permutation is applied to the queries, keys, and values, the output is permuted in the same way (Vaswani et al., 2017):

$$\text{Attention}(\Pi\mathbf{Q}, \Pi\mathbf{K}, \Pi\mathbf{V}) = \Pi\,\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}), \tag{47}$$

where $\Pi \in \{0,1\}^{N \times N}$ is a permutation matrix. Equation (47) means that attention can compare tokens, but does not automatically know whether one token came before or after another. Temporal or positional information must therefore be added explicitly.

Positional Encoding A common method is sinusoidal positional encoding (Vaswani et al., 2017):

$$\text{PE}_{\text{pos},2m} = \sin\left(\frac{\text{pos}}{10000^{2m/d_{\text{model}}}}\right), \qquad \text{PE}_{\text{pos},2m+1} = \cos\left(\frac{\text{pos}}{10000^{2m/d_{\text{model}}}}\right). \tag{48}$$

Here, pos is the position index, m indexes pairs of feature dimensions, and d_{model} is the model dimension. The denominator controls the wavelength. Small values of m produce high-frequency components that distinguish nearby positions, while larger values produce low-frequency components that vary more slowly. Instead of representing time as a single integer, the model receives a multi-frequency code for temporal position.

The Geometric Logic of Sinusoidal Encoding Sinusoidal encodings are useful because they give the attention mechanism a structured notion of relative position. Without positional grounding, the model treats inputs as an unordered set of features. The sinusoidal construction in Equation (48) helps in three ways:

- **Relative translation.** For a fixed offset k, $\text{PE}(\text{pos}+k)$ can be represented as a linear transformation of $\text{PE}(\text{pos})$. This helps the model reason about relative temporal distances Δt.
- **Dot-product interaction.** Since attention depends on $\mathbf{QK}^\top$, positional information should remain visible through inner products. Products of sinusoidal terms involve expressions such as $\cos(\theta_1 - \theta_2)$, making similarity depend partly on relative position.
- **Multi-scale time representation.** High-frequency components distinguish adjacent frames, while low-frequency components provide slower variation across the sequence. This is useful because motion prediction depends on both short-term changes and broader temporal context.

In Figure 7, positional encodings are added both to the latent spatial features and to the learnable queries. For the latent space, they help preserve temporal and spatial identity. For the queries, they prevent all learned proposals from collapsing into a single indistinguishable state during self-attention.

Enforcing Causality via Masking A causal mask is required when an attention layer processes a sequence containing future positions that should not be visible to the current prediction. This is done by adding a mask matrix $\mathbf{M}$ before the softmax:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{QK}^\top}{\sqrt{d_k}} + \mathbf{M}\right), \tag{49}$$

where

$$\mathbf{M}_{ij} = \begin{cases} 0, & j \le i, \\ -\infty, & j > i. \end{cases} \tag{50}$$

Here, i indexes the query position and j indexes the key position. Since $\exp(-\infty) = 0$, forbidden future positions receive zero attention weight after the softmax. In this thesis, the input history $I_{t_i-3:t_i}$ already contains only past and current observations. Therefore, causal masking is mainly relevant if the architecture internally processes future tokens during training, or if it is used autoregressively.

2.2.7 Query-Based Decoding over the Latent Representation

In the original Transformer, the encoder processes the input sequence using self-attention, and the decoder generates the output using self-attention and cross-attention. This structure is powerful, but full self-attention over a large latent scene representation can be expensive. If the latent representation contains N tokens, full self-attention requires an $N \times N$ pairwise comparison.

To reduce this cost, UniAD uses a *query-based decoder* (Hu et al., 2023), shown in Figure 7. Instead of applying full self-attention over every latent token, the model starts from a smaller set of learned query vectors (refer to Application Example in section 2.2.5 for definition of unified queries) and lets these queries read from the latent representation. This assumes that the latent representation already contains the relevant scene information, and that the decoder only needs to retrieve what is useful for the downstream prediction task.

The key simplification is how the queries are generated. In a standard encoder, queries are derived from the input data itself, creating the $N \times N$ bottleneck. UniAD instead uses a small fixed matrix of learnable queries, with $M \ll N$. For a statistician, these M vectors can be understood as learned proposal slots or latent questions. Rather than asking every latent token to compare with every other token, the model asks M learned questions about the scene. The M tokens start as generic learned questions and are transformed by the attention mechanism by adding revelant context from the latent representation. These transformed queries then contain specific and rich context about the questions the planner needs answers for before it can predict a plausible trajectory. For example, the question might about the location of other surrounding actors in the scene. Without this high context information a safe and plausible trajectory is impossible.

Including cross-attention between queries and latent tokens, and self-attention among the queries, the approximate complexity changes from

$$\mathcal{O}(N^2 d_{\text{model}}) \quad \longrightarrow \quad \mathcal{O}(MNd_{\text{model}} + M^2 d_{\text{model}}). \tag{51}$$

Here, N is the number of latent tokens, M is the number of learned queries, and d_{model} is the feature dimension. The term MNd_{model} comes from cross-attention between queries and latent tokens, while $M^2 d_{\text{model}}$ comes from self-attention among the queries. Since $M \ll N$, this is cheaper than full self-attention over all latent tokens (Hu et al., 2023).

Statistically, the query-based decoder acts as an efficient conditional feature extractor. It allows a small number of learned query vectors to retrieve spatial and temporal evidence directly from the latent representation. A useful analogy is asking a small set of well-designed questions about a large image, instead of comparing every part of the image with every other part. This preserves the main advantage of attention, namely selective information retrieval, while keeping the computation feasible for large latent representations.

2.3 Transfer Learning

Before defining the specific adaptation procedure, we first motivate why transfer learning is used instead of training the full architecture from random initialization. In statistical machine learning, Transfer Learning Yosinski et al., 2014; Zhuang et al., 2020 uses information learned by training on a source domain to improve learning in a related target domain. In this thesis, the source domain is a large passenger-car driving dataset, while the target domain is a smaller heavy-truck driving dataset.

Let

$$\mathcal{D}_S = \{(x_i^S, y_i^S)\}_{i=1}^{n_S} \sim P_S, \qquad \mathcal{D}_T = \{(x_i^T, y_i^T)\}_{i=1}^{n_T} \sim P_T. \tag{52}$$

Here, $\mathcal{D}_S$ and $\mathcal{D}_T$ denote the source and target datasets, n_S and n_T are their sample sizes, x_i^S, x_i^T are inputs, y_i^S, y_i^T are supervised labels, and P_S and P_T are the corresponding data-generating distributions. In this section, the main supervised output is the future trajectory, denoted by Y^{traj}.

Training from scratch estimates all model parameters using only the target dataset $\mathcal{D}_T$. For a high-capacity transformer-based model, this is statistically difficult because the target dataset may be too small to constrain millions of parameters without overfitting. Transfer learning instead starts from parameters learned on the larger source dataset:

$$\widehat{\omega}_S \in \arg\min_{\omega} \frac{1}{n_S} \sum_{i=1}^{n_S} L_S\big(y_i^S, f_\omega(x_i^S)\big), \qquad \widehat{\omega}_T \in \arg\min_{\omega} \frac{1}{n_T} \sum_{i=1}^{n_T} L_T\big(y_i^T, f_\omega(x_i^T)\big) \tag{53}$$

$\omega_0 = \widehat{\omega}_S$. Here, ω denotes the trainable model parameters, f_ω is the prediction model, L_S and L_T are the source and target losses, $\widehat{\omega}_S$ is the pre-trained source solution, and ω_0 is the initialization used for target-domain fine-tuning.

The intuition is similar to that of a student who has already learned general calculus before studying a specialized engineering application. The student still needs to adapt to the new problem, but does not need to relearn algebra, derivatives, and basic functions from the beginning. Similarly, early visual and geometric features, such as edges, object shapes, and spatial layout, can be partly reused across

driving domains. Transfer learning, therefore, acts both as an initialization strategy and as a regularization strategy: it starts optimization from a useful region of parameter space and reduces the amount that must be learned only from the smaller target dataset.

In this thesis, we start from a UniAD (Hu et al., 2023) model pre-trained on source domain data and, after adapting the architecture, we fine-tune the pre-trained parameters further on target domain using low learning rates. The motivation behind low learning rates is that we do not want to destroy the common mappings source domain and target domain have, we only want to adapt it to target domain.

2.3.1 Datasets

This study uses two datasets representing the source and target domains. The source domain, denoted by $\mathcal{D}_S$, is the **nuScenes** dataset (Caesar et al., 2020). It is a large-scale benchmark for urban autonomous driving, collected using a passenger vehicle with a multi-sensor setup. Its empirical distribution contains many dense urban scenes, including pedestrians, cyclists, intersections, and low-speed traffic. It consists of 1000 annotated driving scenes, each of 20 seconds length. Samples from this domain are used for the initial pre-training of the architecture.

The target domain, denoted by $\mathcal{D}_T$, is the proprietary **MAN TruckScenes** dataset (Fent et al., 2024). It is collected from a heavy-duty commercial truck operating mainly in highway environments and logistics hubs. It consists of 747 annotated driving scenes, each of 20 seconds length. The target samples represent the distribution on which the transfer-learning procedure is evaluated and fine-tuned. The central problem is that the source and target distributions are related but not identical.

2.3.2 Domain Gap

Deploying a model trained on nuScenes directly on MAN TruckScenes creates a domain adaptation problem. The joint distributions of inputs and trajectory labels differ:

$$P_S(X, Y^{\text{traj}}) \neq P_T(X, Y^{\text{traj}}). \tag{54}$$

Here, X denotes the model input and Y^{traj} denotes the future trajectory label. This joint shift can be decomposed into three main effects:

$$P_S(X) \neq P_T(X), \qquad \text{covariate shift,} \tag{55}$$

$$P_S(G) \neq P_T(G), \qquad \text{scene-prior shift,} \tag{56}$$

$$P_S(Y^{\text{traj}} \mid X) \neq P_T(Y^{\text{traj}} \mid X), \qquad \text{conditional or concept shift.} \tag{57}$$

In Equation (56), G denotes a high-level scene type, such as urban intersection, highway lane-following, or logistics-yard driving. Covariate shift means that the visual inputs change. Scene-prior shift means that the relative frequencies of scene types change. Concept shift means that the same type of observed situation may require a different future trajectory because the vehicle platform and driving environment have changed.

The main source-to-target shifts are described below. For each shift, we separate the description from the statistical implication, because the important point is not only what changes, but why that change affects the learned model.

- **Sensor topology: number and placement of cameras.**

  Description. nuScenes uses a six-camera surround-view system, while TruckScenes uses four cameras (Caesar et al., 2020; Fent et al., 2024):

  $$X_S^{\text{cam}} \in \mathbb{R}^{6\times H\times W\times C}, \qquad X_T^{\text{cam}} \in \mathbb{R}^{4\times H\times W\times C}. \tag{58}$$

  Here, X_S^{cam} and X_T^{cam} denote the camera-image inputs in the source and target domains, H and W are image height and width, and C is the number of colour channels.

  Statistical implication. This is a structural covariate shift. The pre-trained model has learned to combine six perspective views into a shared latent representation. When two views are removed, the input is not merely missing pixels; the geometry of the multi-camera fusion problem changes. If Z_S and Z_T denote the source and target latent representations, then one generally expects

  $$\mathbb{E}_{P_S}[Z_S] \neq \mathbb{E}_{P_T}[Z_T], \qquad \text{Var}_{P_S}(Z_S) \neq \text{Var}_{P_T}(Z_T). \tag{59}$$

  Here, $\mathbb{E}$ denotes expectation and Var denotes variance. The downstream trajectory module is trained to process latent activations with source-domain statistics. If the latent statistics shift too much, it receives out-of-distribution features. Intuitively, this is like learning to navigate using six mirrors and then suddenly being given only four: the task is still related, but the learned spatial reference system must be realigned.

  Adaptation implication. The camera embeddings should therefore not be randomly reinitialized. Instead, source embeddings corresponding to shared camera positions can be reused, while unavailable source views are removed. This preserves the parts of the multi-camera prior that remain valid.

- **Camera intrinsics and field of view.**

Description. TruckScenes uses wider-angle cameras than nuScenes. Under a simplified pinhole projection model (Fent et al., 2024),

$$u = f\frac{x}{z} + c_x, \qquad v = f\frac{y}{z} + c_y, \tag{60}$$

where $(x, y, z)^\top$ is a 3D point in camera coordinates, (u, v) is its image location, f is the focal length, and (c_x, c_y) is the principal point.

Statistical implication. Changing the field of view changes the mapping between physical size and pixel size. Objects at the same physical distance may appear smaller, more distorted, or differently shaped in the target domain. Therefore, the distribution of visual scales changes:

$$P_S\big(\text{scale}(X)\big) \neq P_T\big(\text{scale}(X)\big), \tag{61}$$

where $\text{scale}(X)$ denotes image-level object scale features extracted from the input. This matters because the visual encoder is trained on source-domain scale statistics. A distant object that was large enough to be salient in nuScenes may become a small, low-texture region in TruckScenes. The model may then treat it as background unless the visual representation is adapted.

A real-world analogy is switching from a normal camera lens to a wide-angle action camera. The scene is still the same physical world, but distances, object sizes, and distortions look different. A visual system trained on the first lens cannot automatically interpret the second lens perfectly.

Adaptation implication. The latent representation and spatial attention layers must be fine-tuned so that visual features from the wider field of view are mapped to the correct physical scene representation.

- **Camera extrinsics and mounting height.**

  Description. Truck cameras are mounted higher than passenger-car cameras and may have a different downward pitch. The camera projection depends not only on intrinsics but also on the extrinsic transformation (Fent et al., 2024):

$$\widetilde{p}_{\text{img}} \propto K \begin{bmatrix} R & t \end{bmatrix} p_{\text{world}}. \tag{62}$$

  Here, p_{world} is a 3D world point in homogeneous coordinates, $\widetilde{p}_{\text{img}}$ is the projected image point, K is the intrinsic matrix, R is the rotation matrix, and t is the translation vector.

  Statistical implication. The model learns an implicit relationship between image coordinates and physical ground-plane locations. If the camera height and pitch change, the same pixel location can correspond to a different physical location. Let D denote physical depth and let $U = (u, v)$ denote an image

coordinate. The learned source-domain depth relationship need not match the target-domain relationship:

$$\mathbb{E}_{P_S}[D \mid U] \neq \mathbb{E}_{P_T}[D \mid U]. \tag{63}$$

This means that a model trained on passenger-car geometry may systematically estimate objects as too close or too far away when used on truck images. In trajectory prediction, such errors matter because the future path depends strongly on where obstacles and road boundaries are in physical space.

Adaptation implication. Fine-tuning the latent representation is necessary to recalibrate the pixel-to-ground mapping. Without this step, the end-to-end trajectory model may be forced to operate on geometrically biased scene summaries.

- **Information asymmetry: missing deterministic spatial priors.**

  Description. In some source-domain training setups, the model can use external spatial information such as lane boundaries, drivable areas, or intersection layouts. TruckScenes does not provide the same deterministic spatial priors. Thus, the target model must rely more heavily on sensor-derived features.

  Statistical implication. The prediction task changes from a more constrained conditional distribution to a less constrained one:

$$P_S(Y^{\text{traj}} \mid X_{\text{sensors}}, X_{\text{map}}) \quad \longrightarrow \quad P_T(Y^{\text{traj}} \mid X_{\text{sensors}}). \tag{64}$$

  Here, X_{sensors} denotes sensor-derived input features and X_{map} denotes external spatial map information. Removing X_{map} increases uncertainty because the model must infer road structure from the visual latent representation alone. This is similar to navigating with and without a detailed map. With a map, many impossible paths are ruled out immediately. Without a map, the model must infer which areas are drivable from what it sees.

  Adaptation implication. The latent representation must preserve stronger visual evidence about road structure, lane-like geometry, and free space. The final end-to-end stage can then adapt the trajectory output to this sensor-only representation.

- **Speed distribution and latent-grid compression.**

  Description. nuScenes primarily represents lower-speed urban driving, while TruckScenes contains higher-speed highway driving (Caesar et al., 2020; Fent

et al., 2024). The physical reason this matters can be illustrated by

$$E_{\text{kin}} = \frac{1}{2}mv^2, \qquad d_{\text{stop}} \approx \frac{v^2}{2a}. \tag{65}$$

Here, E_{kin} is kinetic energy, m is vehicle mass, v is speed, d_{stop} is an idealized stopping distance, and a is the magnitude of deceleration. These equations are not used as a physical simulator; they only illustrate why higher speeds and heavy vehicles require longer-range perception and different trajectory dynamics.

Statistical implication. A larger physical perception range must be represented by a finite latent grid. If a 200×200 latent grid represents 50 m in the source domain but 150 m in the target domain, then the physical side length represented by each grid cell changes as

$$\Delta_T = \frac{150}{200} = 3\frac{50}{200} = 3\Delta_S. \tag{66}$$

Here, Δ_S and Δ_T denote the physical side length represented by one grid cell in the source and target domains. Since the grid represents a two-dimensional ground plane, the physical area per cell increases by a factor of nine. This creates a spatial quantization problem: more physical space must be summarized by the same number of latent cells.

The implication is increased risk of feature collision, where distinct physical objects or road structures are compressed into the same or nearby latent locations. In simple terms, this is like drawing a large highway scene on the same small sheet of graph paper: each square of the graph now represents a larger real-world area, so fine details become harder to separate.

Adaptation implication. The latent encoder must learn a more aggressive compression mapping. Fine-tuning should therefore first focus on the spatial representation before the complete model is adapted end-to-end.

- **Scene and contextual shift.**

  Description. The operating environment changes from dense urban roads to highways and logistics hubs. This changes both the frequency of scene types and the visual contexts in which trajectories are observed (Caesar et al., 2020; Fent et al., 2024):

$$P_S(G) \neq P_T(G), \qquad P_S(X \mid G) \neq P_T(X \mid G). \tag{67}$$

  Here, G denotes a high-level scene type and X denotes the input features.

Statistical implication. The scene prior $P(G)$ affects what the model expects to see. In nuScenes, dense urban intersections and low-speed interactions are common. In TruckScenes, highway lanes, commercial vehicles, guardrails, and logistics environments become more prominent. The contextual shift $P(X \mid G)$ is equally important: even if the task is still trajectory prediction, the visual evidence surrounding the trajectory changes.

A simple analogy is recognizing the same driving rule in two countries with different road layouts. The rule may be conceptually similar, but the surrounding cues are different, so a learner initially trained in one environment may make systematic mistakes in the other.

Adaptation implication. The target-domain training stages must adapt both the latent representation and the final trajectory mapping so that the model no longer relies too strongly on urban source-domain scene priors.

Table 1: Summary of source-to-target domain shifts and corresponding adaptation responses.

Shift	**Statistical implication**	**Adaptation response**
Camera topology	Latent activation statistics change because the number of camera views changes	Slice and realign camera embeddings
Camera intrinsics	Object scale and distortion statistics change	Fine-tune visual and spatial representation layers
Camera extrinsics	Pixel-to-ground mapping becomes biased	Recalibrate latent spatial representation
Missing map priors	Conditional prediction becomes less constrained and more uncertain	Strengthen sensor-derived latent features
Speed and spatial horizon	Larger physical range must be compressed into the same latent grid	Fine-tune spatial compression before full model training
Scene context	Source and target scene priors differ	End-to-end adaptation on target-domain trajectories

2.3.3 Bridging the Domain Gap: A Two-Stage Adaptation Curriculum

A natural but inefficient strategy would be to fine-tune the entire architecture end-to-end immediately on TruckScenes. The difficulty is that several shifts occur at the same time: the number of cameras changes, camera geometry changes,

map priors are missing, the spatial horizon expands, and vehicle dynamics differ. If all components are updated at once, the optimizer receives competing signals from many sources of mismatch. This can lead to unstable gradients and poor adaptation.

We therefore use a two-stage adaptation curriculum: first latent representation fine-tuning, then end-to-end fine-tuning. The model parameters are conceptually partitioned as

$$\omega = (\phi, \theta), \tag{68}$$

where ϕ denotes the parameters responsible for constructing the latent representation, and θ denotes the remaining trainable parameters used to map the latent representation to the final trajectory output. Freezing a parameter group means that it is held fixed during backpropagation. The corresponding part of the model still produces outputs in the forward pass, but its weights are not updated by the optimizer.

The motivation for staging is simple: first make the model see the target domain correctly, and only then train the full prediction system. This is similar to adapting from a small car to a large truck. Before refining the full driving strategy, one must first adjust to the new viewpoint, blind spots, and spatial scale.

Stage 1: Latent representation fine-tuning. The first stage adapts the visual and geometric representation to the target domain. The downstream trajectory parameters are frozen, and only the representation parameters are updated:

$$\widehat{\phi} \in \arg\min_{\phi} \mathcal{L}_{\text{rep}}(\phi; \mathcal{D}_T), \qquad \theta \text{ frozen.} \tag{69}$$

Here, $\mathcal{L}_{\text{rep}}$ is the representation loss on the target dataset $\mathcal{D}_T$. The goal is to make the latent space geometrically meaningful for the truck domain before relying on it for end-to-end trajectory prediction.

- **Topological realignment via embedding slicing.** The source and target camera sets are

$$\mathcal{C}_S = \{\text{FL}, \text{F}, \text{FR}, \text{BL}, \text{B}, \text{BR}\}, \qquad \mathcal{C}_T = \{\text{FL}, \text{FR}, \text{BL}, \text{BR}\}. \tag{70}$$

  Here, FL, F, FR, BL, B, and BR denote front-left, front, front-right, back-left, back, and back-right camera positions. If $E_S(c)$ denotes the source camera embedding for camera c, the target embeddings are initialized by

$$E_T = \big(E_S(\text{FL}), E_S(\text{FR}), E_S(\text{BL}), E_S(\text{BR})\big). \tag{71}$$

  Here, E_T denotes the target-domain camera embedding tensor. The source front and back embeddings are dropped because the corresponding cameras

are not available in the target topology. The statistical implication is that the target model begins with embeddings already aligned to comparable physical viewpoints, rather than learning all camera identities from noise.

- **Geometric recalibration of the latent space.** The intrinsics and extrinsics shifts in Equations (60) and (62) mean that the same visual pattern can correspond to a different physical position in TruckScenes than in nuScenes. Stage 1 therefore focuses on recalibrating the latent representation. The statistical implication is that the model first learns a target-domain pixel-to-scene mapping before the final trajectory loss is allowed to modify the entire model.

- **Latent-space compression and spatial horizon.** TruckScenes requires a larger physical perception horizon because of higher speeds and longer stopping distances. Expanding the latent grid from 200×200 to 600×600 would preserve metric resolution but would be computationally expensive because attention cost grows with the number of tokens. Therefore, the latent grid is kept fixed while its physical coverage increases. The statistical implication is that each latent cell must summarize more physical space, making representation learning harder. Stage 1 isolates this compression problem so that the model can adapt the latent space before the final trajectory objective is optimized end-to-end.

Stage 2: End-to-end fine-tuning. After the latent representation has been adapted, the second stage unfreezes the full model. This stage addresses the conditional shift in Equation (57): the same type of observed scene may imply different future trajectories for a heavy truck than for a passenger car. The target-domain objective is

$$(\widehat{\phi}, \widehat{\theta}) \in \arg\min_{\phi,\theta} \mathcal{L}_{\text{total}}(\phi, \theta; \mathcal{D}_T). \tag{72}$$

Here, $\mathcal{L}_{\text{total}}$ is the full training loss on the target dataset, including the representation and trajectory objectives used by the architecture.

The rationale is that representation and trajectory prediction are not independent. If only the final trajectory module were fine-tuned, it would have to predict truck-specific trajectories from features still optimized for passenger-car urban driving. Conversely, if only the latent representation were fine-tuned, the model would not fully adapt its final trajectory mapping to truck-domain dynamics. End-to-end fine-tuning allows both parts to adjust jointly.

Trajectory forecasting can be written abstractly as learning

$$P_T\left(Y^{\text{traj}}_{t+1:t+H} \mid X_{1:t}\right), \tag{73}$$

where $X_{1:t}$ is the observed input history up to time t, $Y^{\text{traj}}_{t+1:t+H}$ is the future trajectory over horizon H, and P_T is the target-domain distribution.

- **Joint adaptation of representation and trajectory prediction.** Fine-tuning on $\mathcal{D}_T$ encourages the model to reduce reliance on source-domain urban dynamics and instead match the empirical transition patterns of highway trucking. The statistical implication is that the learned conditional distribution $P_T(Y^{\text{traj}} \mid X)$ should reflect target-domain motion patterns rather than source-domain ones.

- **Data-driven kinematic priors.** Instead of hard-coding truck-specific physical constraints into the prediction solver, the model is allowed to learn these regularities from target-domain trajectories. The statistical implication is that quantities such as longer deceleration profiles and smoother steering behaviour are absorbed through the empirical loss. This does not guarantee physical feasibility in all cases, but it encourages the learned trajectory distribution to better match the kinematics observed in heavy-truck data.

Throughout both stages of the adaptation curriculum, the model parameters are updated using the Adam optimizer (Kingma & Ba, 2014). Due to the substantial memory footprint required to process high-resolution multi-camera sequences and maintain continuous dense latent representations, the training is conducted with a micro-batch size of 4. This ensures that the optimizer can reliably compute gradients for the computationally heavy attention mechanisms without exceeding hardware limitations.

Overall, the two-stage curriculum adapts the model from representation to prediction. Stage 1 aligns the latent space with the target-domain sensors and geometry. Stage 2 jointly fine-tunes the complete trajectory-prediction system on TruckScenes. This ordering reduces the burden on end-to-end fine-tuning by ensuring that the final trajectory objective receives a more stable and target-relevant latent representation.

2.4 Uncertainty Quantification using Conformal Prediction

2.4.1 Uncertainty

In this thesis, uncertainty refers to the fact that the future target is not known exactly, even after observing the input. Statistically, uncertainty can be represented by a probability distribution, a variance, a quantile, or a prediction set. A standard supervised model often returns only a point prediction,

$$\hat{y} = \hat{f}(x), \tag{74}$$

where x is the input, $\hat{f}$ is the fitted prediction model, and $\hat{y}$ is the predicted target. A point prediction is useful, but it does not describe how plausible nearby or distant alternatives are.

A useful analogy is travel-time prediction. Saying "the trip will take 20 minutes" is a point prediction. Saying "the trip will probably take between 18 and 30 minutes" also communicates uncertainty. The second statement is more useful for planning because it describes both the central estimate and the possible error.

Uncertainty Quantification (UQ) aims to report more than a single point estimate. Depending on the method, this may be a predictive distribution, a variance estimate, a confidence interval, or a prediction set. For example, a probabilistic model may describe uncertainty through a predictive distribution $p(y \mid x, \mathcal{D})$, where y is the target, x is the input, and $\mathcal{D}$ is the training data. A highly concentrated predictive distribution indicates low uncertainty, whereas a diffuse distribution with large spread, heavy tails, or high entropy indicates high uncertainty.

Relevance in Autonomous Driving In the driving prediction problem, relying only on deterministic point estimates can be limiting. The same observed scene may allow several plausible future motions, especially when other road users are present, visibility is poor, or the model observes a rare situation. A prediction and planning system that produces only one trajectory provides no direct information about how uncertain it is about that predicted trajectory.

In this thesis, uncertainty estimates should be interpreted as statistical reliability information, not as direct safety guarantees. A calibrated interval can tell us how often the true future coordinate is expected to fall inside the interval under the assumptions stated later, but it cannot by itself guarantee collision-free or safe behaviour. Its role is to make uncertainty visible. For example, if the prediction interval is wide, a downstream module could choose to behave more cautiously, such as by reducing speed or leaving a larger spatial margin. This thesis does not evaluate such a downstream controller; it focuses just on constructing statistically calibrated prediction uncertainty intervals.

Aleatoric Uncertainty: The Irreducible Noise Aleatoric uncertainty describes variability that remains even if the model class and parameters were known. It is uncertainty caused by the data-generating process itself, conditional on the information available to the model. A common regression model is

$$Y = f^\star(X) + \varepsilon, \qquad \mathbb{E}[\varepsilon \mid X] = 0. \tag{75}$$

Here, X is the input random variable, Y is the target random variable, $f^\star$ is the unknown regression function, and ε is the part of Y not explained by X. In this

model, aleatoric uncertainty is represented by the conditional noise variance

$$\mathrm{Var}(\varepsilon \mid X = x). \tag{76}$$

If this quantity is constant in x, the noise is called homoscedastic. If it changes with x, the noise is called heteroscedastic.

- **Homoscedastic noise**: the noise variance is the same across the input space, for example $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, where σ^2 is constant.
- **Heteroscedastic noise**: the noise variance depends on the input, for example $\varepsilon \mid X = x \sim \mathcal{N}(0, \sigma^2(x))$, where $\sigma^2(x)$ changes with x.

A simple example is depth estimation from images. Objects that are far away are often harder to localize precisely than objects close to the camera. The input x therefore affects the amount of measurement noise. More data can help estimate this noise more accurately, but it does not necessarily remove the noise itself (Hüllermeier & Waegeman, 2021).

Epistemic Uncertainty: The Reducible Ignorance Epistemic uncertainty describes uncertainty caused by limited knowledge of the model or its parameters. It can arise from finite training data, poor coverage of some regions of the feature space, or a model class that is too simple for the true relationship. Unlike aleatoric uncertainty, epistemic uncertainty can often be reduced by collecting more representative data or using a more suitable model.

In Bayesian methods, epistemic uncertainty is commonly represented by the posterior distribution $p(\theta \mid \mathcal{D})$, where θ denotes model parameters and $\mathcal{D}$ denotes the observed data. If the posterior is spread out, many parameter values remain plausible; if it is concentrated, the data strongly identify the parameters. Under regular parametric Bayesian assumptions, such as correct model specification, identifiability, and a fixed-dimensional parameter space, one may have posterior concentration of the form

$$p(\theta \mid \mathcal{D}_N) \xrightarrow[N\to\infty]{} \delta_{\theta^\star}, \tag{77}$$

where $\mathcal{D}_N$ is a dataset of size N, $\theta^\star$ is the true parameter value, and $\delta_{\theta^\star}$ is a point mass at $\theta^\star$. Equation (77) should be interpreted as an idealized statistical principle, not as a universal guarantee for modern deep neural networks (Hüllermeier & Waegeman, 2021).

Uncertainty types in terms of Parametric Variance Bounds In Bayesian predictive modeling, the total predictive variance of a future target Y^* at a new input x^* can be decomposed using the law of total variance (Depeweg et al., 2018):

$$\begin{aligned}\mathrm{Var}(Y^* \mid x^*, \mathcal{D}) = &\underbrace{\mathbb{E}_{\theta\sim p(\theta|\mathcal{D})}\left[\mathrm{Var}(Y^* \mid x^*, \theta)\right]}_{\text{Aleatoric Uncertainty}} \\ &+ \underbrace{\mathrm{Var}_{\theta\sim p(\theta|\mathcal{D})}\left[\mathbb{E}(Y^* \mid x^*, \theta)\right]}_{\text{Epistemic Uncertainty}}\end{aligned} \tag{78}$$

Here, Y^* is the unknown future target, x^* is the corresponding input, $\mathcal{D}$ is the training data, and θ is a random parameter drawn from the posterior distribution $p(\theta \mid \mathcal{D})$. The first term is often associated with aleatoric uncertainty because it averages the remaining output noise for fixed parameter values. The second term is often associated with epistemic uncertainty because it measures how much the prediction changes when plausible parameter values are varied.

This variance decomposition is theoretically useful, but a variance alone does not automatically provide a calibrated prediction interval. A common but restrictive practice is to assume approximately Gaussian errors and use an interval such as

$$\hat{y} \pm 1.96\hat{\sigma}, \tag{79}$$

where $\hat{y}$ is the point prediction and $\hat{\sigma}$ is an estimated predictive standard deviation. This interval is appropriate only under strong assumptions, such as approximate normality and a reliable variance estimate. In complex prediction tasks, errors may be skewed, heavy-tailed, or input-dependent. This motivates **distribution-free methods** that provide finite-sample coverage guarantees under **exchangeability**, without requiring a Gaussian error model.

2.4.2 Exchangeability

A finite sequence of random variables $Z_1, Z_2, \ldots, Z_n$, taking values in a measurable space $\mathcal{Z}$, is exchangeable if its joint distribution is invariant to permutations(Angelopoulos et al., 2024). Mathematically, for every permutation $\pi \in \mathrm{Perm}(n)$,

$$(Z_1, \ldots, Z_n) \overset{d}{=} (Z_{\pi(1)}, \ldots, Z_{\pi(n)}). \tag{80}$$

Here, $\mathrm{Perm}(n)$ is the set of all permutations of $\{1, \ldots, n\}$, and $\overset{d}{=}$ means equality in distribution. Intuitively, exchangeability means that the order of the observations does not carry statistical information about their joint distribution. Shuffling the observations changes their order, but not the probability law assigned to the sequence.

Independent and identically distributed (IID) observations are exchangeable (Angelopoulos et al., 2024). However, exchangeability is weaker than IID, exchangeable observations must be identically distributed, but they do not need to be mutually independent. This distinction matters because conformal prediction relies on exchangeability rather than full independence.

Exchangeability in Scene-Based Driving Datasets The finite-sample guarantees of conformal prediction require the calibration and test examples, or more precisely their nonconformity scores, to be exchangeable. Standard statistical pipelines often assume IID observations to satisfy this condition. For sequential driving data, this assumption is not automatically valid at the frame level.

Consecutive frames from the same driving sequence are generally not exchangeable because their order carries information about motion. For example, the state at time t is strongly related to the state at time $t-1$. Reversing or randomly permuting frames from one sequence would usually not produce a sequence with the same joint distribution.

To make the exchangeability assumption more plausible, this thesis uses scene-level splitting. Entire scenes or driving logs, rather than individual frames, are treated as the primary units when separating training, calibration, and test data. Frames within one scene may still be temporally correlated, but different scenes are not directly dependent. Thus, scene-level exchangeability is a working assumption: it is not automatically guaranteed, but it is more defensible than frame-level exchangeability.

In practice, this means that frames from the same scene should not be divided across training, calibration, and test splits. The analogy is grouping exam questions by topic: if several near-duplicate questions come from the same topic, placing some in training and some in testing would make the test look easier than a genuinely independent test. Scene-level splitting reduces this leakage and better matches the assumptions needed for conformal calibration.

2.4.3 Conformal Prediction

Traditional uncertainty quantification often relies on parametric assumptions, such as Gaussian error distributions. These assumptions can be inaccurate when errors are skewed, heavy-tailed, or strongly input-dependent. Conformal Prediction (CP) (Angelopoulos et al., 2024; Vovk et al., 2005) addresses this issue by avoiding a parametric model for the error distribution.

Rather than estimating the full conditional density $p(y \mid x)$, CP wraps around an existing predictor and converts its outputs into prediction sets. In regression, the prediction set is often an interval. The term *conformal* refers to the idea that a new observation should look similar, according to a chosen score, to previous

calibration observations. A useful analogy is quality control: instead of assuming a machine's errors are Gaussian, we measure errors on a calibration batch and choose a tolerance large enough to cover most future errors.

Let

$$\mathcal{D}_{\text{cal}} = \{(X_1, Y_1), \ldots, (X_n, Y_n)\} \tag{81}$$

be a hold-out calibration dataset of size n, where X_i is an input and Y_i is the corresponding target. Let X_{n+1} be a new test input with unknown target Y_{n+1}. The theoretical validity of CP relies on the exchangeability of

$$Z_i = (X_i, Y_i), \qquad i = 1, \ldots, n+1. \tag{82}$$

Under this assumption, CP can construct prediction sets with finite-sample marginal coverage without knowing the true data-generating distribution (Angelopoulos et al., 2024; Vovk et al., 2005).

Theoretical Guarantees of Marginal Coverage Given a target miscoverage level $\alpha \in (0, 1)$, CP constructs a prediction set $\widehat{C}(X_{n+1})$. For scalar regression, this set is often an interval $[L, U]$, where L and U are the lower and upper endpoints. The defining guarantee is (Angelopoulos et al., 2024)

$$\mathbb{P}\left\{Y_{n+1} \in \widehat{C}(X_{n+1})\right\} \geq 1 - \alpha. \tag{83}$$

The probability in Equation (83) is taken over the random calibration sample and the random test point. The guarantee is called *marginal* because it averages over all possible test inputs X_{n+1}.

For common conformal constructions with continuous nonconformity scores, the achieved coverage is also bounded above:

$$1 - \alpha \leq \mathbb{P}\left\{Y_{n+1} \in \widehat{C}(X_{n+1})\right\} \leq 1 - \alpha + \frac{1}{n+1}. \tag{84}$$

The term $1/(n+1)$ appears because the rank of the test score among the n calibration scores and the one test score can take only $n+1$ discrete values. The upper bound in Equation (84) controls the achieved coverage probability, not the length of the prediction interval. Short and informative intervals still require a useful underlying prediction model and a suitable score function.

The Limitation of Marginal Coverage The guarantee in Equation (83) does not imply exact conditional coverage for every individual input x. In general, CP does not guarantee

$$\mathbb{P}\left\{Y_{n+1} \in \widehat{C}(X_{n+1}) \mid X_{n+1} = x\right\} \geq 1 - \alpha \quad \text{for every } x. \tag{85}$$

This distinction is very important. Marginal coverage can be correct on average even if the errors are unevenly distributed across the feature space. For example, intervals may over-cover easy cases and under-cover difficult cases while still achieving the desired average coverage. Exact distribution-free conditional coverage for every x is impossible in general without additional assumptions, restrictions, or relaxations. Practical alternatives include approximate conditional coverage, group-conditional coverage, locally weighted conformal methods, or stronger modeling assumptions.

2.4.4 The Split Conformal Methodology

Full conformal prediction evaluates candidate labels y for a new test input X_{n+1} by appending (X_{n+1}, y) to the data and recomputing conformity scores. For many learning algorithms, this requires refitting the model for many candidate values of y. For large neural networks, this is computationally infeasible.

Split Conformal Prediction (SCP) (Lei et al., 2018) avoids this bottleneck by separating model training from calibration. The available data are divided into two disjoint subsets: a training set $\mathcal{D}_{\text{train}}$ and a calibration set $\mathcal{D}_{\text{cal}}$. The prediction model $\hat{f}$ is trained once on $\mathcal{D}_{\text{train}}$. Because $\mathcal{D}_{\text{cal}}$ is not used to fit $\hat{f}$, the calibration scores can be treated as exchangeable with the future test score, conditional on the trained model.

The Nonconformity Score and the Empirical CDF A nonconformity score measures how unusual or inaccurate or non-conformal a prediction is. Let

$$s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R} \tag{86}$$

be a score function, where $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the target space. For scalar regression, a common choice is the absolute residual

$$s(x, y) = |y - \hat{f}(x)|. \tag{87}$$

Here, x is an input, y is the observed target, and $\hat{f}(x)$ is the point prediction.

For a vector-valued trajectory, one may instead use a norm-based score,

$$s(x, y) = \|y - \hat{f}(x)\|_2, \tag{88}$$

where $\|\cdot\|_2$ is the Euclidean norm. Another option, used later in this thesis, is to calibrate separate scalar scores for each coordinate and prediction horizon.

For each calibration point (X_i, Y_i), define

$$s_i = s(X_i, Y_i), \qquad i = 1, \ldots, n. \tag{89}$$

The empirical cumulative distribution function (CDF) of these scores is

$$\widehat{F}_n(s) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(s_i \le s). \tag{90}$$

Here, $\mathbb{I}(\cdot)$ is the indicator function, which equals 1 if the condition inside is true and 0 otherwise. Equation (90) counts the fraction of calibration errors that are no larger than a candidate threshold s.

The Finite-Sample Threshold Correction The goal is to choose a threshold $\hat{r}$ such that future scores are below $\hat{r}$ with probability at least $1-\alpha$. The finite-sample correction comes from a rank argument. Under exchangeability, the rank of the test score s_{n+1} among

$$s_1, \ldots, s_n, s_{n+1} \tag{91}$$

is uniformly distributed over $\{1, \ldots, n+1\}$. Therefore, to guarantee marginal coverage at least $1-\alpha$, we use the rank

$$k = \lceil (n+1)(1-\alpha) \rceil, \tag{92}$$

where $\lceil \cdot \rceil$ is the ceiling function. Equivalently, the adjusted empirical probability level is

$$p_{\text{target}} = \frac{\lceil (n+1)(1-\alpha) \rceil}{n}. \tag{93}$$

Let $s_{(1)} \le \cdots \le s_{(n)}$ denote the ordered calibration scores. If $k \le n$, the split conformal threshold is

$$\hat{r} = s_{(k)}. \tag{94}$$

Equivalently, using the empirical CDF,

$$\hat{r} = \inf\left\{ s \in \mathbb{R} : \widehat{F}_n(s) \ge \frac{\lceil (n+1)(1-\alpha) \rceil}{n} \right\}. \tag{95}$$

If $k > n$, a finite threshold cannot be obtained from the calibration scores alone; one may set $\hat{r} = \infty$, or note that a finite split conformal threshold requires sufficiently large n relative to α.

Constructing the Set and the Homoscedastic Limitation With the calibrated threshold $\hat{r}$, the split conformal interval for a new scalar input X_{n+1} is

$$\widehat{C}(X_{n+1}) = \left[\hat{f}(X_{n+1}) - \hat{r},\ \hat{f}(X_{n+1}) + \hat{r} \right]. \tag{96}$$

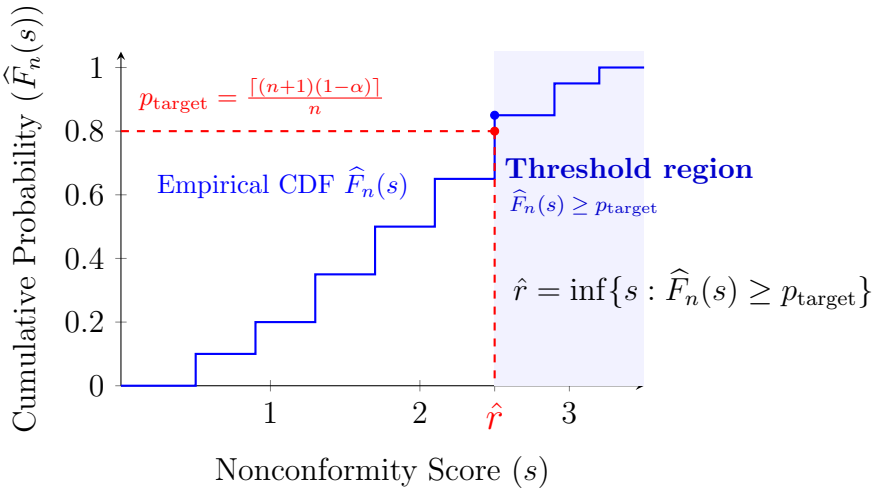

Figure 8: Calculation of the split conformal threshold $\hat{r}$. The threshold is the smallest score at which the empirical cumulative distribution function $\widehat{F}_n(s)$ reaches the finite-sample target probability p_{target}($p_{\text{target}} = 0.8$ here).

The true target Y_{n+1} lies inside this interval if and only if the test residual score s_{n+1} is no larger than $\hat{r}$. Therefore, under exchangeability, the interval in Equation (96) satisfies the marginal coverage guarantee in Equation (83).

The limitation is that standard SCP uses one global threshold $\hat{r}$. Therefore, the interval width $2\hat{r}$ is constant across all inputs. This is valid marginally, but it may be inefficient and locally miscalibrated in heteroscedastic problems. In easy regions of the input space, the interval may be unnecessarily wide; in difficult regions, it may be too narrow. This motivates combining conformal calibration with an interval model whose width can vary with the input (Romano et al., 2019).

2.4.5 Quantile Regression

Standard Split Conformal Prediction is a post-hoc calibration method that applies a global margin around a point predictor. To obtain intervals that adapt to local variability, we use Conditional Quantile Regression (QR)(Romano et al., 2019). Instead of estimating only the conditional mean, QR estimates boundaries of the conditional distribution given the data.

For a quantile level $\gamma \in (0, 1)$, the conditional γ-quantile of Y given $X = x$ is

$$Q_\gamma(x) = \inf \left\{ y \in \mathbb{R} : F_{Y|X=x}(y) \geq \gamma \right\}. \tag{97}$$

Here, $F_{Y|X=x}$ is the conditional CDF of Y given $X = x$. The value $Q_\gamma(x)$ is the smallest threshold below which at least a fraction γ of the conditional probability mass lies.

For a nominal $1-\alpha$ interval, one often estimates the equal-tailed lower and upper quantiles

$$\gamma_\ell = \frac{\alpha}{2}, \qquad \gamma_u = 1 - \frac{\alpha}{2}. \tag{98}$$

The learned quantile functions are denoted by $\widehat{Q}_\ell(x;\theta)$ and $\widehat{Q}_u(x;\theta)$, where θ denotes trainable model parameters. The resulting QR interval is

$$C_{\mathrm{QR}}(x;\theta) = \left[\widehat{Q}_\ell(x;\theta), \widehat{Q}_u(x;\theta)\right]. \tag{99}$$

These bounds are not necessarily symmetric around the mean or median. Their main advantage is that the interval width can vary with x. In driving prediction, this means that a familiar, nearly straight scene can receive a narrow interval, while a more ambiguous or rare scene can receive a wider interval.

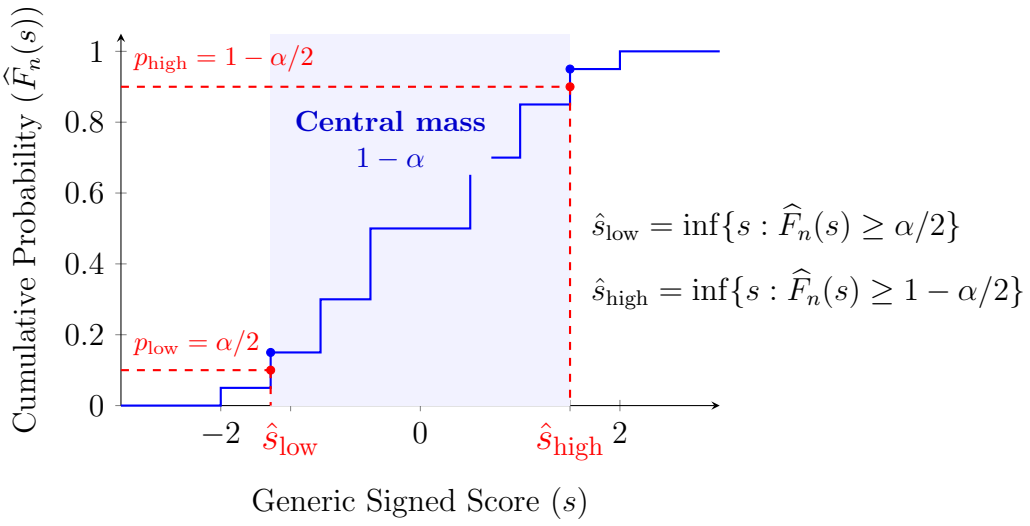

Figure 9: Illustration of lower and upper empirical quantiles for a generic signed scalar score. This figure explains equal-tailed quantiles. The standard CQR procedure used later applies one conformal correction to a learned lower–upper quantile interval.

Estimating Quantiles via the Pinball Loss The true conditional CDF $F_{Y|X=x}$ is unknown, so the quantile functions in Equation (97) cannot be computed directly. Instead, the model learns them from data using the pinball loss (Romano et al., 2019), also called the check loss. This loss is explained in detail in the next subsection.

The Theoretical Trade-off Quantile Regression and Split Conformal Prediction solve complementary problems. QR can learn intervals whose widths vary with the input, which is important in heteroscedastic settings. However, QR coverage is model-dependent and, in finite neural-network training, learned quantiles may be miscalibrated. SCP gives finite-sample marginal coverage under exchangeability, but the basic absolute-residual version produces intervals with a global width. Conformalized Quantile Regression combines these strengths: QR provides the input-dependent interval shape, and conformal calibration applies a finite-sample correction to recover marginal coverage.

2.4.6 The Pinball Loss: Asymmetric Optimization

To estimate conditional quantiles, we replace the symmetric regression objective by an asymmetric one. For quantile level $\gamma \in (0, 1)$, quantile regression can be written as the empirical risk minimization problem

$$\hat{\theta}_\gamma \in \arg\min_\theta \frac{1}{m} \sum_{i=1}^{m} \rho_\gamma \left(Y_i, f(X_i; \theta) \right) + \mathcal{R}(\theta). \tag{100}$$

Here, m is the number of training examples used for quantile regression, X_i is the input, Y_i is the target, $f(X_i; \theta)$ is the model prediction, θ denotes trainable parameters, ρ_γ is the pinball loss, and $\mathcal{R}(\theta)$ is a regularization term.

Mathematical Formulation For a true observation y and a predicted value $\hat{y}$, define the residual

$$e = y - \hat{y}. \tag{101}$$

The pinball loss (Romano et al., 2019) is

$$\rho_\gamma(y, \hat{y}) = \begin{cases} \gamma(y - \hat{y}), & \text{if } y - \hat{y} > 0, \\ (1 - \gamma)(\hat{y} - y), & \text{otherwise.} \end{cases} \tag{102}$$

The case $y - \hat{y} > 0$ means the prediction is too low. The alternative means the prediction is too high or exactly correct. In terms of the residual e, the same loss can be written compactly as

$$\rho_\gamma(e) = e \left(\gamma - \mathbb{I}(e < 0) \right). \tag{103}$$

Here, $\mathbb{I}(e < 0)$ is one if $e < 0$ and zero otherwise.

The derivative with respect to the prediction $\hat{y}$, away from $y = \hat{y}$, is

$$\frac{\partial \rho_\gamma(y, \hat{y})}{\partial \hat{y}} = \begin{cases} -\gamma, & \text{if } y > \hat{y}, \\ 1 - \gamma, & \text{if } y < \hat{y}. \end{cases} \tag{104}$$

At $y = \hat{y}$, one may use any subgradient between $-\gamma$ and $1 - \gamma$.

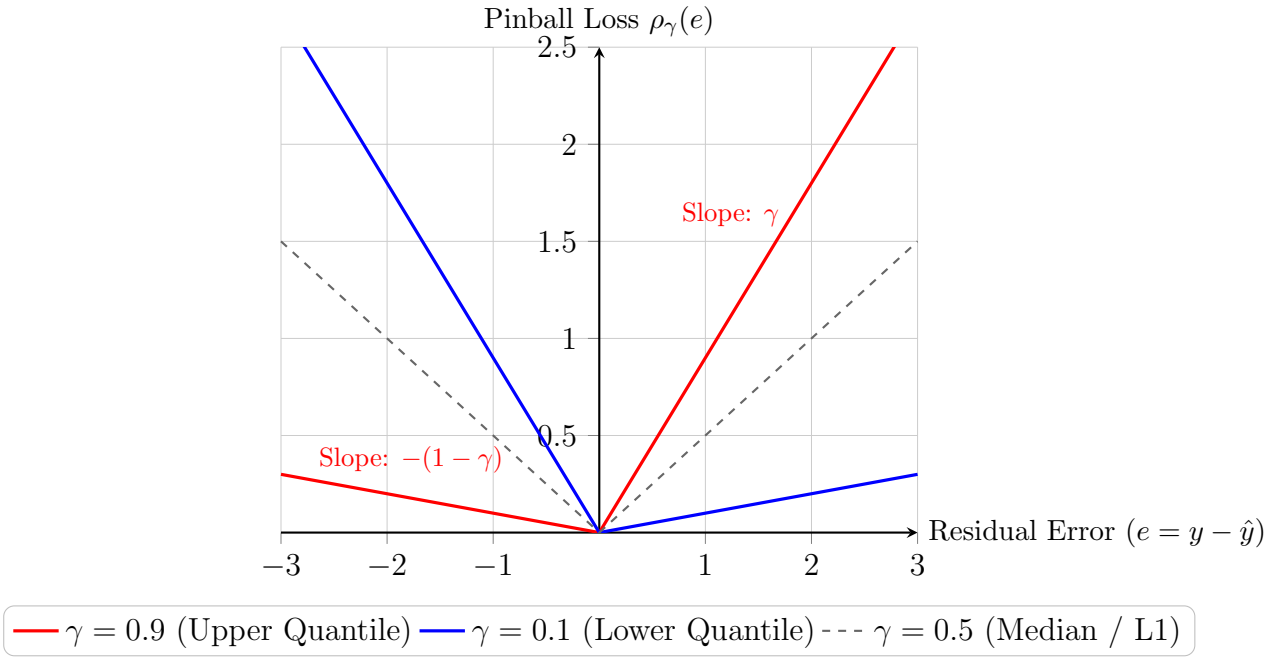

Figure 10: The optimization landscape of the pinball loss. A positive residual $e > 0$ means the prediction is too low, while a negative residual $e < 0$ means the prediction is too high. The slopes determine the gradients used during optimization.

The Mechanics of Asymmetric Penalties The key idea is that the pinball loss penalizes underprediction and overprediction differently (Romano et al., 2019). For $\gamma = 0.5$, the two slopes have equal magnitude, and the loss reduces to a scaled L_1 loss. Minimizing it estimates the conditional median (Romano et al., 2019).

For $\gamma = 0.9$, underprediction is penalized with slope 0.9, while overprediction is penalized with slope 0.1. The optimizer therefore moves the prediction upward until approximately 90% of the conditional observations lie below the predicted value and 10% lie above it. This is exactly the defining property of the conditional 0.9-quantile.

Robustness and Gradient Dynamics The piecewise linear shape in Figure 10 also makes the pinball loss less sensitive to extreme residuals than Mean Squared Error (MSE). For MSE,

$$\frac{\partial}{\partial \hat{y}}(y - \hat{y})^2 = -2(y - \hat{y}). \tag{105}$$

Thus, the MSE loss grows quadratically in $|y - \hat{y}|$, and the gradient magnitude grows linearly in $|y - \hat{y}|$. Large residuals can therefore dominate the update. In contrast, the pinball-loss gradient in Equation (104) is piecewise constant. Large residuals still matter, but their gradient magnitude does not increase without bound. This is useful when prediction errors are heavy-tailed or contain rare unusual cases.

2.4.7 Conformalized Quantile Regression (CQR)

Conformalized Quantile Regression (CQR) (Romano et al., 2019) combines the local adaptivity of Quantile Regression with the finite-sample marginal coverage guarantee of split conformal prediction. QR supplies the input-dependent interval shape, and conformal calibration corrects finite-sample miscalibration.

Phase I: Quantile Regression as the Heuristic Foundation First, a baseline QR model is trained on a training subset $\mathcal{I}_1$. For a desired miscoverage level α, the lower and upper quantile levels are

$$\gamma_\ell = \frac{\alpha}{2}, \qquad \gamma_u = 1 - \frac{\alpha}{2}. \tag{106}$$

The model estimates two conditional quantile functions just like standard QR,

$$\widehat{Q}_\ell(x; \hat{\theta}) \quad \text{and} \quad \widehat{Q}_u(x; \hat{\theta}), \tag{107}$$

where $\widehat{Q}_\ell$ is trained at level γ_ℓ, $\widehat{Q}_u$ is trained at level γ_u, and $\hat{\theta}$ denotes the fitted parameters. This stage gives a heteroscedastic interval whose width can change with the input. However, because the interval is learned by a finite neural network on finite data, it is not guaranteed to have exact $1 - \alpha$ coverage.

Phase II: Scoring the Boundaries The learned quantile interval is evaluated on a separate calibration subset $\mathcal{I}_2$. The main difference from standard split conformal prediction is the nonconformity score. Instead of measuring distance from a point prediction, CQR measures how far the observed target lies outside the learned interval:

$$E_i = \max\left\{\widehat{Q}_\ell(X_i; \hat{\theta}) - Y_i,\ Y_i - \widehat{Q}_u(X_i; \hat{\theta})\right\}, \qquad i \in \mathcal{I}_2. \tag{108}$$

Here, E_i is the CQR calibration score. If $E_i > 0$, then Y_i lies outside the learned interval. If $E_i \leq 0$, then Y_i lies inside the learned interval. Negative scores therefore indicate that the learned interval had extra slack.

Phase III: Adaptive Calibrated Inversion Let $n = |\mathcal{I}_2|$ be the number of calibration examples and define

$$k = \lceil (n+1)(1-\alpha) \rceil . \tag{109}$$

Let $E_{(1)} \leq \cdots \leq E_{(n)}$ be the ordered CQR scores. If $k \leq n$, the CQR correction is

$$\hat{\eta} = E_{(k)} = \text{Quantile}\left(E_1, \ldots, E_n; \frac{\lceil (n+1)(1-\alpha) \rceil}{n} \right). \tag{110}$$

Here, $\hat{\eta}$ is a scalar correction. It does not define the whole interval width by itself. Instead, it expands or shrinks the learned QR interval by a uniform amount. The final CQR interval is

$$\widehat{C}_{\text{CQR}}(X_{n+1}) = \left[\widehat{Q}_\ell(X_{n+1}; \hat{\theta}) - \hat{\eta},\ \widehat{Q}_u(X_{n+1}; \hat{\theta}) + \hat{\eta} \right]. \tag{111}$$

Under exchangeability of the calibration and test examples, this interval satisfies

$$\mathbb{P}\left\{ Y_{n+1} \in \widehat{C}_{\text{CQR}}(X_{n+1}) \right\} \geq 1 - \alpha. \tag{112}$$

Thus, CQR preserves the input-adaptive shape learned by QR while adding a finite-sample marginal coverage guarantee through conformal calibration.

2.4.8 CQR Model Architecture and Implementation inside UniAD

To use CQR inside the UniAD (Hu et al., 2023) trajectory prediction setting, we add a parallel uncertainty head. The primary planning head predicts a center trajectory, while the uncertainty head predicts input-dependent extents around that center. This decoupling is useful because the center trajectory and the uncertainty interval serve different purposes: the center gives the nominal prediction, while the extents describe how much error should be allowed around it.

The Parallel Uncertainty Network For prediction horizon $h \in \{1, \ldots, H\}$, the primary planning head outputs the center coordinate

$$(c_{x,h}, c_{y,h}) \in \mathbb{R}^2. \tag{113}$$

Here, $c_{x,h}$ and $c_{y,h}$ are the predicted lateral and longitudinal coordinates at horizon h. The uncertainty head outputs four non-negative extents,

$$(\Delta_{x,L,h}, \Delta_{x,R,h}, \Delta_{y,B,h}, \Delta_{y,F,h}) \in \mathbb{R}^4_{\geq 0}. \tag{114}$$

The symbols L, R, B, and F denote left, right, backward, and forward extents relative to the predicted center. Non-negativity can be enforced in implementation by a positive output activation such as softplus.

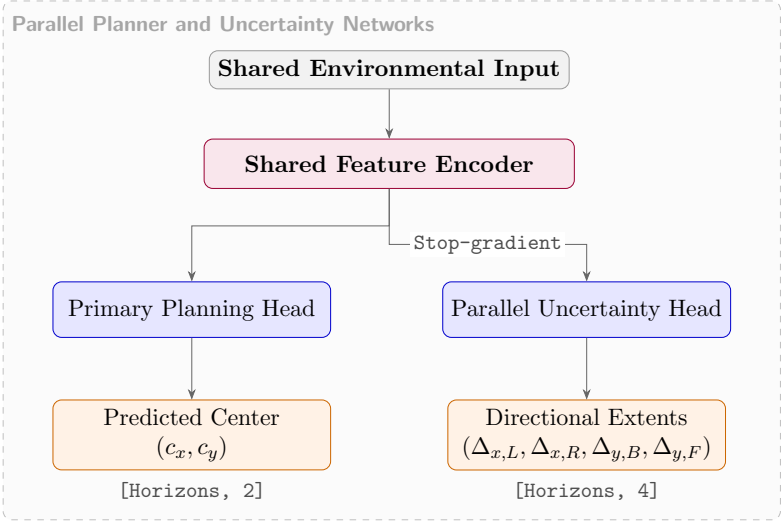

Figure 11: Dual-stream architecture used for trajectory prediction and uncertainty estimation. The primary head predicts the center trajectory, while the uncertainty head predicts non-negative directional extents around it. The stop-gradient prevents the uncertainty loss from updating the shared representation.

The resulting box is not forced to be symmetric. The model may predict a larger extent forward than backward, or a larger extent to one side than the other. A simple analogy is drawing an adjustable buffer around a planned path: the centerline gives the nominal path, while the four extents say how much room is needed in each direction.

Directional Pinball Loss and Gradient Isolation The primary planning head is trained using the standard trajectory loss described earlier. The uncertainty head is trained using a directional one-sided pinball loss (Romano et al., 2019). Rather than predicting absolute lower and upper coordinates directly, it predicts distances from the center trajectory.

For calibration or training sample i and horizon h, let

$$e_{x,i,h} = Y_{x,i,h} - c_{x,i,h}, \qquad e_{y,i,h} = Y_{y,i,h} - c_{y,i,h}. \tag{115}$$

Here, $Y_{x,i,h}$ and $Y_{y,i,h}$ are the ground-truth coordinates, and $c_{x,i,h}$, $c_{y,i,h}$ are the predicted center coordinates. The required non-negative distances to cover the ground truth in each direction are

$$\begin{aligned} d_{x,L,i,h} &= \max(-e_{x,i,h}, 0), & d_{x,R,i,h} &= \max(e_{x,i,h}, 0), \\ d_{y,B,i,h} &= \max(-e_{y,i,h}, 0), & d_{y,F,i,h} &= \max(e_{y,i,h}, 0). \end{aligned} \tag{116}$$

For example, if $e_{x,i,h} > 0$, the ground truth lies to the right of the predicted center, so $d_{x,R,i,h} = e_{x,i,h}$ and $d_{x,L,i,h} = 0$.

For a nominal two-sided coordinate interval with miscoverage α, each side should leave approximately $\alpha/2$ probability outside the interval. Therefore, we train each directional extent at the one-sided quantile level

$$\gamma = 1 - \frac{\alpha}{2}. \tag{117}$$

For $\alpha = 0.1$, this gives $\gamma = 0.95$. For a target distance $d \geq 0$ and predicted extent $\hat{d} \geq 0$, the directional pinball loss is

$$\mathcal{L}_\gamma(d, \hat{d}) = \max\left\{\gamma(d - \hat{d}),\ (\gamma - 1)(d - \hat{d})\right\}. \tag{118}$$

Since γ is strictly between 0 and 1, therefore the max function inherently selects the positive γe penalty for positive errors and the positive $(\gamma - 1)e$ penalty for negative errors, Thereby replicating the piecewise logic in Equation 102.

Here, underestimating the required distance is penalized more strongly than overestimating it. With $\gamma = 0.95$, underestimation receives weight 0.95, while overestimation receives weight 0.05.

The **uncertainty-head training loss** can be written as

$$\begin{aligned}\mathcal{L}_{\text{unc}} = \frac{1}{|\mathcal{I}_1|H} \sum_{i \in \mathcal{I}_1} \sum_{h=1}^{H} \Big[&\mathcal{L}_\gamma(d_{x,L,i,h}, \Delta_{x,L,i,h}) + \mathcal{L}_\gamma(d_{x,R,i,h}, \Delta_{x,R,i,h}) \\ &+ \mathcal{L}_\gamma(d_{y,B,i,h}, \Delta_{y,B,i,h}) + \mathcal{L}_\gamma(d_{y,F,i,h}, \Delta_{y,F,i,h}) \Big].\end{aligned} \tag{119}$$

Here, $\mathcal{I}_1$ is the training index set for the uncertainty head and H is the number of prediction horizons.

Without gradient isolation, the uncertainty loss could update the center trajectory in a way that reduces interval loss but harms the accuracy of the nominal trajectory. To avoid this interaction, the center prediction is detached when constructing the residuals in Equation (115). In addition, gradients from the uncertainty loss are stopped before reaching the shared encoder, as shown in Figure 11. Thus, the uncertainty objective updates the uncertainty head, while the primary planning objective remains responsible for the center trajectory and shared representation.

Per-Axis Conformal Calibration After training the uncertainty head, **conformal calibration** (Romano et al., 2019) is applied separately for each axis and horizon. Let $\mathcal{I}_{\text{cal},h}$ be the set of valid calibration indices at horizon h, and let $n_h = |\mathcal{I}_{\text{cal},h}|$. For calibration sample i and horizon h, define the x-axis CQR score

$$E_{x,i,h} = \max\left\{(c_{x,i,h} - \Delta_{x,L,i,h}) - Y_{x,i,h},\ Y_{x,i,h} - (c_{x,i,h} + \Delta_{x,R,i,h})\right\}, \tag{120}$$

and the y-axis CQR score

$$E_{y,i,h} = \max\left\{(c_{y,i,h} - \Delta_{y,B,i,h}) - Y_{y,i,h},\ Y_{y,i,h} - (c_{y,i,h} + \Delta_{y,F,i,h})\right\}. \tag{121}$$

A positive score means that the ground truth lies outside the learned interval for that axis. A negative score means that the ground truth lies inside the learned interval. Negative scores are useful because they allow conformal calibration to shrink intervals if the learned QR intervals are overly conservative (Romano et al., 2019).

For each horizon h, define

$$k_h = \lceil (n_h + 1)(1 - \alpha) \rceil. \tag{122}$$

The conformal corrections are then

$$\hat{\eta}_{x,h} = \text{Quantile}\left(\{E_{x,i,h} : i \in \mathcal{I}_{\text{cal},h}\}; \frac{k_h}{n_h}\right), \tag{123}$$

$$\hat{\eta}_{y,h} = \text{Quantile}\left(\{E_{y,i,h} : i \in \mathcal{I}_{\text{cal},h}\}; \frac{k_h}{n_h}\right). \tag{124}$$

Here, $\hat{\eta}_{x,h}$ and $\hat{\eta}_{y,h}$ are fixed calibration constants after calibration. They are not recomputed for each new test input. During inference, the neural network produces input-dependent extents, and these fixed conformal corrections are applied to them.

For a new input X, the final calibrated x-axis interval at horizon h is

$$C_{x,h}(X) = \left[(c_{x,h} - \Delta_{x,L,h}) - \hat{\eta}_{x,h},\ (c_{x,h} + \Delta_{x,R,h}) + \hat{\eta}_{x,h}\right], \tag{125}$$

and the final calibrated y-axis interval is

$$C_{y,h}(X) = \left[(c_{y,h} - \Delta_{y,B,h}) - \hat{\eta}_{y,h},\ (c_{y,h} + \Delta_{y,F,h}) + \hat{\eta}_{y,h}\right]. \tag{126}$$

Here, $c_{x,h}$, $c_{y,h}$, and the four Δ-terms are produced by the neural network for the new input X, while $\hat{\eta}_{x,h}$ and $\hat{\eta}_{y,h}$ are obtained from the calibration set.

Marginal versus Joint Coverage This formulation provides marginal coverage separately for each axis and horizon. For a target miscoverage level α, the conformal guarantees are

$$\mathbb{P}\left\{Y_{x,n+1,h} \in C_{x,h}(X_{n+1})\right\} \geq 1 - \alpha, \tag{127}$$

$$\mathbb{P}\left\{Y_{y,n+1,h} \in C_{y,h}(X_{n+1})\right\} \geq 1 - \alpha. \tag{128}$$

Here, $Y_{x,n+1,h}$ and $Y_{y,n+1,h}$ are the true future coordinates of the test example at horizon h. For $\alpha = 0.1$, Equations (127) and (128) state that each coordinate

is covered with probability at least 0.90, marginally over the calibration and test distribution.

These statements do not imply 90% joint coverage of the full two-dimensional rectangle

$$C_{x,h}(X_{n+1}) \times C_{y,h}(X_{n+1}). \tag{129}$$

Let

$$A_{x,h} = \{Y_{x,n+1,h} \in C_{x,h}(X_{n+1})\}, \qquad A_{y,h} = \{Y_{y,n+1,h} \in C_{y,h}(X_{n+1})\}. \tag{130}$$

If $A_{x,h}$ and $A_{y,h}$ were independent, then

$$\mathbb{P}(A_{x,h} \cap A_{y,h}) = \mathbb{P}(A_{x,h})\mathbb{P}(A_{y,h}) \approx (1-\alpha)^2. \tag{131}$$

For $\alpha = 0.1$, this gives approximately $0.9^2 = 0.81$. However, independence between lateral and longitudinal errors is not guaranteed. Without independence, the union bound gives

$$\mathbb{P}(A_{x,h} \cap A_{y,h}) \geq 1 - \alpha_x - \alpha_y, \tag{132}$$

where α_x and α_y are the marginal miscoverage levels used for the two axes. If both are 0.1, this lower bound is 0.8.

Thus, the intervals in this thesis should be interpreted as per-axis marginally calibrated intervals, not as 90% joint two-dimensional prediction boxes. If a 90% joint rectangle were required, one could calibrate each axis at 95% and use the union bound, or define a single multivariate nonconformity score and calibrate the full two-dimensional prediction region directly. The per-axis approach is still useful because it gives dimension-specific uncertainty information: lateral and longitudinal errors can have different scales and different physical causes.

3 Results

3.1 Results: Fine-Tuning the Latent Representation and Object Detection

Before analyzing detection metrics, we first examine the optimization behaviour of the representation fine-tuning stage. TruckScenes (Fent et al., 2024) consists of 747 sequential driving scenes, out of which 90 percent were used for training and 10 percent were set as validation set for this experiment. Figure 12 shows the training and validation loss curves over the 18-epoch adaptation period. During this stage, the downstream planning module is frozen, and only the representation parameters are updated using the representation objective

$$L_{\text{stage1}}(\phi) = L_{\text{representation}}(\phi; \mathcal{D}_T). \tag{133}$$

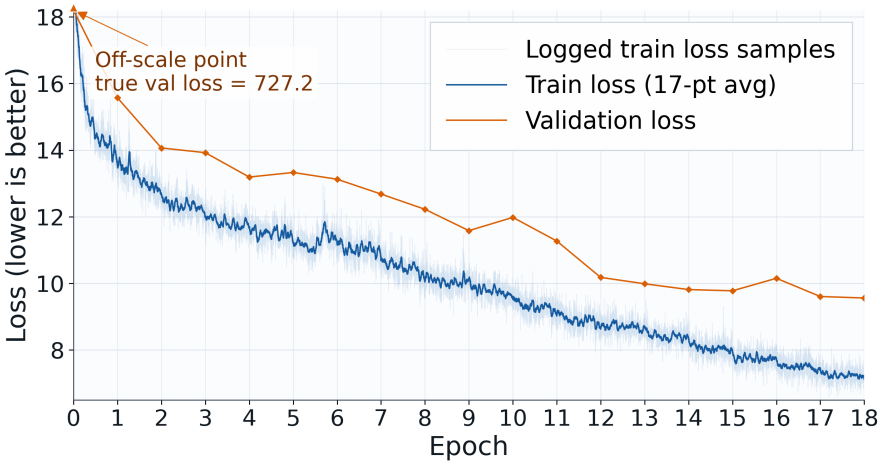

Figure 12: Latent representation training and validation loss during 18 epochs of fine-tuning on TruckScenes. The x-axis denotes epoch and the y-axis denotes the representation loss. The faint blue curve shows raw training loss, the dark blue curve shows a 17-point smoothed training average, and the orange curve shows validation loss. The initial validation loss at epoch 0 is off-scale and is annotated separately. The parallel decrease of training and validation losses indicates stable target-domain optimization with no clear train–validation divergence.

Here, ϕ denotes the trainable parameters of the latent representation module, $\mathcal{D}_T$ denotes the TruckScenes target-domain training data, $L_{\text{representation}}$ is the representation loss, and L_{stage1} is the objective optimized during this first adaptation stage. Freezing the downstream planning module isolates the representation-learning problem, the model is encouraged to adapt its internal scene representation to the target-domain sensor geometry before the final trajectory module is updated.

The curves in Figure 12 suggest a stable adaptation process. The validation loss follows the same overall decreasing trend as the training loss and does not show a clear divergence. This is important because the target domain differs from the source domain in camera topology, mounting geometry, and scene distribution. If the representation update were unstable, one would expect large oscillations or a widening gap between training and validation loss. The observed behaviour, therefore, indicates that the representation module adapted to the target-domain inputs without obvious optimization collapse. However, the loss curve alone does not prove that the representation thus learned is good; that require validation metrics, qualitative checks, or source-domain evaluation after fine-tuning.

A well-behaved loss curve is still only an indirect indicator. The latent repre-

sentation is a high-dimensional tensor, so its geometric quality (the locations of surrounding actors and objects) cannot be inspected directly. We therefore evaluate it through a downstream proxy of object detection. The intuition is similar to checking a hidden map by asking whether important landmarks appear in the correct places. If the latent representation fails to encode useful spatial and semantic information, the detection head is unlikely to produce accurate object boxes. Conversely, improved detection performance provides evidence that the adapted latent space contains more useful target-domain information. This should be interpreted as proxy evidence, not as definitive proof that every aspect of the latent geometry is correct.

To quantify this proxy task, we report two detection metrics on the TruckScenes validation set: mean Average Precision (mAP) and the nuScenes Detection Score (NDS) (Caesar et al., 2020). Under the nuScenes-style 3D detection protocol, mAP measures detection and classification performance using center-distance matching between predicted and ground-truth boxes around surrounding actors and other map elements. NDS is broader: it combines mAP with true-positive errors for translation, scale, orientation, velocity, and object attributes. These auxillary tasks such as predicting velocity, orientation, size etc. of detected objects is also done by the detector. The auxillary tasks are absolutely essential for driving because these attributes affect driving behavior directly. Thus, mAP mainly measures whether objects are detected and classified correctly, while NDS additionally reflects the geometric and kinematic quality of the predicted 3D boxes. These metrics are useful perception-quality proxies, but they should not be interpreted as direct guarantees of downstream driving safety.

Table 2: Object detection performance on the TruckScenes validation setting. Higher values are better for both mAP and NDS.

Model / paradigm	Training or adaptation regime	mAP ↑	NDS ↑
UniAD, zero-shot	Without fine-tuning	0.0009	0.0505
PETR (Liu et al., 2022)	Trained for 48 epochs	0.0200	0.1200
TruckScenes baseline (Fent et al., 2024)			
UniAD, this work	Fine-tuned for 18 epochs	**0.1100**	**0.2200**

Table 2 summarizes the quantitative effect of representation fine-tuning. The low performance of the zero-shot model isn't just an engineering failure. It mathematically demonstrates a severe divergence between the source and target marginal

distributions, $P_S(Z)$ and $P_T(Z)$. In particular, the representation learned from the source-domain camera geometry and scene distribution does not transfer directly to the TruckScenes setting. This is consistent with the domain-gap analysis above. Changes in camera topology, camera mounting, field of view, and driving environment can all alter the latent features received by the detection and planning heads.

After 18 epochs of representation fine-tuning relative to the zero-shot UniAD baseline, the performance gap is

$$\frac{\mathrm{mAP}_{\mathrm{UniAD,ft}}}{\mathrm{mAP}_{\mathrm{UniAD,zs}}} = \frac{0.1100}{0.0009} \approx 122.2, \tag{134}$$

$$\frac{\mathrm{NDS}_{\mathrm{UniAD,ft}}}{\mathrm{NDS}_{\mathrm{UniAD,zs}}} = \frac{0.2200}{0.0505} \approx 4.36. \tag{135}$$

Here, the subscript ft denotes the fine-tuned model, and the subscript zs denotes the zero-shot model. mAP metric improves by roughly two orders of magnitude, while NDS improves by more than a factor of four. Together with the loss curves in Figure 12, these validation results suggest that the representation fine-tuning successfully adapted the model to the target-domain camera geometry and feature statistics. The result should be interpreted as strong empirical evidence of target-domain adaptation.

The comparison with the TruckScenes PETR(Fent et al., 2024; Liu et al., 2022) camera baseline further supports the usefulness of transfer learning. The PETR baseline, trained for 48 epochs, achieves 0.0200 mAP and 0.1200 NDS, whereas the fine-tuned UniAD model achieves 0.1100 mAP and 0.2200 NDS after 18 epochs of representation adaptation. The relative improvements are

$$\frac{\mathrm{mAP}_{\mathrm{UniAD,ft}}}{\mathrm{mAP}_{\mathrm{PETR}}} = \frac{0.1100}{0.0200} = 5.5, \tag{136}$$

$$\frac{\mathrm{NDS}_{\mathrm{UniAD,ft}}}{\mathrm{NDS}_{\mathrm{PETR}}} = \frac{0.2200}{0.1200} \approx 1.83. \tag{137}$$

This comparison should not be described as a formal statistical significance result, since no confidence intervals or repeated training runs are reported. Instead, it provides comparative evidence that initializing from a pre-trained UniAD representation is more data-efficient than relying only on target-domain training. From a Bayesian perspective, transferring the pre-trained source weights $\hat{\omega}_S$ acts as a highly informative prior distribution $p(\phi)$. Training PETR from scratch assumes a non-informative prior, which requires an immense amount of target data to narrow the posterior distribution. Fine-tuning simply updates the prior to the posterior $p(\phi|\mathcal{D}_T)$, requiring far fewer samples to converge on a low-error bound because the model is only searching a localized sub-space for target-specific geometric transformations.

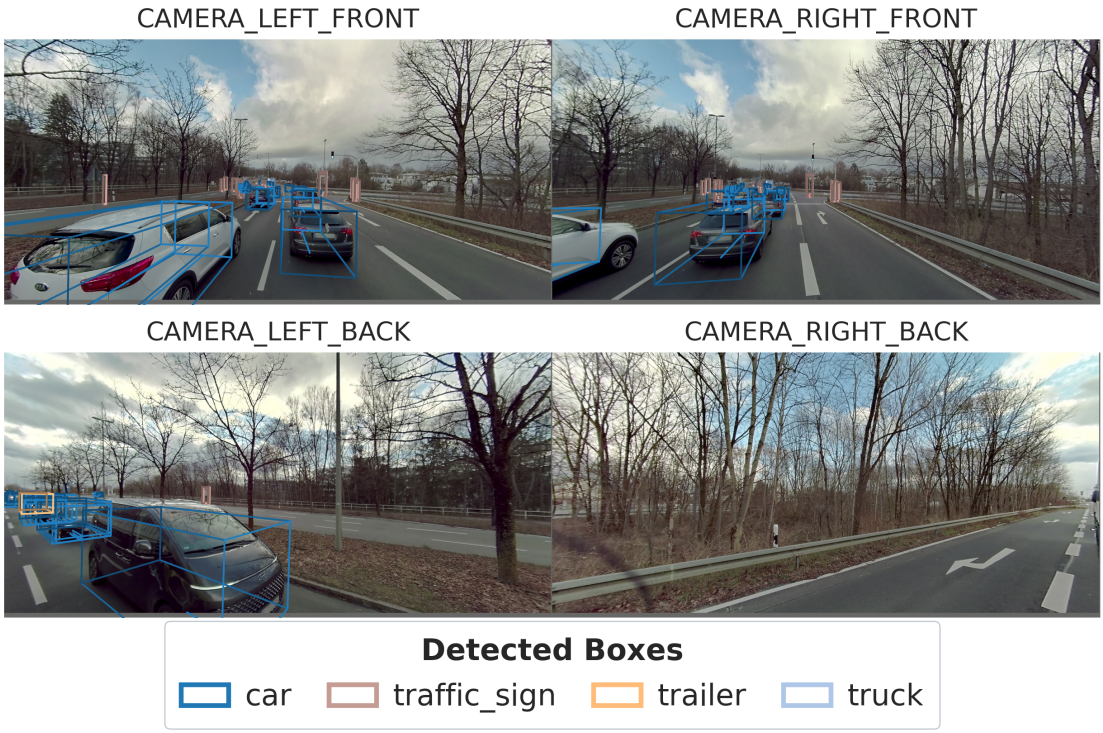

Figure 13: Qualitative 4-camera detection visualization in a clear-weather multi-lane highway scene. Projected 3D detections are shown in the camera views. Blue boxes indicate dynamic actors such as vehicles, while brownish-red boxes indicate static traffic signs. The example illustrates that the fine-tuned model produces spatially coherent detections in dense traffic with overlapping vehicles.

The qualitative visualizations in Figures 13 and 14 support the quantitative results. In the clear-weather highway example, the model produces plausible detections for several nearby and partially overlapping vehicles, suggesting that the adapted representation can fuse information across the four camera views. In the rainy example, the model still produces plausible detections for vehicles and traffic signs despite degraded visibility. These examples do not establish robustness for all dense-traffic or rainy scenarios, but they provide useful visual evidence that the validation improvements in Table 2 correspond to coherent perception outputs. Overall, the stable loss curves, improved validation metrics, and qualitative examples indicate that the representation fine-tuning stage produced a latent representation better aligned with the TruckScenes target domain.

Figure 14: Qualitative 4-camera detection visualization under rainy, low-visibility conditions. Detections with confidence greater than 0.25 are projected into the camera views. Blue boxes indicate dynamic actors, while brownish-red boxes indicate static traffic signs. The detections remain geometrically plausible despite visual degradation from rain, wet roads, and reduced visibility. This figure provides qualitative evidence of robustness in this example, but not a general robustness guarantee across all adverse weather conditions.

3.2 Results for UniAD Motion Prediction/Planning

After adapting the latent representation to the TruckScenes sensor geometry, the second fine-tuning stage updates the full UniAD pipeline with end-to-end (full model) training. In contrast to the representation-only stage, where the downstream planning components are frozen, this stage allows both the representation and the final trajectory prediction modules to adapt jointly to the target domain. The objective optimized in this stage can be written as

$$L_{\text{stage2}}(\phi, \theta) = L_{\text{total}}(\phi, \theta; D_T),$$

where D_T denotes the TruckScenes target-domain data, ϕ denotes the representation parameters, and θ denotes the downstream prediction and planning parameters. The purpose of this stage is not only to make the visual representation

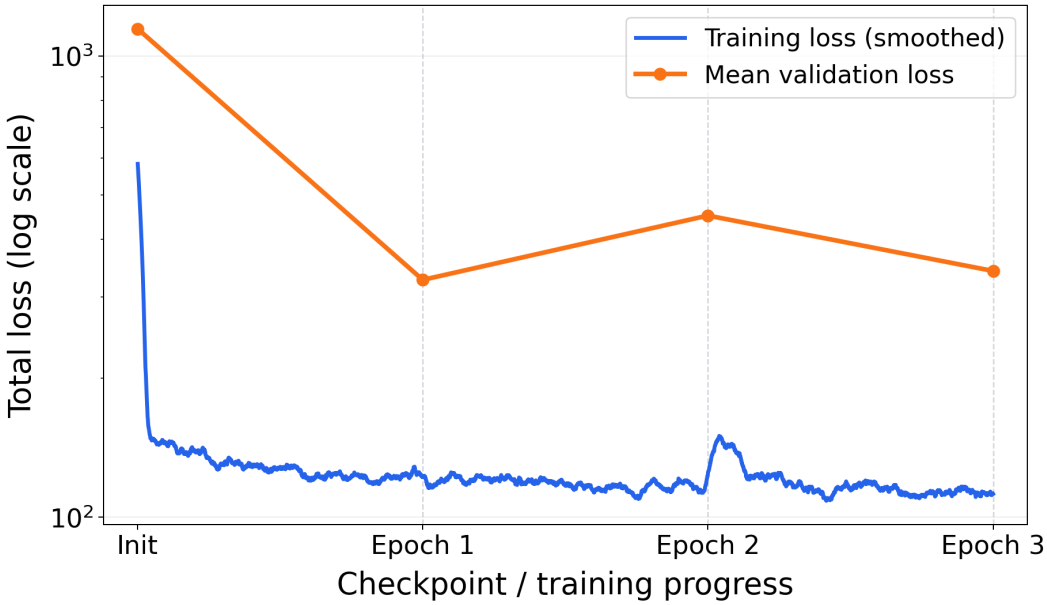

Figure 15: End-to-end UniAD fine-tuning loss on TruckScenes. The horizontal axis shows training progress over epochs, and the vertical axis shows the total loss on a logarithmic scale. The legend box identifies the blue curve as the training loss and the orange curve as the validation loss; the grey dashed vertical lines mark epoch boundaries. The figure shows a rapid initial decrease followed by an early plateau, with validation loss no longer clearly improving by the third epoch. The main takeaway is that end-to-end fine-tuning reached diminishing returns quickly under the available training budget.

compatible with the truck domain, but also to adapt the final trajectory output to the speed, road layout, and motion patterns observed in TruckScenes.

Figure 15 shows the training and validation losses during the end-to-end fine-tuning run. The training loss decreases very rapidly at the beginning of optimization and then enters a much narrower range. The average validation loss also drops strongly during the first part of training, but by the third epoch it no longer shows a clear improvement. Thus, the available run suggests that the end-to-end objective reached an early plateau after only three epochs. Because this stage was computationally expensive and the remaining project time was limited, training was not extended beyond this point. This stopping decision should therefore be interpreted as a practical constraint rather than as evidence that the model had reached a globally optimal solution. Longer training, additional learning-rate tuning, or repeated runs are likely to still improve the final result, but they were outside the available time allowed for the thesis.

Table 3: Mean L2 distance trajectory error before and after end-to-end fine-tuning. The zero-shot model is the nuScenes-trained UniAD model evaluated directly on TruckScenes, while the fine-tuned model is adapted end-to-end on TruckScenes. Lower values are better.

Horizon	Zero-shot UniAD [m]	Fine-tuned UniAD [m]	Relative reduction
1 s	5.440	2.456	54.9%
2 s	10.344	5.278	49.0%
3 s	14.686	7.245	50.7%
Simple mean	10.157	4.993	50.8%

To evaluate the effect of fine-tuning directly on trajectory prediction, we report the L2 distance between the predicted and ground-truth ego trajectory at the 1 s, 2 s, and 3 s horizons separately. For horizon h, the metric is

$$d_h = \frac{1}{N_h} \sum_{i=1}^{N_h} \left\| \hat{P}_{i,h} - P_{i,h} \right\|_2 ,$$

where $P_{i,h}$ is the ground-truth waypoint for sample i at horizon h, $\hat{P}_{i,h}$ is the corresponding predicted waypoint, and N_h is the number of valid evaluation samples at that horizon. Lower values indicate better trajectory accuracy.

Table 3 shows a clear improvement after end-to-end adaptation. Assuming isotropic Gaussian noise, minimizing the L_2 distance is equivalent to Maximum Likelihood Estimation. Therefore, halving this error demonstrates that the model's conditional expectation, $\mathbb{E}[Y|X]$, has successfully adapted from passenger-car dynamics to the heavy-duty physics of the target domain. This improvement is important because the planning task is affected by both perception-domain and motion-domain shifts.

The qualitative examples in Figures 16 and 17 provide additional context for the numerical improvement. In both examples, the fine-tuned model retains a useful road-driving prior. Its predicted trajectories remain smooth, road-aligned, and physically plausible. This is important because end-to-end fine-tuning can sometimes damage previously learned structure if the target-domain data is limited. Here, the qualitative outputs suggest that the model does not forget the basic geometry of road driving during adaptation.

Figure 16 shows an easy trajectory driving example. The zero-shot model predicts a trajectory with too little forward progress, which is consistent with a lower-speed source-domain prior learned from nuScenes. The direction of travel is not completely unreasonable, but the predicted trajectory is too short compared with the ground truth. After fine-tuning, the prediction is much closer to

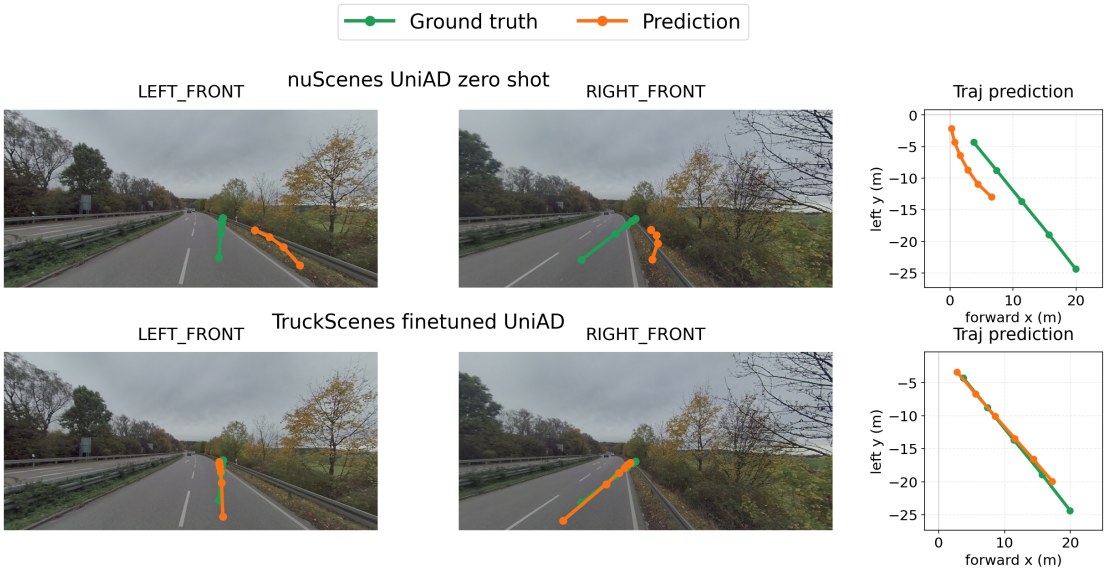

Figure 16: Unchallenging trajectory example after end-to-end fine-tuning. The camera panels show the LEFT_FRONT and RIGHT_FRONT views. The top row shows the nuScenes-trained UniAD model evaluated zero-shot on TruckScenes, and the bottom row shows the TruckScenes fine-tuned UniAD model. In the trajectory plots on the right, the horizontal axis is forward displacement x in meters and the vertical axis is lateral displacement y to the left in meters. The legend identifies the green curve as the ground truth and the orange curve as the prediction. The small grey boxed label in the camera view gives the result index. The zero-shot model predicts a low-speed trajectory with insufficient forward progress, while the fine-tuned model better matches the ground-truth speed and direction. The main takeaway is that end-to-end fine-tuning corrects much of the source-domain speed bias in simple road-following scenes.

the ground-truth trajectory. This indicates that the fine-tuned model successfully aligned its conditional expectation of longitudinal kinematics with the target distribution, $P_T(Y|X)$, while preserving the structural geometric priors that constrain the trajectory to a smooth, physically valid manifold..

Figure 17 shows a more challenging scenario. In this case, the ground-truth trajectory changes lane, likely because the human driver must have responded to the vehicles up ahead in the original lane. The fine-tuned model does not reproduce this lane-change maneuver and instead continues along a lane-following trajectory. This is a meaningful failure mode. The model misses the maneuver decision, however, the prediction is still not an absurd or physically implausible trajectory. It remains smooth, forward-moving, and compatible with the road

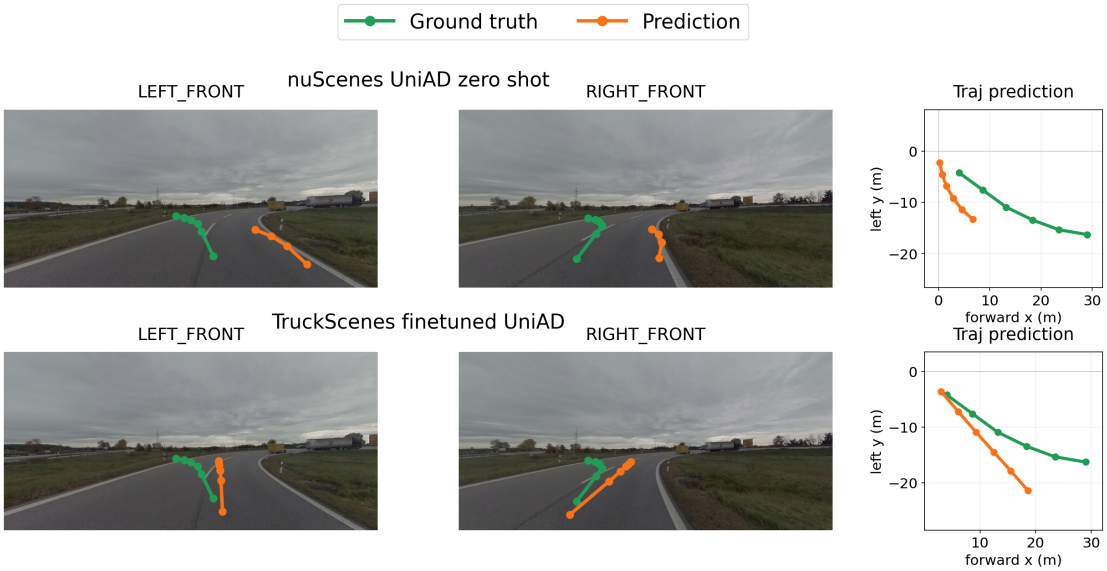

Figure 17: Challenging trajectory example after end-to-end fine-tuning. The camera panels show LEFT_FRONT and RIGHT_FRONT views for the zero-shot model in the top row and the TruckScenes fine-tuned model in the bottom row. In the right-hand trajectory plots, the horizontal axis is forward displacement x in meters and the vertical axis is leftward lateral displacement y in meters. The legend identifies the green curve as the ground truth and the orange curve as the prediction. The small grey boxed label in the camera view gives the result index. The ground truth performs a lane-change-like maneuver, while the fine-tuned prediction remains in a lane-following mode. The main takeaway is that fine-tuning preserves plausible road-driving behaviour, but the model can still miss more interaction-dependent maneuvers.

layout. The error is therefore better interpreted as a missed interaction-dependent maneuver rather than a complete breakdown of road-scene understanding. Also the prediction is much closer to Ground truth than the zero shot predicted trajectory.

In conclusion, the end-to-end fine-tuning results demonstrate a robust positive transfer, successfully mitigating the distributional shift between the passenger-car source and the heavy-duty target domain. The consistent reduction in empirical risk across all temporal horizons indicates that the optimization did not merely overfit to immediate, short-term state transitions, but effectively aligned the full joint predictive distribution, $P_T(Y^{traj}_{t+1:t+H}|X_{1:t})$, with the target domain's empirical measure. Qualitatively, the model achieves this alignment without catastrophic forgetting of the structural priors learned from the source data. It successfully retains the manifold of physically valid, road-aligned trajectories while shifting its

conditional expectation to accurately reflect heavy-truck kinematics. Together, these metrics and visual outputs empirically validate the two-stage transfer learning framework thereby demonstrating that the pre-trained weights functioned as a highly informative prior that enabled efficient adaptation. While the residual variance indicates that the target distribution is not yet perfectly approximated, extended training is theoretically expected to yield an even tighter convergence to the true target distribution.

3.3 Results for Conformalized Quantile Regression

The main purpose of the CQR experiments is to evaluate whether the learned uncertainty intervals are both statistically calibrated and meaningful for sequential trajectory forecasting. We evaluate three properties, i.e., empirical marginal coverage, growth of uncertainty across the prediction horizon, and input-adaptivity of interval width across simple and complex scenes (Heteroscedasticity).

In a time-series prediction problem, uncertainty is expected to grow as the forecast moves further away from the observed context. Near-term predictions are strongly conditioned on the most recent observations, while later predictions depend on more unobserved future factors, such as acceleration, braking, steering changes, and interactions with surrounding actors. Therefore, even when all future waypoints are predicted directly rather than recursively, the conditional distribution of the future position is expected to become wider as the prediction horizon increases.

A useful uncertainty model should also be data-dependent. In statistical terms, the conditional spread of the response distribution $Y \mid X$ is generally not constant across all inputs. Here, X denotes the observed input context and Y denotes the future trajectory response. Some inputs correspond to familiar and low-variance outcomes, while others correspond to rare, ambiguous, or noisy outcomes. Therefore, a good interval model should be heteroscedastic, i.e., it should produce narrow intervals for predictable samples and wider intervals for samples where the future is harder to infer. This is important because a single global uncertainty margin would ignore the fact that different regions of the input space can have very different levels of predictive difficulty.

Due to limited time, we choose to do Uncertainty Quantification (UQ) on the nuScenes trained UniAD model only and not on TruckScenes adapted UniAD model. If we intend to do UQ on a TruckScenes finetuned model as future work, we can copy the full implementation of CQR model in UniAD as is, because the model architecture remains the same.

Throughout the trajectory plots in this subsection, x denotes lateral position

and y denotes longitudinal position in the ego-vehicle coordinate frame. Thus, the horizontal axis in the figures shows longitudinal position y in meters, while the vertical axis shows lateral position x in meters. This convention is stated explicitly because many mathematical coordinate systems use x as the horizontal axis; here, the notation follows the coordinate convention used in the trajectory output.

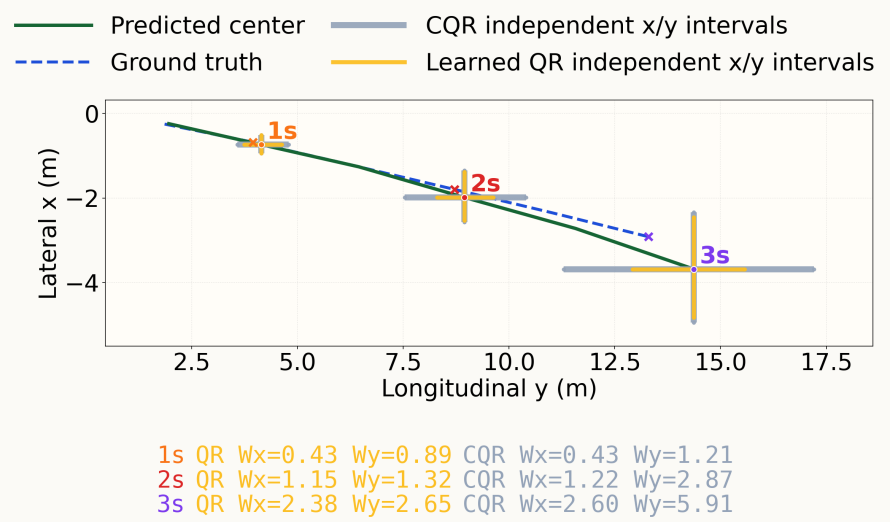

Figure 18: Separate per-axis interval visualization. The horizontal axis shows longitudinal position y in meters, and the vertical axis shows lateral position x in meters. The solid green curve is the predicted trajectory center, while the dashed blue curve is the ground-truth trajectory. Grey line segments show the CQR intervals separately along the x- and y-axes, and light orange segments show the learned QR intervals before conformal calibration. The orange, red, and purple markers correspond to the 1 s, 2 s, and 3 s prediction horizons.

It is important to interpret Figure 18 as a separate per-axis uncertainty visualization. In these experiments, not only the QR intervals are predicted separately for the lateral and longitudinal coordinates, but, conformal calibration is also applied separately for the two axes as well. Therefore, for a nominal coverage level of 90%, the guarantee is marginal per axis:

$$\mathbb{P}\left(Y_{x,h} \in C_{x,h}(X)\right) \geq 0.90, \qquad \mathbb{P}\left(Y_{y,h} \in C_{y,h}(X)\right) \geq 0.90. \tag{138}$$

Here, $h \in \{1, 2, 3\}$ denotes the prediction horizon in seconds, X is the observed input, $Y_{x,h}$ and $Y_{y,h}$ are the true lateral and longitudinal coordinates at horizon h, and $C_{x,h}(X)$ and $C_{y,h}(X)$ are the calibrated CQR intervals for those coordinates.

Equation (138) means that the lateral and longitudinal coordinates are each calibrated to achieve approximately 90% marginal coverage. It does not mean

that the complete two-dimensional waypoint is guaranteed to lie inside a joint x-y region with 90% probability. If

$$A_{x,h} = \{Y_{x,h} \in C_{x,h}(X)\}, \qquad A_{y,h} = \{Y_{y,h} \in C_{y,h}(X)\}, \tag{139}$$

then the event that the full two-dimensional waypoint is covered at horizon h is $A_{x,h} \cap A_{y,h}$. The per-axis guarantees in Equation (138) do not imply

$$\mathbb{P}(A_{x,h} \cap A_{y,h}) \geq 0.90. \tag{140}$$

If $A_{x,h}$ and $A_{y,h}$ were independent, the joint coverage would be approximately $0.9^2 = 0.81$. Without independence, the union bound only gives

$$\mathbb{P}(A_{x,h} \cap A_{y,h}) \geq 1 - 0.1 - 0.1 = 0.8. \tag{141}$$

Similarly, per-horizon coverage does not imply that all future waypoints are covered simultaneously with probability 90%. A joint guarantee over the full trajectory would require either a multivariate trajectory-level nonconformity score or a multiple-testing correction across axes and horizons. The cross-shaped intervals in Figure 18 are therefore the most direct representation of the actual statistical guarantee, i.e., one calibrated interval for lateral error and one calibrated interval for longitudinal error at each horizon.

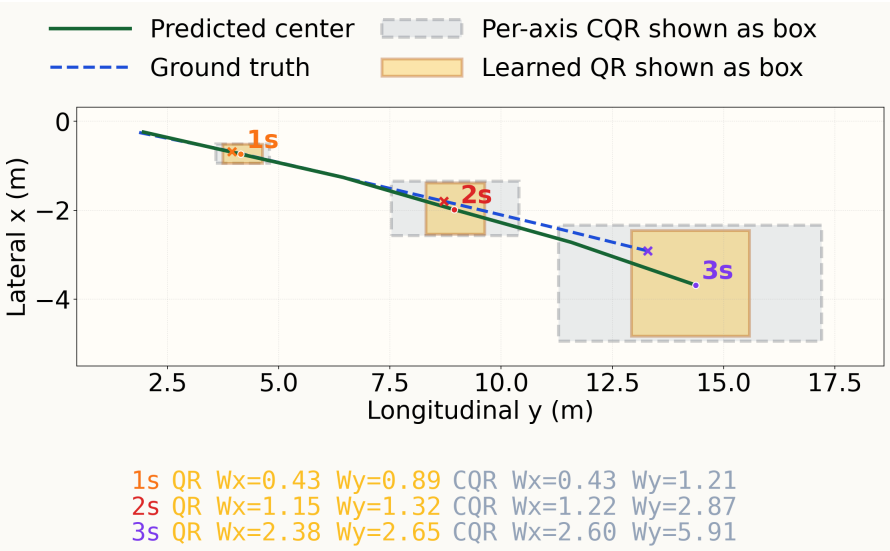

Figure 19: Box visualization of the same sample shown in Figure 18. The horizontal axis is longitudinal position y in meters, and the vertical axis is lateral position x in meters. The solid green line denotes the predicted center trajectory, and the dashed blue line denotes the ground truth. The dashed pale boxes visualize the conformalized per-axis CQR intervals as rectangles, while the solid orange boxes show the learned QR intervals before calibration. The labeled points indicate the 1 s, 2 s, and 3 s horizons.

Figure 19 shows the same sample but using rectangular boxes. This view is useful because it is easier to interpret spatially as each rectangle looks like an uncertainty region around the future waypoint. However, the rectangles are only the Cartesian visualization of the separately calibrated x- and y-intervals. They should not be interpreted as strict 90% joint two-dimensional prediction boxes. The important result in the box view is therefore not the rectangular shape itself, but the fact that the conformalized CQR intervals expand with time and are wider than the learned QR intervals where the raw quantile model is too optimistic.

For Figure 19 and Figure 18, both visualizations show the expected horizon-dependent behaviour. At the 1 s horizon, the predicted center and ground truth are close, and the intervals are small. By the 2 s and 3 s horizons, the intervals become wider, especially in the longitudinal direction. Intuitively, this is similar to everyday driving: it is usually easier to estimate where a vehicle will be in the next moment than after several seconds, because small changes in braking, acceleration, or steering can noticeably change the later position.

The difference between lateral and longitudinal widths is also important. In general, a larger interval along one coordinate means that the model assigns larger calibrated predictive uncertainty to that coordinate under the available information. The uncertainty therefore does not need to be the same in all directions. Let

$$w_{x,h} = |C_{x,h}(X)|, \qquad w_{y,h} = |C_{y,h}(X)|, \qquad A_h = w_{x,h} w_{y,h}. \tag{142}$$

Here, $w_{x,h}$ is the lateral CQR interval width, $w_{y,h}$ is the longitudinal CQR interval width, and A_h is the area of the rectangular visualization at horizon h. For Figure 19, the sample-level widths are shown in Table 4.

Table 4: Sample-level CQR interval widths for Figure 19. The area is the product of the separately calibrated lateral and longitudinal widths and is shown only as a spatial visualization aid, not as a joint 90% coverage guarantee.

Horizon	Lateral width $w_{x,h}$ [m]	Longitudinal width $w_{y,h}$ [m]	Rectangle area A_h [m^2]
1 s	0.43	1.21	0.52
2 s	1.22	2.87	3.50
3 s	2.60	5.91	15.37

Table 4 shows that the interval widths increase with horizon for this sample. The longitudinal width increases more strongly than the lateral width, indicating that the model assigns more uncertainty to forward progress than to side-to-side position. This is intuitive in driving. When a vehicle is following a road, its lateral movement is often restricted by the lane, road boundary, and steering smoothness.

In contrast, its longitudinal position after a few seconds can change substantially depending on whether the driver accelerates, maintains speed, or brakes. A simple analogy is driving behind another car: one can often predict that the vehicle will remain in its lane, but it is harder to know exactly how far forward it will be after three seconds because that depends on speed changes. Thus, larger longitudinal intervals mainly reflect uncertainty in timing and forward progress, while larger lateral intervals would indicate uncertainty in steering direction, road alignment, or maneuver choice.

Table 5: Empirical coverage of the learned QR intervals and conformalized CQR intervals across prediction horizons. The coverages are evaluated on held-out validation samples after conformal corrections are computed from the calibration split. Higher coverage is not always better by itself: useful intervals should be calibrated without becoming unnecessarily wide.

Horizon	Valid samples	QR x	QR y	CQR x	CQR y
1 s	5719	89.61%	81.83%	89.82%	89.77%
2 s	5419	87.14%	64.66%	89.87%	89.72%
3 s	5119	85.43%	64.80%	89.96%	89.92%

Table 5 shows why conformal calibration is necessary. Quantile Regression already produces heteroscedastic intervals, meaning that the interval width can change from one input to another. However, QR by itself does not provide a finite-sample coverage guarantee. Its calibration relies on the model, the optimization procedure, and the amount of available data being sufficient to learn the correct conditional quantiles. In a finite neural-network setting, this can fail even when the intervals look visually plausible.

The empirical CQR coverages are close to the nominal 90% level for both axes and all three horizons. Some values are slightly below 90%, such as 89.72% at the 2 s longitudinal horizon. This is not unexpected when coverage is measured on a held-out validation set, because empirical coverage is a sample estimate. If $\widehat{p}_h$ denotes the empirical coverage at horizon h computed from n_h held-out samples, its approximate binomial standard error is

$$\mathrm{SE}(\widehat{p}_h) = \sqrt{\frac{\widehat{p}_h(1-\widehat{p}_h)}{n_h}}. \tag{143}$$

Here, $\widehat{p}_h$ is the observed coverage proportion and n_h is the number of valid evaluation samples at horizon h. With n_h around 5,000 and $\widehat{p}_h \approx 0.90$, this standard error is approximately 0.4 percentage points. Therefore, held-out empirical coverages

such as 89.7% are close to the nominal target within ordinary validation-sample variability.

The raw QR intervals are much closer to the target coverage in the lateral x-direction than in the longitudinal y-direction. This is consistent with the qualitative behaviour seen in Figure 18. Lateral motion is often constrained by lane geometry and steering smoothness, whereas longitudinal motion depends strongly on future speed, acceleration, and braking. The raw QR model therefore underestimates the longitudinal error distribution, especially at the 2 s and 3 s horizons. Additional training, stronger temporal modeling, or better loss balancing might improve QR calibration, but more training alone would not guarantee finite-sample coverage. This is precisely why conformal calibration is needed.

The implication of low QR coverage is overconfidence. If an interval is intended to represent a 90% uncertainty band but covers the ground truth much less often, then the model is presenting an uncertainty estimate that is too narrow. In trajectory prediction, this is problematic because the predicted centerline may appear more reliable than it actually is. CQR corrects finite-sample miscalibration by using the calibration set to enlarge or adjust the learned QR intervals. Under exchangeability, this yields marginal coverage at the target level, while the empirical validation coverage is expected to be close to the target up to sampling variability. Thus, the QR head provides the adaptive shape of the uncertainty, while the conformal step makes that uncertainty statistically calibrated at the per-axis level.

Table 5 evaluates calibration, but calibration alone does not measure interval efficiency. A trivial interval covering the entire road would have high coverage but would not be informative. Efficiency asks how wide the interval must be to achieve the desired coverage. In this subsection, efficiency is illustrated through the sample-level widths in Table 4 and the qualitative examples below.

Contrasting low- and high-uncertainty scenarios. The next qualitative examples evaluate whether the interval width changes in a meaningful way across different driving contexts. Low uncertainty is expected when the current input lies in a well-represented and predictable region of the data distribution. In such cases, the conditional response distribution is relatively concentrated. High uncertainty is expected when the input lies in a rarer, more ambiguous, or noisier region of the data distribution. In such cases, there may be less training support, or several future outcomes may be plausible.

The qualitative context images in Figures 21 and 23 are six-camera visualizations from the trajectory-prediction setting. Their purpose here is to provide visual context for the uncertainty intervals, not to evaluate the detection pipeline.

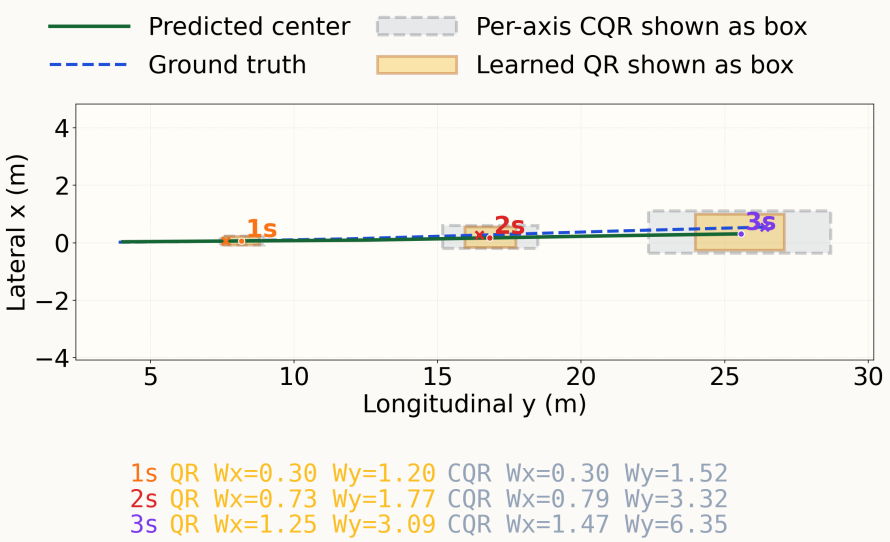

Figure 20: Low-uncertainty trajectory validation example. The horizontal axis shows longitudinal position y in meters, and the vertical axis shows lateral position x in meters. The solid green curve is the predicted center trajectory, and the dashed blue curve is the ground-truth trajectory. The dashed pale boxes show the CQR intervals, and the solid orange boxes show the learned QR intervals before conformal calibration.

Figures 20 and 21 show a **low-uncertainty case**. The predicted trajectory and the ground-truth trajectory remain almost aligned across the horizon, and the CQR intervals remain compact, especially laterally. The six-camera view verifies it. The ego vehicle is traveling in a relatively open scene, the road geometry is mostly straight, and there are few nearby actors directly influencing the ego path. Such a scene corresponds to a more common driving pattern.

The result is meaningful because the uncertainty is not inflated uniformly across all samples. The model assigns narrow intervals to the lateral evolution of the trajectory because the path is visually and geometrically constrained. The longitudinal interval still increases with time, which is expected even in simple driving, since the exact future position along the road depends on the future speed profile, acceleration, and braking. Thus, the sample is not interpreted as having low maneuver ambiguity and lower apparent predictive uncertainty relative to more complex cases. This is an indication that both aleatoric and epistemic uncertainty are lower in this case meaning the scene is less complex and the model has trained on it enough to make it relatively sure about its prediction.

Figures 22 and 23 show a **high-uncertainty case**. Before focusing on the specific scene, it is useful to interpret what a larger interval means more gen-

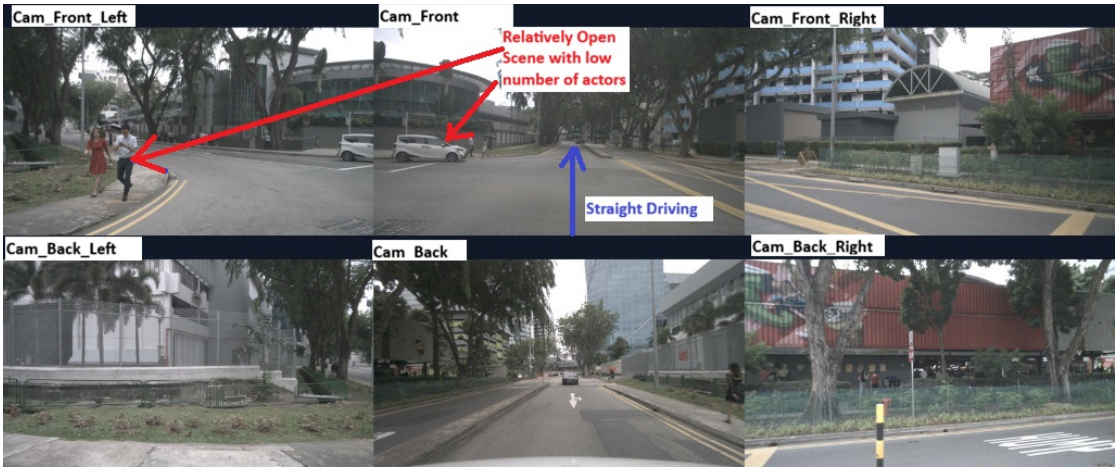

Figure 21: Six-camera scene view corresponding to the low-uncertainty example in Figure 20. The image grid shows synchronized front-left, front, and front-right views on the top row, and back-left, back, and back-right views on the bottom row. The scene is relatively open, the road geometry is mostly straight, and there are few nearby actors directly constraining the ego path.

erally. A wider CQR interval indicates that, under the learned representation and calibration data, the true future coordinate has historically been harder to contain tightly for similar model outputs. This can be consistent with epistemic uncertainty, where the model has limited experience with similar scenarios, or with aleatoric uncertainty, where the scene itself admits several plausible future behaviours, more sensor noise, etc. CQR does not by itself decompose uncertainty into epistemic and aleatoric components. It provides a calibrated predictive interval, not a causal explanation of the uncertainty source.

In this example, the camera view is consistent with the larger uncertainty intervals. The ego vehicle appears to be moving through a dense, low-speed scene with many nearby cars, parked vehicles, a bus, roadside objects, occlusions, and possible pedestrian activity. This is less like regular open-road lane following and more like a constrained exit or dense urban maneuver. Such a scene requires the model to infer many complex things such as available free space under partial occlusion and consider a wide variety of possible interactions with numerous nearby actors. It is also plausibly a lower-frequency situation compared with straight road-following scenes, which are more common in typical driving data. Therefore, the larger CQR intervals can be interpreted as the model assigning wider calibrated uncertainty in a more difficult and less stereotypical driving context.

The contrast between Figures 20–21 and Figures 22–23 is the main qualitative result. In statistical terms, the uncertainty is heteroscedastic: the interval width

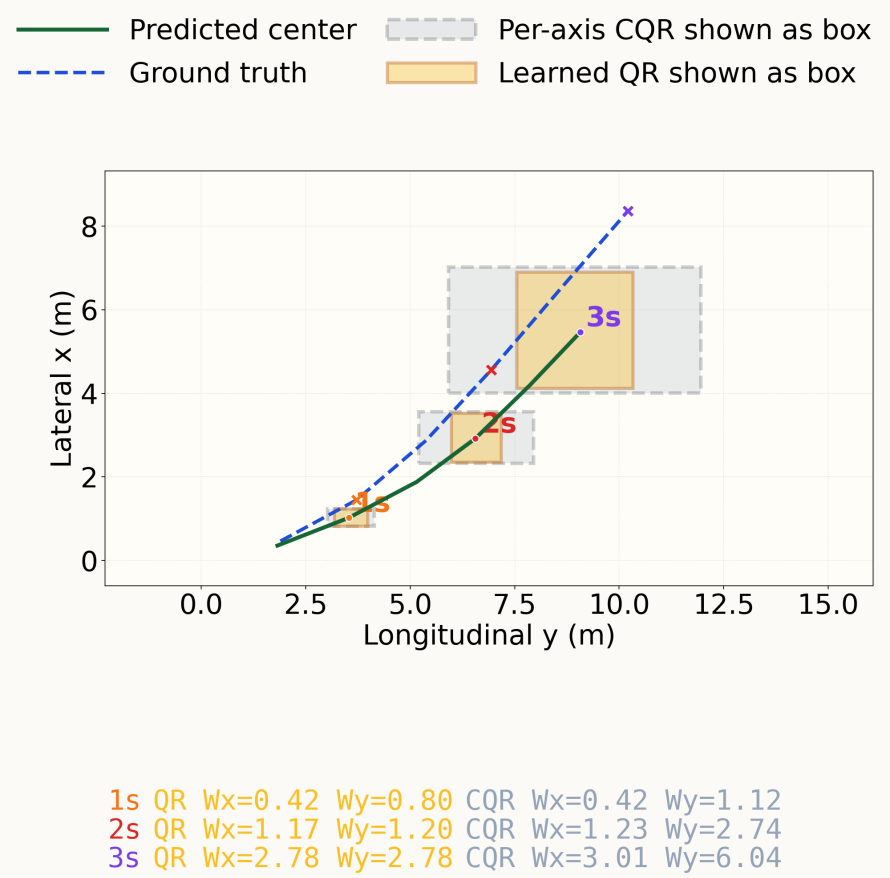

Figure 22: High-uncertainty trajectory example. The horizontal axis shows longitudinal position y in meters, and the vertical axis shows lateral position x in meters. The solid green curve represents the predicted center trajectory, and the dashed blue curve represents the ground-truth trajectory. The dashed pale boxes show the conformalized CQR intervals, while the solid orange boxes show the learned QR intervals before calibration. The orange, red, and purple labels mark the 1 s, 2 s, and 3 s horizons.

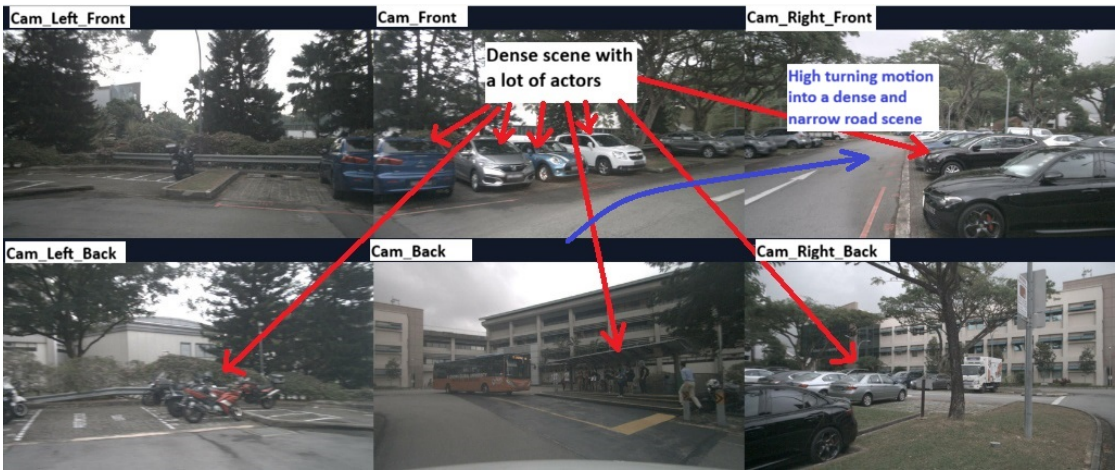

Figure 23: Six-camera scene view corresponding to the high-uncertainty example in Figure 22. The image grid shows synchronized front-left, front, and front-right views on the top row, and back-left, back, and back-right views on the bottom row. The ego vehicle is surrounded by a denser low-speed environment with nearby vehicles, parked cars, a bus, roadside objects, occlusions, and possible pedestrian activity.

changes with the input instead of remaining fixed for all samples. This matters because real prediction errors are not uniform across the feature space. Some samples are easy because the input strongly constrains the output, while other samples are difficult because the input is rare, ambiguous, noisy, or interaction-heavy. A fixed uncertainty margin would ignore this difference. CQR, by contrast, keeps the interval compact for a familiar and simple scene, and expands it for a dense and less frequent scene.

The qualitative figures show that the learned interval widths vary in a visually reasonable way. Compact intervals occur in a simple and regular scene, while wider intervals occur in a denser and more ambiguous scene. This behaviour has a natural risk-monitoring interpretation. Low uncertainty means that the model and calibration procedure assign a narrow set of plausible future ego positions to the scene. This is appropriate when the road is open, the maneuver is almost straight, and the visual evidence strongly supports one future trajectory. High uncertainty means that the future ego position is less reliable if represented only by the predicted centerline.

The result instead show that CQR provides a calibrated signal that could be used by a future safety-aware downstream module to identify when the nominal trajectory is less certain. Such a module could use high uncertainty as an addi-

tional input for risk assessment, fallback logic, or decision monitoring. Developing an explicitly uncertainty-aware planning system is outside the scope of this thesis, but the results here show why such a direction is relevant.

The high-uncertainty cases also suggest where uncertainty could be reduced in future work. If the uncertainty is mainly epistemic, it may be reduced by collecting more representative training and calibration data from similar low-speed, dense, actor-rich scenarios. Targeted data collection, scenario reweighting, active learning, and augmentation of rare situations would expose the model to more examples of these difficult contexts. If the uncertainty is partly aleatoric, it cannot be eliminated completely. However, it may still be better characterized by adding richer context, such as longer temporal history, stronger actor-interaction modeling, map or lane priors where available, and multimodal trajectory hypotheses. CQR does not remove the underlying uncertainty; rather, it reveals where the model assigns wider calibrated predictive intervals and makes that uncertainty statistically calibrated at the per-axis level.

The results show that CQR improves the reliability of the learned QR intervals while preserving their usefulness for interpreting driving scenes. The empirical coverage becomes close to the desired 90% level separately for each axis, the intervals grow naturally across the prediction horizon, and the interval size adapts to the complexity of the scene. The main caveats remain that the guarantee is marginal per axis rather than joint over the full two-dimensional box, and it is marginal over the calibration distribution rather than conditional for every individual scene. Nevertheless, the qualitative and quantitative results indicate that CQR provides a calibrated uncertainty layer on top of the trajectory prediction model, with larger intervals appearing in the types of scenes where a point prediction alone would be less trustworthy.

4 Discussion

Our aim at the start of this thesis was to study whether a transformer-based autonomous driving model could be transferred from a passenger-car source domain to a heavy-truck target domain, and whether calibrated uncertainty intervals could be added to nuScenes (Caesar et al., 2020) trained UniAD's (Hu et al., 2023) trajectory predictions. The results show that both parts of the problem are important. Direct zero-shot transfer baseline from nuScenes to TruckScenes (Fent et al., 2024) performed poorly, which confirms that the domain gap is too large to ignore. Differences in camera topology, camera geometry, speed profile, vehicle type, and scene distribution all affect the latent representation and the final trajectory prediction.

The first stage of transfer learning, where only the latent representation was fine-tuned, improved object detection performance substantially. This indicates that the pre-trained UniAD model still contained useful visual and geometric knowledge, but that this knowledge had to be realigned to the TruckScenes sensor setup. The improvement over both the zero-shot model and the PETR (Liu et al., 2022) TruckScenes (Fent et al., 2024) baseline suggests that transfer learning was more effective than learning the representation only from the smaller target-domain dataset. This supports the idea that the source-domain model provides a strong initialization, while target-domain fine-tuning adapts it to the new camera geometry and visual statistics.

The second stage, improved the actual trajectory prediction task. The L2 trajectory error was reduced by roughly half at all evaluated horizons, showing that the model did not only learn a better perception representation, but also adapted its prediction behaviour to the TruckScenes domain. The qualitative examples support this interpretation. After fine-tuning, the model produced smoother and more target-domain appropriate trajectories than in the zero-shot case, especially in terms of forward progress and speed. The challenging example shows that fine-tuning does not solve all planning cases and the model can still miss interaction-dependent manoeuvres, such as lane changes caused by vehicles ahead, even when its predicted trajectory remains physically plausible.

The uncertainty quantification results show that point predictions alone are not sufficient. Raw Quantile Regression (Romano et al., 2019) produced intervals that were not reliably calibrated, especially in the longitudinal direction and at longer horizons. After conformal calibration, the empirical coverage was close to the desired 90% level for both axes and all horizons. This demonstrates the value of Conformalized Quantile Regression (Romano et al., 2019), the quantile model gives input-dependent interval widths, while conformal calibration corrects the finite-sample miscalibration. The qualitative uncertainty examples also behaved as expected, with wider intervals in more ambiguous or difficult scenes and narrower intervals in simpler scenes.

However, the uncertainty results should be interpreted carefully. The conformal guarantee is marginal and per axis. It does not guarantee conditional coverage for every scene type (Romano et al., 2019). The intervals also do not make the autonomous driving system safe by themselves; they only provide calibrated information about prediction uncertainty. To make practical use of this information, a downstream planner would need to react to wide intervals, for example, by choosing a more conservative action. This is left as future work and is outside the scope of this thesis.

There are broader limitations as well. The end-to-end fine-tuning stage was stopped after three epochs because the validation loss seemed to plateau, and

due to limited project time, longer training may still improve the model. The experiments also do not include repeated runs or a full ablation study, so the results should be treated as empirical evidence from this case study rather than final benchmark claims. Finally, the detection metrics are only a proxy for latent representation quality, and L2 distance does not capture all aspects of planning quality. Future work should therefore include longer training, repeated experiments, stronger evaluation of interaction-heavy scenes, conformal regions, and integration of uncertainty estimates into an actual planning or safety module.

5 Conclusion

Our study shows that adapting autonomous driving models across domains requires more than applying a pre-trained model directly to new data. The results suggest that a staged transfer-learning procedure is a practical way to reuse knowledge from a large source-domain model while still adapting to the geometry, appearance, and motion patterns of a heavy-truck target domain. In this sense, the main contribution is not only the improved performance after fine-tuning, but the demonstration that representation adaptation and end-to-end adaptation address different parts of the domain shift.

It also highlights the importance of uncertainty in trajectory prediction. A single predicted path gives an incomplete description of the future, especially in ambiguous driving scenes. By adding calibrated prediction intervals, the model output becomes more informative. It indicates not only where the vehicle is expected to go, but also how reliable that prediction is. Overall, the work supports the view that future autonomous driving systems should combine transfer learning, calibrated uncertainty estimation, and downstream risk-aware planning rather than relying only on deterministic point predictions.

References

Angelopoulos, A. N., Barber, R. F., & Bates, S. (2024). Theoretical foundations of conformal prediction. *arXiv preprint arXiv:2411.11824*.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Caesar, H., et al. (2020). *Nuscenes: A multimodal dataset for autonomous driving*. arXiv.

Chen, L., Wu, P., Chitta, K., Jaeger, B., Geiger, A., & Li, H. (2024). End-to-end autonomous driving: Challenges and frontiers. *IEEE Trans. Pattern Anal. Mach. Intell.*, *46*(12), 10164–10183. https://doi.org/10.1109/TPAMI.2024.3435937

Codevilla, F., Miiller, M., López, A., Koltun, V., & Dosovitskiy, A. (2018). End-to-end driving via conditional imitation learning, 1–9. https://doi.org/10.1109/ICRA.2018.8460487

Depeweg, S., Hernandez-Lobato, J. M., Doshi-Velez, F., & Udluft, S. (2018). Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. *International Conference on Machine Learning*, 1184–1193.

Fent, F., Kuttenreich, F., Ruch, F., Rizwin, F., Juergens, S., Lechermann, L., Nissler, C., Perl, A., Voll, U., Yan, M., et al. (2024). Man truckscenes: A multimodal dataset for autonomous trucking in diverse conditions. *Advances in Neural Information Processing Systems*, *37*, 62062–62082.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://doi.org/10.1007/978-0-387-84858-7

Hu, Y., Yang, J., Chen, L., Li, K., Sima, C., Zhu, X., Chai, S., Du, S., Lin, T., Wang, W., et al. (2023). Planning-oriented autonomous driving. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 17853–17862.

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, *110*(3), 457–506.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., & Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, *113*(523), 1094–1111.

Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Yu, Q., & Dai, J. (2024). Bevformer: Learning bird's-eye-view representation from lidar-camera via

spatiotemporal transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *47*(3), 2020–2036.

Liu, Y., Wang, T., Zhang, X., & Sun, J. (2022). Petr: Position embedding transformation for multi-view 3d object detection. *European conference on computer vision*, 531–548.

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability and Its Applications*, *9*, 141–142.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *International conference on machine learning*, 1310–1318.

Romano, Y., Patterson, E., & Candes, E. (2019). Conformalized quantile regression. *Advances in neural information processing systems*, *32*.

Ross, S., Gordon, G., & Bagnell, D. (2011, November). A reduction of imitation learning and structured prediction to no-regret online learning. In G. Gordon, D. Dunson, & M. Dudík (Eds.), *Proceedings of the fourteenth international conference on artificial intelligence and statistics* (pp. 627–635, Vol. 15). PMLR. https://proceedings.mlr.press/v15/ross11a.html

Tishby, N., Pereira, F. C., & Bialek, W. (2000). The information bottleneck method. *arXiv preprint physics/0004057*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Vovk, V., Gammerman, A., & Shafer, G. (2005). *Algorithmic learning in a random world*. Springer.

Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, *27*, 3320–3328.

Zhang, A., Lipton, Z. C., Li, M., & Smola, A. J. (2023). *Dive into deep learning*. Cambridge University Press.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.