CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Cut-in Detection from Videos

Master's thesis in Computer science and Engineering

Apoorva Udayakumar
Aditya Padmanabhan Varma

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

Cut-in Detection from Videos

Apoorva Udayakumar
Aditya Padmanabhan Varma

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

Apoorva Udayakumar
Aditya Padmanabhan Varma

Supervisor: Ashkan Panahi, Department of Data Science and AI
Advisor: Joakim Johnander, Zenseact
Examiner: Peter Damaschke, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer science and engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Apoorva Udayakumar
Aditya Padmanabhan Varma
Department of Computer science and engineering
Chalmers University of Technology and University of Gothenburg

Abstract

For autonomous driving, it is crucial to anticipate the behaviour of other road users and act accordingly. One important scenario is when a vehicle cuts into the lane of the ego-vehicle with or without sufficient indicator cue. This thesis studies such a scenario and investigates the application of deep learning techniques for understanding and predicting when the cut-in maneuver is performed by the vehicles ahead. As a first step, indicator cues from videos of vehicles performing cut-ins are detected successfully with an F1 score of of 83% and recall value of 85%. We achieve this result by employing ResNet-18 and CNN-LSTM with a tuned level of context around the target vehicle. Further we predict the estimates of interest such as start and end of cut-in intention using the same architectures and discuss the challenges.

Keywords: cut-in, Zenseact, autonomous driving, indicator, intent prediction

Acknowledgements

We want to thank our supervisors Joakim Johnander, Ashkan Panahi, Bruno Augusto, Fredrik Hoxell, and examiner Peter Damaschke for their help and guidance during the project and writing of the thesis.

Apoorva Udayakumar & Aditya Padmanabhan Varma

Gothenburg, 2023-10-09

Contents

List of Figures

List of Tables

List of Abbreviations

2D-CNN 2D Convolutional Neural Networks

3D-CNN 3D Convolutional Neural Network

AD Autonomous Driving

ADAS Advanced Driver Assistance System

ACC Adaptive Cruise Control

AUSE Area Under Sparsification Error

AVs Autonomous Vehicles

CNN Convolutional Neural Network

DL Deep Learning

ECE Expected Calibration Error

GPS Global Positioning System

HA Highway Autopilot

HD High Definition

LDW Lane Departure Warning System

LiDAR Light Detection and Ranging

LSTM Long Short-Term Memory Networks

MRI Magnetic Resonance Imaging

NLL Negative Log Likelihood

ODD Operational Design Domain

OOD Out of Distribution

RADAR RAdio Detection And Ranging

RoI Region of Interest

RNN Recurrent Neural Networks

SAE Society of Automotive Engineers International

TTLC Time To Lane Change

VAR Video Action Recognition

V2V Vehicle-to-Vehicle

V2X Vehicle-to-Everything

1

Introduction

In the recent years, the focus of car manufacturers has shifted towards safety. While safety was primarily defined in the past by physical elements inside the car, such as airbags, the modern view emphasizes assisting the drivers and, in extreme cases, taking over control to prevent fatal accidents. Human error, like inappropriate speed and failure to yield the right of way, has been identified as a significant cause of on-road accidents by the German Federal Statistical Office [1]. A similar study by the U.S. National Highway Traffic Safety Administration [2] also found that human error was estimated to account for a staggering 94% of the critical reasons[1] between 2005 and 2007. Critical driver-related reasons were recognition and decision-based errors. Recognition errors refer to the driver's inattention to details or distractions, and decision-based errors include misreading the intention of surrounding drivers and illegal maneuvers. Advanced Driver Assistance System (ADAS) and Autonomous Driving (AD) is one solution to this problem since it reduces the risk of human errors and assists drivers in critical situations. ADAS/AD has the potential not only to reduce accidents on the road but also to make the commute more efficient and cheap [3].

To attain safe mobility, Autonomous Vehicles (AVs) should have a powerful perception and understanding of their environment. This includes detecting static objects like the roads, signboards, etc., and dynamic entities like the surrounding vehicles and their intentions. At present, the perception modules are well equipped with sensors such as LiDAR, RADAR, Camera, and GPS. RADAR and LiDAR are used to map the objects surrounding the AVs, GPS is used to locate the vehicle across the globe, and the camera to view the surroundings. The success of autonomous technologies depends on the effective use of the data collected by the perception modules.

To drive safely and comfortably, it is essential to anticipate the behavior of other drivers. Lane change is a common and important lateral maneuver to keep track of. When the lane change is into the ego lane, predicting the intent becomes more necessary than advantageous since it directly affects the user. Lane change intent prediction is a persistent problem due to the broad and complex driver behavior observed on the road. Detecting and tracking vehicles that perform cut-in yields a

[1] *"The critical reason is the immediate reason for the critical pre-crash event and is often the last failure in the causal chain of events leading to the crash. Although the critical reason is an important part of the description of events leading to the crash, it is not interpreted as the cause of the crash, nor as the assignment of the fault to the driver, vehicle, or environment"* [2]

naïve solution, leaving little time for an appropriate reaction from ego. However, using turn signals to confirm intent and other features like drift towards ego lane would enable lane change to be predicted sufficiently early. In this thesis, we aim to establish a benchmark on an internal Zenseact dataset focusing on turn indicators as primary confirmation of intent and develop guidelines for future work on the systems based on the challenges faced.

1.1 Deep Learning in Automobile Industry

In recent years, Deep Learning (DL) techniques have aided various applications within autonomous driving to achieve remarkable results. The areas that utilize deep learning in autonomous vehicles are:

- Perception − Being aware of the environment and recognizing obstacles.
- Localisation − Locating the vehicle's position in the world.
- Planning − Defining a trajectory using localization and perception.

Deep learning has helped accomplish extraordinary results in perception tasks for autonomous driving. The data from sensors like cameras and Light Detection and Ranging (LiDAR) is used by deep learning algorithms for tasks such as lane detection, road segmentation, object detection, road mark detection, and traffic sign detection. A specific crucial application is to identify possible lane change scenarios by vehicles ahead, which provide valuable information to the drive control systems to adjust the trajectory accordingly, which is also the focus of this thesis.

2

Background and Related Work

The interest in AD/ADAS or AVs has been on the rise. It is projected to grow into a market of USD 196 billion by 2030 from USD 25 billion in 2021[1]. This growth is accelerated by standardization by car manufacturers through systems such as Adaptive Cruise Control (ACC) and Lane Departure Warning System (LDW). Optimizing these models brings the ultimate goal of AD closer and safer. One of the crucial problems, especially in highway scenarios, that needs to be addressed is predicting the intent of other vehicles. This helps to accurately detect and estimate characteristics of the maneuver, such as duration and trajectory of the targets ahead of the ego. In the remainder of the work, lane change refers to the action performed by the target.

Detecting surrounding vehicles is a well-researched problem with approaches ranging from using single sensor [4] to fusing information from an array of sensors on-board [5], [6]. One of the most recent works by Wei *et al.* [7] combines multiple sensors with Kalman filter to achieve high detection accuracy, reaching 97.5% across diverse scenarios. Kim *et al.* [5] performs 2D object detection using RGB images and projecting LiDAR point cloud into the camera plane to obtain the additional parameters such as depth and height. The most common approach combines data from multiple modalities to improve prediction quality. However, this comes with the additional overhead of processing and storing possible redundant information. Redundancy in this context refers to sensor data that adds no value to existing information from other sensors. So, it is necessary to make sure that the array only includes sensors that are complementary to each other. Cameras are a very intuitive candidate for the sensor array as it enables environment perception close to a human driver. With reference to this particular work, cameras would help the perception systems make informed decisions based on visual cues like turn indicators or drift toward the lane markers to confirm the target intention of lane change.

All existing systems from simple ACC to most sophisticated Highway Autopilot (HA) that fall under Society of Automotive Engineers International (SAE) L3-4 [8] consider cut-in scenarios to be extremely critical [9]. In this work, cut-in is defined as a specific lane change maneuver by the target from the adjacent lane to the ego lane. Early prediction of this intent is critical for safety since it can prepare the ego to

[1]https://www.globenewswire.com/en/news-release/2022/10/10/2531178/0/en/Autonomous-Car-Market-Size-to-Reach-196-97-Billion-by-2030-CAGR-25-7-Confirms-Strategic-Market-Research.html

slow down and leave sufficient gaps for the vehicle to move in without compromising the safety of both parties. Due to the highly dynamic driving behavior and different road conditions, detecting a target performing a maneuver is challenging. A reliable system adds to the safety barrier and can assist the ego to be prepared in advance.

Access to high-bandwidth and low-latency solutions is helping the development of Vehicle-to-Vehicle (V2V) [10] or more general Vehicle-to-Everything (V2X) communications, increasing connectivity both within and outside vehicles. V2V allows road users to easily communicate their intent in advance to surrounding entities, especially vehicles that are directly affected. In such cases, intent prediction systems such as cut-in detection modules in AVs can potentially act as a reliable backup.

2.1 Related Work

Different ways exist to classify existing works in vehicle motion modeling and planning. Lefèvre *et al.* [11] propose a classification in three levels, in increasing order of complexity. First, physics-based methods where the underlying assumption is that the motion only depends on the dynamic states. Second, maneuver-based methods where future motion is based on the specific maneuver the driver intends to perform, and finally, interaction-aware models that consider the inter-dependencies between the different vehicles' maneuvers to make the final prediction. Another approach was put forth by Song *et al.* [12] compiling prior works in lane change maneuver prediction of surrounding vehicles. This study classifies all existing works into four buckets broadly based on their approach – model-based, generative, discriminative, and neural network-based. Each approach has its own set of merits and demerits. Model-based approaches, for instance, are highly interpretable and require less data but come at the cost of weak assumptions and low tuning capabilities.

Most works in intent prediction, focusing on detecting lane change by other vehicles, specifically to the ego vehicle lane, make use of a wide array of sensors, both individually like camera [13] and in an ensemble such as LiDAR [14] and Radar [15] along with Camera. There are also works that use physical features such as lateral and longitudinal position and velocity estimates of the target vehicle [16]–[18].

One of the most recent works focuses on lane change detection of surrounding vehicles using only camera data in a two-stream technique and late fusion [13]. The authors examined multiple architectures, such as Slow-Fast networks, using identical network designs but varying frame rates of the video input for both streams and Spatio-temporal multiplier networks that try to isolate the two aspects in their respective streams and learn them individually. They achieved a prediction accuracy exceeding 90% in time horizons (anticipation) between 1-2s before the maneuver.

In an ensemble of sensors, the data generated by each sensor captures different characteristics. Multimodal architectures leverage such ensembles. The UMD-DMED model [19] uses the multimodal driving context of the ego vehicle (i.e., vehicle signals and videos from the front view camera) to understand the vehicle's surroundings and capture the relation between the driving context and driving maneuvers. The model

achieves 90.4% precision and 89.9% recall with a 5s threshold for early detection of driving maneuvers.

Specific works focusing on turn signal detection have explored architectures such as CNN layers followed by LSTM [20], and with an attention module in the early stages to estimate driver intent [21].

Representing the environment more efficiently by overlapping consecutive frames from the same video feed and using motion flow [22] have also been considered. In such works, the authors try to condense the spatial and temporal aspects into a single representation, which is then passed as input to the network. For instance, the authors stored the grayscale version of the image in the red channel, target vehicle recognition in the blue channel, and surrounding vehicles recognition in the green channel. This way, the entire sequence was converted into a single image. The same authors have also previously worked with validating CNN and LSTMs for the specific use case of lane change intent prediction [23] by storing the motion history in one of the channels of the image, preserving the original size and depth, working around the requirement to stack multiple images to the model.

Most works in CNN also make use of backbone networks. Izquierdo *et al.* [22] include an extensive comparison of backbone networks in their experiments, which include GoogleNet, ResNet101, ResNet50, ResNet18, SqueezeNet, AlexNet, and ShuffleNet. It was shown that the ResNet variants were ahead of the other alternatives for the specific dataset and experiment. Other networks, such as YOLOv3-tiny, have also been used in a different capacity [24], such as feature fusion across multiple depths and spatial pyramid pooling to enrich deep features.

Only a few works specifically focus on cut-ins since they are subsets of the much broader target lane change instances. Modules that can predict lane changes can be restricted for adjacent lane targets to be used as cut-in predictors. Among the limited works, Wang *et al.* [25] developed an algorithm based on a comprehensive behavioral study on cut-in to extract events from the Shanghai Naturalistic Driving Study. Nodine *et al.* [26] studied truck following behavior, including distances and headways at which cut-ins were observed. Kim *et al.* [27] looked at the opposite end of the scenario where the driver behavior (decision making) to different cut-in instances was studied. In all these cases, specific characteristics of cut-in, such as lateral velocity for confirmation, driver behavior, the relative position of the surrounding vehicles, etc., are used to describe the maneuver quantitatively. However,Wang *et al.* [25] worked on empirically studying the different specifics of cut-in to extract more samples. Nalcakan *et al.* [28] work specifically with LSTM networks and condense the features of interest to only the center, height, and width of the bounding box. According to the authors, selecting bounding box parameters as features helps reduce processing time for prediction. However, the method does not assume variable road layouts, in which the bounding boxes of vehicles could have movements similar to changing lanes. The authors also completely discount visual cues like turn indicators, which are a direct confirmation of intent to change lanes and depend only on the movement of vehicles towards the ego lane center. In this thesis, we aim to include a combination of both these features, which allows for a more robust system.

2.2 Goals and Scope

1. *Successfully constructed benchmark for cut-in detection with a focus on turn signals based on internal data where lane changes have been annotated*

   Since the problem is data-specific, the benchmark obtained will be useful to drive future work. The internal data consisted of lane change annotations confirmed using turn indicators.

2. *Investigate and compare spatio-temporal architectures and their reliability for the use-case*

   Since cut-in or lane change detection can be classifier as Video Action Recognition (VAR), it was necessary to explore models that can capture both spatial and temporal features. A detailed study based on popular architectures – 3D-CNN and CNN-LSTM was performed. The specific responses of the model to the dataset were also included in the study.

3. *Evaluate limits of predicting estimates of interest – start and end of target lane change. Also, explored methods to best parameterize output predictions.*

   Driver behavior is diverse. Using target-level features such as turn indicators, in general, to predict when the lane change would start or end was assumed to be a challenging problem. The particular limits of estimating the metrics were quantified and studied in this thesis.

4. *Experiment with techniques to best estimate the uncertainty in predictions for both classification and regression problems*

   For models with critical use cases such as this, it is necessary also to obtain a measure of surety in predictions so that relevant responses can be taken after considering both the prediction and the confidence in the same. Techniques to estimate uncertainty for classification and regression models were explored.

5. *Model and optimize the surrounding vehicle context to improve the inference before actual lateral confirmation of the lane change*

   The context provided to the model can be varied from the entire scene to only the target information. Depending on this, the features learned would also be different since passing only the target information would force the model to learn vehicle-specific features such as turn indicators, whereas passing the entire scene would also give information about the environment and lane markers. Experiments were carried out to evaluate the performance of models with multiple levels of additional context around the target, and the results were tabulated. Guidelines were proposed to tune vehicle context as a feature based on the observations.

3

Theory

This chapter discusses the concepts and theories related to the thesis. The chapter starts with defining cut-in scenarios, followed by turn indicators and how they aid lane change intent prediction, and then moves to the different architectures and evaluation metrics explored during the course of the thesis. Finally, uncertainty is explored from a generic and DL perspective.

3.1 Cut-in

In this work, cut-in is defined as a lane change maneuver performed by an adjacent vehicle into the ego lane. In Figure 3.1, the boxes represent vehicles, and the dashed lines separate the lanes in which the vehicles are located. Vehicle V1 is performing a lane change maneuver from its lane into the ego lane and wants to position itself in front of the ego vehicle.

Wang *et al.* [25], defines the duration of cut-in as the time span from the target's initiation of lateral movement to stabilization in the ego lane. The authors observed that in their dataset, which consisted of 5608 instances, the duration varied from 0.7s to 12.4s with a mean and variance of 3.91s and 2.34s, respectively. Since it involves vehicles moving directly into the ego lane, it is very important to track cut-in as it happens and, if possible, in advance. This helps other autonomous drive systems be prepared for the action. In this work, the start time of the cut-in is defined as the point when the vehicle confirms the intent to move towards the ego lane with the turn indicator, and it ends when the vehicle is appropriately positioned in the ego lane.

3.2 Turn Indicators as Cue

Drivers use turn indicators to indicate their intention to leave the current lane. Although usually represented by orange blinking lights, they vary across vehicle types and manufacturers (as visible in Fig 3.2)

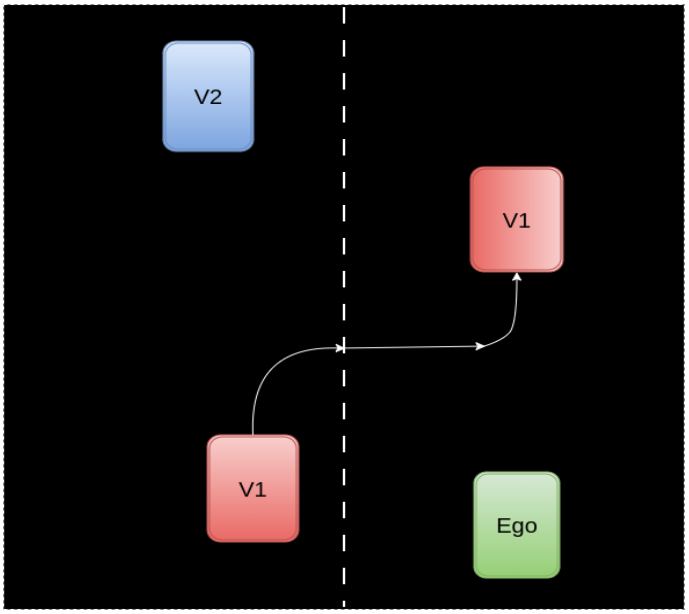


Figure 3.1: Illustration of a cut-in scenario. Vehicle, V1, drives behind vehicle V2 and performs a cut-in in front of the ego vehicle. The curved path with the arrows represents the cut-in path of vehicle V1.

Figure 3.2: Challenging Indicators. The first two sub-figures show a particular turn indicator that lights up sequentially instead of blinking. The last two sub-figures show an extreme case where the indicator lights are very small relative to the vehicle

In Figure 3.1, if vehicle V1 uses the turn indicator at or before the start of the cut-in process, it will help the ego vehicle sense the cut-in and plan its trajectory, i.e. maintain a safe distance to allow the target vehicle to move into the ego lane. Sole usage of turn indicators does not suffice to predict a cut-in intention. It suffers from in-between driver variability, such as timing or even usage. Therefore, it becomes necessary to capture not only indicators but also other possible visual cues that suggest a lane change intention, such as drift towards the ego lane center.

In-between driver variability and lack of standard design across vehicles (as in Figure 3.2) make turn indicators a challenging yet helpful feature for the application.

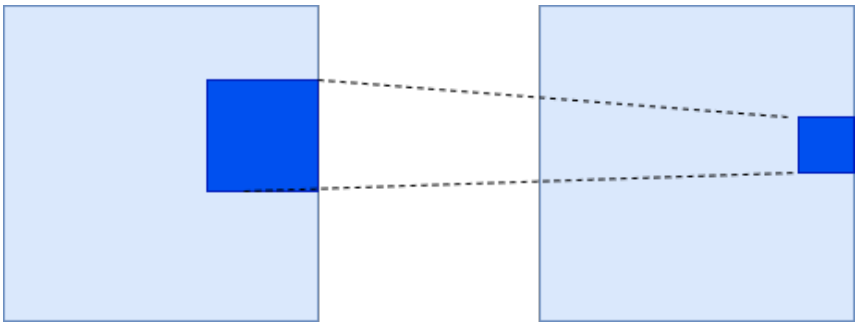


Figure 3.3: 2D Convolutions

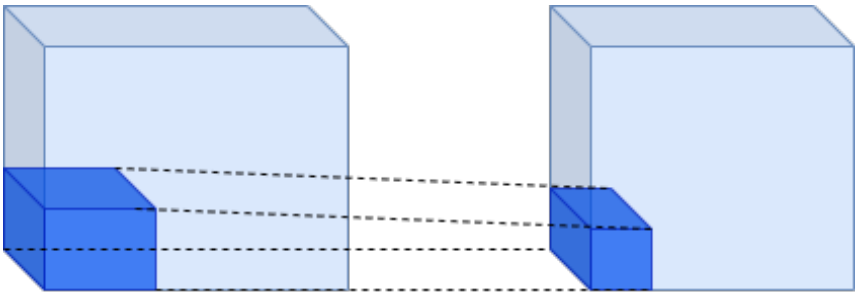


Figure 3.4: 3D Convolutions

3.3 Spatio-Temporal Architectures

Architectures that capture either spatial information (CNNs for image recognition) or temporal information (such as Recurrent Neural Networks (RNN)) have been well-researched and used to set benchmarks in respective applications. VAR tasks are a subset that requires models that capture both types of features. The spatio-temporal features can be learned either simultaneously or sequentially. Mänttäri *et al.* [29] studied the modeling of temporal data between spatio-temporal architectures. The authors observed that 3D-CNNs, which capture the features simultaneously, works best for capturing linear transitions in time and shorter action events in the input sequence, focusing on spatial features concentrated at the center of the frame. CNN-LSTM models, which combine the features sequentially, can, however, model non-linear transitions in time with the hidden state and focus on smaller patches of the spatial feature. In this thesis, we used both models to understand the nature of the maneuver better and comprehensively compare the performance for the specific dataset.

3.3.1 3D Convolutional Neural Network (3D-CNN)

A 3D-CNN is a deep learning architecture used for processing and analyzing three-dimensional data such as videos and medical images like MRI etc. It is an extension of a traditional 2D Convolutional Neural Networks (2D-CNN). In the case of 2D-CNN, the model learns features in individual images (Figure 3.3). However, in 3D-CNN, the model learns features both in the spatial and temporal domain (Figure 3.4). The spatio-temporal information is vital to understand the change in motion or action

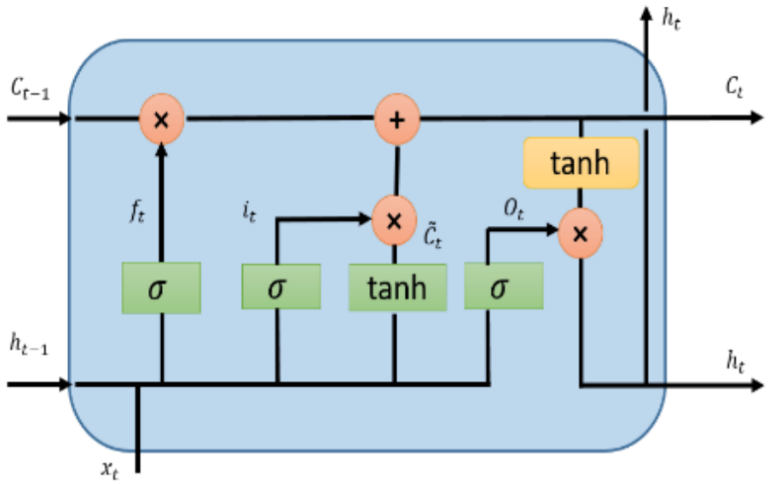


Figure 3.5: Long Short-Term Memory Networks (LSTM) Diagram (Source : Alom *et al.* [30]). At each time step t, the unit works with the current input, x_t, previous cell state, C_{t-1} and the previous hidden state, h_{t-1}

over time.

The input to a 3D-CNN can be volumetric data such as a video (a sequence of images or frames). The kernels in the layers slide across the input in all three directions to capture the features simultaneously. In this work, 3D-CNN could learn the specific part of the frame to check for a change in intensity across time to be a definite feature to identify the action.

3.3.2 CNN-LSTM

The spatio-temporal features for VAR tasks can be extracted simultaneously using 3D-CNNs or individually by combining architectures that best learn both features. The latter is achieved by CNN-LSTM networks that combine 2D convolutional neural networks and LSTM networks. The CNN primarily captures local patterns as features in two-dimensional data like images. Therefore, they are used in image recognition and classification tasks. The LSTM [31] is a RNN that captures temporal dependencies over sequential data. This network has the ability to remember or forget information over time intervals based on the input data. This is used in tasks like video analysis, speech recognition, etc.

The CNN part of architecture uses traditional 2D Convolutions(Figure 3.3) in combination with layers like pooling and fully connected layers to extract frame-wise local features from the input. The hyper-parameters that can be adjusted include the size of the convolving kernel, the stride length, and dilation parameters that control the receptive fields of the network and help capture specific features.

LSTMs are a type of RNN, developed to solve the challenge of vanishing long-term gradients when using traditional RNNs for long sequences. Bengio *et al.* [32] stated that RNNs are only effective for short-term sequences and ultimately face a trade-off scenario between efficient learning by gradient descent and latching long-term

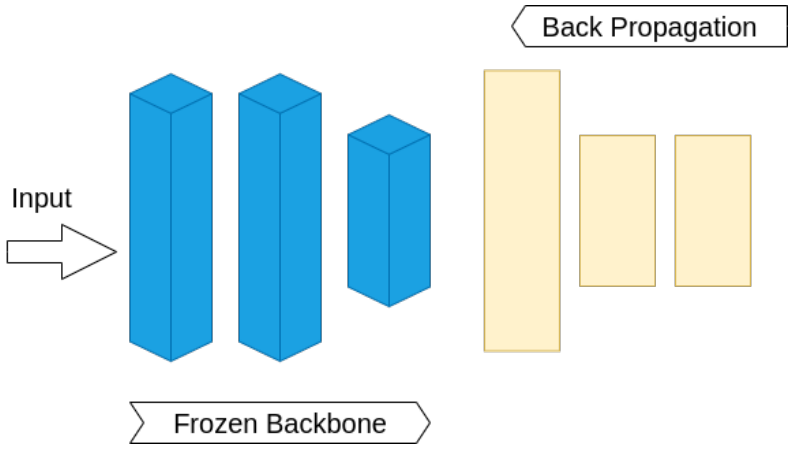


Figure 3.6: Transfer learning with frozen backbone

information. This is solved by the cell state and a series of gates - which control the level of information retained or discarded within each time step. There are three gates in an LSTM cell - namely input, output, and forget gates. Figure 3.5 shows an LSTM unit and its various components. At any given time step, the unit has access to h_{t-1}, previous hidden state, C_{t-1}, previous cell state and x_t, current input. The gates selectively learn the level of information to be retained or discarded during training.

The CNN-LSTM architecture combines the strengths of CNN and LSTM by capturing spatial and temporal features in the given data. The local features captured by the CNN are fed to the LSTM that captures the temporal features along the input sequence. In this case, the CNN can focus on learning the specific spatial features (such as turn indicators), and the LSTM can focus on the spatial maps produced to have a much easier pattern recognition across frames.

3.4 Transfer Learning

DL models require abundant data to learn reliable generic features. For most applications, however, there is a lack of sufficient data to build models. Transfer Learning helps work around this challenge by augmenting learned features from a model trained on a large dataset. This helps the new model converge faster and leads to better generalization in the final model. In other words, in transfer learning, a network trained in the source domain for a specific task is repurposed as a feature extractor in the target domain for a different task. The source domain here refers to the original context of the dataset on which the model was trained, and the target domain refers to the context of the dataset on which it is applied. The models used for transfer learning are also referred to as the backbone network.

There exist different networks which set the respective benchmarks in their domains. The most popular models used for transfer learning include ResNet-18 [33], YOLOv3 [34] and GoogleNet [35].

Depending on the volume of the target dataset, the backbone model weights can

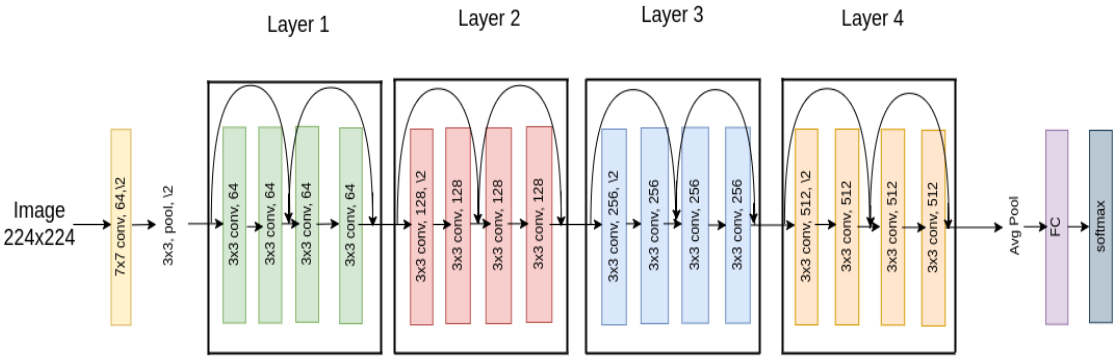


Figure 3.7: ResNet-18 architecture

be either frozen (Figure 3.6) or fine-tuned for the specific application. If the target dataset is sufficiently large, the backbone can be fine-tuned to improve the features.

3.4.1 ResNet-18

ResNet was introduced in He *et al.* [36] to ease the training of extremely deep networks by using shortcut connections that perform identity mapping. Consequently, even though the architecture was approximately 8x deeper than VGG Networks, it maintained lower complexity. The network topped the leader-board across multiple prevalent dataset challenges like ImageNet [37] in the ILSVRC-2015[1] in detection and localization and COCO. Figure 3.7 shows the architecture of ResNet-18.

3.5 Loss

The loss function measures the deviation of the model predictions to observations. It is used to find an optimal set of parameters that minimize the disparity between the two values. In this section, we will introduce various loss functions used in our experiments.

3.5.1 Cross Entropy

Cross Entropy loss is among the most popular loss functions for classification problems [38], [39]. It is defined as

$$L_{CE}(p_i, t_i) = -\sum_{i=1}^{n} t_i \log(p_i) \tag{3.1}$$

where t_i is the true label and p_i is the softmax probability for the i^{th} class, predicted by the model. All the classifier models in the thesis are trained with binary cross-entropy loss.

[1] https://image-net.org/challenges/LSVRC/2015/

3.5.2 Laplacian Negative Log Likelihood

Laplacian Negative Log Likelihood (NLL) is used instead of L1 loss for regression models to estimate the mean μ and uncertainty b in prediction for each sample. This loss can be interpreted as the NLL of the annotation under a Laplacian distribution. The probability density function of a Laplacian distribution is given by

$$p_{\mu,b}(y) = \frac{1}{2b} e^{\frac{-|y-\mu|}{b}} \tag{3.2}$$

where b is the scale parameter, μ is the mean of the distribution and y is the value of the random variable. In a supervised learning setup, μ and b depend on the input x. Hence they can be written as $\mu(x), b(x)$, respectively. For n statistically independent samples, the joint likelihood is given by :

$$L(\mu, b; x) = \prod_{i=1}^{n} \frac{1}{2b(x_i)} e^{\frac{-|y_i-\mu(x_i)|}{b(x_i)}} \tag{3.3}$$

where x_i is the input features, $b(x_i)$ is the scale parameter, $\mu(x_i)$ is the regressed mean and y_i is the true value for the i^{th} sample. We may also write the NLL as:

$$\ln(L(\mu, b; x)) = \sum_{i=1}^{n} -\ln(2b(x_i)) - \frac{|y_i - \mu(x_i)|}{b(x_i)}$$

$$= \sum_{i=1}^{n} -1 \left(\ln(2) + \ln(b(x_i)) + \frac{|y_i - \mu(x_i)|}{b(x_i)} \right)$$

Which leads to the modified Laplacian loss as

$$l(\mu, b; x) = \ln(b(x)) + \frac{|y - \mu(x)|}{b(x)} \tag{3.4}$$

where μ and b are predicted by the model for each sample i with input feature x and true label y.

Laplacian NLL is treated similarly to L1 loss and preferred in situations where the outliers are required to be safely ignored.

3.5.3 Gaussian Negative Log Likelihood

Gaussian NLL loss is the L2 counterpart for regressor models which also predict uncertainity estimate σ alongside mean μ. Similar to Laplacian NLL, the Gaussian loss is interpreted as the NLL of the annotation under a Gaussian distribution defined by μ and σ.

The probability density function for a Gaussian distribution is given by

$$p_{\mu,\sigma}(y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2} \tag{3.5}$$

where σ is the standard deviation, μ is the mean of the distribution and y is the value of the random variable.

Then the joint likelihood of n statistically independent samples depending on input x_i is

$$L(\mu, \sigma; x) = \prod_{i=1}^{n} \frac{1}{\sigma(x_i)\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y_i-\mu(x_i)}{\sigma(x_i)}\right)^2} \tag{3.6}$$

$$= \frac{1}{\sqrt{2\pi}} \prod_{i=1}^{n} e^{-\ln(\sigma(x_i))-\frac{1}{2}\left(\frac{y_i-\mu(x_i)}{\sigma(x_i)}\right)^2}$$

where $\sigma(x_i)$ is the standard deviation, $\mu(x_i)$ is the regressed mean estimated from input feature x_i and y_i is the true value for the label of the i^{th} sample.

$$L(\mu, \sigma; x) = \frac{1}{\sqrt{2\pi}} e^{\sum_{i=1}^{n} -\ln(\sigma(x_i))-\frac{1}{2}\left(\frac{y_i-\mu(x_i)}{\sigma(x_i)}\right)^2}$$

This gives us the modified Gaussian loss as:

$$l(\mu, \sigma; x) = 2\ln(\sigma(x)) + \left(\frac{y-\mu(x)}{\sigma(x)}\right)^2 \tag{3.7}$$

where the model predicts σ and μ for each sample with features, x and true label y

3.6 Evaluation Metrics

In our experiments, we analyze lane change detection and regression of the timings. The former problem is treated as a binary classification task, with the positive classes being lane change instances with the turn indicator switched on. For the latter problem, we take the start and end of lane change as output variables. In the following, we explain the evaluation metrics for these two problems.

3.6.1 Classification

3.6.1.1 Accuracy

Accuracy is a generic performance measure for classification problems. It is given by,

$$\text{Accuracy} = \frac{TP + TN}{N} \tag{3.8}$$

where TP denotes true positives, TN denotes true negative and N is the total count of classified samples ($TP + TN + FP + FN$).

Although easy to interpret, accuracy comes with its own challenges, like the inability to account for the distribution of samples in the dataset. If the classes are highly imbalanced in the training set, accuracy as a standalone metric fails as a performance gauge. In such cases, or when the application considers one class more important than others, alternate measures like precision, recall, or F1 score can be used.

3.6.1.2 Precision

Precision is used as a measure in applications where the cost of false positives is higher than false negatives. In this case, false positives can be inferred as predicting indicator action or lane change when the vehicle is not performing a maneuver. This can cause the system to be triggered unnecessarily.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.9}$$

where TP and FP refer to true positives and false positive counts, respectively.

3.6.1.3 Recall

Recall, on the other hand, is used as a measure in settings where the cost of false negatives is higher than false positives. With respect to the thesis, a false negative instance is where the target vehicle performs a lane change or uses the indicator, but the model fails to identify and correctly classify it and can result in an accident in the worst case.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.10}$$

where TP and FN denote true positive and false negative counts respectively.

3.6.1.4 F1

In cases where a balance between precision and recall is desirable in an imbalanced dataset, F1 can be used to gauge the performance of the models.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.11}$$

3.6.2 Regression

3.6.2.1 MAE

Mean Absolute Error, or MAE, is computed by averaging the absolute differences of the model's predictions from the actual values for the mini-batch.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{3.12}$$

where $\hat{y}_i$ is the predicted value and y_i is the true value of the sample.

3.6.2.2 RMSE

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{3.13}$$

where $\hat{y}_i$ is the predicted value and y_i is the true value for the sample.

3.6.3 Uncertainty Estimation

Autonomous Vehicle systems have to deal with a broad set of unpredictable scenarios due to the nature of the highly dynamic operational environment. Operational Design Domain (ODD) was introduced to consider a set of constrained operational environments consisting of only a subset of all possible situations the system can encounter [40]. Safety requirements for EVs are built over the ODD specifications. However, additional sources can introduce uncertainty to the system apart from the environment. Specifically for perception modules, these sources include sensor properties, model uncertainty, and scenario coverage [41].

Uncertainty can be broadly classified as aleatoric and epistemic [42], [43]. Aleatoric uncertainty or the irreducible variance is caused by the inherent randomness between the data samples [44]. In contrast, epistemic or model uncertainty refers to model ignorance due to insufficient information to describe the ODD [45]–[47]. Model uncertainty can grow arbitrarily high when considering scenarios outside the ODD, such as variations in weather, and can be reduced by including more data that explains the environment better. In most cases, the two types of uncertainties are entangled and referred to as predictive uncertainty [42]. Depending on the application, it can be beneficial to differentiate between the components. For example, in active learning settings, the aleatoric component can be ignored while the epistemic component can be used to measure the usefulness of the sample [48]. Different disentangling techniques [49] such as MC-Dropout [50], Ensembles [51], Flipouts [52], etc can be used if the application demands a separation of the two components.

For the purpose of evaluating the quality of uncertainty estimation, different metrics can be used depending on the nature of the task. Predictive entropy and mutual information [53] can be used to evaluate classification models, while NLL can be used for classification and regression settings. However, to calibrate the models, more intuitive visual metrics such as Expected Calibration Error (ECE) [54], [55] and Area Under Sparsification Error (AUSE) [56] are easier.

3.6.3.1 Expected Calibration Error (ECE)

ECE is used to measure the deviation of the expected confidence from the expected accuracy. Confidence is defined as the likelihood of the model to make the correct

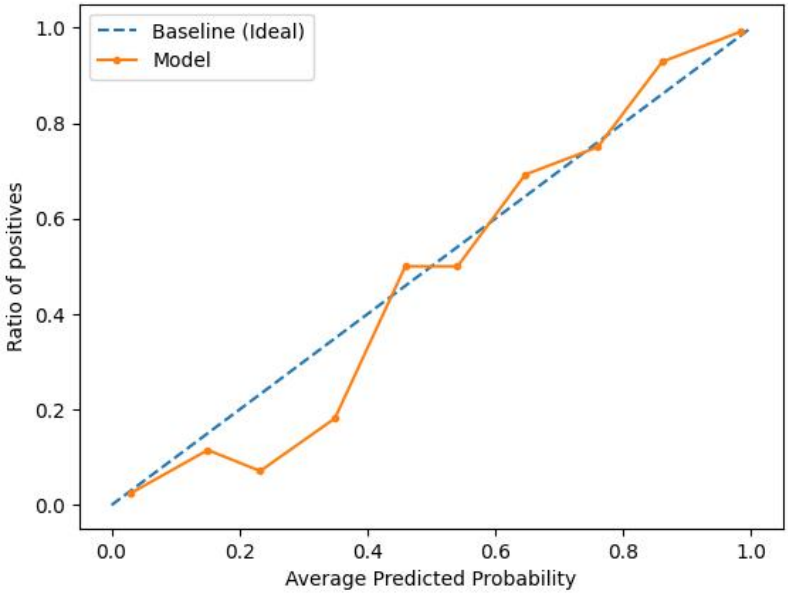


Figure 3.8: Example of Calibration Plot. In this case, the model is less confident about samples with a predicted positive probability of less than 0.5 but well-calibrated for higher values.

predictions. This contrasts with accuracy, defined as the proportion of correct predictions in a given validation set. Ideally, in a well-calibrated model, the model confidence and accuracy are proportional. The prediction with a predicted positive probability of 20% is expected to be correct 20% on average. To compute the measure, the predictions from the models are sorted into K fixed bins ($K = 10$ in our experiments). The ECE is then calculated empirically by

$$\text{ECE} = \sum_{i=1}^{K} P(i) \cdot |o_i - e_i| \tag{3.14}$$

where P_i is the fraction of all instances that fall in bin i, o_i is the true fraction of positive instances in bin i and e_i is the mean of post-calibrated probabilities in bin i. For a well-calibrated model, the ECE value is close to 0. Figure 3.8 shows an example ECE plot where the model is calibrated well only for the positive class (predicted positive probability > 0.5).

3.6.3.2 Area Under Sparsification Error (AUSE)

AUSE is a relative measure extracted from a sparsification plot consisting of an oracle and the model curve. The sparsification plot is used to quantify the closeness of the uncertainty predicted by the model to the ideal uncertainty. The ideal uncertainty is computed by using the deviation of the predicted metric from the actual label as a proxy. The predicted uncertainty is directly obtained from the model for each prediction. To build the oracle curve, the samples are all sorted in descending order

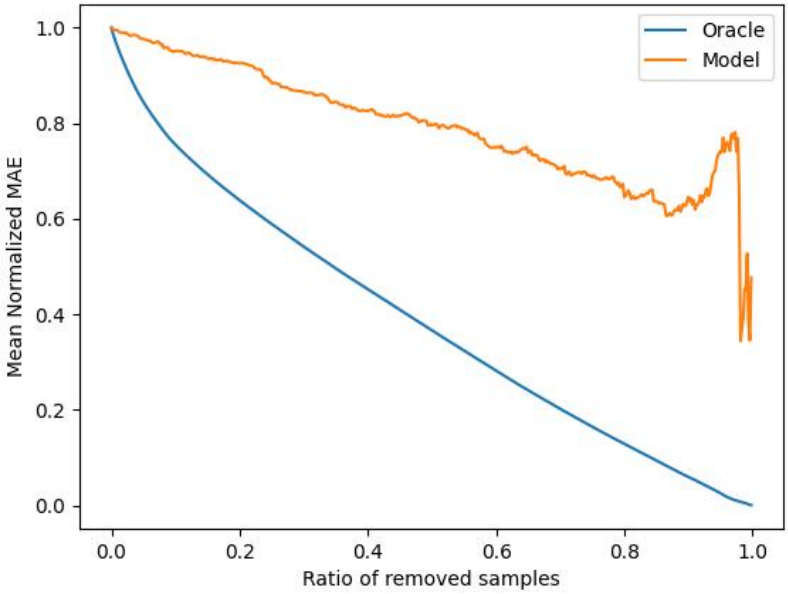


Figure 3.9: Example of Sparsification plot with the Oracle and Model curves. The gap between the curves shows that when sorting and removing entries with predicted uncertainties, it deviates from the expected ideal behavior. The closer the curves, the better the estimate.

of ideal uncertainty and then sequentially dropped from the set. The average error of the remaining samples is computed at each step and normalized to obtain the oracle curve. The process is repeated with the predicted uncertainty to obtain the model curve. Ideally, the curves should coincide. To quantify the deviation of the predicted uncertainty from the ideal nature, the area between the curve is computed and presented as AUSE. The lower the value, the better the model predicts the uncertainty. Figure 3.9 shows an example sparsification plot, and the AUSE is computed as the difference in area between the curves.

4

Methods

The chapter includes details about the dataset used for the experiments, followed by the model architectures chosen, with emphasis on the specific layers and the implementations for lane change detection with turn signals and timing predictions.

4.1 Data

For the purpose of the thesis, internal Zenseact data was used. It comprised manual annotations for lane change instances of vehicles ahead of the ego vehicle made during various test runs of the development vehicle. The annotations include the ego vehicle information and target vehicle ID followed by the maneuver's start-time and end-time. The annotations are specifically made for lane change instances performed with turn signals since they are considered a significant cue for cut-ins. The negative instances included instances where the target vehicle kept the same lane.

For both positive and negative instances, a buffer period of 5s before and after the marked times was considered to extract the video from the logs (Fig. 4.1). This is then passed through the internal object detection pipeline to track the target vehicle across the frames using the vehicle ID stored in the annotation. The final stage of the pipeline saves the video frames for the selected duration along with an additional file that includes information from the object detection pipeline such as the bounding box across the frames for the target vehicle and an active flag which is set to True if the object was detected in the respective frame.

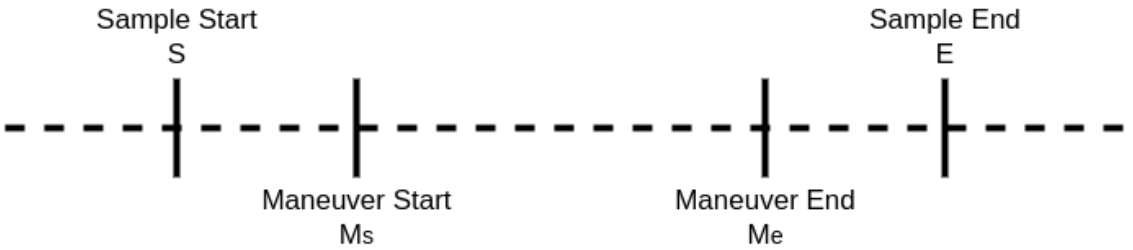


Figure 4.1: Timing diagram. M_s and M_e are marked using indicators as confirmation of intent

Perfect negative instances, i.e., where the exact vehicle is shown to keep and change lanes are also included in the dataset apart from the exclusive samples.

4.1.1 Pre-Processing

Since the data points were video instances, the pre-processing stage involved mining the instances from the storage using the annotations and processing the frames individually for the models. Once the data files were extracted and split into train and validation sets (3:1), the subsets were generated according to the appropriate strategy for the experiment depending on the model (classifier or regressor). Table 4.1 shows the final distribution across the binary class.

One of the aims was to identify the level of context, also referred to as RoI (Region of Interest), necessary for the optimal performance of the network. The smallest possible crop as input to the model would only include the bounding box around the target vehicle while the highest context is obtained by passing the entire scene. The crop is resized by scaling it using bicubic interpolation, to the closest preset dimension. Then it is zero-padded to meet the other dimension requirement while keeping the vehicle centered.

The output from the object detection pipeline also included an active flag for instances where the bounding box information was missing. This was useful for instances where the target vehicle was obstructed by other vehicles or when it moved out of the frame. It was discarded for the chosen subset if the number of missing flags was more than 50% of the duration. For the other instances, the bounding box for the missing frames was copied from the previously known frame. The assumption was that the vehicle movement would be minimal between consecutive frames since the sensors, the camera, in this case, function at a high frequency.

The dataset class included hyper-parameters such as control over the level of additional context, augmentation, and dataset restriction for regressors. The number of possible variations in the road scene, including the target type, weather conditions, road layouts, etc., implies that having a dataset exhaustively describes the problem is also challenging. Considering this and the number of unique data points from the dataset, data augmentation was included to address over-fitting. Random brightness augmentation and horizontal flips were performed for all frames in a given video sample. The data for each batch was finally stacked as a 5D tensor (N, D, C, H, W), where N is the batch size, D is the depth or length of the video subset, C is the number of channels (3 in the case of RGB images), H is the height and W, the width of the image.

Class	Count
Positive	418
Negative	312

Table 4.1: Raw dataset distribution

4.1.2 Subsets

Training VAR models are made easy using fixed-length video subsets. To make the best use of limited data, each video instance was used to generate multiple subsets

of pre-defined length. Depending on the experiment, the subset generation strategy varied slightly.

4.1.2.1 Classifier Dataset

For each sample, the random subset is generated by sampling between M_s and M_e and then extending the sample equally until the required duration is attained. Sampling the midpoint was also restricted such that the resulting subset would be within [M_s, M_e]. Table 4.2 shows the final distribution of the dataset for classifier models.

Class	Train	Validation
positive	2836	948
negative	2133	675

Table 4.2: Classifier dataset distribution with 10 subsets per sample

4.1.2.2 Regressor Dataset

The original dataset was first filtered only to include positive samples. Since the start and end times were regressed separately, depending on the requirement, the sampling strategy was also different.

- Regressing Time To Lane Change (TTLC) Start: In this case, the midpoint for the subset is sampled from [0, $M_s - t/2$] where t is the subset duration. Table 4.4 shows the properties of the final dataset used for regressor models that estimate lane change start.

- Regressing Time To Lane Change (TTLC) End: To generate subsets to regress the end of the maneuver, the midpoint for the subset is sampled from [M_s, $M_e - t/2$]. Table 4.3 shows the properties of the final dataset used for regressor models that estimate lane change end.

In both cases, the generated subset was confirmed to be a valid subset (checking the adequate number of bounding box detections across frames).

Measure	Train	Validation
Count	2902	960
Mean	50.0713	44.0843
SD	86.1733	38.8939

Table 4.3: Summary of TTLC End dataset with 10 subsets per sample.

Measure	Train	Validation
Count	2282	696
Mean	26.8698	29.4094
SD	20.3828	21.0612

Table 4.4: Summary of TTLC Start dataset with 10 subsets per sample

4.2 Models

4.2.1 3D-CNN

The first objective was to train a model end-to-end on the dataset that can simultaneously capture spatial and temporal features. The chosen architecture was one layer of 3D-CNN [57] followed by the classification or regression head. Tran *et al.* [57] have experimentally shown that small filters, 3x3x3 perform well in capturing the necessary video features. The choice of not having too deep a network is also to prevent over-fitting by forcing the model to learn more generic features.

4.2.2 3D-CNN with ResNet Backbone

The next variation was to introduce a network that could provide a focus on spatial features before introducing the 3D-CNN. The objective was to improve performance by improving the features used by the 3D kernels.

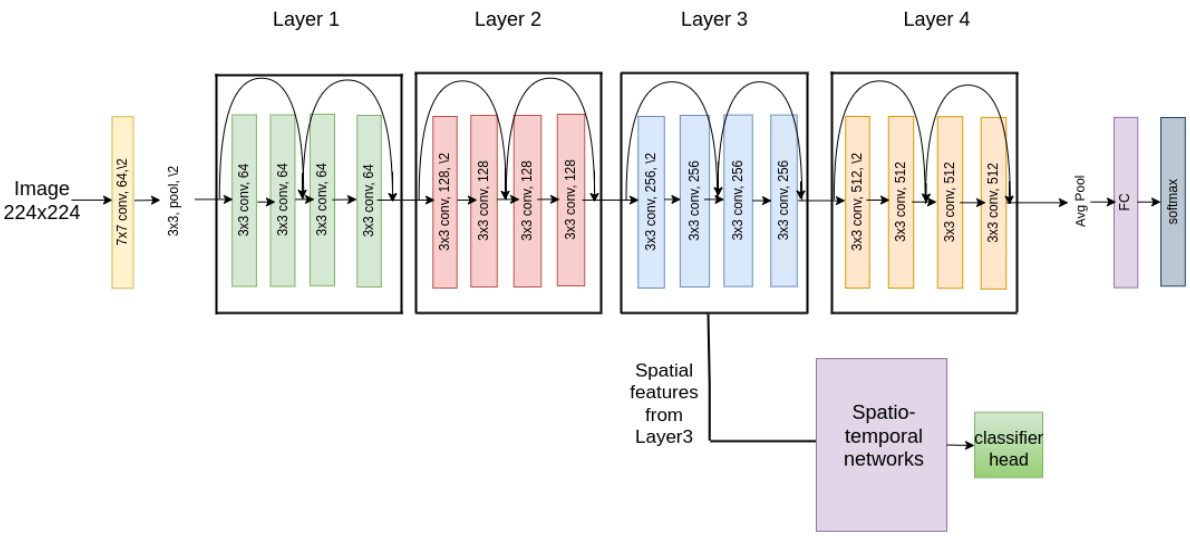


Figure 4.2: 3D-CNN with ResNet-18 Layer 3 architecture

More rich and complex features are extracted at deeper layers of convolutional networks. Therefore, to understand the impact of the complexity of features captured by the model for this task, three different variants, i.e., features extracted from Layers 2, 3, and 4 (shown in Fig. 3.7) are used as feature extractors for the spatio-temporal networks.

Fig. 4.2 shows an example of an experiment where the resized RoI is passed through the feature extractor, ResNet-18 Layer 3, and then used as input by 3D-CNN.

4.2.3 CNN-LSTM

As opposed to 3D-CNN, CNN-LSTM models give individual focus to both spatial and temporal features. This allows for more controlled experiments, where the effect of the features can be adjusted to understand better what kind of features impact the final results more.

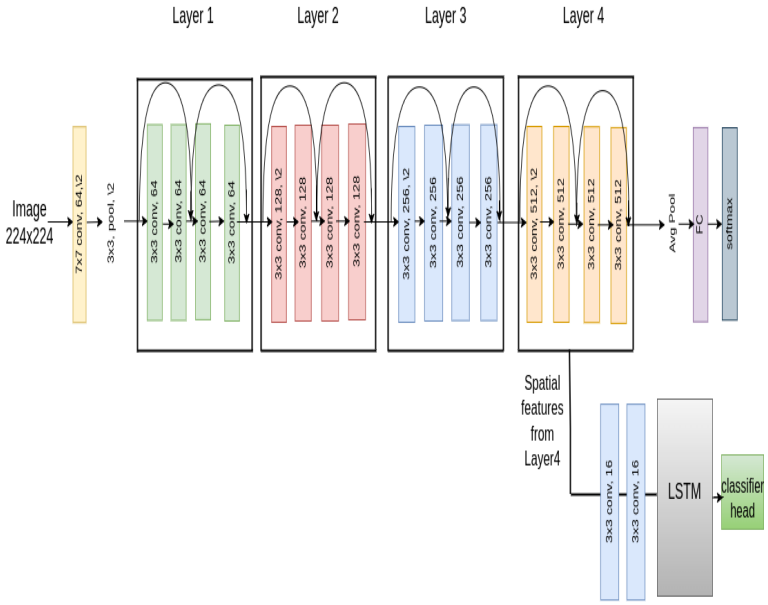


Figure 4.3: CNN-LSTM architecture

The chosen architecture for the backbone was pre-trained ResNet-18 followed by 2D-CNNs and LSTM, as seen in Figure 4.3. The input to the above architecture is the set of Region of Interest (RoI) from the video frames. Layer 4 of the pre-trained ResNet-18 extracted the spatial features passed to the LSTM to capture the temporal features. The entire model was trained with frame-wise loss as explained in 4.4.2.2.

4.2.4 Regression with Uncertainty

Generic regression problems have a single node in the last layer, which is used to regress the mean of the value of interest. However, this gives no information about the confidence of the prediction. With AD/ADAS application, where the actions taken on the predictions of these models have a serious impact, it is advantageous also to have a confidence or uncertainty metric along with the regressed value. This allows other systems down the pipeline to find a balance between confident predictions and very uncertain predictions. For this purpose, Laplacian and Gaussian NLL losses (as specified in Section 3.5.2 and 3.5.3) are used with two nodes in the final layer - the first node regressing the mean and the second node predicting the uncertainty parameter.

When using the Gaussian NLL, the first node is defined to regress the TTLC (μ), while the second node is used to predict the standard deviation (σ). Similarly, when implementing Laplacian NLL, the first node regresses μ, and the second node predicts the scale parameter, b.

4.3 List of Experiments

The thesis made use of ResNet-18 as a feature extractor based on the success of Izquierdo *et al.* [22], where multiple popular networks such as ResNet variants, GoogleNet, and AlexNet were benchmarked against each other for the specific task of lane change intent prediction. The authors found no significant performance difference across the different ResNet variants (ResNet-18, ResNet-50, ResNet-101), and ResNet variants were ahead of the other networks in terms of anticipation and accuracy of lane change maneuver prediction.

The final list of experiments and their details are as follows.

4.3.1 Classification

The classification models aimed to detect lane change instances from the manually annotated dataset with turn indicators.

1. 3D-CNN on raw frames
2. 3D-CNN with ResNet-18 Layer 2 Features
3. 3D-CNN with ResNet-18 Layer 3 Features
4. 3D-CNN with ResNet-18 Layer 4 Features
5. CNN-LSTM with ResNet-18 Layer 2 Features
6. CNN-LSTM with ResNet-18 Layer 3 Features
7. CNN-LSTM with ResNet-18 Layer 4 Features

4.3.2 Regression with Uncertainty

Regressors aimed to predict the start and end of the maneuver from only the positive instances. The experiments are repeated for both TTLC start and end values independently.

1. 3D-CNN on raw frames
2. 3D-CNN with two layers on raw frames
3. 3D-CNN with ResNet-18 Layer 2 Features
4. 3D-CNN with ResNet-18 Layer 3 Features
5. 3D-CNN with ResNet-18 Layer 4 Features
6. CNN-LSTM with ResNet-18 Layer 2 Features

7. CNN-LSTM with ResNet-18 Layer 3 Features
8. CNN-LSTM with ResNet-18 Layer 4 Features

4.4 Implementation

4.4.1 Settings

PyTorch Lightning was the preferred framework to run the experiments. The various hyper-parameters that were controlled are as follows.

- Subset length - On average, the video instances consist of approximately 180 frames. However, we would only use a small subset of the original video to train the model. A sliding-window approach can be used to execute the model on a longer video. From the literature survey, it was concluded that 16 frames would be an ideal duration for the subset.
- Subset count - Multiple subsets from the same video instance were sampled to best use the dataset. Data augmentation over the subsets increased the number of data points effectively used to train the models. However, this increased the chances of the model overfitting the training data.
- Dimension - Since the input to the models was a sequence of frames, the image dimension was one of the parameters that could be tuned. The smallest possible image dimension among 50x50, 100x100, and 150x150 is chosen depending on the architecture and filter sizes. However, for models with ResNet-18 backbone, the image dimensions were fixed at 224x224.
- Region of Interest (RoI) - For the initial implementations, the bounding box was the RoI. However, part of the research goal is also to identify the optimum RoI beyond the target vehicle, which is necessary to produce good inferences. This was achieved by centering the target vehicle and increasing the sides while keeping the aspect ratio constant. Fig. 4.4 shows the different levels from no additional context to 50% additional context around the target.
- Batch Size - Batch size for the experiments was maintained at 8 and 16.
- Early Stopping - The training and validation metrics were tracked across epochs, and validation loss was used to halt the training process with a patience parameter of 10 epochs. A scheduler was also set up to reduce the learning rate by a factor of 0.1 when performance stagnates.

4.4.2 Loss Function

During the course of the experiments, we used two implementations of Cross Entropy Loss for classification tasks – generic and frame-wise. Mean reduction is used by default to decouple the learning rate from the mini-batch size.

Figure 4.4: Increasing context : 0-50% additional RoI with padding. This is an example of a cut-in scenario with additional context. Increasing the RoI also shows the lane markers being crossed by the vehicle that could be used as a feature

4.4.2.1 Generic Loss

This refers to the generic method where the loss is computed using only the final output. This is used for all regressor models and classifier models without LSTM layers.

4.4.2.2 Frame Wise Loss

Since the input to the models were video subsets, all the frames were assigned the same label and used to compute the cross entropy loss for each step. This allows for a frame-wise evaluation and also understand prediction performance across time steps. This implementation is restricted to only LSTM-based models.

4.4.3 Evaluation

Different techniques were used to evaluate the models depending on the experiment category. Classifiers were primarily evaluated based on the F1 Score and other metrics like Precision, Recall, and ECE value. A naive approach was to use the final prediction metrics such as accuracy, recall, and precision values. Suppose the architecture allowed evaluation on a frame level. In that case, these are also computed at intermediate points, both as a sanity check and to understand better the network's earliness to make accurate predictions. All metrics of interest, including confusion matrices and loss values, are logged at every step of the training and validation cycle.

5

Results and Analysis

5.1 Classifier

5.1.1 Summary of Performance

To measure the performance of the classification models, multiple metrics, including accuracy, precision, recall, and F1 were tracked. The values shown are derived from 3 replicate runs for each model. The models were trained by sampling at most ten subsets from each raw annotated instance.

Model	Accuracy	
	Mean	Stand. Dev
3DCNN	0.6839	0.0089
ResNet18-L2-3DCNN	0.7217	0.0200
ResNet18-L3-3DCNN	0.7344	0.0120
ResNet18-L4-3DCNN	0.7428	0.0046
ResNet18-L2-LSTM	0.7369	0.0089
ResNet18-L3-LSTM	**0.7697**	**0.0028**
ResNet18-L4-LSTM	0.7539	0.0151

Table 5.1: Summary of performance - Accuracy

The dataset was inherently biased towards positive samples, so accuracy was not the preferred metric for measuring performance. However, it provided a very broad overview of the model performance. The accuracy value had saturated at 77% across models, and to better understand the mis-classifications, the other metrics were helpful.

Model	F1-score	
	Mean	Stand. Dev
3DCNN	0.7520	0.0101
ResNet18-L2-3DCNN	0.7801	0.0135
ResNet18-L3-3DCNN	0.7899	0.0077
ResNet18-L4-3DCNN	0.7901	0.0060
ResNet18-L2-LSTM	0.7883	0.0053
ResNet18-L3-LSTM	**0.8137**	**0.0023**
ResNet18-L4-LSTM	0.7971	0.0153

Table 5.4: Summary of performance - F1 Score

Model	Precision	
	Mean	Stand. Dev
3DCNN	0.6941	0.0101
ResNet18-L2-3DCNN	0.7251	0.0201
ResNet18-L3-3DCNN	0.7350	0.0164
ResNet18-L4-3DCNN	0.7557	0.0180
ResNet18-L2-LSTM	0.7439	0.0117
ResNet18-L3-LSTM	**0.7713**	**0.0202**
ResNet18-L4-LSTM	0.7683	0.0054

Table 5.2: Summary of performance - Precision

Model	Recall	
	Mean	Stand. Dev
3DCNN	0.8206	0.0152
ResNet18-L2-3DCNN	0.8446	0.0310
ResNet18-L3-3DCNN	0.8544	0.0203
ResNet18-L4-3DCNN	0.8294	0.0316
ResNet18-L2-LSTM	0.8386	0.0097
ResNet18-L3-LSTM	**0.8611**	**0.0035**
ResNet18-L4-LSTM	0.8284	0.0288

Table 5.3: Summary of performance - Recall

Analyzing precision and recall values showed that false positives were more common than false negatives. In this case, the false positives refer to samples incorrectly predicted to have turn indicators and changed lanes. False negatives are samples that

were predicted negative when it was performing lane change with turn indicators. The results are promising since there were fewer false negatives, and the application domain (ADAS) would consider this a higher priority. Identifying all positive instances is more critical than misclassifying negative instances.

Features extracted from ResNet-18 Layer 3, followed by the CNN-LSTM model, had the best performance across all four metrics for classification tasks. The results were consistent with a standard deviation of 0.0023 for the F1 score.

Overall, the complex models surpassed the baseline easily for the specific task. The ResNet models were marginally ahead of the 3D-CNN equivalents. This could imply that the 3D-CNN's technique of capturing the spatial and temporal features from the data was only less effective by very little margin. This could also give a brief idea about the nature of the action. CNN-LSTM models are more capable of focusing on smaller patches of spatial features across time which could explain the marginal improvement. The lack of standard indicator design and the position of the same towards the edge of the bounding box could also cause a difference in performance as convolutions do not preserve edge and border features as easily as central features.

5.1.2 Effect of Additional RoI

To better understand the effect of increasing context around the vehicles for the classification task, the RoI was varied from no additional context to 50% extra context around the vehicle. The metrics were measured across different levels of context, and the trends were analyzed.

Both accuracy (Figure 5.1) and F1 scores (Figure 5.2) improved with more context around the vehicle. On deeper analysis, we found that the bounding boxes could occasionally be wrapped too tight around the object of interest, in this case, the target vehicle. This would mean that essential information could be lost when cropping the bounding coordinates from the scene. Since indicators are placed close to the vehicles' edge and are the major cue to be picked up by these models, a few misclassifications can be potentially avoided by having additional context. All three CNN-LSTM variants were consistently better than their 3D-CNN counterparts.

It is also essential to note the overall smooth trend upward without extreme differences in performance for all models. This implies that the features from the edges, which now move closer toward the center with additional context, are more definitive. If the additional context allowed the model to learn radically new features, it is more likely to observe a significant change in performance.

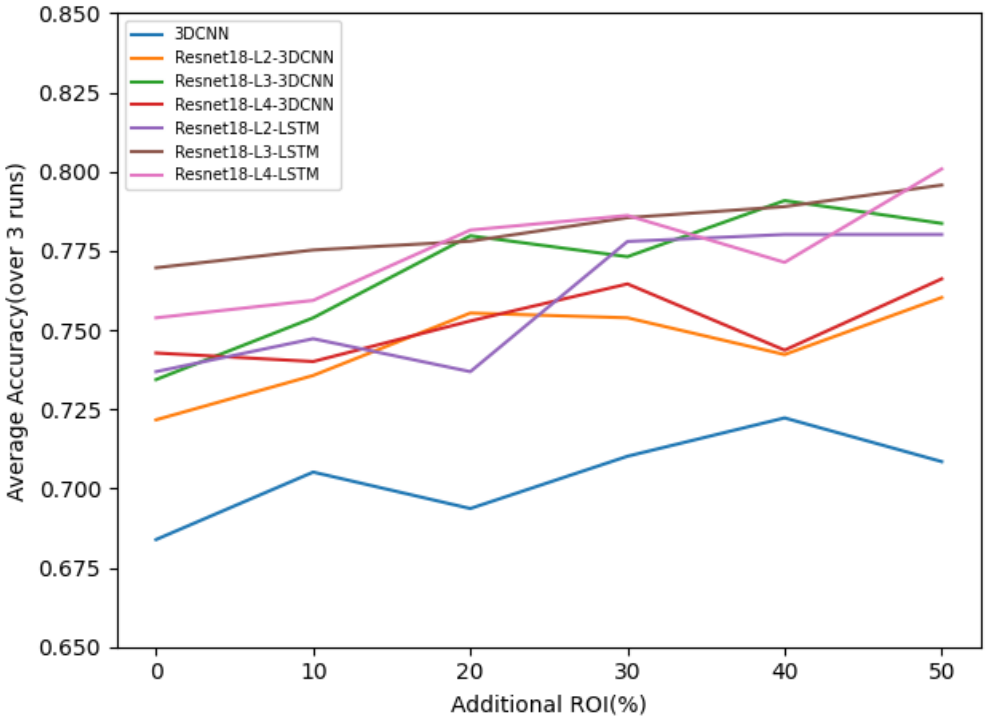


Figure 5.1: Accuracy vs Additional RoI

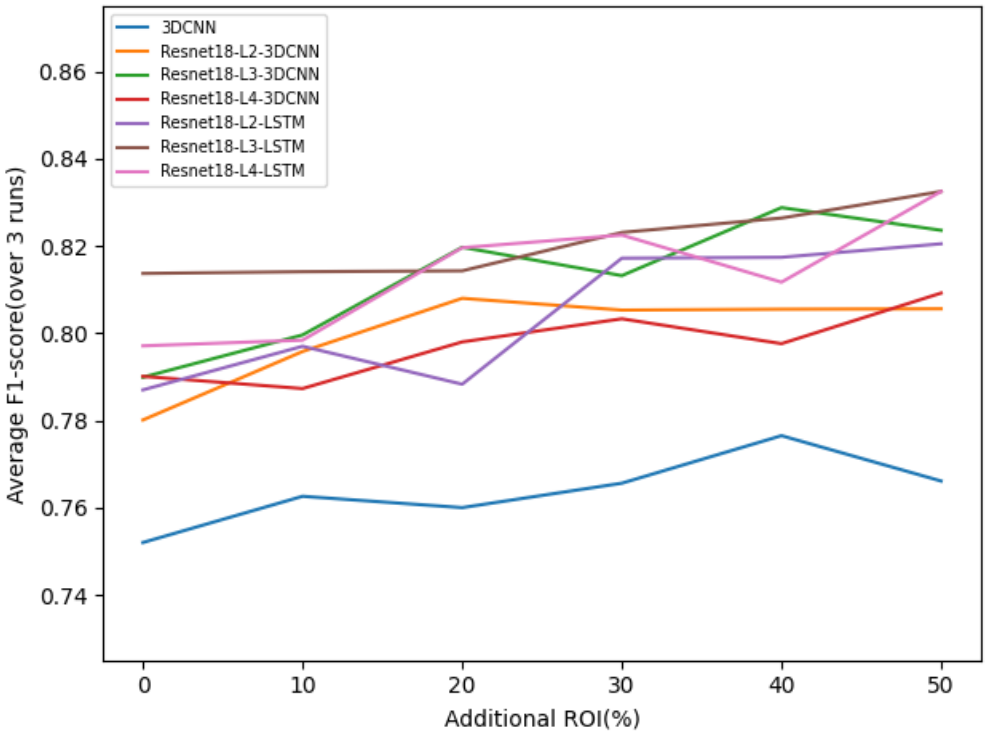


Figure 5.2: F1-score

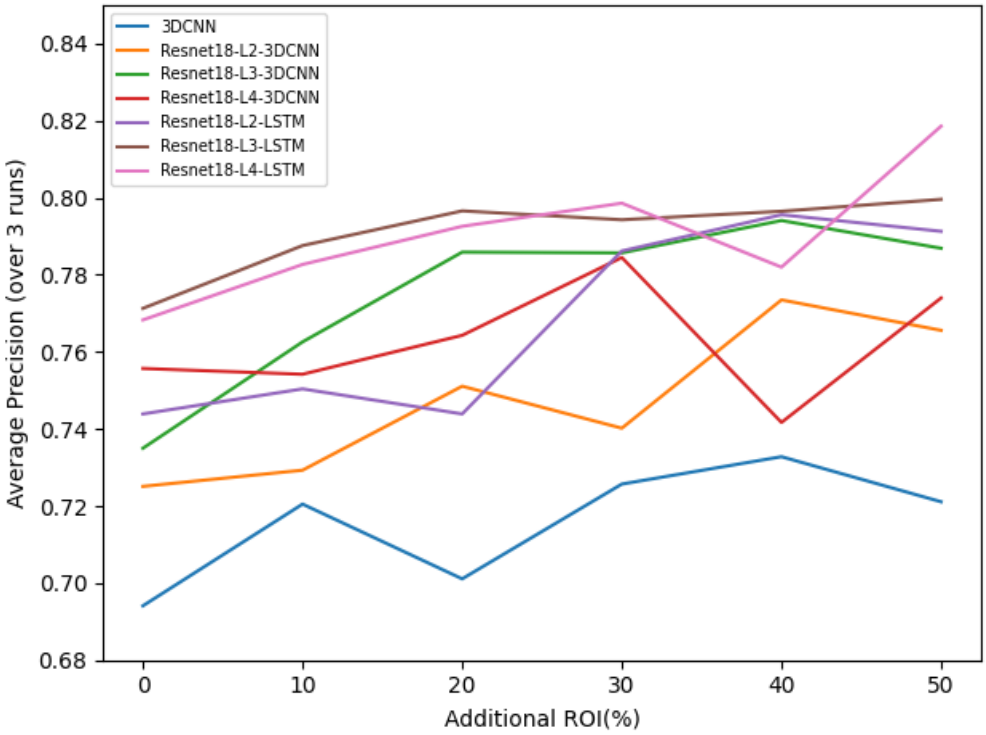


Figure 5.3: Precision

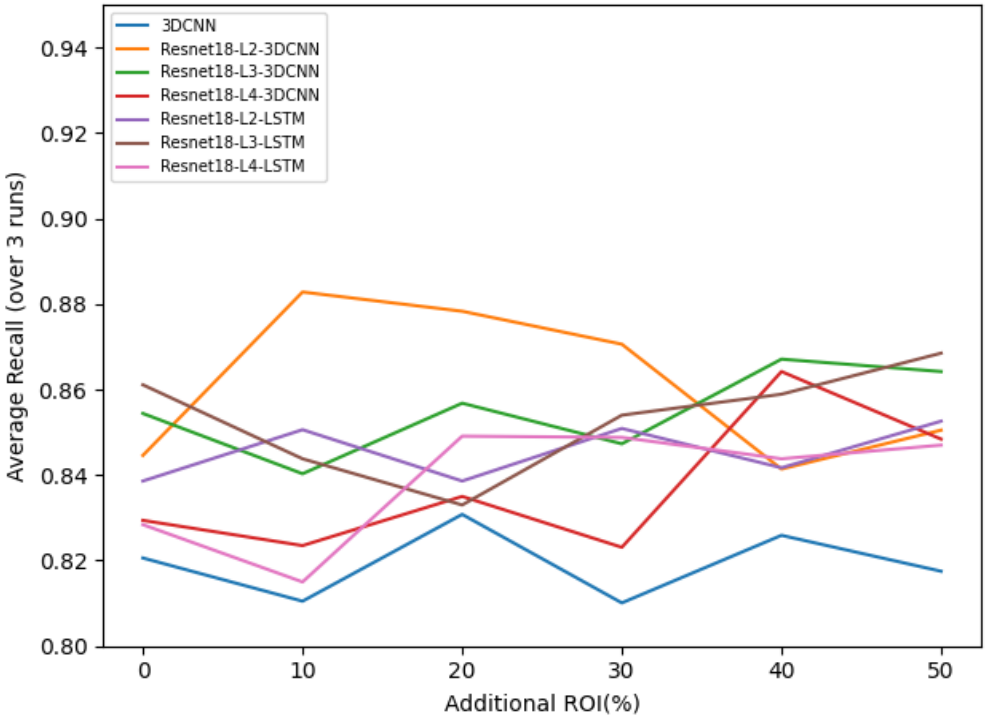


Figure 5.4: Recall

For precision curves (Figure 5.3), it was observed that the addition of context improved performance to 30% for all variants, after which ResNet-18-L4-3DCNN had a very sharp reduction. Recall (Figure 5.4) had a more flat response to the extra context. ResNet-18-Layer2-3DCNN had an interesting performance graph where it improved significantly for the first step. However, the performance continued to degrade from that point.

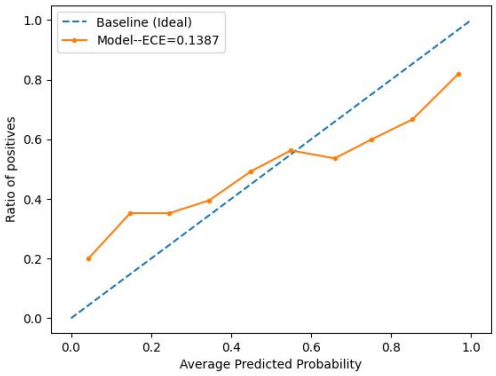
The fine balance in additional context is based on several factors - primarily the number of vehicles around the target in close proximity, distance to the target, and the ego lane position with respect to the road layout. Increasing context comes with the risk of including more noise sources, such as lights from on-coming vehicles. Vehicles in close proximity could also be challenging when expanding context since focusing on the right target would be challenging. This could imply that adding context, specifically around the target, needs to be catered to the different traffic scenarios.

5.1.3 Best Performing Model - ECE

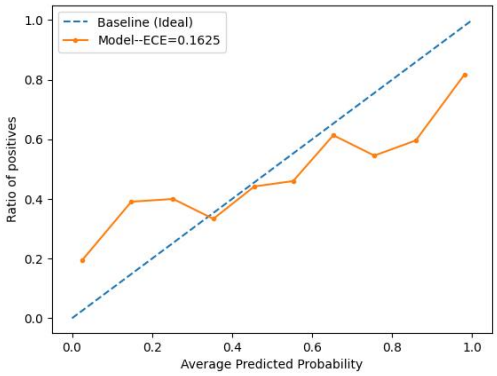
The best-performing model among the experiments was chosen to be further analyzed from a calibration point of view since it was not only sufficient to produce the right results but also to have a measure of confidence in the results. CNN-LSTM model over ResNet-18 Layer 4 features was chosen as it produced the best results with an additional context of 0.5 between all the other variants. Since the model was LSTM-based, it was possible to evaluate it on a frame level instead of a sample level.

Regarding absolute values, Frame 0 inferences were significantly better than the other steps (ECE=0.1325). Predicting if a vehicle has its indicators turned on or making a lane change maneuver from just one frame is impossible since the class has to be confirmed through blinking action or moving towards the center of the adjacent lane; both require at least one more frame for reliability. This could, however, give more information about the possible nature of the subsets where the turn indicator actions were more prominent in the earlier frames than later. The model could have learned this bias and based the predictions on this specific characteristic of the dataset. The low ECE value obtained from the first frame predictions can also be a limitation of the chosen metric.

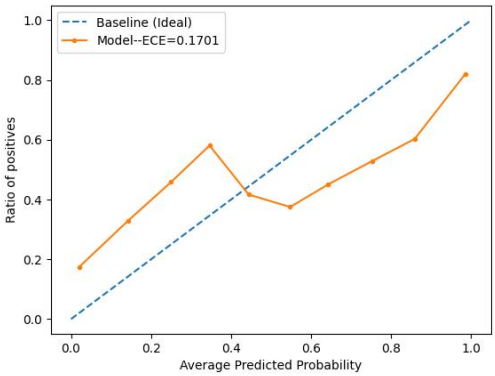
Overall, the calibration is observed to deteriorate until frame eight and then slowly improve with abrupt changes in performance until the very last frame. It also consistently overestimates the positive predictions beyond 70% predicted positive probability from all frames. Figures 5.5, 5.6, and 5.7 show the frame-wise ECE plots for the best-performing classifier model.



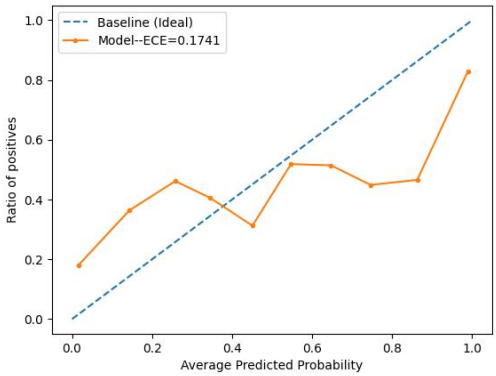
(a) Frame 0



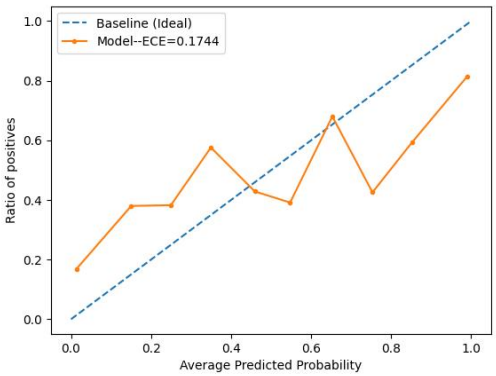
(b) Frame 1



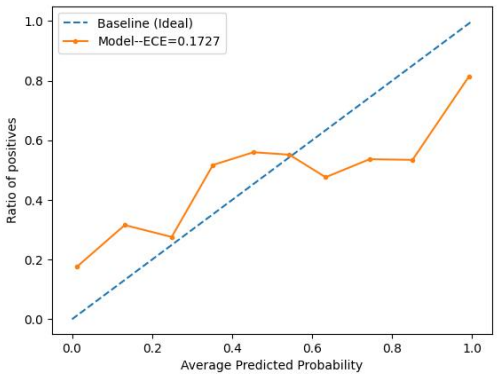
(c) Frame 2



(d) Frame 3

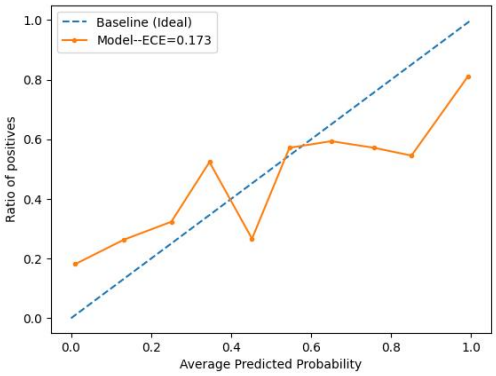


(e) Frame 4

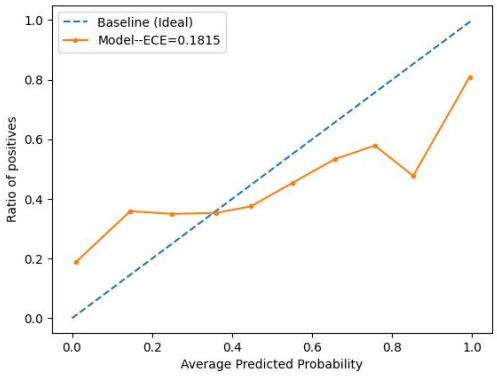


(f) Frame 5

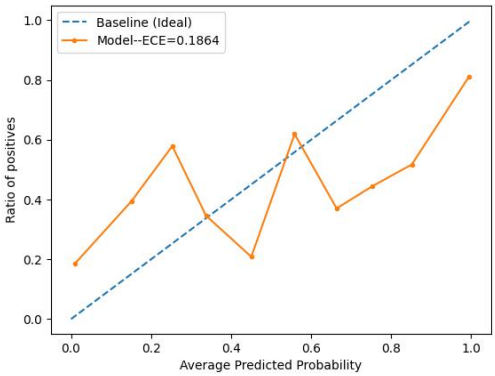
Figure 5.5: Frame level analysis of best-performing classifier - ECE (frames 0-5)



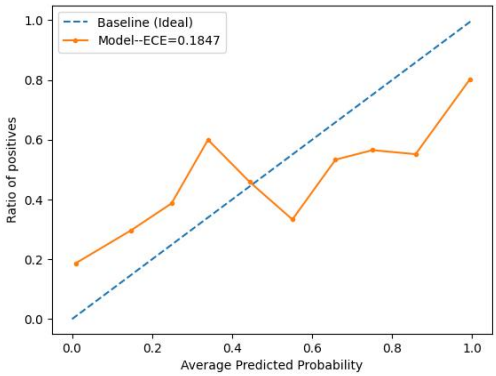
(a) Frame 6



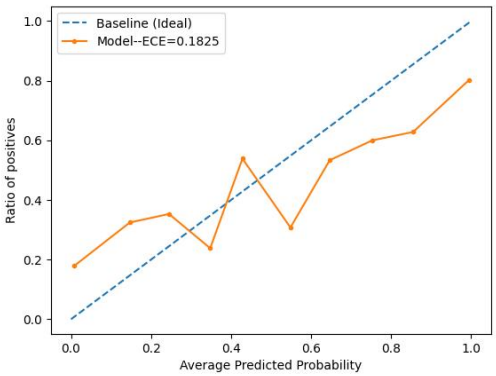
(b) Frame 7



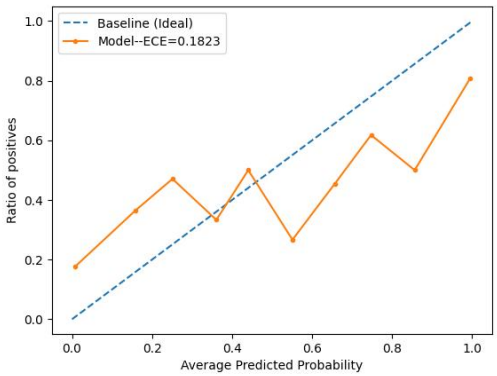
(c) Frame 8



(d) Frame 9

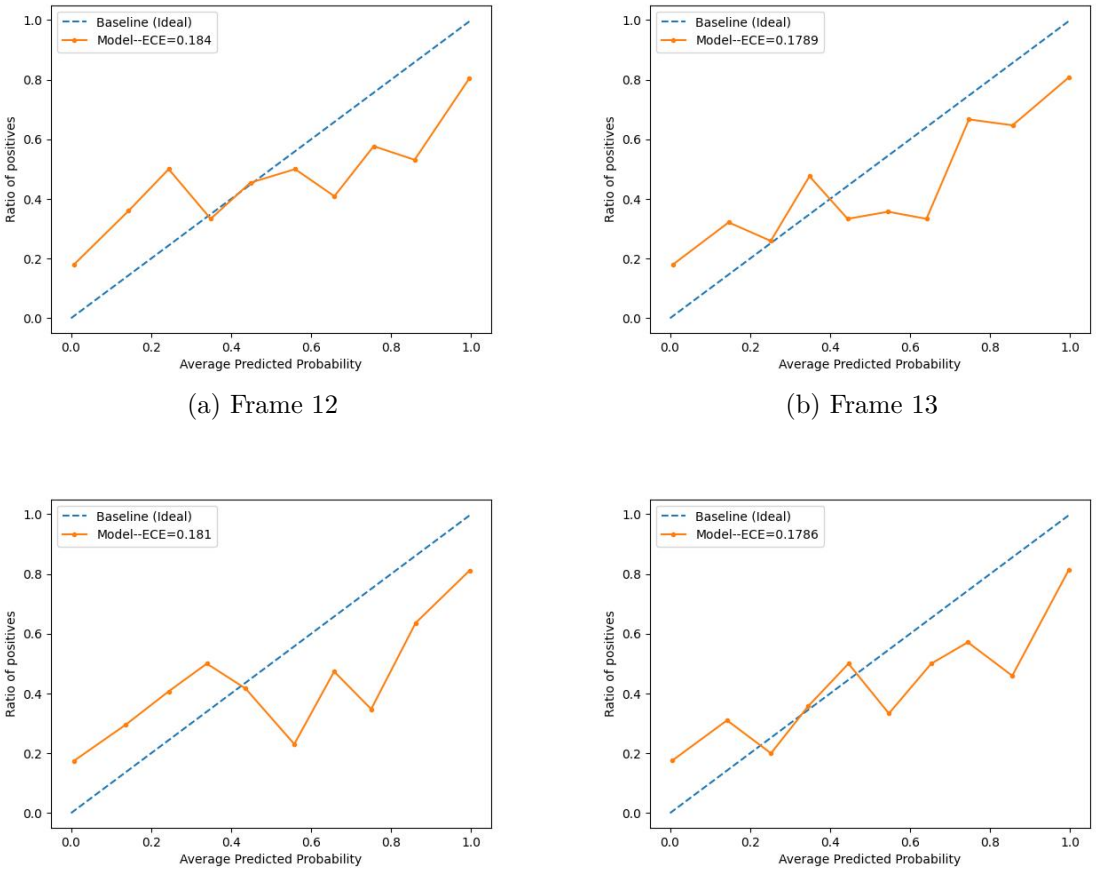


(e) Frame 10



(f) Frame 11

Figure 5.6: Frame level analysis of best-performing classifier - ECE (frames 6-11)



(a) Frame 12

(b) Frame 13

(c) Frame 14

(d) Frame 15

Figure 5.7: Frame level analysis of best-performing classifier - ECE. From the plots, it is clear that it consistently overestimates the positive predictions beyond 70% predicted positive probability

5.1.4 Analysing Common Misclassifications

Analyzing misclassifications is one way to identify the features picked up by the models on a broad level, especially since spatio-temporal features are complex. To help understand the shortcomings of the dataset and model limitations, three top-performing models from the set were considered. Namely,

- CNN-LSTM with ResNet-18 Layer 3 features with additional RoI of 50%
- CNN-LSTM with ResNet-18 Layer 4 features also with additional RoI of 50%
- 3D-CNN with ResNet-18 Layer 3 features with additional RoI of 40%

It was noted that eleven common false positives and seven common false negatives were found in a dataset consisting of 72 negative and 104 positive instances (the validation set was generated by sampling one subset per raw sample). Figure 5.9 and 5.8 shows a few examples of the common mis-classifications. Although there are proper mis-classifications (Figure 5.9 (c) instance), there are also challenging

situations. In the case of Figure 5.9 (a), where the car has a few cycles attached to the rear, the situation is very uncommon, and the turn indicator is very hard to spot if the model does not face such situations (obstructed rear view) during training. Figure 5.9 (b) is a strong case for the need for ODD. Target vehicles too far away from the ego were a common occurrence in the dataset. When the bounding boxes are cropped and resized, the image becomes very noisy to the extent that the vehicle is no longer visible. Hence strict working criteria have to be defined so that the model can be trained on a dataset that closely adheres to the restrictions and produces more reliable results.

In the case of false positives, the misclassifications are more interesting. In Figure 5.8 (a), it can be observed that the target is surrounded by other vehicles, especially oncoming. The only dynamic element in the sample is the flickering headlights of the oncoming vehicle as it passes beyond the separator, miming a blinking light. Figure 5.8 (b) is a slightly more challenging sample where the target travels through an illuminated tunnel. The lights reflected off the target also exhibit a change that can be inferred as blinking, confusing the model.

Figure 5.8: Common false positive classification examples. In the top slides, the divider and the headlights from the oncoming vehicle simulate blinking lights which the model misreads as a turn indicator. In the bottom slides, the lights inside the tunnel being reflected off the vehicle also confuse the model as it blinks.

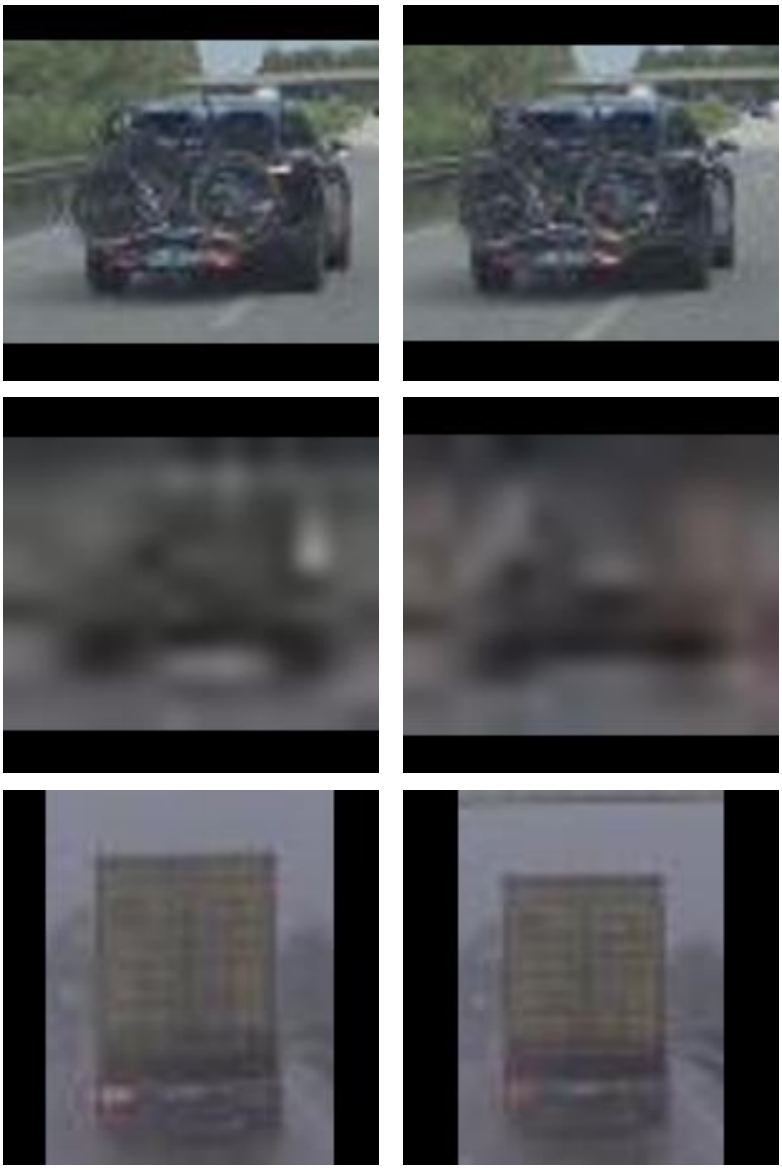


Figure 5.9: Common false negative classification example

5.1.5 Synthetic Validation Samples

To further understand the features learned by the model, synthetic data points were generated from a validation sample, imitating different scenarios. The validation sample with a high predicted negative probability (0.9922) was chosen for this task. The experiment was performed by painting over the sample to mimic turn indicators, brake lights, and headlights from oncoming vehicles.

The number of synthetic samples and the position of the feature were varied in an attempt to study the features in detail. The turn indicator was first analyzed by painting over the rear lights of the target in 1, 2, 3, and 4 frames for both left and right indicators separately. It was observed that the predicted positive probability was highest (0.2293) for four alternative painted frames at the beginning of the sample for right-turn indicators. Having the synthetic samples at the middle or end

Figure 5.10: Examples of synthetic samples imitating right turn, left turn, target brake lights, and non-target brake lights. The model prediction was easily influenced by the turn indicators, both left and right, whereas the other two additions affected the model very little.

of the sequence had very little impact compared to the initial setting. This could also imply an inherent bias in the dataset where the turn signals are concentrated towards the first few frames. In the case of left turn indicators, a similar performance was observed. For robustness to other lights, brake light scenarios for both target and non-target vehicles were also simulated. It was observed that the model performance was affected very little by these augmentations to the sample.

This confirms that the indicators were considered a major feature in the models, and the model was robust enough to identify brake lights apart from turn indicators. The position of the turn indicators played a huge role in deciding the final classification, as turn signals detected towards the beginning of the video instance had a much higher chance of being classified as positive.

5.1.5.1 Spatial Features Analysis

Since the best-performing model was 2D-CNN based, it is possible to manually inspect the features picked up by the model when synthetic samples are passed through it.

In Fig. 5.12, it was clear that most filters, primarily filters 1, 6, and 7, were focused on indicators since they were the most dynamic when introducing turn indicators.

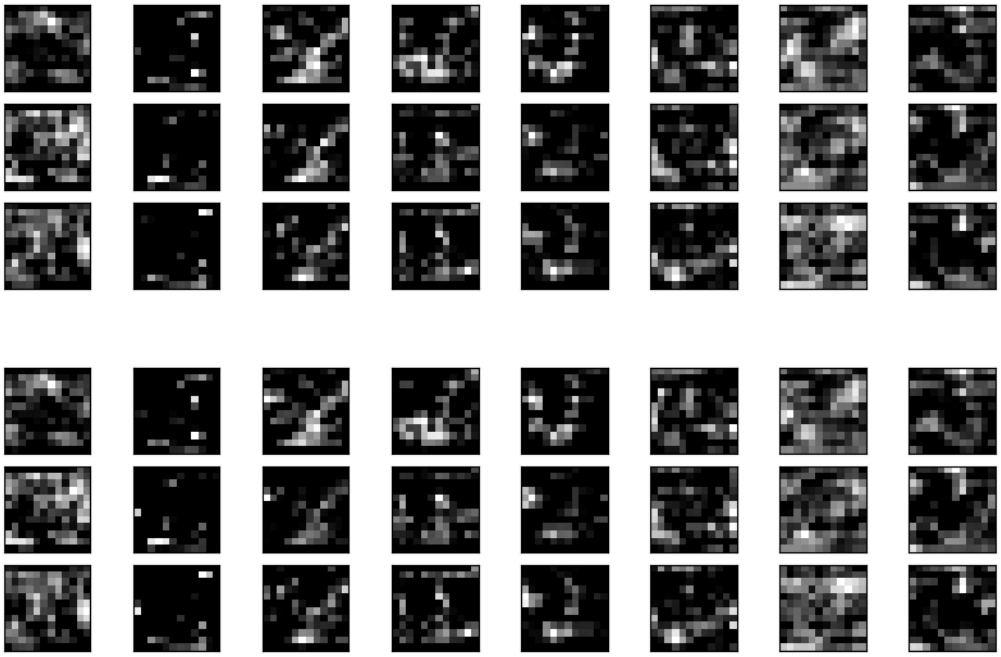


Figure 5.11: Headlights from oncoming vehicle – synthetically introduced in last three frames of the negative sample. The top three rows show the result of the convolution of 8 filters over the original sample, and the bottom three rows show the same on the synthetic sample. The changes in the maps are very minimal (filter 6)
.

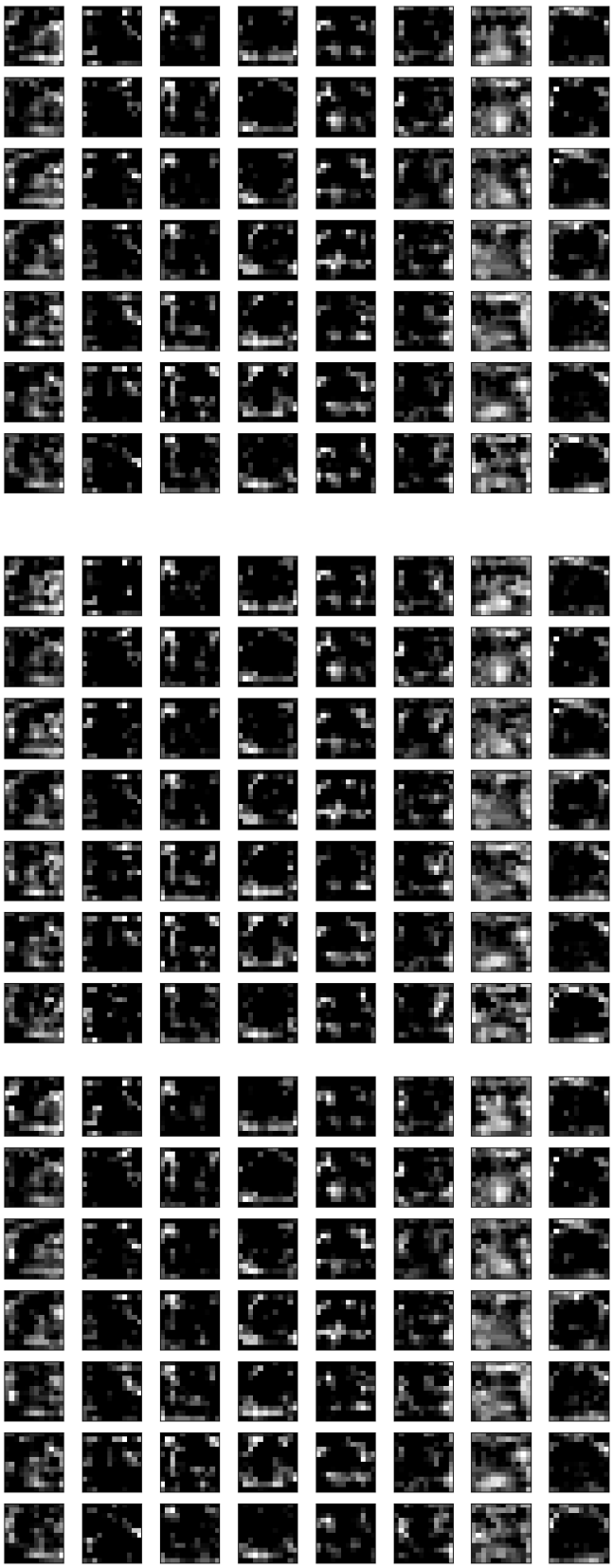


Figure 5.12: The spatial maps for synthetic turn indicator sample. The top six rows show the original negative sample, the next six rows show the synthetic sample with the right turn indicator, and the last six show the left turn indicator sample. The turn indicator is introduced in alternate frames starting from the first frame.

5.2 Regressor with Uncertainty

The objective was to design and validate regressors to predict the start and end of the lane change maneuver from context cues like turn indicators and lane markers. Predicting both these values was a challenge since it depended greatly on the driver. There could be situations where the driver either turns off the indicator too early or starts it too late, in which case the model has to weigh the other cues higher to make a reliable estimate of the maneuver. For this purpose, the models and loss functions were designed to regress the mean of start and end (through different models) and predict an uncertainty estimate. Ideally, the lower bound for the uncertainty estimate, depending on the loss chosen, would be proportional to its deviation from the true label. The estimate is used to train and evaluate the model on a known validation set so that the values obtained when deploying the model in the test set are easy to interpret.

The experiments were broadly split into four categories based on the loss function chosen (Laplacian NLL, Gaussian NLL) and the parameter regressed (TTLC start, TTLC end). The choice of model architectures was the same, with the addition of a deeper 3D-CNN.

5.2.1 Summary of Performance

Laplacian NLL performed best with simpler models (Table 5.5 and 5.7), whereas Gaussian NLL worked well with the heaviest model in terms of parameters (Table 5.6 and 5.8). Assuming the sensors work at 15Hz, the errors can be converted to absolute time units. Predicting the time to lane change start and end was assumed to be a challenging problem as it depends on the environment around the target vehicle, driver, their requirements, etc. The best-performing model reported an MAE of 17.1445 when predicting lane change start, translating to approximately 1s error. The performance is, however, very specific to the dataset considered.

For TTLC end, among the variants, fully trained deep 3D-CNN with Laplacian NLL was observed to perform better among all the other models. In the case of TTLC start, however, the CNN-LSTM model over ResNet-18 Layer 4 features implemented with Gaussian NLL is preferred.

It is also interesting to note the difference in performance when regressing TTLC end using Gaussian and Laplacian NLL. This can be attributed to the high variance training subset generated for the models (refer Table 4.3). Gaussian NLL loss tends to penalize the extremities higher, while Laplacian NLL loss is more robust.

However, the regressed values were close to the mean from the training set, and the models only provided a marginal improvement over the naive baseline. This could imply that the features learned across the models, specifically target-level features, were insufficient to describe the intent of drivers to change lanes.

These results are only based on the regressed mean of the metric of interest. To better understand the quality of the predicted uncertainty parameter, the AUSE values across different context levels were also logged for the models.

5.2.1.1 Regressing TTLC End

Model	MAE		RMSE	
	Mean	SD	Mean	SD
3DCNN	27.4032	0.1197	39.9304	0.4824
3DCNN-2	**27.0781**	0.0700	**38.9386**	0.1405
ResNet_L2-3DCNN	28.5563	0.1906	40.1320	0.0286
ResNet_L3-3DCNN	28.5454	0.0574	39.7802	0.2541
ResNet_L4-3DCNN	29.6244	0.3532	40.2460	0.1415
ResNet_L2-CNN_LSTM	28.7917	0.1053	41.3028	0.3981
ResNet_L3-CNN_LSTM	28.3020	0.3463	41.0539	0.3715
ResNet_L4-CNN_LSTM	28.2955	0.3989	41.0042	0.1375

Table 5.5: Summary of performance: TTLC End with Laplacian NLL loss

Model	MAE		RMSE	
	Mean	SD	Mean	SD
3DCNN	34.1478	1.4503	42.0528	1.3313
3DCNN-2	34.4857	0.8432	42.5490	0.5491
ResNet_L2-3DCNN	32.6324	0.4183	40.5830	0.4290
ResNet_L3-3DCNN	31.5334	0.8907	40.5641	0.5872
ResNet_L4-3DCNN	32.4690	0.1504	41.3496	0.7855
ResNet_L2-CNN_LSTM	30.3075	0.1433	38.7345	0.0807
ResNet_L3-CNN_LSTM	29.7083	0.2912	38.7643	0.0394
ResNet_L4-CNN_LSTM	**29.4114**	0.7671	**38.5379**	0.2628

Table 5.6: Summary of performance: TTLC End with Gaussian NLL loss

5.2.1.2 Regressing TTLC Start

Model	MAE		RMSE	
	Mean	SD	Mean	SD
3DCNN	**17.2650**	0.1013	**21.9077**	0.1095
3DCNN-2	17.5462	0.0593	21.9050	0.3654
ResNet_L2-3DCNN	17.6668	0.0873	21.9456	0.1926
ResNet_L3-3DCNN	17.9112	0.1095	22.3509	0.1498
ResNet_L4-3DCNN	17.7550	0.4913	22.4812	0.5587
ResNet_L2-CNN_LSTM	17.6164	0.0361	21.9914	0.2099
ResNet_L3-CNN_LSTM	17.6497	0.0571	22.0379	0.1392
ResNet_L4-CNN_LSTM	17.9071	0.1380	22.5360	0.1450

Table 5.7: Summary of performance: TTLC Start with Laplacian NLL loss

Model	MAE		RMSE	
	Mean	SD	Mean	SD
3DCNN	17.2720	0.0515	21.4569	0.1094
3DCNN-2	17.4590	0.0507	21.4500	0.0567
ResNet_L2-3DCNN	17.4245	0.1480	21.4527	0.1782
ResNet_L3-3DCNN	17.5099	0.1486	21.5976	0.0624
ResNet_L4-3DCNN	17.1647	0.0649	21.4378	0.0835
ResNet_L2-CNN_LSTM	17.3120	0.0240	21.1493	0.0209
ResNet_L3-CNN_LSTM	17.3381	0.0423	21.1854	0.0117
ResNet_L4-CNN_LSTM	**17.1445**	0.0656	**21.0233**	0.0675

Table 5.8: Summary of performance: TTLC Start with Gaussian NLL loss

5.2.2 AUSE vs Additional RoI

The effect of additional context was also analyzed for the regression setting. Sparsification plots were generated, and the area under the curve was computed using the uncertainty node in both the loss models as the predicted uncertainty for each sample.

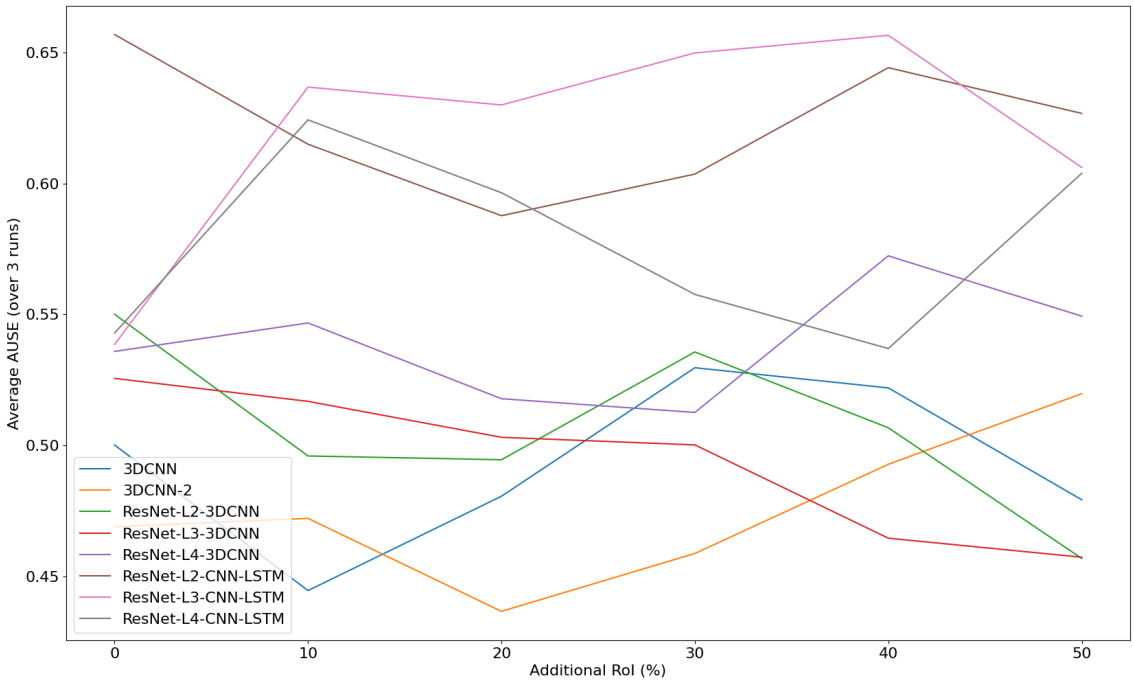


Figure 5.13: AUSE vs Additional RoI - TTLC End regression with Laplacian NLL loss. There are no clear trends to the plot for any curve, except ResNet-L3-3DCNN, which improved with added context. This implies that estimating end-of-lane change with turn indicators is a challenging problem with the given target-level features when estimating uncertainty.

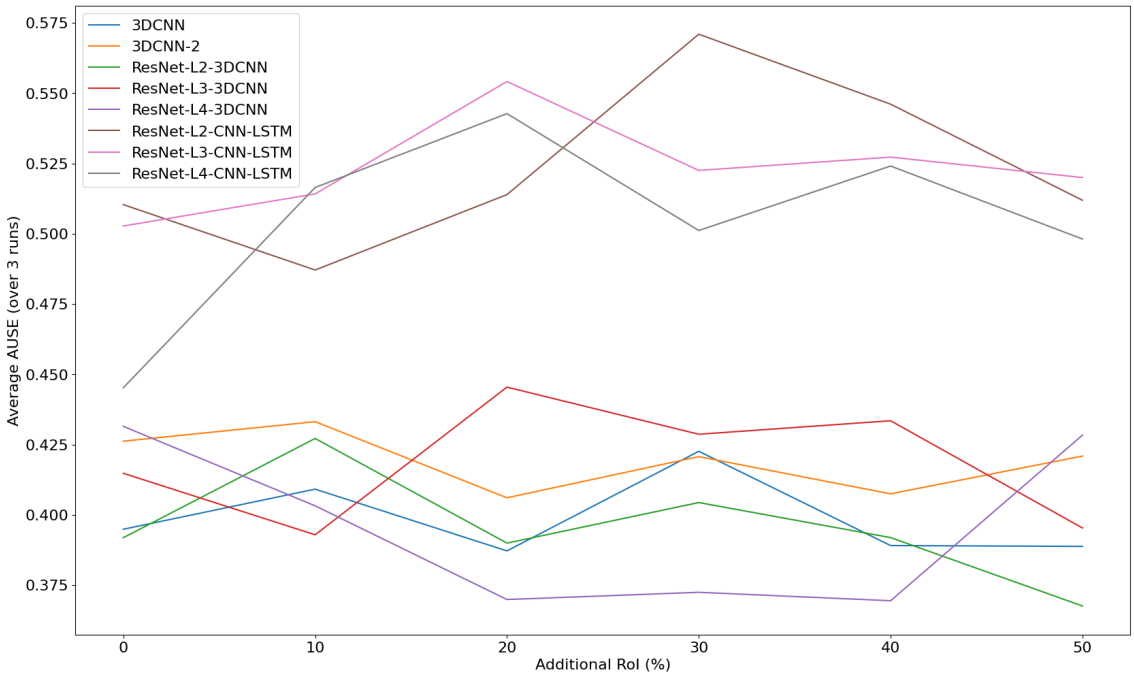


Figure 5.14: AUSE vs. Additional RoI - TTLC End regression with Gaussian NLL loss

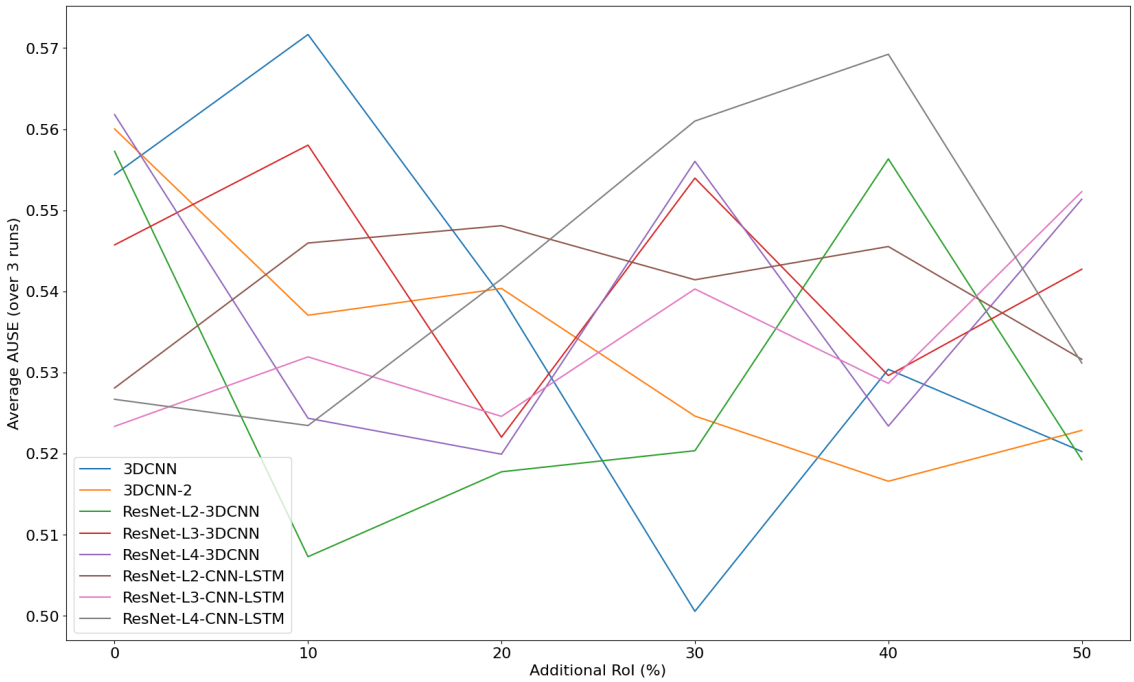


Figure 5.15: AUSE vs. Additional RoI - TTLC Start regression with Laplacian NLL loss

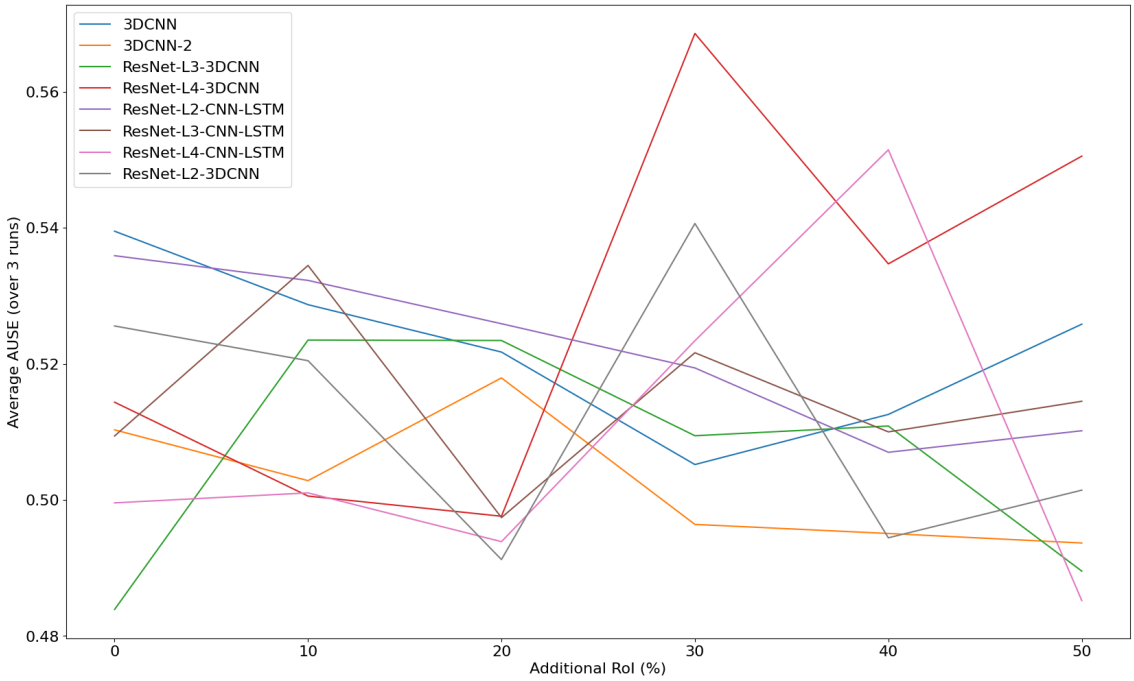


Figure 5.16: AUSE vs. Additional RoI - TTLC Start regression with Gaussian NLL loss

Regressing TTLC End with Laplacian loss did not exhibit any particular trends for any models. ResNet-L3-3DCNN is the only exception where it steadily improved

TTLC Metric	Model	NLL Loss	MAE	RMSE	AUSE
End	3DCNN-2	Laplace	27.0781	38.9386	0.4688
Start	ResNet_L4-CNN_LSTM	Gaussian	17.1445	21.0233	0.4995

Table 5.9: Summary - Best performing models for TTLC start and end

with added context. In all the other cases, an abrupt shift in performance was observed, which implies a very uncertain feature learned by the model.

The uncertainties were more easily distinguishable since the LSTM-based models were separated from the 3D-CNN variants. The uncertainty-accuracy trade-off is also clearly visible where the best-performing model in terms of MAE or RMSE is not the best in predicting the uncertainty parameter and vice versa.

In both cases, no clear trend was observed. However, an interesting observation was that the two extreme models, in terms of complexity, are the most volatile in a Laplacian setting. CNN-LSTM model over ResNet-18 Layer 2 features is an exception in the Gaussian setting as it steadily improves the uncertainty prediction. The result was expected for more complex models since the features learned would be very specific. With more contexts and due to the challenging nature of the task, the feature learned would not be generic enough to be extended to any situation. Table 5.9 summarizes the best-performing models across the regressors for estimating TTLC start and end.

6

Discussion and Future Work

6.1 Discussion and Conclusion

The different spatio-temporal networks, 3D-CNN and CNN-LSTM variants, were explored and benchmarked against each other on the internal dataset. The additional context (over the target) level's effect on these models was also tested and analyzed. An average F1 score of 83.6% and recall of 85% was reported on the internal dataset when used with an additional context of 50%. This was achieved with the CNN-LSTM model over ResNet-18 Layer 4 features.

The best-performing model was further analyzed to study the calibration and it was found that the model was over-confident with samples that had predicted positive probability exceeding 70%.

On comparing the instances misclassified by the top models, it was clear that the task undertaken was challenging and required a very well-defined ODD to produce feasible results. Among the misclassifications, the false positive instances included light sources from surrounding imitating blinking action, which was wrongly detected as turn indicators. The false negative instances showed the limitation of the model and the need for having strict filters for the dataset and the problem as a whole. Synthetic samples were also used to confirm the specific features learned by the model by considering a very strongly negative instance and introducing artificial elements to check the effect on the final prediction. It was confirmed that the turn indicators were a vital feature of the model by considering the spatial maps of the artificial samples.

Estimates of interest, TTLC start and TTLC end, were also explored with an additional uncertainty estimation which helped balance accuracy with prediction confidence. Laplacian and Gaussian NLL loss were derived and used for parallel estimation of the mean and uncertainty parameters. Depending on the regressed value, different models were observed to perform best in different loss settings. TTLC end was best regressed by a deeper 3D-CNN model, whereas TTLC start was best estimated by a more complex CNN-LSTM model that worked with ResNet-18 Layer 4 features. The performance difference was marginal across the models and only slightly improved over the naive baseline.

The effect of additional RoI over the target vehicle affected both classifiers and the regressors differently. In classifiers, it boosted the performance but did not affect

the performance significantly in regressors. This implies that the features learned by the models differ significantly. The level of additional context is directly affected by multiple factors, including the number of surrounding vehicles and longitudinal distance to the target. Adding context in high vehicle density scenarios could trigger false classifications due to additional noise from the other vehicles.

6.1.1 Challenges

- One of the major challenges was to prevent the models from over-fitting on the small dataset. Over-parameterizing was very common during the experiments. Measures were taken to address the challenge, such as forcing down the number of learnable parameters, data augmentation, and progressively decaying the learning rate while training.
- Yet another challenge was the lack of a manual benchmark for the dataset, i.e., have a pool of human drivers annotate samples and measure their accuracy and earliness in predicting lane change. Comparing against this would help in understanding model efficiency more easily. Benchmarking across works is a challenge since the authors use different datasets, which have their own characteristics.
- Balancing accuracy and anticipation to have the optimal model since focusing on one parameter deteriorates the other. It was common that the model that best estimates the uncertainty parameter was not the best in regressing the mean and vice versa.

6.2 Future Work and Proposed Guidelines

- *Sampling negative instances from the dataset effectively :* The ideal negative samples in the dataset would include data points of the same vehicles performing lane change maneuvers. This would make it easier for the model to focus more on the changing elements (turn indicators) and help improve performance.
- *Set up ODD for the module :* A thorough but broad definition of the environmental settings would help the model have reliable and clean data. Proposed conditions that could play a major role in ODD include longitudinal distance to target, road geometry, ego lane position, vehicles surrounding target in close proximity, etc.
- *Exploring more advanced architectures like multiple-stream networks and alternate options for representing the environment around the target vehicle better :* Having multiple streams helps focus on different sparse and continuous features simultaneously that could potentially improve performance. Environment representation includes identifying and tracking non-target vehicles and entities and including them in the model. Efficient environment representation can help model the cause for lane change and potentially have better performance in anticipating.

- *Encoding additional target vehicle information such as motion history and vehicle interactions :* Using target history is one way to model driver behavior. This could be a very useful feature to study how the target reacts in varied traffic settings, and this can be used to hand-pick signals to best work as features.
- *Explore multi-modality and feature fusion :* Multi-modality refers to fusing information from multiple complementary sensors. Currently, the work only focuses on working with camera data, which has its own set of limitations. Adding other sensors as features could help overcome the local limitations of using one sensor.

Bibliography

[1] Jul. 2023. [Online]. Available: https://www.destatis.de/EN/Themes/Society-Environment/Traffic-Accidents/Tables/driver-mistakes.html.

[2] U. D. of Transportation National Highway Traffic Safety Administration, *Critical reasons for crashes investigated in the national motor vehicle crash causation survey, nhtsa*, https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812506.

[3] H. Shi and Y. Li, "Introduction," in *Advanced Driver Assistance Systems and Autonomous Vehicles: From Fundamentals to Applications*, Y. Li and H. Shi, Eds. Singapore: Springer Nature Singapore, 2022, pp. 1–16, ISBN: 978-981-19-5053-7. DOI: 10.1007/978-981-19-5053-7_1. [Online]. Available: https://doi.org/10.1007/978-981-19-5053-7_1.

[4] A. Stroescu, L. Daniel, D. Phippen, M. Cherniakov, and M. Gashinova, "Object detection on radar imagery for autonomous driving using deep neural networks," in *2020 17th European Radar Conference (EuRAD)*, 2021, pp. 120–123. DOI: 10.1109/EuRAD48048.2021.00041.

[5] J. Kim, J. Choi, Y. Kim, J. Koh, C. C. Chung, and J. W. Choi, "Robust camera lidar sensor fusion via deep gated information fusion network," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1620–1625.

[6] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, *Multi-view 3d object detection network for autonomous driving*, 2017. arXiv: 1611.07759 [cs.CV].

[7] P. Wei, Y. Zeng, W. Ouyang, and J. Zhou, "Multi-sensor environmental perception and adaptive cruise control of intelligent vehicles using kalman filter," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 2023. DOI: 10.1109/TITS.2023.3306341.

[8] *SAE Levels of Driving Automation™ Refined for Clarity and International Audience — sae.org*, https://www.sae.org/blog/sae-j3016-update, [Accessed 05-Dec-2022].

[9] *What's new for 2020? | Euro NCAP — euroncap.com*, https://www.euroncap.com/en/vehicle-safety/safety-campaigns/2020-assisted-driving-tests/whats-new/, [Accessed 05-Dec-2022].

[10] D. W. Matolak, "V2v communication channels: State of knowledge, new results, and what's next," in *Communication Technologies for Vehicles*, M. Berbineau, M. Jonsson, J.-M. Bonnin, *et al.*, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–21, ISBN: 978-3-642-37974-1.

[11] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, vol. 1, no. 1, p. 1, Jul. 2014, ISSN: 2197-4225. DOI: 10.1186/s40648-014-0001-z. [Online]. Available: https://doi.org/10.1186/s40648-014-0001-z.

[12] R. Song and B. Li, "Surrounding vehicles' lane change maneuver prediction and detection for intelligent vehicles: A comprehensive review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 6046–6062, 2022. DOI: 10.1109/TITS.2021.3076164.

[13] M. Biparva, D. Fernández-Llorca, R. I. Gonzalo, and J. K. Tsotsos, "Video action recognition for lane-change classification and prediction of surrounding vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 569–578, 2022. DOI: 10.1109/TIV.2022.3164507.

[14] X. Geng, H. Liang, B. Yu, P. Zhao, L. He, and R. Huang, "A scenario-adaptive driving behavior prediction approach to urban autonomous driving," *Applied Sciences*, vol. 7, no. 4, p. 426, Apr. 2017, ISSN: 2076-3417. DOI: 10.3390/app7040426. [Online]. Available: http://dx.doi.org/10.3390/app7040426.

[15] O. Scheel, N. S. Nagaraja, L. Schwarz, N. Navab, and F. Tombari, "Attention-based lane change prediction," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8655–8661. DOI: 10.1109/ICRA.2019.8793648.

[16] J. Schlechtriemen, A. Wedel, J. Hillenbrand, G. Breuel, and K.-D. Kuhnert, "A lane change detection approach using feature ranking with maximized predictive power," *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 108–114, 2014.

[17] Z. Shou, Z. Wang, K. Han, Y. Liu, P. Tiwari, and X. Di, "Long-term prediction of lane change maneuver through a multilayer perceptron," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 246–252. DOI: 10.1109/IV47402.2020.9304587.

[18] A. Benterki, M. Boukhnifer, V. Judalet, and M. Choubeila, "Prediction of surrounding vehicles lane change intention using machine learning," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 2, 2019, pp. 839–843. DOI: 10.1109/IDAACS.2019.8924448.

[19] J. Y. Jun Gao and L. Murphey, "Driving maneuver early detection via sequence learning from vehicle signals and video images," *ELSEVIER*, 2020. DOI: https://dx.doi.org/10.1080/23249935.2021.1936279.

[20] H.-K. Hsu, Y.-H. Tsai, X. Mei, *et al.*, "Learning to tell brake and turn signals in videos using cnn-lstm structure," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317782.

[21] D. Frossard, E. Kee, and R. Urtasun, "DeepSignals: Predicting intent of drivers through visual signals," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, May 2019. DOI: 10.1109/icra.2019.8794214. [Online]. Available: https://doi.org/10.1109%2Ficra.2019.8794214.

[22] R. Izquierdo, Á. Quintanar, J. Lorenzo, *et al.*, "Vehicle lane change prediction on highways using efficient environment representation and deep learning," *IEEE Access*, vol. 9, pp. 119 454–119 465, 2021. DOI: 10.1109/ACCESS.2021.3106692.

[23] R. Izquierdo, A. Quintanar, I. Parra, D. Fernández-Llorca, and M. A. Sotelo, "Experimental validation of lane-change intention prediction methodologies based on cnn and lstm," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3657–3662. DOI: 10.1109/ITSC.2019.8917331.

[24] Q. Li, S. Garg, J. Nie, *et al.*, "A highly efficient vehicle taillight detection approach based on deep learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4716–4726, 2021. DOI: 10.1109/TITS.2020.3027421.

[25] X. Wang, M. Yang, and D. Hurwitz, "Analysis of cut-in behavior based on naturalistic driving data," *Accident; analysis and prevention*, vol. 124, pp. 127–137, Jan. 2019. DOI: 10.1016/j.aap.2019.01.006.

[26] E. Nodine, A. Lam, M. Yanagisawa, and W. Najm, "Naturalistic study of truck following behavior," *Transportation Research Record*, vol. 2615, no. 1, pp. 35–42, 2017. DOI: 10.3141/2615-05. eprint: https://doi.org/10.3141/2615-05. [Online]. Available: https://doi.org/10.3141/2615-05.

[27] S. Kim, J. Wang, D. Guenther, *et al.*, "Analysis of human driver behavior in highway cut-in scenarios," SAE Technical Paper, Tech. Rep., 2017.

[28] Y. Nalcakan and Y. Bastanlar, *Monocular vision-based prediction of cut-in maneuvers with lstm networks*, 2022. arXiv: 2203.10707 [cs.CV].

[29] J. Mänttäri, S. Broomé, J. Folkesson, and H. Kjellström, "Interpreting video features: A comparison of 3d convolutional networks and convolutional lstm networks," in *Computer Vision – ACCV 2020*, H. Ishikawa, C.-L. Liu, T. Pajdla, and J. Shi, Eds., Cham: Springer International Publishing, 2021, pp. 411–426, ISBN: 978-3-030-69541-5.

[30] M. Z. Alom, A. Moody, N. Maruyama, B. Van Essen, and T. Taha, "Effective quantization approaches for recurrent neural networks," Feb. 2018.

[31] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[32] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994. DOI: 10.1109/72.279181.

[33] J. Chen, H. Wang, S. Wang, E. He, T. Zhang, and L. Wang, "Convolutional neural network with transfer learning approach for detection of unfavorable driving state using phase coherence image," *Expert Systems with Applications*, vol. 187, p. 116 016, 2022, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.116016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421013634.

[34] I. Ahmed, G. Jeon, A. Chehri, and M. M. Hassan, "Adapting gaussian yolov3 with transfer learning for overhead view human detection in smart cities and societies," *Sustainable Cities and Society*, vol. 70, p. 102 908, 2021, ISSN: 2210-6707. DOI: https://doi.org/10.1016/j.scs.2021.102908. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210670721001967.

[35] J. Park and W. Yu, "A sensor fused rear cross traffic detection system using transfer learning," *Sensors*, vol. 21, no. 18, 2021, ISSN: 1424-8220. DOI: 10.3390/s21186055. [Online]. Available: https://www.mdpi.com/1424-8220/21/18/6055.

[36] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[37] O. Russakovsky, J. Deng, H. Su, *et al.*, *Imagenet large scale visual recognition challenge*, 2015. arXiv: 1409.0575 [cs.CV].

[38] M. Ben naceur, M. Akil, R. Saouli, and R. Kachouri, "Fully automatic brain tumor segmentation with deep learning-based selective attention using overlapping patches and multi-class weighted cross-entropy," *Medical Image Analysis*, vol. 63, p. 101 692, 2020, ISSN: 1361-8415. DOI: https://doi.org/10.1016/j.media.2020.101692. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841520300578.

[39] B. Shan and Y. Fang, "A cross entropy based deep neural network model for road extraction from satellite images," *Entropy*, vol. 22, no. 5, p. 535, May 2020, ISSN: 1099-4300. DOI: 10.3390/e22050535. [Online]. Available: http://dx.doi.org/10.3390/e22050535.

[40] P. Koopman and F. Fratrik, "How many operational design domains, objects, and events?" In *SafeAI@AAAI*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:67865082.

[41] K. Czarnecki and R. Salay, "Towards a framework to manage perceptual uncertainty for safe automated driving," in *Developments in Language Theory*, Springer International Publishing, 2018, pp. 439–445. DOI: 10.1007/978-3-319-99229-7_37. [Online]. Available: https://doi.org/10.1007%2F978-3-319-99229-7_37.

[42] Y. Gal, "Uncertainty in deep learning," 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:86522127.

[43] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods," *Machine Learning*, vol. 110, no. 3, pp. 457–506, Mar. 2021. DOI: 10.1007/s10994-021-05946-3. [Online]. Available: https://doi.org/10.1007%2Fs10994-021-05946-3.

[44] L. Sluijterman, E. Cator, and T. Heskes, *How to evaluate uncertainty estimates in machine learning for regression?* 2023. arXiv: 2106.03395 [stat.ML].

[45] F. Arnez, H. Espinoza, A. Radermacher, and F. Terrier, *A comparison of uncertainty estimation approaches in deep learning components for autonomous vehicle applications*, 2020. arXiv: 2006.15172 [cs.LG].

[46] A. Kendall, V. Badrinarayanan, and R. Cipolla, *Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding*, 2016. arXiv: 1511.02680 [cs.CV].

[47] K. Lee, K. Saigol, and E. Theodorou, "Early failure detection of deep end-to-end control policy by reinforcement learning," May 2019, pp. 8543–8549. DOI: 10.1109/ICRA.2019.8794189.

[48] V.-L. Nguyen, S. Destercke, and E. Hüllermeier, *Epistemic uncertainty sampling*, 2019. arXiv: 1909.00218 [cs.LG].

[49] M. Valdenegro-Toro and D. Saromo, *A deeper look into aleatoric and epistemic uncertainty disentanglement*, 2022. arXiv: 2204.09308 [cs.LG].

[50] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059. [Online]. Available: https://proceedings.mlr.press/v48/gal16.html.

[51] B. Lakshminarayanan, A. Pritzel, and C. Blundell, *Simple and scalable predictive uncertainty estimation using deep ensembles*, 2017. arXiv: 1612.01474 [stat.ML].

[52] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. Grosse, *Flipout: Efficient pseudo-independent weight perturbations on mini-batches*, 2018. arXiv: 1803.04386 [cs.LG].

[53] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[54] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., ser. Proceedings of Machine Learning Research, vol. 70, PMLR, Jun. 2017, pp. 1321–1330. [Online]. Available: https://proceedings.mlr.press/v70/guo17a.html.

[55] M. Pakdaman Naeini, G. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, Feb. 2015. DOI: 10.1609/aaai.v29i1.9602. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/9602.

[56] E. Ilg, Ö. Çiçek, S. Galesso, *et al.*, *Uncertainty estimates and multi-hypotheses networks for optical flow*, 2018. arXiv: 1802.07095 [cs.CV].

[57] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

A

Appendix 1

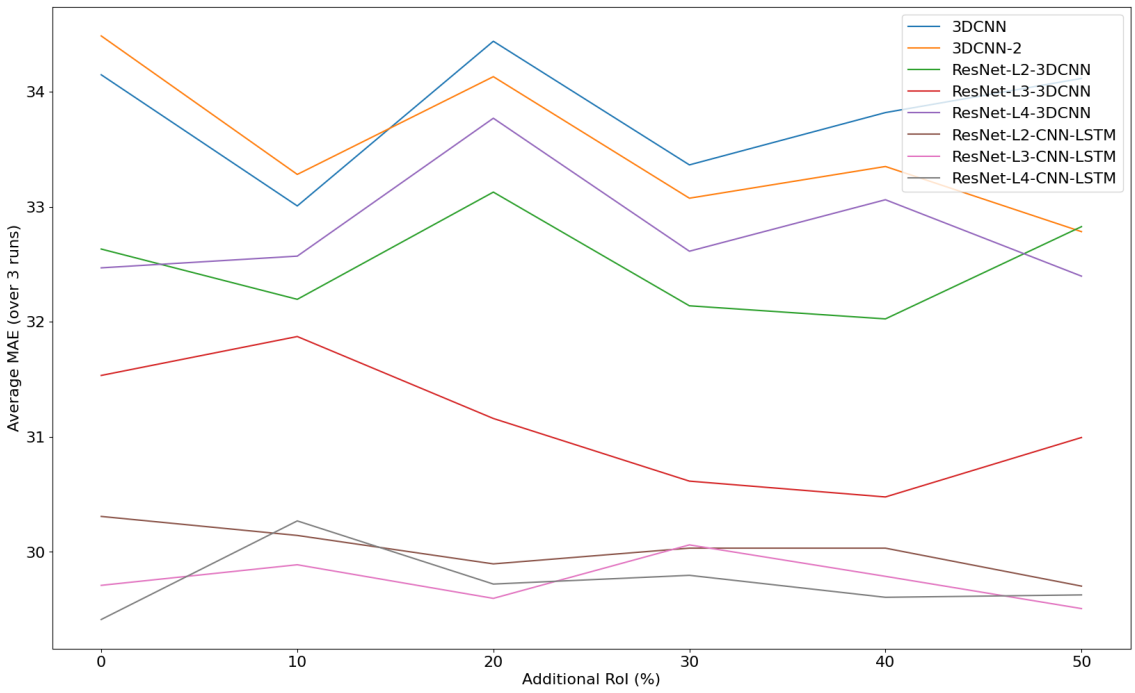


Figure A.1: MAE vs Additional RoI - TTLC End Regression with Gaussian NLL

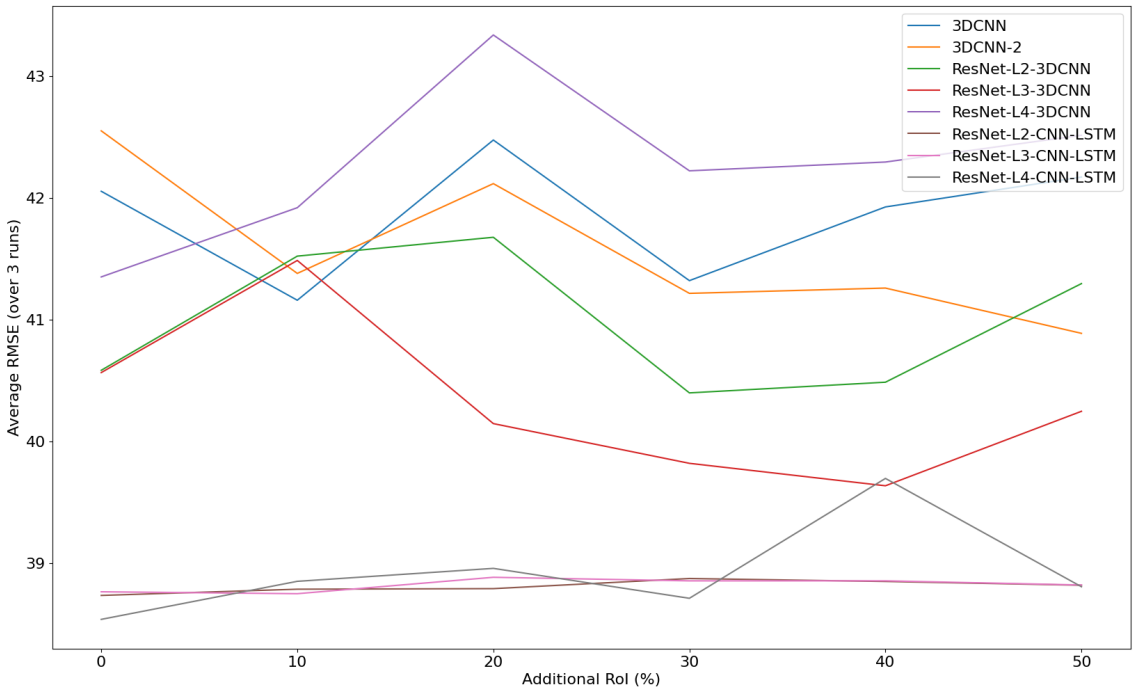


Figure A.2: RMSE vs Additional RoI - TTLC End Regression with Gaussian NLL

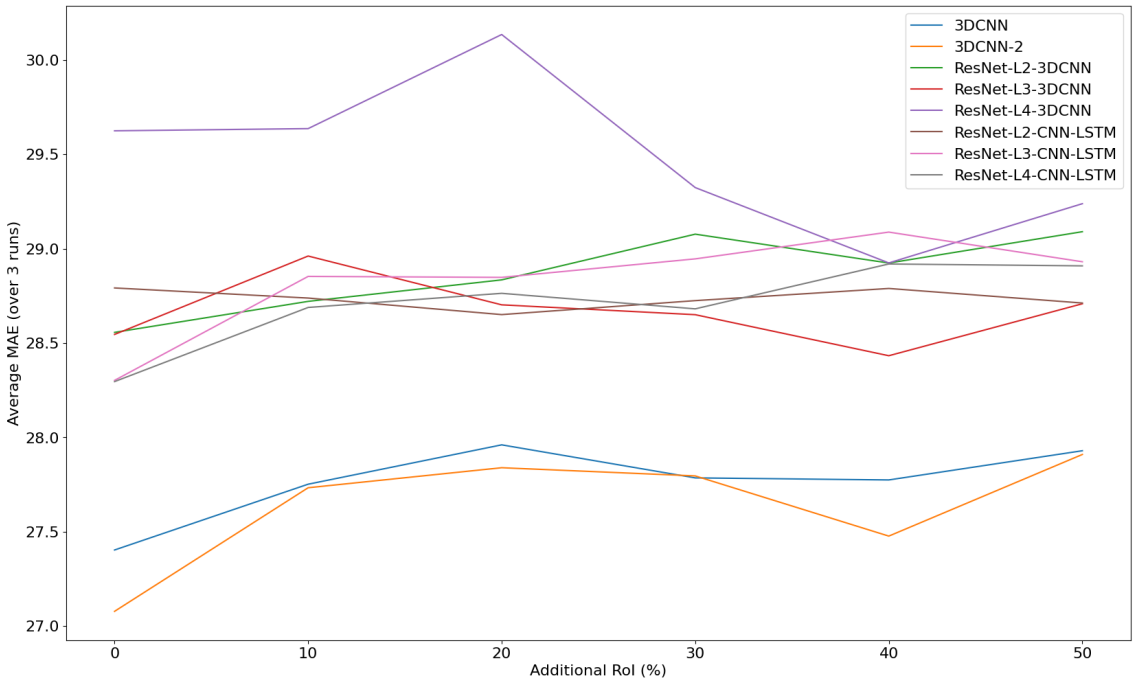


Figure A.3: MAE vs Additional RoI - TTLC End Regression with Laplace NLL

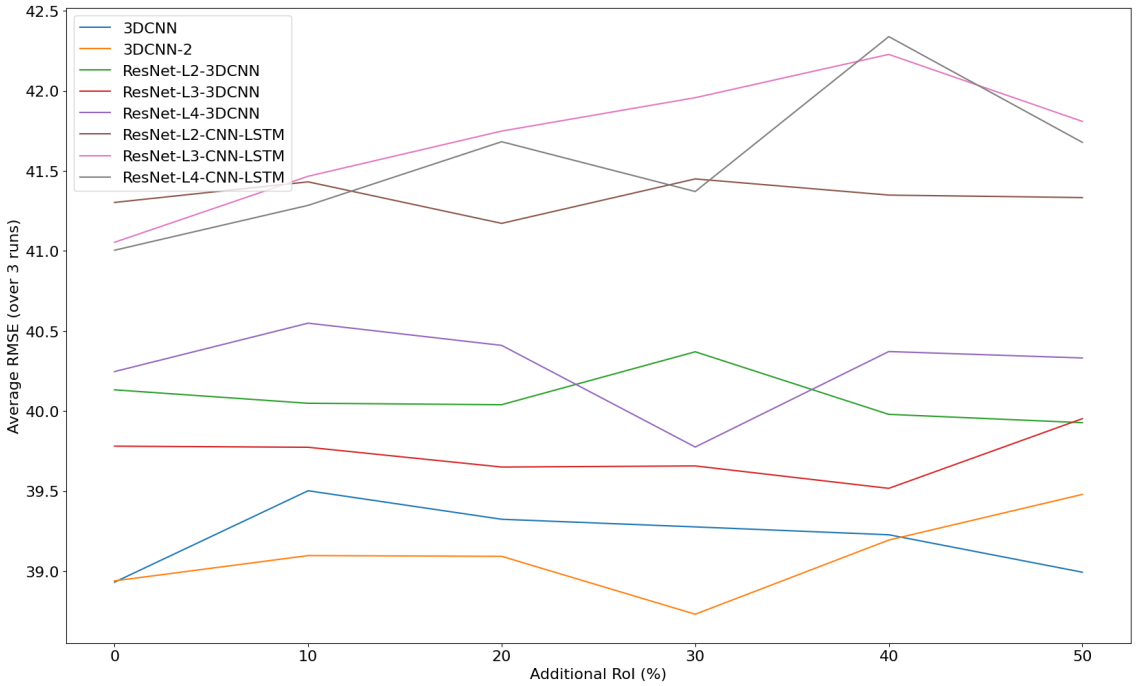


Figure A.4: RMSE vs Additional RoI - TTLC End Regression with Laplace NLL

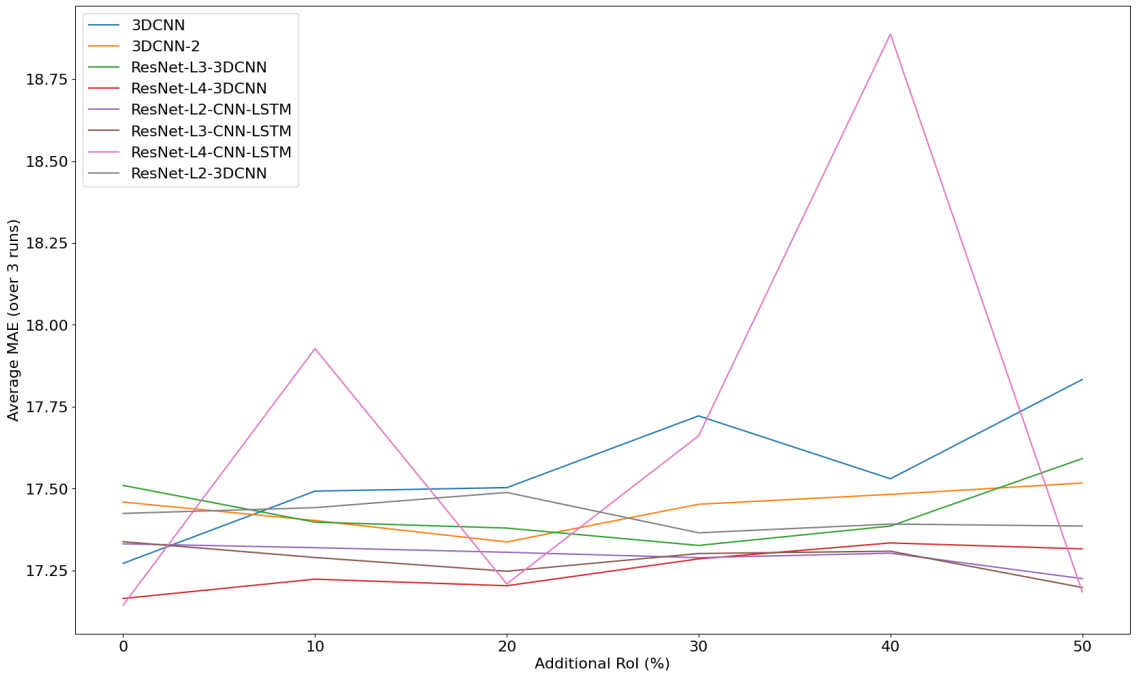


Figure A.5: MAE vs Additional RoI - TTLC Start Regression with Gaussian NLL

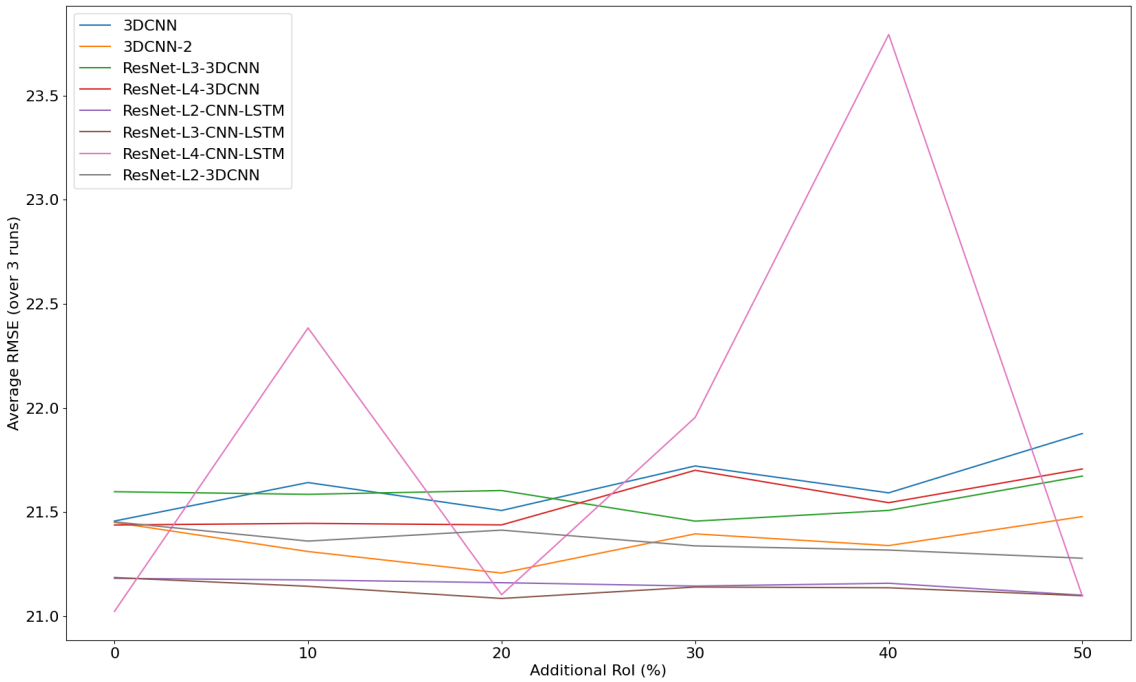


Figure A.6: RMSE vs Additional RoI - TTLC Start Regression with Gaussian NLL

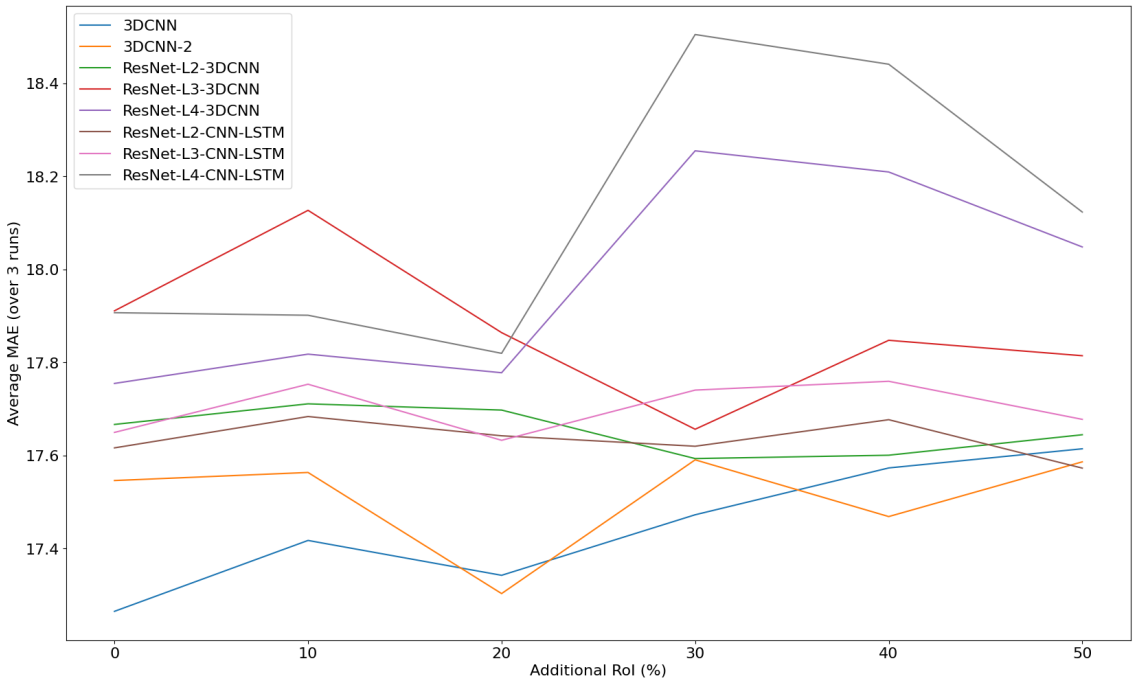


Figure A.7: MAE vs Additional RoI - TTLC Start Regression with Laplace NLL

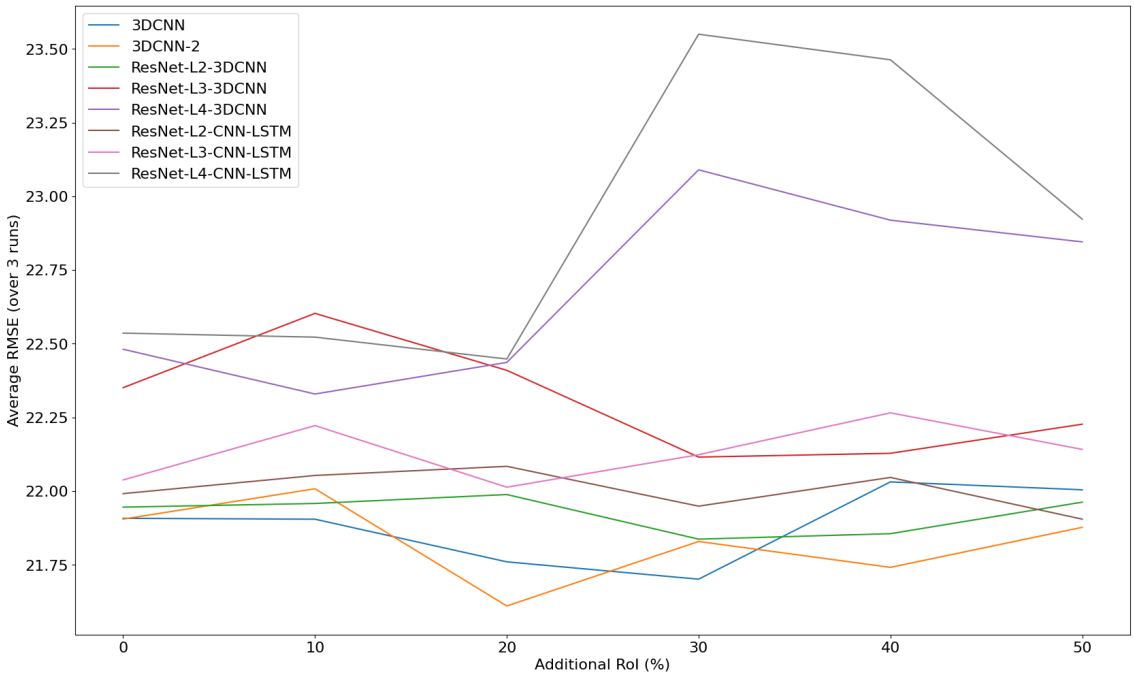


Figure A.8: RMSE vs Additional RoI - TTLC Start Regression with Laplace NLL