CHALMERS
UNIVERSITY OF TECHNOLOGY

Data Hide and Seek is Over!

Automatic annotation of vehicle data from onboard car sources with predefined annotation concepts

Master's thesis in Systems, control and mechatronics

OSKAR ANDERSSON
OTTO ÄRLIG

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Data Hide and Seek is Over!

Automatic annotation of vehicle data from onboard car sources with predefined annotation concepts

OSKAR ANDERSSON
OTTO ÄRLIG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2025

Data Hide and Seek is Over!
Automatic annotation of vehicle data from onboard car sources with predefined annotation concepts
OSKAR ANDERSSON
OTTO ÄRLIG

Supervisor: Nastaran Dashti, Volvo Cars, Safe Vehicle Automation
Examiner: Jonas Sjöberg, Department of Electrical Engineering, Systems and Control

Master's Thesis 2025
Department of Electrical Engineering
Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Visualization of a driving scene from the constructed dataset, showing the camera image alongside its corresponding LiDAR point cloud.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Data Hide and Seek is Over!
Automatic annotation of vehicle data from onboard car sources with predefined annotation concepts.
OSKAR ANDERSSON
OTTO ÄRLIG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

A complete pipeline is presented for automatic annotation and retrieval of multimodal vehicle data, using synchronized front-facing camera images and LiDAR point clouds. Driving scenes are classified across four categories: road condition, road type, lighting, and visibility. A dataset of 1,878 one-minute segments is constructed from over 200 hours of real-world driving. Segments are manually labeled to provide ground-truth annotations for training and evaluation, and selected to ensure an even distribution across all scenario categories.

Separate models are trained for each sensor: a VGG19-based CNN for image classification and a lightweight PointNet for LiDAR point clouds. The best-performing vision model achieves strong results across all categories, while the LiDAR model performs best on road condition and visibility. A fusion model, implemented as a small multilayer perceptron, combines outputs from both sensors and outperforms the individual models, particularly on more difficult scenarios and categories.

Sequence-level aggregation of predictions is applied to reduce frame-level variation and improve accuracy. A proof-of-concept data retrieval interface is also presented, enabling users to filter and inspect data based on predicted labels and confidence scores, and to explore both camera images and LiDAR point clouds for each retrieved segment.

Keywords: Automatic Annotation, Multimodal, Camera, LiDAR, CNN, PointNet, Data Retrieval.

Acknowledgements

We would like to thank everyone at the Data Platform and Management Department at Volvo Cars for their support and interest in our work. We are especially grateful to our supervisor at Volvo Cars, Nastaran Dashti, for her technical guidance and continuous support. Her expertise and involvement helped us make steady progress throughout the project.

We also thank Stefan Flink and Kirill Pruntov at Volvo Cars for their valuable input, particularly in helping us define and frame the problem early in the project.

Lastly, we thank our examiner at Chalmers, Jonas Sjöberg, whose feedback and guidance were particularly helpful during the writing of this thesis.

Oskar Andersson, Gothenburg, June 2025
Otto Ärlig, Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

CLIP	Contrastive Language–Image Pretraining
CNN	Convolutional Neural Network
FN	False Negative
FP	False Positive
LiDAR	Light Detection and Ranging
MLP	Multi-Layer Perceptron
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
TN	True Negative
TP	True Positive
VGG19	Visual Geometry Group 19-layer network

Nomenclature

Below is the nomenclature of indices, parameters, and variables that have been used throughout this thesis.

Indices

i	Index of an individual frame within a sequence
s	Index of a sequence
c	Index of a class

Parameters

N_s	Number of datapoints in sequence s
N_{pts}	Number of points sampled per LiDAR point cloud
B	Batch size during training
C	Number of classes in the classification category

Variables

$L_{i,c}$	Logit, before softmax, for class c on frame i
$\bar{L}_{s,c}$	Sequence-level logit for class c, averaged over all frames in s
$\hat{y}_s$	Predicted class for sequence s, $\hat{y}_s = \arg\max_c \bar{L}_{s,c}$

Contents

List of Figures

List of Tables

1
Introduction

This report presents the design and implementation of a complete pipeline for automatic annotation and retrieval of multimodal vehicle data. The pipeline integrates camera images and LiDAR point clouds to assign reliable labels across four scenario categories: road condition, road type, lighting, and visibility. The system was built to support engineers at Volvo Cars in locating relevant driving scenarios from large sensor datasets.

Manual annotation of multimodal sensor data is time-consuming, subjective, and often infeasible at scale. Many segments in the existing database are either unlabeled or inconsistently annotated, making it difficult for engineers to retrieve data for specific test scenarios. This report addresses the problem by using machine learning models to assign semantic labels based on visual and spatial sensor inputs and demonstrates how these predictions can be used for efficient scenario-based retrieval.

1.1 Background and Motivation

The work was conducted in collaboration with Volvo Cars' Data Platform and Management department, which oversees the company's centralized database of sensor recordings. These recordings are collected from test vehicles equipped with front-facing cameras, LiDAR sensors, and various internal state sensors. The resulting data is therefore multimodal, meaning that each segment contain multiple types of information, synchronized over time.

This database is meant to support the development of sensor-based features and functions. However, effective use of the data requires accurate metadata that describes driving conditions, so that relevant data can be easily retrieved. Currently, many sequences contain inconsistent and unreliable manual annotations, and some lack this sort of metadata entirely. As a consequence, engineers must often resort to manually reviewing large amounts of footage or even collecting new data altogether, which is expensive and inefficient. This motivated the development of a unified pipeline for automatic annotation and retrieval of data.

1.2 Illustrative Example

Consider two engineering teams that both need access to driving data recorded under snowy conditions. One team is developing a LiDAR-based object detection

algorithm and requires data recorded during active snowfall to study how falling snowflakes affect detection performance. The second team is focused on traction control and needs segments with snow or ice on the road surface to analyze slip events.

If the database only contains vague annotations—such as a generic *snow* label that does not specify whether the snow is on the road, on the sides, or falling—it becomes difficult to retrieve relevant data. And if there are no labels at all, the situation is even worse, as neither team can reliably find the data they need. They may need to spend hours manually reviewing footage or even schedule new test drives. To address this challenge, this report describes the development of a tool that automatically assigns clearly defined and accurate class labels in chosen categories, such as road condition and visibility. With the help of these labels, the most relevant data segments can be selected, based on the specific needs and specifications of each engineering team.

1.3 Contributions

This thesis contributes to solving the annotation and retrieval challenge by designing, implementing, and evaluating the following components:

- **Dataset Creation:** We constructed a high-quality dataset of 1,878 one-minute sequences, synchronized across camera and LiDAR modalities. Each sequence was manually annotated across four categories: road condition, road type, lighting, and visibility using a class-balanced sampling strategy. We used this dataset to train and evaluate machine learning models, and it may also serve as a valuable resource for future research.
- **Sensor-Specific Classifiers:** We implemented and evaluated deep learning models for each modality. For camera data, a fine-tuned VGG19 CNN achieved high classification performance. For LiDAR, we trained a custom lightweight PointNet architecture on sampled point clouds, exploring both full-scene and ground-level variants.
- **Fusion and Temporal Aggregation:** We developed a multi-layer perceptron (MLP) to fuse predictions from both sensors into a final decision. In addition, we applied sequence-level aggregation of logits, averaging per-frame predictions to produce a single robust prediction per one-minute segment.
- **Retrieval Interface:** We designed and implemented a web-based proof-of-concept interface that enables engineers to filter sequences based on predicted labels and confidence thresholds. The interface supports multimodal inspection and CSV export for integration into downstream workflows.

2
Dataset for Model Development

This chapter describes how raw vehicle recordings from Volvo Cars database, are transformed into a structured, labeled dataset suitable for developing and evaluating annotation models. We describe how raw camera and LiDAR streams captured during real driving are synchronized, and how segments are selected from the database. We also describe how each of the segments is reviewed and assigned ground-truth labels for road condition, road type, lighting, and visibility. It is then explained how the annotated segments are split into training and validation sets.

2.1 Database and Data Source Sensors

The need for automatic annotations stems from the fact that Volvo Cars governs a large database of vehicle sensor data that lacks sufficient labels, as detailed in Section 1.1. The database is a multi-server PostgreSQL system comprising:

- **Primary server:** Maintains the authoritative archive of all raw data.
- **Replica server:** Offers read-only access for large-scale querying and analysis.
- **Development server:** Used for testing new data ingestion and processing pipelines.

We utilized the Replica server to explore the data through querying and analysis, without risking altering the structure or data used in production. The vehicle sensor data in the database is sourced from a large number of drives with Volvo Cars test vehicles equipped with multiple sensors and data loggers. These drives are in turn divided into shorter, one-minute segments of data recordings, each indexed by timestamp, vehicle identifier, and additional metadata. Out of the sensors on the test vehicles, the ones whose recordings we explore for our automatic annotation pipeline are the following:

- **Camera:** A high-resolution front-facing camera recording images with a capture frequency of 30 Hz.
- **LiDAR:** Light Detection and Ranging sensors that emits rapid laser pulses to generate three-dimensional point clouds, capturing time-of-flight, pulse angle and return reflectance.

2.2 Construction of a Custom Dataset

This section describes how raw LiDAR and camera data are processed, synchronized and manually annotated to support training and evaluation of models for automatic annotation.

2.2.1 Defining Categories and Classes

In order to evaluate the feasibility of automatic annotation based on the raw data, we defined four label categories related to driving conditions to serve as these annotations. These categories were established in consultation with the data governance team at Volvo Cars, with consideration for what might be useful to engineers searching for particular driving scenarios. Labels cannot be too generic but must be sufficiently specific to be useful, as described in Section 1.2. We also took into account that it must be feasible to create reliable ground-truth annotations using these labels—meaning they need to be clearly defined and relatively constant over the one-minute segments in the database. Based on these considerations, the final categories and class labels are presented in Table 2.1.

Table 2.1: Annotation categories and their corresponding classes.

Category	Class 1	Class 2	Class 3
Road condition	Dry	Wet	Snow/Ice
Road type	City Road	Country Road	Highway/Large Road
Lighting	Light	Dark	–
Visibility	Clear	Not Clear	–

The *Road condition* category captures the surface state of the road to facilitate data retrieval from scenarios with varying traction. *Road type* distinguishes between urban, rural, and highway environments, which may be relevant since each corresponds to distinct driving patterns and potential hazard risks. *Lighting* indicates whether the recording was made during daytime or nighttime. *Visibility* groups the weather conditions rainfall, fog, and snowfall under the class *Not Clear*, to facilitate retrieval of data recorded under different visibility conditions, which could be useful for evaluation of certain sensors.

2.2.2 Extracting our Dataset from the Database

When developing our custom dataset, the focus was on extracting sequences from the database across a wide range of driving conditions and locations, in order to ensure data diversity. This resulted in 1878 one-minute sequences from a large number of unique drives, evenly distributed over more than 200 driving hours, where one one-minute sequence was sampled every five minutes in the original data. Two sets of such sequences were created, one for each of the two sensors used: camera and LiDAR. These sequences were extracted at matching timestamps to ensure synchronization between the two modalities. Within each sequence, one image–LiDAR

pair was sampled per second, resulting in 60 such pairs per sequence. In total, this yielded a dataset containing 112 680 synchronized image–LiDAR pairs.

2.2.3 Processing of our extracted Dataset

To prepare the data for usage in the development of the automatic annotation models, the data had to be processed. The camera data in the database is stored in the proprietary `.zvlf` archive format. This data cannot be viewed or interpreted using standard applications, and the images were therefore decoded and saved as JPEG files. The LiDAR data is stored in HDF5 files, from which we extracted the individual point clouds and saved them as arrays in NumPy files. These arrays consist of one 4-dimensional vector per point in the point cloud, where the four dimensions correspond to the x, y, z coordinates and the reflectance intensity recorded by the LiDAR sensor. The dataset was organized using a master CSV file, where each row corresponds to one sequence and includes information about file paths to the LiDAR and camera data, together with other metadata from the database such as timestamps.

2.2.4 Manual Ground-Truth Annotation of our Dataset

The dataset was manually annotated with ground-truth labels, using the labels defined in Section 2.2.1. Annotations were performed per sequence by manually observing a sample of images from each sequence and determining which labels best corresponded to it across the four categories. Sequences were excluded if the camera view was obscured, LiDAR data were incomplete, or labeling confidence was low, in order to maintain high accuracy and confidence in the labels. The result is a diverse dataset with 1,878 one-minute sequences, carefully and manually annotated with ground-truth labels.

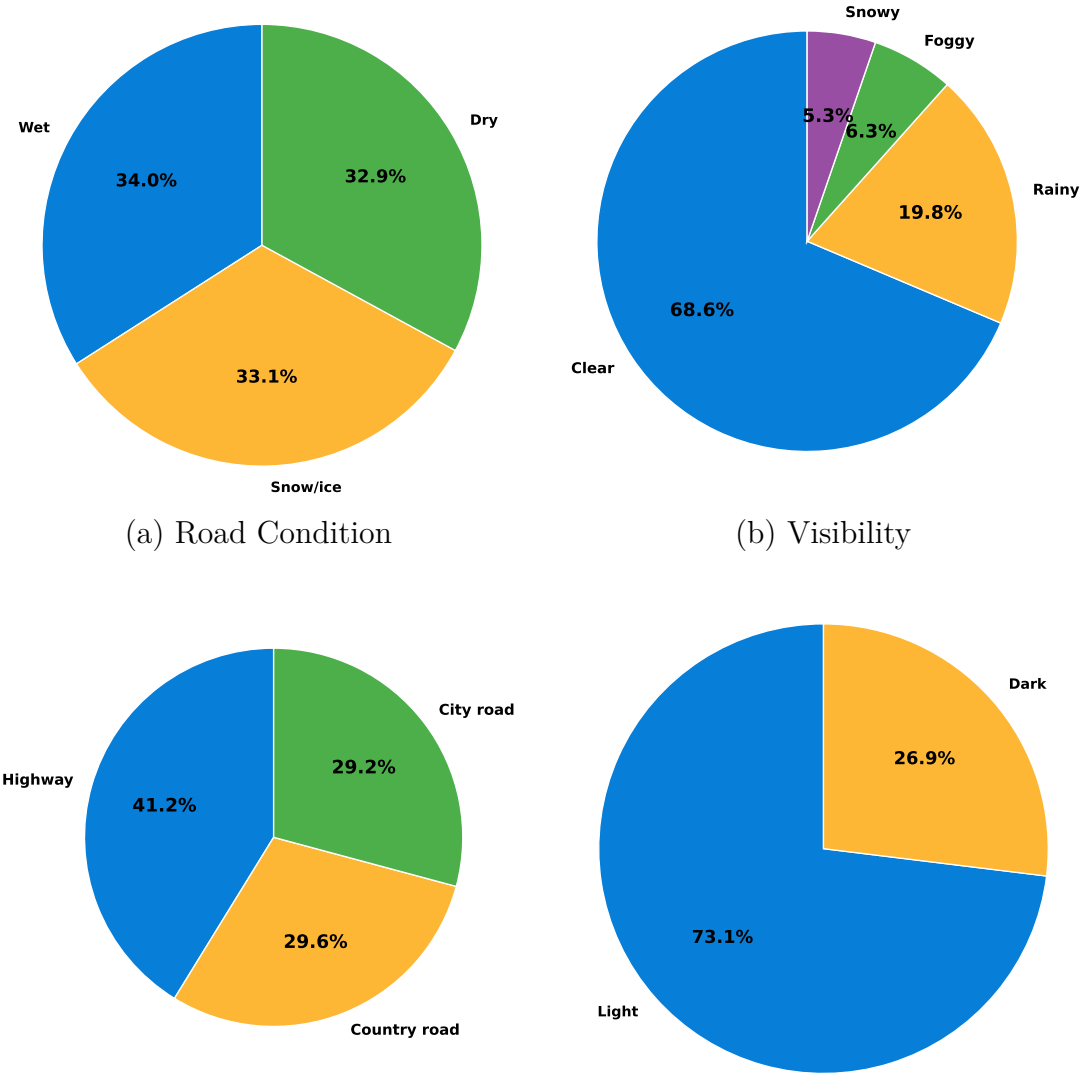
2.2.5 Training and Validation Split

The manually annotated dataset was split into a training set and a validation set to support both training and evaluation of automatic annotation models. This ensures that the validation set serves as an independent benchmark, measuring how well the model performs on data it has never seen during training. The validation split was roughly 10% of the full annotated dataset, corresponding to 199 one-minute sequences, while the remaining 1679 sequences form the training set.

The split was performed at the level of entire drives, meaning that all sequences from a particular drive were assigned to either the training set or the validation set. This approach ensures that the validation set has no overlap with the training set, allowing for a proper assessment of annotation models abilities to generalize to unseen data.

2.2.6 Class Distribution Over our Dataset

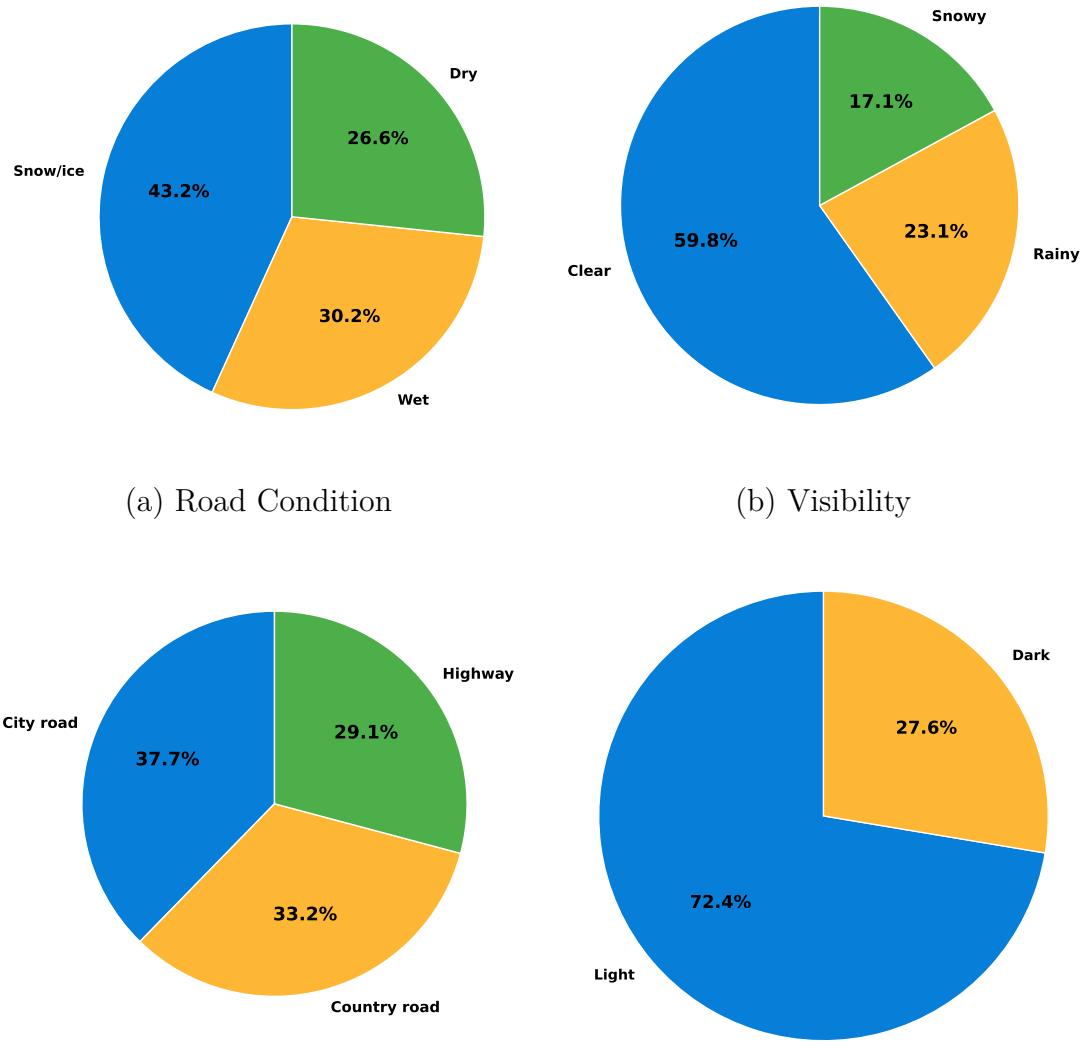
The distribution of classes over all 1878 sequences in our dataset is presented in in Figure 2.1. Similarly, distribution of classes over the 199 sequences in the validation set is seen in the Figure 2.2.



(a) Road Condition (b) Visibility

(c) Road Type (d) Lighting

Figure 2.1: Pie chart showcasing the class distributions for the full dataset.



(a) Road Condition

(b) Visibility

(c) Road Type

(d) Lighting

Figure 2.2: Class distributions for the validation subset.

3

Vision-Based Classification Models

This chapter details the training and evaluation of three vision-based classifiers on the manually annotated dataset from Chapter 2. A two-stage training strategy is described, showing how classification heads were attached to a frozen VGG19, trained, and then fine-tuned to optimize automatic annotation of road condition, visibility, road type, and lighting. The performance of each model is evaluated on the validation set.

3.1 Vision Model Architectures

To extract information from images such as the ones in our dataset, the most common approach is to use a Convolutional Neural Networks (CNN). CNN's leverage stacks of convolutional filters to hierarchically extract features from images [1, 2]. Early convolutional layers detect patterns like edges, corners, textures, while deeper layers combine these into object-level representations. This hierarchical feature learning, coupled with pooling operations for translation invariance and shared weights for parameter efficiency, enables CNNs to generalize across visual tasks, such as automatic annotation [3, 4].

We use the pre-trained feature extractor VGG19 to extract features that are used for automatic annotation. This is accomplished by transfer learning. Transfer learning retains a network's convolutional layers and replaces its top layers for new tasks [5, 6]. By freezing the parameters of VGG19, we preserve generic visual representations while adding classification heads tailored for our annotation categories, as defined in Section 2.2.1. An overview of the multi-head architecture is shown in Figure 3.1.

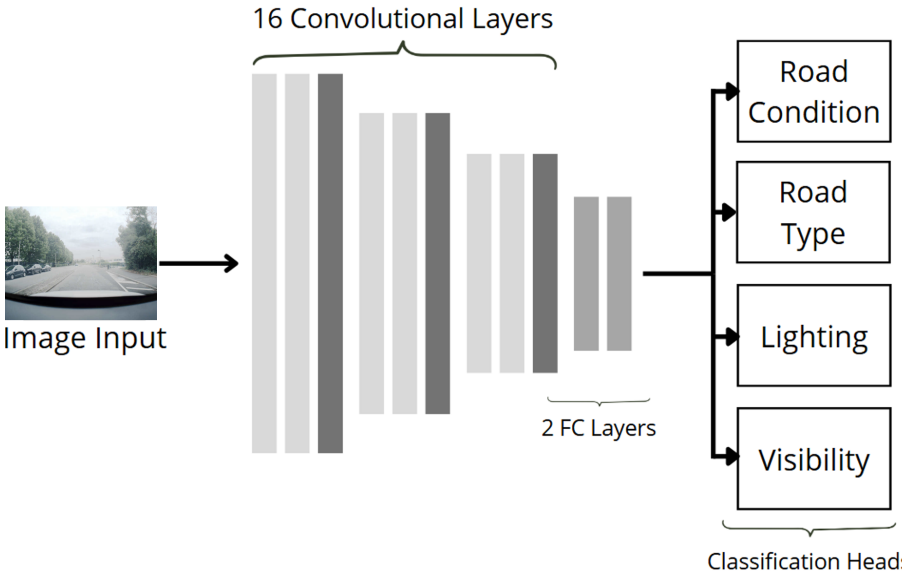


Figure 3.1: Model overview: An input image is processed by a VGG19 backbone consisting of 16 convolutional layers followed by two fully connected layers to produce a single feature vector. The feature vector is then fed into four parallel classification heads that predict the corresponding class within each category: road condition, road type, lighting, and visibility.

3.2 Experiment with Masking Images

An experiment was conducted to investigate whether predictions can be improved by masking certain parts of the images, we employ a semantic segmentation mask on input images generated by Detectron2 [7, 8]. By isolating the road and sky regions, a CNN trained on masked images will not be influenced by features on the side of the roads, as illustrated in Figure 3.2.

(a) Original image (b) Masked image showing road and sky

Figure 3.2: Comparison of the same scene: (a) the full original image and (b) the same image with a mask applied to show only road and sky regions.

3.3 Training Classification Heads and Fine-Tuning

From the pre-trained VGG19 model, classification heads were created and added as the new output layers for our annotation categories. These heads were then trained on the training portion of the dataset constructed in Chapter 2. During this training, all convolutional layers of the pre-trained VGG19 feature extractor remained frozen so that only the parameters of the newly added heads were updated.

Each input image is first downsized to 224 x 224 pixels, and normalized. The input is then propagated through the frozen feature extractor and collapsed by global average pooling into a 4096-dimensional feature vector. That vector is fed independently into each head, two heads in the Masked-CNN variant for road condition and road type, and four heads in the Full-Frame-CNN variant adding visibility and lighting. This parallel head structure ensures that adjusting parameters in one head does not influence the others. Training is carried out for a predefined number of epochs, with checkpoints assessed on the validation set to identify the best-performing epoch.

After the heads reach their best performance, all convolutional layers of the backbone are unfrozen and training resumes in a process known as fine-tuning. The same annotated images continue to drive learning, but now both the original backbone and the added heads adjust their parameters. Learning rates are reduced and regularization is applied to prevent overfitting. After each training epoch, the updated model is evaluated on the validation set to determine the best-performing epoch. The hyperparameters used during training are listed in Appendix A.

3.4 Evaluation of Vision Models

In this section, the procedure for evaluating all classification models is outlined. Performance metrics are calculated for each Vision Model and the results for the are presented and compared.

3.4.1 Performance Metrics

To assess each model's performance across the four classification categories, four metrics are computed:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}},$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}},$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

These metrics are calculated using the values:

- **True Positives (TP):** Sequences where the model correctly predicts the class assigned in the ground-truth label.
- **False Positives (FP):** Sequences where the model predicts a class that does not match the ground-truth label.
- **False Negatives (FN):** Sequences where the model fails to predict the class assigned in the ground-truth label.
- **True Negatives (TN):** Sequences where the model correctly identifies that a specific class is not the ground-truth label.

Of the presented metrics, the F1-score is used as the primary performance indicator, since it combines precision and recall into a single measure and thus provides the most balanced overview of model performance.

3.4.2 Validation and CNN Evaluation

Model performance was assessed on the validation set by comparing each model's predicted labels against the manually annotated ground truth. For every category, predictions were scored using the metrics defined in Section 3.4.1, and confusion matrices were generated to highlight patterns of misclassifications.

Both the Masked CNN and Full-Frame CNN variants were evaluated in this manner. Each model processed its respective input, masked or full-frame images, and produced one label per category. Those labels were then directly compared to the ground truth annotations, yielding per-category performance scores and confusion matrices that reveal how each architecture handles different classes.

3.4.3 CLIP Evaluation

In addition to CNN-based classifiers, we also investigated whether a pre-trained Contrastive Language-Image Pretraining (CLIP) model could be feasible to use for automatic labeling, without any additional training. CLIP pairs an image encoder and a text encoder, learning to align their outputs in a shared embedding space through contrastive pretraining on large-scale image–text datasets [9].

For each of the four annotation categories, a set of descriptive text prompts, one per class, is created. During inference, each validation image is passed through CLIP's image encoder to produce an image embedding. Cosine similarities are then computed between this embedding and each class's text-prompt embedding and the highest-scoring class is selected as the prediction. The prompts used for the CLIP model can be found in Appendix A.

These prediction are evaluated with the same metrics defined in Section 3.4.1, enabling a direct comparison of CLIP's performance against that of the CNN variants.

3.4.4 Comparison of Vision Model Variants

To determine the strongest performing vision model, the following five models were evaluated: two Full-Frame CNNs under frozen and fine-tuned configurations, two Masked CNNs under frozen and fine-tuned configurations, and a CLIP model. For each variant, the checkpoint achieving the highest average F1 score on the validation set was selected for comparison. Table 3.1 reports the per-category and overall F1 results.

Evaluation shows that the Full-Frame CNN under frozen configuration delivers the highest overall average F1, driven by its performance on road condition, lighting and visibility. The fine-tuned variant of the same architecture yields a improvement on road type but shows a slight decline in the other categories. The masked CNN benefit significantly from fine-tuning in the road condition and road type annotaion. The CLIP model, while offering competitive performance on road condition, underperforms compared to all CNN variants across the remaining categories.

Table 3.1: F1 scores by model and category (FF = Full-Frame, M = Masked).

Model	**Road Condition (%)**	**Road Type (%)**	**Lighting (%)**	**Visibility (%)**	**Average (%)**
FF CNN[a]	**88.07**	85.23	**99.10**	**61.04**	**83.36**
FF CNN[b]	87.26	**86.44**	98.45	60.10	83.06
M CNN[a]	75.13	80.56	–	–	–
M CNN[b]	78.64	81.17	–	–	–
CLIP	81.40	63.47	87.47	40.20	68.13

[a] frozen configuration [b] fine-tuned configuration

Based on these results, the frozen Full-Frame CNN is chosen as the vision model for the annotation pipeline. Its confusion matrix in Figure 3.3, further illustrates strong diagonal accuracy across most classes.

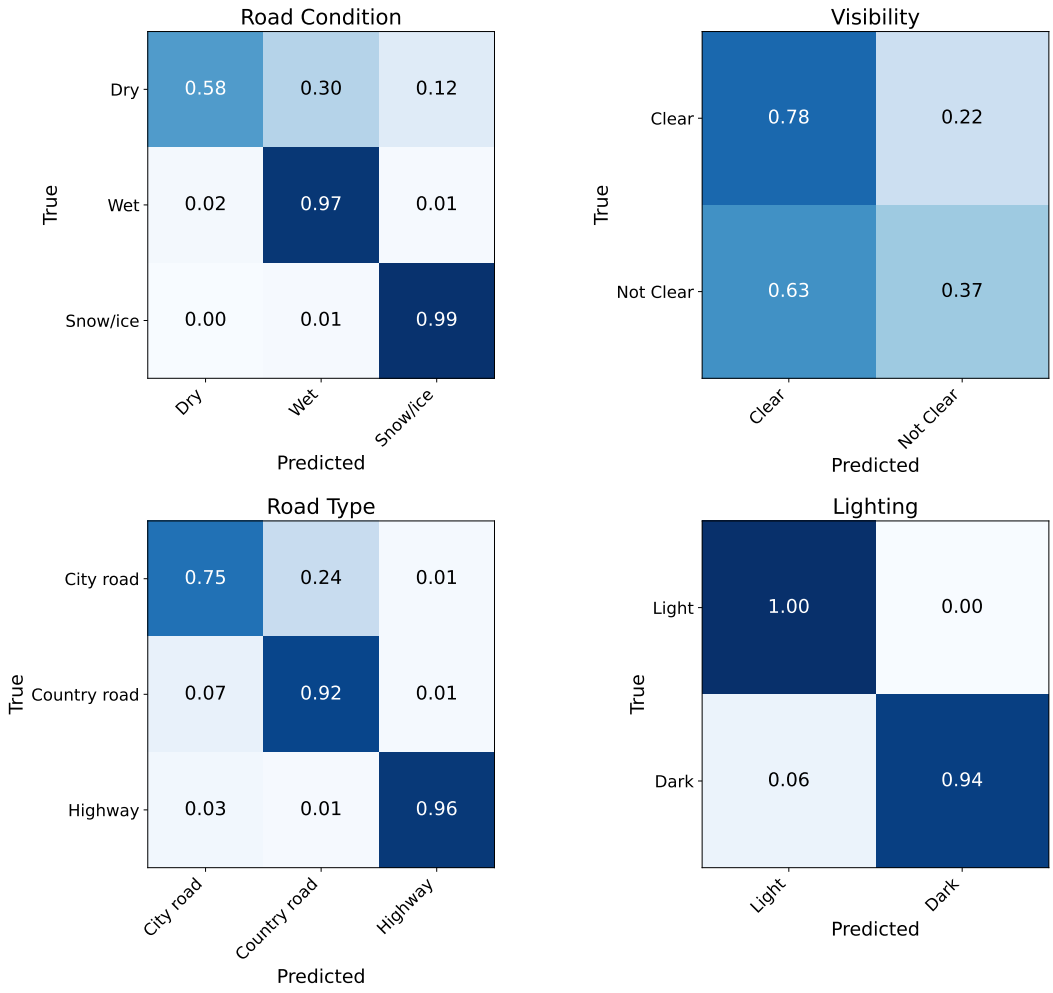


Figure 3.3: Confusion matrix for the Full-Frame CNN without fine-tuning on the validation set.

4

LiDAR-Based Classification Model

To evaluate whether the LiDAR point clouds in our dataset can effectively contribute to automatic annotation, we train and evaluate a PointNet-based classifier that predicts the four target categories—road condition, road type, lighting and visibility. We explore different sampling strategies for the raw LiDAR point clouds, evaluating both points sampled from the full-scenes, and points sampled exclusively from the ground level.

All variants share the same network architecture and are evaluated on the same validation set, detailed in Section 2.2.5. By comparing accuracy, generalization behavior and the feasibility of each sampling strategy given our dataset, we assess which configuration is most suitable for the annotation pipeline. The following sections describe point-cloud sampling, the PointNet model architecture, training process and final performance.

4.1 Model Architectures

To efficiently utilize the LiDAR data in our dataset for automatic annotation, a model architecture capable of processing raw 3D point clouds was required. Point-based models are therefore a natural choice, since they operate directly on unordered point sets, like the ones present in our dataset. Among these methods, PointNet has emerged as a straightforward yet effective approach [10]. It is efficient, scales to large point sets, and preserves each scene's geometric structure, making it well suited for annotating driving scenarios directly from LiDAR point clouds.

4.1.1 Theory behind PointNet

PointNet is a neural network architecture designed specifically for unordered point clouds, such as those recorded by LiDAR [10]. It works by applying a series of shared multilayer perceptrons (MLPs) independently to each point, producing per-point feature vectors. These features are then combined into a single representation of the entire cloud. To do this, PointNet uses a symmetric aggregation function, typically max-pooling, that takes the maximum activation over all points for each feature dimension.

Because max-pooling is permutation invariant, the resulting global feature does not depend on the order in which points are presented. Permutation invariance

means that any order of the same set of points yields the same aggregated output, which is essential for point clouds that have no ordered structure. This aggregated feature therefore captures the overall shape and structure of the scene in a consistent manner, and the resulting global descriptor can be used for downstream prediction tasks such as scene classification.

4.1.2 Customized PointNet

We created a custom version of PointNet, tailored for multi-task classification. The model follows the general theory of PointNet by applying shared transformations to each point and aggregating them into a global representation using a symmetric function. Specifically, we implement the shared MLPs using 1×1 convolutional layers combined with batch normalization and ReLU activations, as this is equivalent to applying fully connected layers independently to each point. This produces per-point features that are then aggregated using a global max-pooling operation over all N_{pts} points.

The resulting global feature vector of size 512 is used as a common representation across categories. This vector is passed to multiple fully connected heads, one for each classification category: road condition, road type, lighting, and visibility. Each head outputs class logits corresponding to its category. This modular design allows the model to learn shared geometric features while also specializing to the individual categories.

The input size of the network depends on the number of points N_{pts} sampled from each point cloud. During data loading, point clouds with fewer than N_{pts} points are discarded, and those with more are downsampled randomly to exactly N_{pts} points.

Table 4.1: Architecture of the multi-task PointNet model.

Component	Layer	Output Shape
Backbone	1×1 conv (4→64), batch norm, ReLU	$64\times N_{\text{pts}}$
	1×1 conv (64→128), batch norm, ReLU	$128\times N_{\text{pts}}$
	1×1 conv (128→256), batch norm, ReLU	$256\times N_{\text{pts}}$
	1×1 conv (256→512), batch norm, ReLU	$512\times N_{\text{pts}}$
	Max-pool over points	512
Heads	Fully connected (512→3) for road condition	3
	Fully connected (512→3) for road type	3
	Fully connected (512→2) for lighting	2
	Fully connected (512→2) for visibility	2

Table 4.1 outlines the layers and output shapes of the model. The shared backbone transforms each input point independently and produces a global feature through max-pooling. This feature vector is then used for classification by each of the category-specific heads. The design leverages the strengths of the original PointNet framework while adapting it for efficient multi-task learning on large point clouds.

4.2 LiDAR Point Cloud Sampling

LiDAR point clouds naturally vary in size because the number of returns recorded by the sensor depends on the environment. For example: urban scenes with many objects produce dense clouds, while rural roads over open fields produce more sparse clouds. Before training any neural network, this variability must be addressed, since neural networks generally requires a fixed input size for every sample.

To enforce a consistent input shape, we sample a fixed number of points N_{pts} from each cloud. Retaining more points naturally preserves more details but increases memory use and computation time. Furthermore, raising N_{pts} forces more clouds to be discarded as only those with at least N_{pts} points can be used, meaning that there is a trade-off between model performance and dataset coverage.

We examined the distribution of number of points per cloud across all LiDAR data in our dataset, as visualized in Figure 4.1. Based on this analysis, we chose $N_{\text{pts}} = 16\,384$ as our upper limit because it retains 93.8% of the dataset. Any higher threshold would have forced us to discard a substantial number of samples, reducing the amount of data available for training and limiting the portion of the dataset that could be automatically annotated. The red bars in Figure 4.1 show the amount of LiDAR data that that needs to discarded with the chosen threshold of $N_{\text{pts}} = 16\,384$.

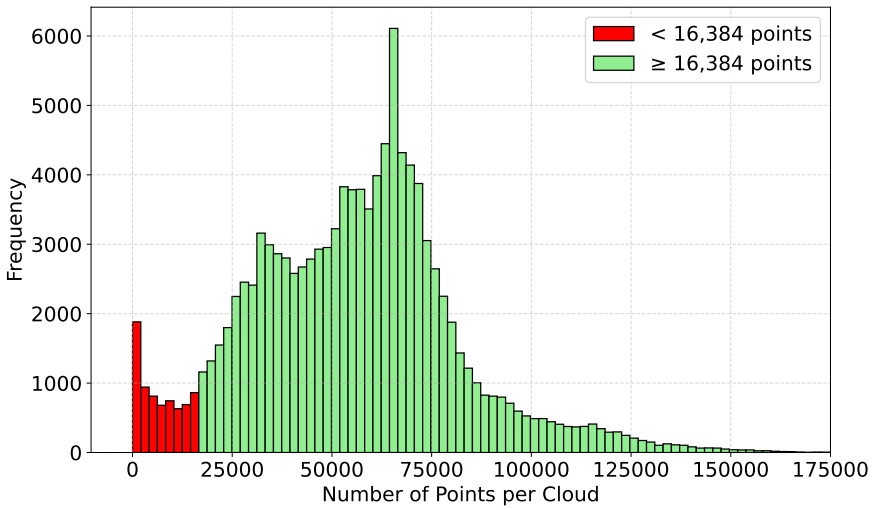


Figure 4.1: Distribution of number of points over all point clouds, with no particular filtering. Bins displayed in red correspond to point clouds with less than 16 384 points, that were discarded when loading the dataset for the model using 16 384 points.

To evaluate differences in prediction accuracy across sampling strategies, two full-scene models were selected to determine the best way to sample the LiDAR point clouds. One used our chosen maximum number of points $N_{\text{pts}} = 16\,384$ but was trained only on every fourth point cloud to reduce training time. The other used

$N_{\text{pts}} = 2\,048$, allowing training on every point cloud in the training set. Both models were trained to predict all four categories: road type, road condition, lighting and visibility.

An additional experiment investigated whether focusing exclusively on road-surface points could yield better results for predicting road condition and road type. We found that the LiDAR sensor used to record the LiDAR data is mounted roughly two meters above ground at the front center of the vehicles, with the point-cloud origin aligned to the sensor. Based on this setup, we applied a filter that removes all points outside a lateral range of ±10 m, eliminating points from objects at the side of the road. Similarly, every point with height $z > -1.25$ m was removed to exclude points more than 75 cm above the estimated road plane.

Filtered clouds containing fewer than $N_{\text{pts}} = 2\,048$ points were excluded, and the remaining clouds were randomly downsampled to that size. We chose 2 048 points after examining the distribution of post-filter counts shown in Figure 4.2, this threshold retains the majority of ground-level clouds. Using a higher threshold would have removed an excessive number of point clouds, reducing training coverage and a lower threshold would have sacrificed geometric detail. Because this setup focuses exclusively on road-surface points, the model was trained only on the two most relevant categories for this case: road condition and road type.

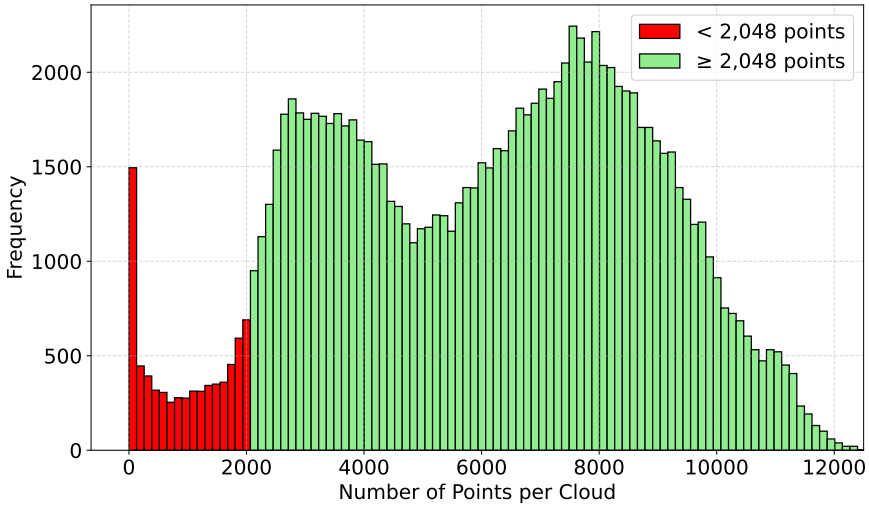


Figure 4.2: Distribution of number of points over all point clouds where clouds have been filtered to retain only the ground-level points. Bins displayed in red correspond to point clouds with less than 2 048 points, that were discarded when loading the data for the model trained on ground-level points.

4.3 Training of the PointNet Model

Training of our customized LiDAR PointNet model is performed over 30 epochs with a learning rate of 1×10^{-3}. We use the Adam optimizer which is well-suited for

handling sparse gradients and noisy data as commonly found in point cloud classification tasks [11]. We apply the same network architecture across all three sampling strategies, and use the predefined train and validation split from Section 2.2.5, ensuring that validation sequences come from drives unseen during training. This setup provides a realistic measure of each sampling strategy's generalization.

In each iteration, we sum the classification losses from all output heads to update model weights. After every epoch, we record validation performance and save the checkpoint with the highest average accuracy across categories. All metrics are logged for subsequent analysis.

Figure 4.3 shows the training and validation curves for the full-scene model with $N_{\text{pts}} = 16\,384$. Training accuracy steadily increases for all categories, while validation accuracy fluctuates more. Road condition validation closely follows training, indicating good generalization. In contrast, road type, visibility and especially lighting show some divergence, suggesting mild overfitting.

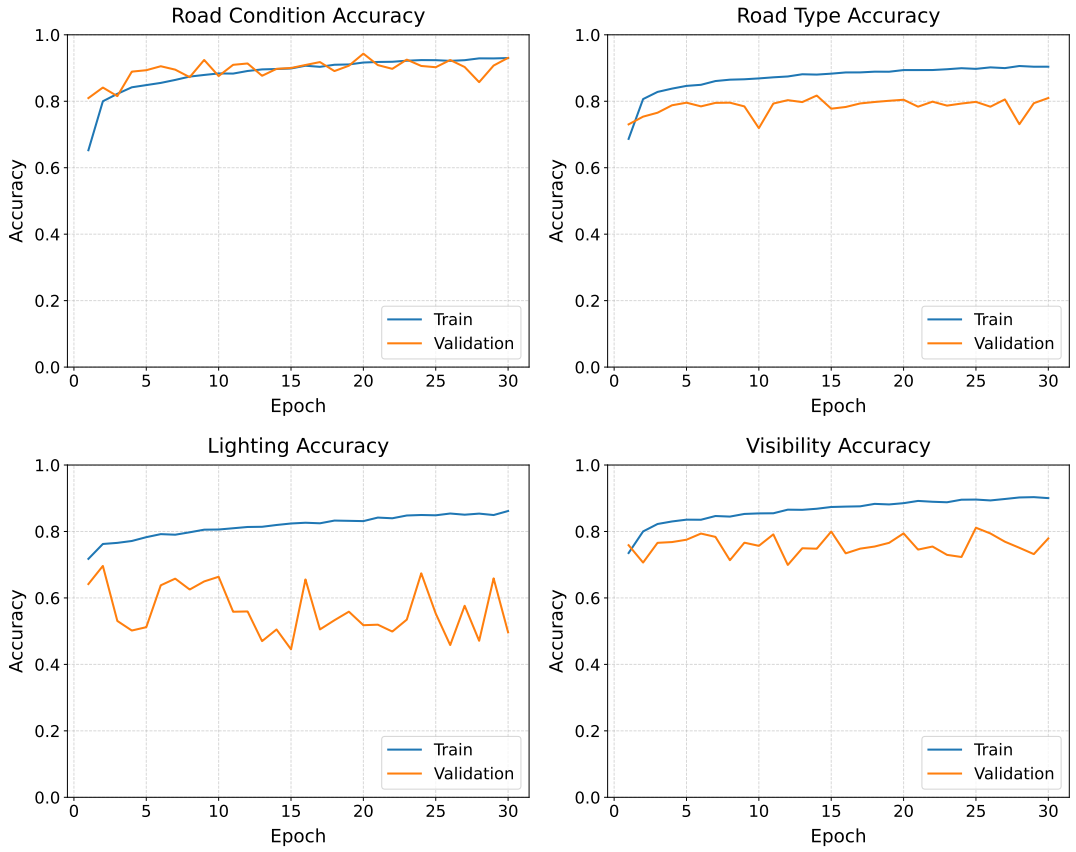


Figure 4.3: Training and validation accuracy for the full-scene model with $N_{\text{pts}} = 16\,384$.

Figure 4.4 presents the curves for the full-scene model with $N_{\text{pts}} = 2\,048$, trained on every point cloud. Road condition and road type achieve stable convergence, but

lighting validation accuracy drops sharply after about five epochs, indicating overfitting. Visibility follows the training trend more closely but still shows fluctuations.

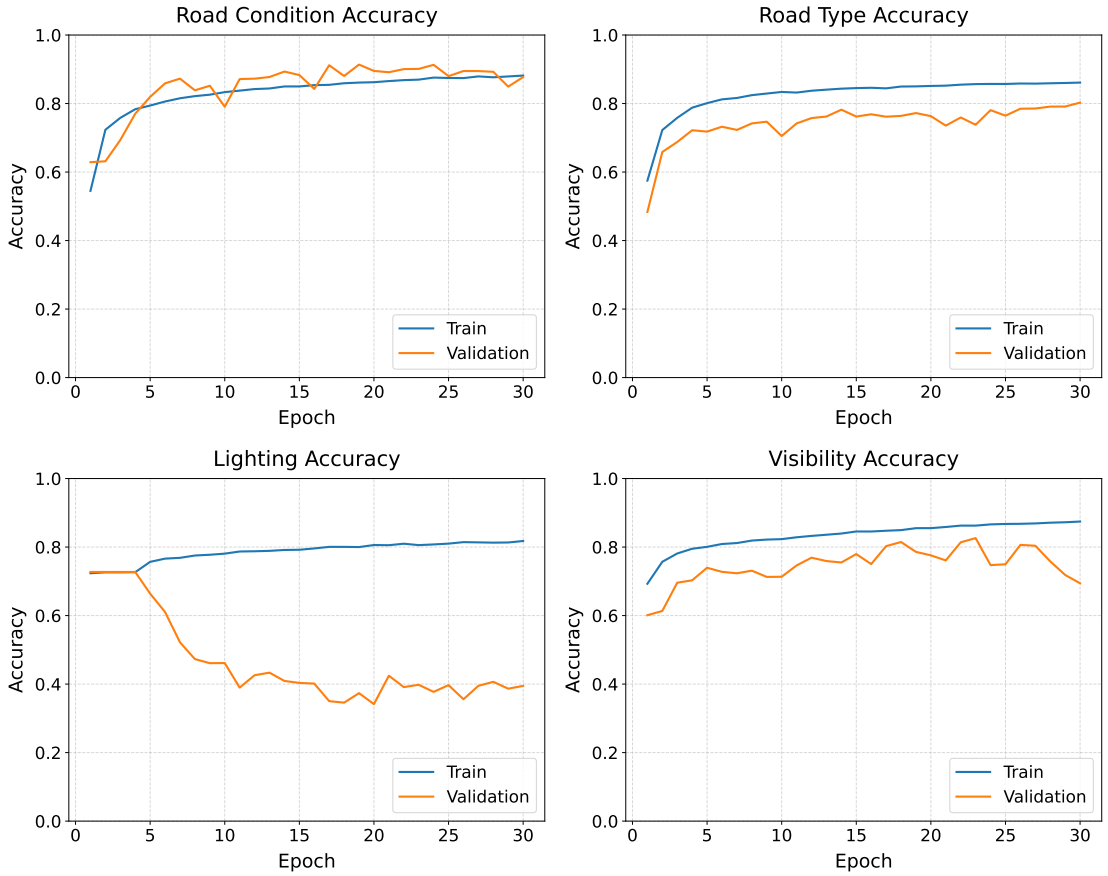


Figure 4.4: Training and validation accuracy for the full-scene model with $N_{\text{pts}} = 2\,048$.

Figure 4.5 shows training for the ground-only model with $N_{\text{pts}} = 2\,048$. This configuration converges rapidly, and validation closely tracks training for both road condition and road type, reflecting that focusing on surface points reduces noise for those tasks.

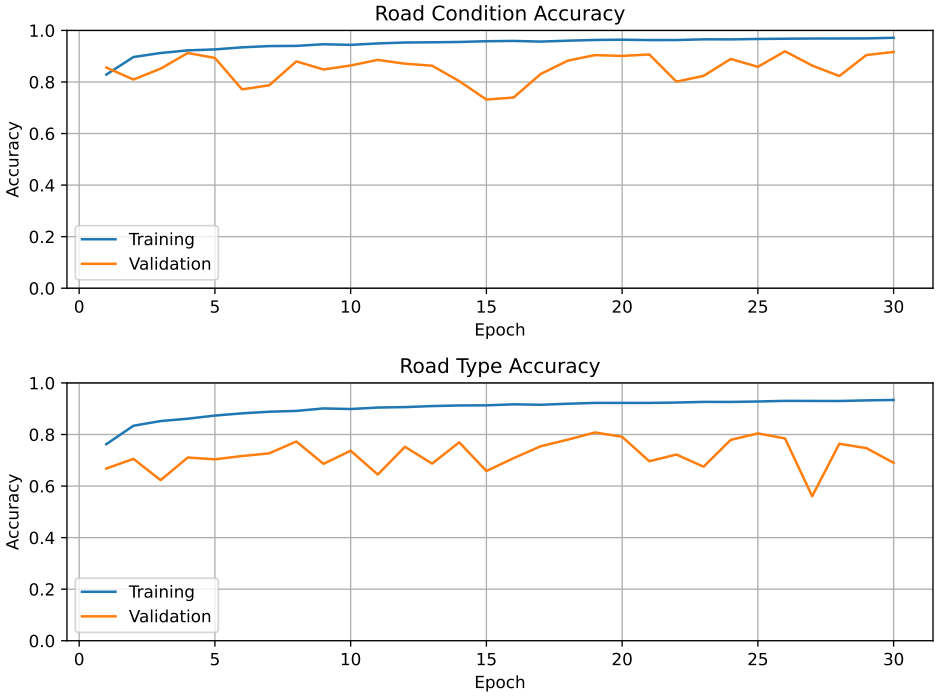


Figure 4.5: Training and validation accuracy for the ground-only model with $N_{\text{pts}} = 2\,048$.

4.4 Comparison

For each sampling strategy, we retain the checkpoint with the highest validation accuracy and generate predictions on the full validation set. We compute F1-score as our primary metric, balancing precision and recall, as detailed in Section 3.4.1. Table 4.2 summarizes performance across categories.

Table 4.2: Validation F1-scores for each LiDAR sampling strategy.

Model	Road Condition	Road Type	Lighting	Visibility	Average
Full scene ($N_{\text{pts}} = 16\,384$)	88.73	79.92	**40.57**	**75.72**	**71.24**
Full scene ($N_{\text{pts}} = 2\,048$)	82.51	72.10	40.05	73.28	66.99
Ground only ($N_{\text{pts}} = 2\,048$)	**89.73**	**80.92**	—	—	—

The results show that the sampling strategy using full-scene clouds with $N_{\text{pts}} = 16\,384$ outperforms the reduced full-scene strategy ($N_{\text{pts}} = 2\,048$) across all categories. However, for road condition and road type, the ground-only strategy slightly outperforms both full-scene strategies. The gain is marginal, around 1% compared to the larger full-scene approach, and since the ground-only strategy cannot address lighting or visibility, it would need to be combined with a full-scene strategy in a production setup.

Given this trade-off, the full-scene sampling strategy with $N_{\text{pts}} = 16\,384$ was chosen for integration in the annotation pipeline, as further described in Chapter 5.

5
Fusion Model

To determine whether combining camera and LiDAR predictions can contribute to more accurate automatic annotations, we develop and evaluate a fusion model that ingests outputs from the vision model detailed in Chapter 3, and the LiDAR model detailed in Chapter 4. These models produces raw logits for our four category labels and the fusion layer learns to integrate these complementary signals into a single prediction.

We present a lightweight Fusion MLP and compare it against three baseline models, logistic regression, support vector machine and decision tree. All fusion methods are evaluated on the same validation set defined in Section 2.2.5, with a focus on performance, generalization, and practical integration into our annotation pipeline. Subsequent sections present the fusion architectures, training procedure, and final evaluation results.

5.1 Fusion Model Architecture

In order to leverage both the images and LiDAR point clouds available in our dataset for automatic annotation, a fusion model was developed to combine the outputs of the vision model presented in Chapter 3 and the LiDAR model presented in Chapter 4. An overview of the automatic annotation pipeline is shown in Figure 5.1. When used independently, each classifier applies an arg-max to its raw output scores, called logits, to select the most likely class per category. However, to preserve the full predictive information, the fusion module instead uses the complete vector of raw logits from each model rather than their arg-maxed predictions.

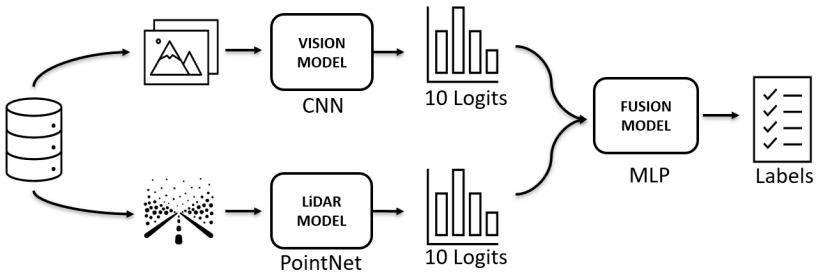


Figure 5.1: Automatic annotation pipeline: 10 logits each from the CNN vision model and the PointNet LiDAR model are fused by an MLP to produce the final labels.

Both the LiDAR and Vision models predict four categories: road condition (3 classes), road type (3 classes), lighting (2 classes) and visibility (2 classes), yielding a total of 10 raw logits per model. Consequently, the fusion module receives 20 input features: 10 logits from the Vision model and 10 from the LiDAR model. A compact MLP then combines these inputs to produce a unified prediction per scene. Separate MLPs are trained for each classification category. The chosen architecture of the MLP used for fusion is presented in Table 5.1.

Prior to training, each block of logits is normalized via softmax and then standardized using a scaler fitted on the training data, ensuring consistent preprocessing across all fusion models.

Table 5.1: Structure of each decision MLP. C is the number of classes for the given category (2 or 3).

Component	Layer	Output Shape
Hidden transformation	Fully connected ($20 \rightarrow 16$), ReLU	16
Regularization	Dropout (p=0.5)	16
Output projection	Fully connected, no activation	C

As a baseline comparison, three simpler decision models were also evaluated: logistic regression, support vector machine, and decision tree. These models were trained on the same 20-dimensional input vector. As with the MLP, logits from each sensor were first passed through a softmax function to normalize them to pseudo-probabilities, and then standardized.

Logistic regression is a linear model that estimates class probabilities using a softmax over linear combinations of input features. The support vector machine with an RBF kernel is a non-linear classifier that separates classes by maximizing the margin in a high-dimensional feature space. The decision tree classifier splits the feature space into axis-aligned regions by recursively selecting thresholds that maximize class separation. Each of these models was trained independently for each category. These comparisons serve as baseline alternatives to the MLP.

5.2 Training of the MLP Models

All four category-specific fusion MLPs were optimized with the Adam optimizer. Early stopping with a patience of eight epochs, monitored on validation accuracy, prevented overfitting, and the best checkpoint for each category was saved for evaluation.

Figure 5.2 shows the training and validation accuracy curves for each category. Across road condition, road type, and lighting, training and validation accuracy track closely, indicating stable generalization. The visibility model's validation accuracy begins to diverge after roughly 20 epochs, at which point training was halted.

Because of early stopping, the total number of epochs varies by category, but in every case the model state with peak validation accuracy was retained.

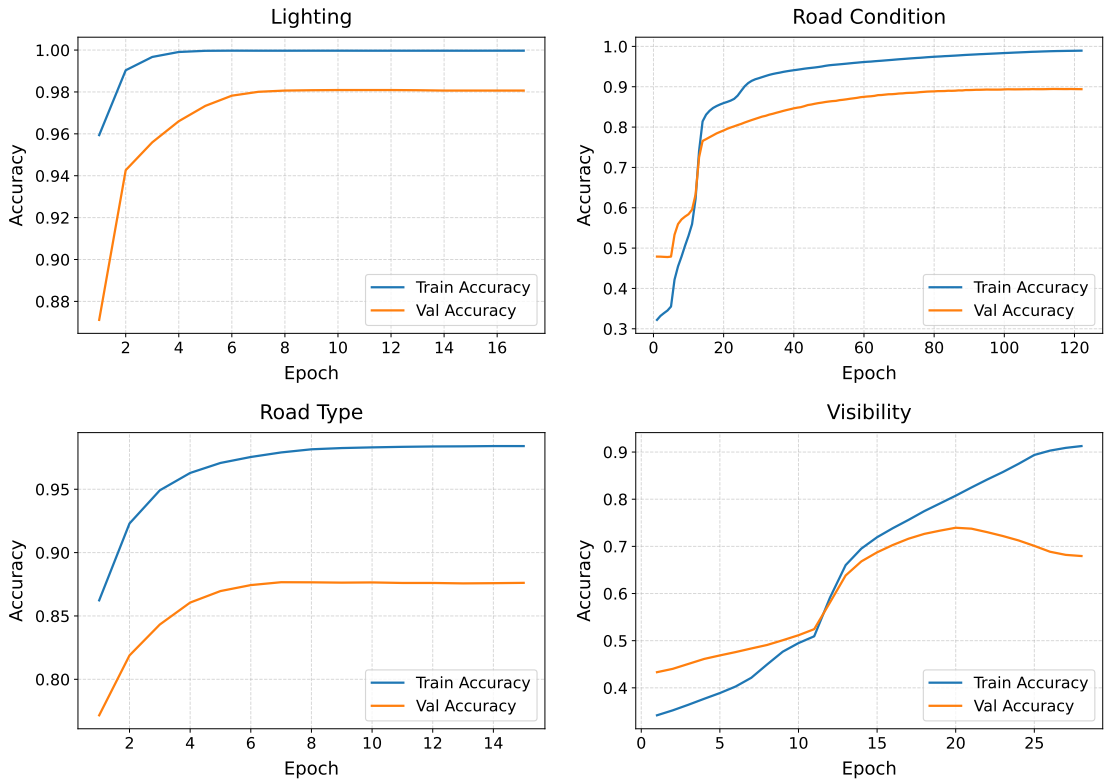


Figure 5.2: Training and validation accuracy per epoch for the MLP for each category.

5.3 Fusion Model Evaluation

The performance of all four evaluated fusion methods is summarized in Table 5.2, which reports the validation F1-score per category and the overall average. The MLP achieved the highest overall performance, especially on the visibility category, which proved challenging for other models.

Table 5.2: Validation F1-scores (%) for each category and overall average for four classification models.

Model	Road Cond. (%)	Road Type (%)	Lighting (%)	Visibility (%)	Overall (%)
Logistic Regression	85.4	**88.0**	97.6	57.6	82.1
Support Vector Machine	85.0	**88.0**	**97.7**	57.7	82.1
Decision Tree	84.5	87.2	97.5	57.5	81.7
MLP	**86.1**	86.4	97.6	**74.4**	**86.1**

The MLP achieved the best overall average across all categories, 86.1% compared to the seconds best model at 82.1%. It also outperformed all baseline classifiers on the most difficult category, visibility, with a validation F1-score of 74.4%. This strong performance, combined with its fast training time and capacity to model interactions between input features, makes the MLP the most suitable decision layer in the annotation pipeline.

Further insight into the MLP's performance is given by the confusion matrix in Figure 5.3.

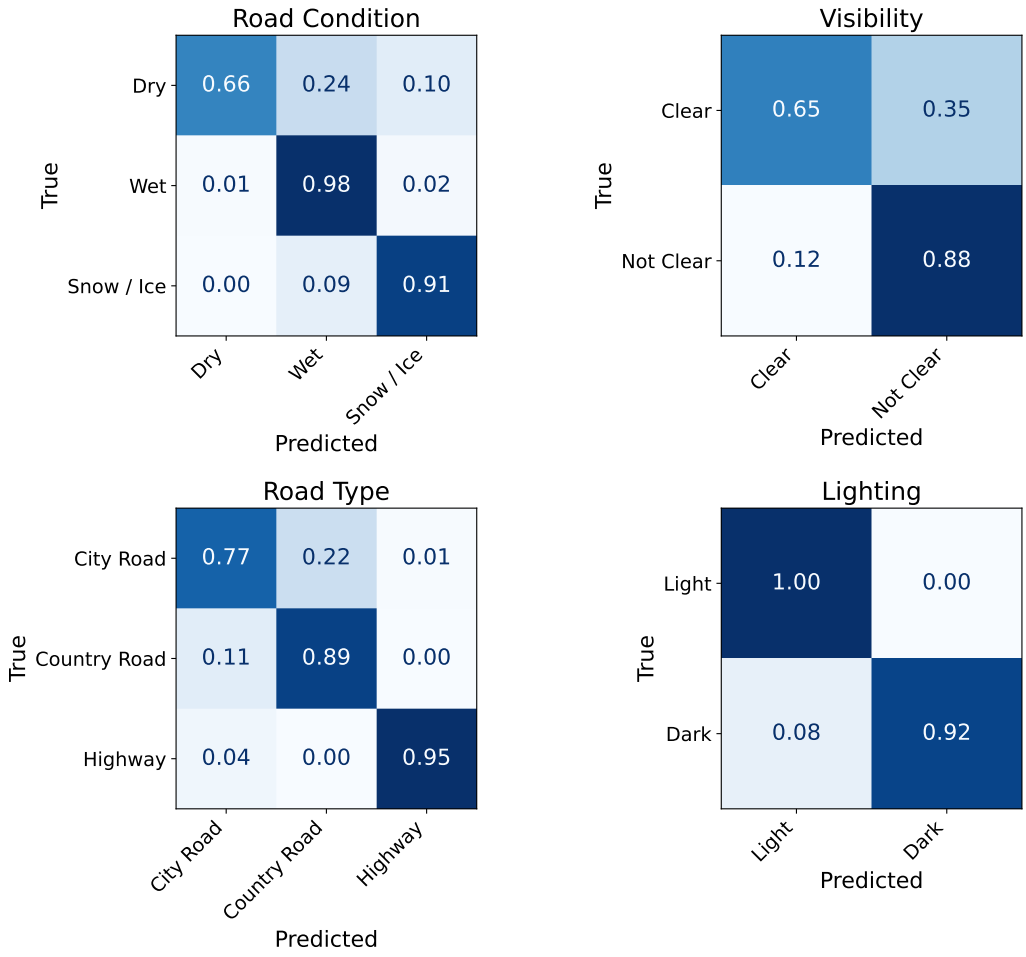


Figure 5.3: Confusion matrix for the MLP fusion model on the validation set, showing per-class accuracy.

A detailed breakdown of the MLP's performance on the validation set is provided in Table 5.3. Accuracy, precision, recall, and F1-score are reported for each category. These results indicate that fusing vision and LiDAR inputs yields improved performance compared to using a single sensor modality.

Table 5.3: Performance of the MLP fusion model on validation sets.

Category	Accuracy (%)	Precision (%)	Recall (%)	F1 Score (%)
Road Condition	86.4	88.6	86.4	86.1
Road Type	86.3	86.8	86.3	86.4
Lighting	97.7	97.7	97.7	97.6
Visibility	74.3	78.6	74.3	74.4
Average	86.1	87.9	86.1	86.1

6

Sequence Prediction Aggregation and Model Comparisons

This chapter describes a sequence-level prediction aggregation method that transforms per-frame logits into a single, one-minute prediction by averaging logits. It then reports performance metrics for the Vision, LiDAR and Fusion models, highlighting the improvements gained by smoothing frame-level noise. Finally, a direct comparison of all three approaches to demonstrate the additional gains achieved through multi-sensor fusion.

6.1 Sequence Prediction Aggregation

The Vision, LiDAR, and Fusion models all predict a category label for each individual scene, whereas the ground-truth labels were annotated for entire one-minute sequences comprising 60 scenes each, as the labels remain constant throughout the sequence. To align model outputs with the annotation format, we leverage prediction aggregation to convert the 60 scene-level predictions into a single prediction per sequence. The scene-level outputs from the models consist of raw logits, one logit per class label.

Aggregation of scene-level predictions is performed by averaging the logits over the entire sequence, followed by applying softmax to obtain the final sequence-level prediction. This approach not only matches the format of the ground-truth labels but also improves predictive reliability. As demonstrated in [12], averaging logits across multiple samples yields better confidence calibration and higher overall accuracy than either majority voting or direct averaging of class probabilities.

Mathematically, the aggregation works by averaging the logits for each class across all frames in the sequence. For each driving sequence s, consisting of data points $i = 1, \ldots, N_s$, we let $L_{i,c}$ denote the logit for class c on frame i. The sequence-level logit is then computed as:

$$\bar{L}_{s,c} \;=\; \frac{1}{N_s} \sum_{i=1}^{N_s} L_{i,c},$$

and the predicted class is selected as

$$\hat{y}_s \;=\; \arg\max_{c} \; \bar{L}_{s,c}.$$

Averaging in this way smooths out frame-by-frame noise and yields a single, robust prediction per category for the entire sequence.

6.2 Prediction Aggregation for Vision Model

To evaluate the effect on the vision model of aggregating predictions over sequences, we apply this aggregation to the predictions from the best vision model from Chapter 3, namely the Full-Frame CNN. We use this model to perform inference on the validation set and record the raw logits for each scene. These predictions are then aggregated across sequences, and the resulting accuracies per category are shown in Figure 6.1.

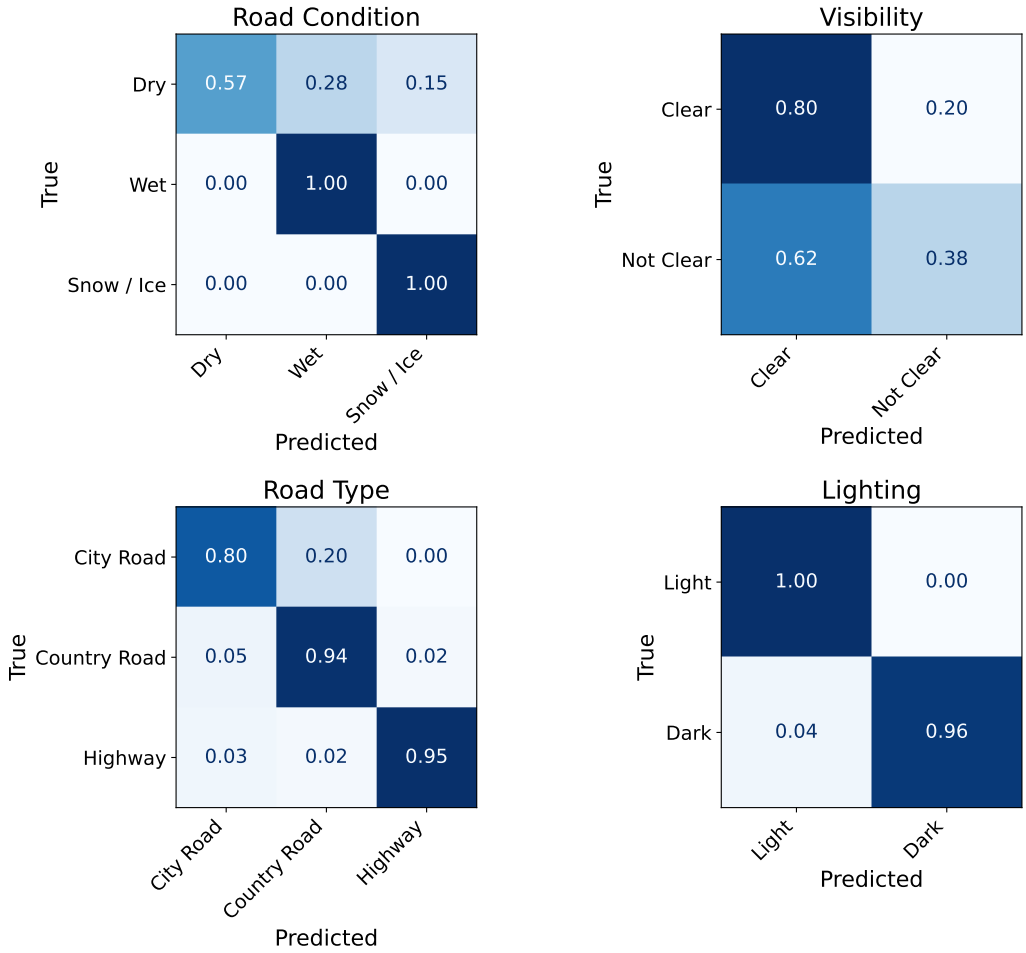


Figure 6.1: Confusion matrices for the Full-Frame CNN after sequence-level logit averaging.

Table 6.1 presents the sequence-level evaluation metrics for the Vision model following the application of logit averaging. Figure 6.2 illustrates the absolute changes in F1-score for each annotation category when comparing sequence-level predictions against their single-frame counterparts. The most obvious improvement occur in the

road type category and in the overall average F1-score, whereas the road condition and lighting categories show slight reductions.

Table 6.1: Sequence-level performance of the Full-Frame CNN after logit averaging and softmax.

Category	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Road Condition	88.44	90.29	88.44	87.35
Road Type	88.94	89.78	88.94	88.99
Lighting	98.99	99.01	98.99	98.99
Visibility	62.81	61.51	62.81	61.04
Average	84.80	85.15	84.80	84.09

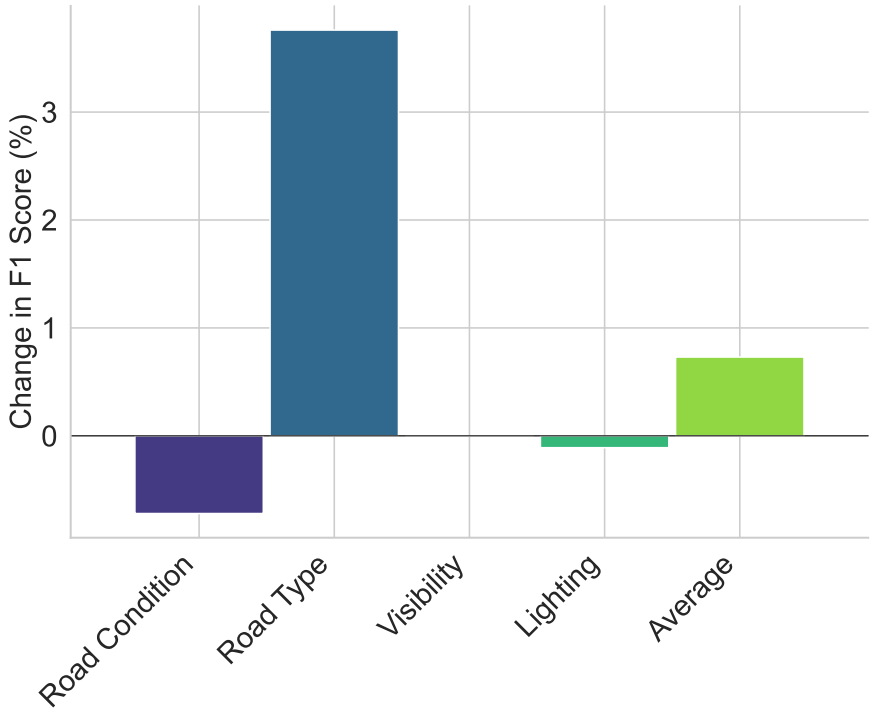


Figure 6.2: Absolute change in F1-score (%) per category for the Full-Frame CNN after logit averaging.

6.3 Prediction Aggregation for LiDAR Model

Similar to the Vision model, we apply sequence-level prediction aggregation to evaluate its effect on the LiDAR-based classifier. The model used is the best-performing LiDAR model from Chapter 4, specifically the full-scene PointNet trained on 16 384 points per scan.

The aggregation is performed in the same way as for the Vision model, by averaging logits across each one-minute sequence to produce a single prediction per category. The resulting sequence-level predictions are visualized as confusion matrices in Figure 6.3, and the corresponding accuracy, precision, recall, and F1-score per category are reported in Table 6.2.

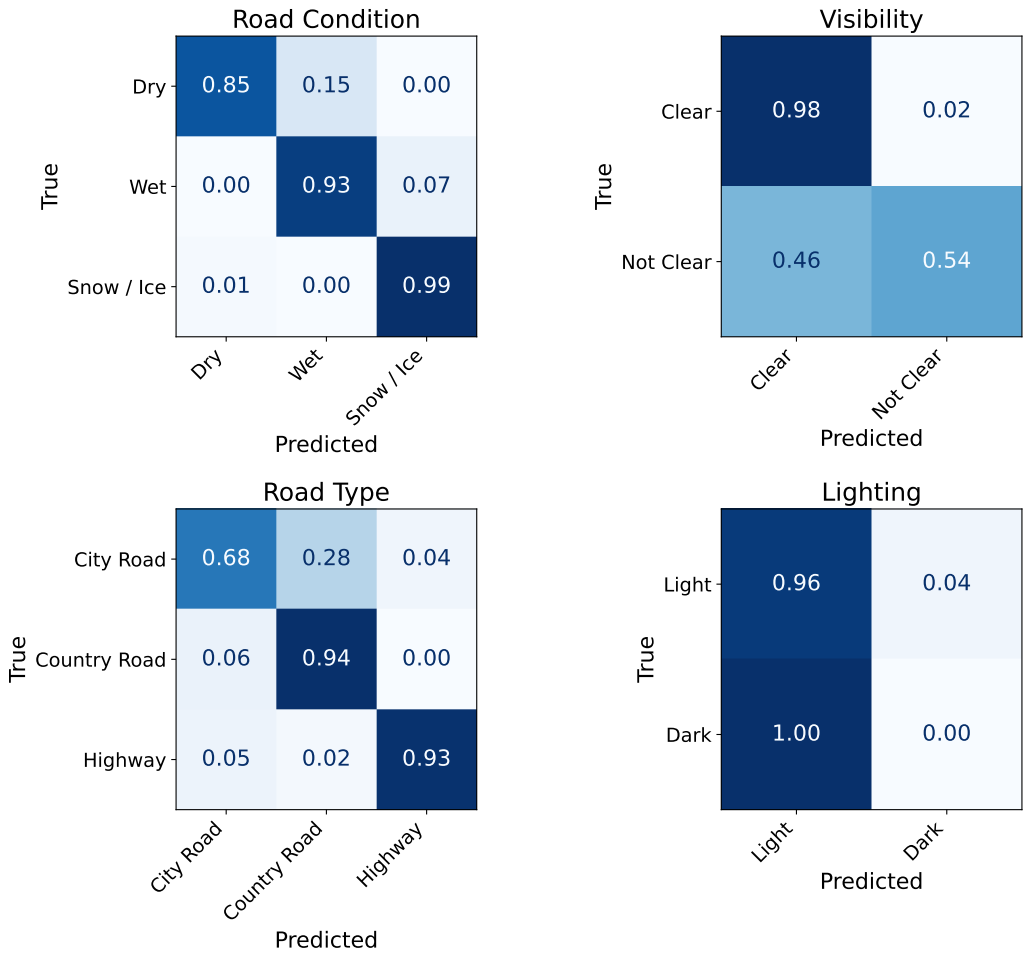


Figure 6.3: Confusion matrices for the LiDAR model after sequence-level logit averaging.

Table 6.2: Sequence-level performance of the LiDAR model after logit averaging.

Category	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Road Condition	93.47	93.71	93.47	93.43
Road Type	83.92	85.23	83.92	83.69
Lighting	69.35	51.74	69.35	59.26
Visibility	80.40	83.85	80.40	78.91
Average	81.79	78.88	81.79	78.82

Figure 6.4 shows the absolute change in F1-score for each annotation category when comparing sequence-level predictions to their per-scan counterparts. Road condition improves by roughly 5 percentage points, road type by about 4 points, and visibility by around 3 points. The most substantial gain occurs in the lighting category, with an increase of approximately 18 points. On average, sequence-level aggregation boosts the F1-score by about 7.5 points, highlighting its effectiveness in producing more reliable LiDAR-based annotations.

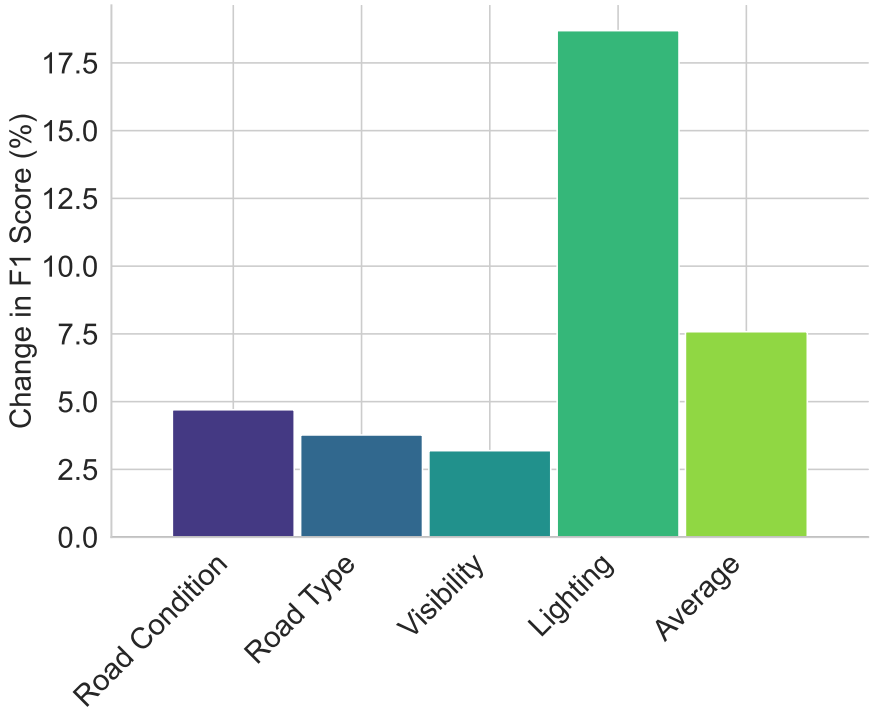


Figure 6.4: Absolute change in F1-score (%) per category for the LiDAR PointNet model after logit averaging.

6.4 Prediction Aggregation for Fusion Model

Similar to the vision and LiDAR models, we apply sequence-level prediction aggregation to evaluate its effect on the fusion model. The classifier used is the fusion MLP described in Chapter 5, which combines per-frame logits from the Vision and LiDAR models.

The aggregation is performed in the same way as for the other models, by averaging logits across each one-minute sequence to produce a single prediction per category. The resulting sequence-level predictions are visualized as confusion matrices in Figure 6.5, and the corresponding accuracy, precision, recall, and F1-score per category are reported in Table 6.3.

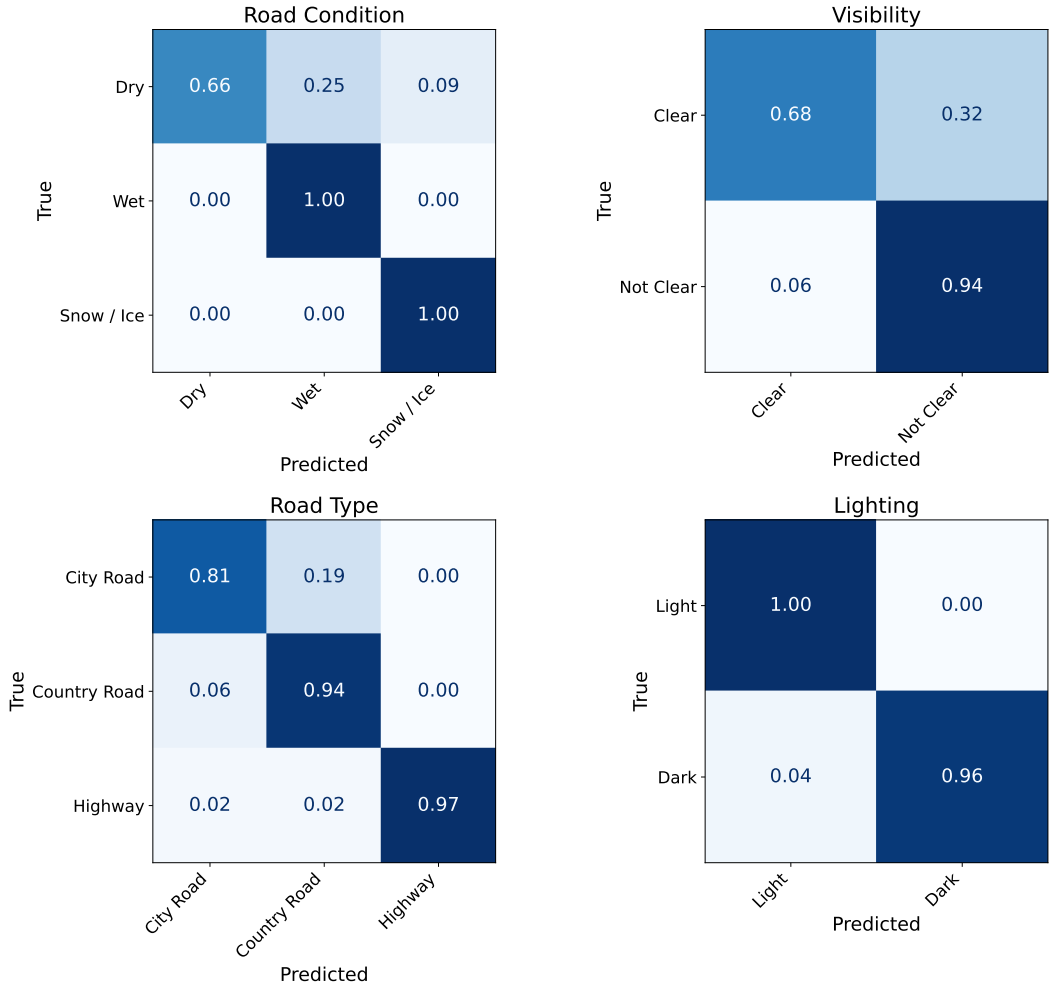


Figure 6.5: Confusion matrices for the fusion model after sequence-level logit averaging.

Table 6.3: Sequence-level performance of the fusion model after logit averaging.

Category	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-score (%)**
Road Condition	90.91	92.22	90.91	90.34
Road Type	89.90	90.63	89.90	89.95
Lighting	98.99	99.00	98.99	98.98
Visibility	78.28	82.97	78.28	78.41
Average	89.27	91.20	89.27	89.42

Figure 6.6 shows the absolute change in F1-score for each annotation category when comparing sequence-level predictions to their per-datapoint counterparts. Road condition improves by approximately 4 percentage points, road type by about 3.5 points, visibility by roughly 4 points, and lighting by around 1.5 points. On aver-

age, sequence-level aggregation increases the F1-score by approximately 3.5 points, indicating that fusion benefits similarly from smoothing of per-frame noise.

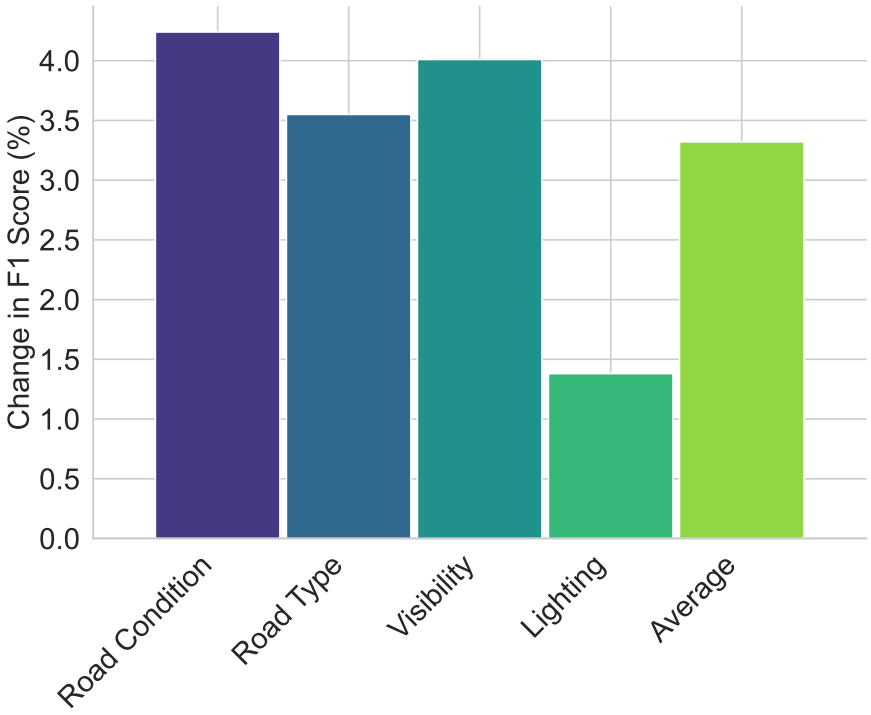


Figure 6.6: Absolute change in F1-score (%) per category for the fusion model after logit averaging.

6.5 Comparative Model Performance

To conclude our evaluation, we compare the final sequence-level F1-scores of the best-performing models from each modality: the Full-Frame CNN for vision, the full-scene PointNet for LiDAR, and the Fusion MLP. All models are evaluated after applying the sequence-level prediction aggregation described in Section 6.1.

Table 6.4 presents the resulting per-category F1-scores side by side. This comparison summarizes the results of the report and highlights the performance gains from both aggregation and multi-sensor fusion.

Table 6.4: Sequence-level F1-scores for the best Vision, LiDAR, and Fusion models after logit averaging.

Model	**Road Condition (%)**	**Road Type (%)**	**Lighting (%)**	**Visibility (%)**	**Average (%)**
Vision	87.35	88.99	**98.99**	61.04	83.84
LiDAR	**93.43**	83.69	59.26	**78.91**	78.82
Fusion	90.34	**89.95**	98.98	78.41	**89.42**

The fusion model achieves the highest average F1-score across all categories, outperforming both the vision and LiDAR models at the sequence level. By leveraging the complementary strengths of camera and LiDAR data—combining visual context with spatial geometry—it provides a more balanced and reliable basis for classification. Its performance is further improved through sequence-level aggregation, which helps reduce frame-level noise and increases prediction stability.

Given its strong overall performance and ability to combine information from both sensor modalities, the fusion model is well suited to serve as the final output of our automatic annotation pipeline and is integrated accordingly. To further improve robustness, sequence-level aggregation is applied on top of the fusion output.

7

Proof-of-Concept Data Retrieval Interface

This chapter introduces a proof-of-concept user interface designed to facilitate the retrieval of multimodal vehicle data using annotations from the automatic annotation pipeline. By combining query filters with visualizations of camera frames and LiDAR point clouds, the interface enables users to quickly locate, inspect, and validate driving scenarios.

7.1 Interactive Query Construction and Filtering

To facilitate exploration of large-scale driving data, a user interface was developed that leverages annotations from our pipeline. For every one-minute sequence, the pipeline assigns a label for each of the four categories: road condition, road type, lighting, and visibility. Each label is accompanied by a confidence score. These confidence scores are computed by applying the softmax function across all classes within a category. This produces scores between 0 and 1 for each possible label, where higher values indicate greater certainty in the prediction.

The interface provides controls to select one or more of these predicted annotations in each category and to specify minimum confidence thresholds. Each annotation–threshold pair defines a filter for that category, and the interface returns only the sequences that satisfy all selected filters. For instance, a user may retrieve sequences annotated as wet at or above 0.8 confidence while simultaneously requiring highway at or above 0.7 confidence. Only sequences that meet every specified criterion are included in the results.

Results are displayed in a table listing all sequences that match the selected criteria. Confidence scores for model predictions are shown alongside each label to support informed selection of sequences for retrieval. To facilitate downstream analysis, the table can be exported as a CSV file.

7.2 Multimodal Record Inspection

In addition to tabular summaries, the user interface provides a record-level inspection view. Upon selecting an entry, the corresponding camera frame and LiDAR point cloud are displayed side by side. The camera view shows the image frame,

while the LiDAR view renders the three-dimensional point cloud with interactive rotation and zoom controls. By situating each prediction within its original multi-modal context, the interface enables users to visually inspect the raw sensor data alongside the model's predicted annotations.

By combining flexible filtering controls with synchronized image and point cloud views, this proof-of-concept interface demonstrates how users can locate and validate driving scenarios using predicted annotations from our annotation pipeline. A screenshot of the interface can be seen in Appendix B.

8

Discussion

This chapter discusses the dataset composition, annotation quality, and model performance across the four scenario categories. It includes an analysis of class balance, annotation errors, and the potential for reusing the dataset in other research. It also discusses the results from the Vision, LiDAR, and Fusion models, with a focus on how their performance is affected by sensor characteristics and sequence-level aggregation. The chapter ends with suggestions for future work, including extensions to the dataset, annotation categories, and integration into Volvo's data pipeline.

8.1 Dataset Discussion

This section discusses class balance, edge cases, and broader reuse potential of our dataset, which is introduced in Chapter 2. We examine how the four annotation categories are distributed across our one-minute segments, highlight examples of annotation errors revealed by model predictions, and outline how the synchronized camera–LiDAR data can support future research.

8.1.1 Class Balance Across Categories

When assembling our dataset from Volvo Cars test-drive database, driving sessions were selected to maximize diversity and balance across our four annotation categories. Figure 2.1 in Chapter 2 shows that Road condition and Road type each have near-equal representation among the 1878 one-minute segments, reflecting the fact that each class for these categories had good representation in the database.

In contrast, the proportion of classes for the categories Visibility and Lighting is more skewed. Only 31.4% of segments are labeled "Not Clear" as a consequence of the fact that very few non-clear recordings were available in the database. This imbalance may contribute to the lower F1-score for visibility. Regarding the lighting category, "Light" recordings comprise 73.1% of the dataset due to the fact that we limited the number of "Dark" scenes, as producing accurate ground-truth annotations at night proved more challenging. Despite this imbalance, the clear contrast between light and dark scenes likely explains the high accuracy achieved by the vision model for this category.

However, for the Visibility category, the class imbalances may have affected model performance negatively, as the models struggle with this category and the differences

between the classes are not as clear to see. The models would likely have benefited from more evenly distributed classes, meaning more data in the class "Not Clear". Another way to address this could have been through loss weighting, which could have been applied to underrepresented classes.

8.1.2 Annotation Errors

While the evaluation metrics provide insights into the model's performance, they inherently rely on the accuracy of the ground-truth labels. As such, it is important to consider the potential for labeling inconsistencies. Despite the careful manual annotations, it can be assumed that a small number of segments might still be mislabeled. For instance, in the validation set, the fusion model sometimes predicts "Wet" when the ground-truth label is "Dry." We observed that one such case occurred at the very onset of rainfall: the road surface was transitioning from dry to wet, and initial droplets on the windshield signaled wet conditions. The ground-truth annotation, however, remained "Dry," making this an instance where the fusion model's prediction was more accurate than the assigned label. This suggests that there may be a small number of cases where the model is correct despite being different from the ground-truth label.

8.1.3 Broader Reuse of the Constructed Dataset

In addition to supporting the development of our classification models, the constructed dataset has potential value for a broader range of research applications. Beyond its immediate role in training the vision and LiDAR models, our synchronized camera–LiDAR dataset could likely offer value across multiple research directions. Because each image–point-cloud pair is precisely time-aligned and manually annotated, the dataset provides a strong benchmark for evaluating other multimodal architectures, such as LidarCLIP [13], that fuse visual and spatial information. Moreover, the raw, synchronized data can facilitate unsupervised and self-supervised approaches: models can exploit the natural correspondence between images and point clouds to learn joint representations without requiring manual labels.

8.2 Discussion of Classification Results

This section discusses the performance of the Vision, LiDAR, and Fusion models across the four annotation categories. We explore factors such as sensor characteristics, data distribution, and temporal aggregation that likely explain each model's strengths and weaknesses.

8.2.1 Model Comparison and Category-Level Analysis

The Vision model performs best on the lighting and road type categories, as seen by its high F1-scores in Table 6.1. This is likely because the images contain useful information, such as overall brightness for lighting and road markings, or road width for road type. Its scores on visibility and surface condition are lower. This may be

because rain, snow, or subtle pavement details are harder to identify in camera data, especially since the images are downsampled before being fed into the Vision model, causing some fine-grain detail to be lost, see Section 3.3.

The LiDAR model instead performs best on road condition and visibility, as shown in Table 6.2. This might be because point clouds are able to capture changes in surface geometry and return reflectance values that indicate wet or snowy roads, and they are not blurred or obscured as images can be in certain conditions. The LiDAR model performs poorly on lighting, likely because LiDAR does not measure ambient illumination and its point clouds are therefore unaffected by environmental light.

As concluded in Chapter 6, combining the LiDAR and Vision models using a simple MLP fusion yields better predictions than the individual models. It can be assumed that the Fusion model is able to draw on the Vision model's strengths in lighting and road type and the LiDAR model's strengths in road condition and visibility. This results in more balanced performance across all four categories, as seen in Table 6.4.

8.2.2 Analysis of Sequence Aggregation

It was found that aggregating frame-level prediction logits over each one-minute sequence consistently improved performance for the models, as shown in Figures 6.2, 6.4, 6.6. This aggregation serves as a smoothing function and can be assumed to reduce sensor noise, such as sun glare, sparse LiDAR returns or momentary occlusions and produces more stable confidence estimates.

The two-step process of first fusing camera and LiDAR logits, then averaging those fused scores over each one-minute sequence, yields the strongest results. The fusion MLP integrates complementary information, visual cues for lighting, and geometry cues for surface state into a single per-frame prediction. Sequence-level aggregation then smooths out any remaining frame-to-frame noise, producing the most accurate and stable scenario labels across all four categories.

8.3 Future Work

This thesis has demonstrated that multimodal automatic annotation using camera and LiDAR data is effective. However, there are still several ways to improve and expand this pipeline.

8.3.1 Additional Modalities and Annotation Categories

The presented annotation pipeline serves as a proof of concept. To become a more functional and fully customizable tool for engineering applications, it would need to support a wider range of scenario categories, as well as additional sensor inputs. Expanding both the sensor suite and the annotation schema would cover more use cases and potentially further improve accuracy under varied driving conditions.

8.3.2 Dataset Construction for Expanded Annotations

To support additional scenario categories such as traffic density, road curvature, or weather conditions, the dataset would need to be expanded to include these new labels. This could be done by applying the existing preprocessing and annotation workflow to new data containing the relevant conditions. Such data may be sourced from the current database or collected through new test drives. Once selected, the data would need to be annotated according to the extended label set to enable training and evaluation of an updated model.

8.3.3 Integration into Volvo Car's Production Environment

Moving from a proof of concept to a production environment would require investigating how to embed the inference, fusion, and smoothing steps into Volvo's data flow. It would need to be determined where and when in the process of injecting new data, the annotation pipeline would come into play. The process for storing generated labels alongside sensor data in the central database also needs to be defined. Finally, appropriate monitoring tools and retraining workflows must be developed so the models can be updated as new sensors are added or annotation requirements change.

9
Conclusion

It has been demonstrated that integrating camera and LiDAR sensors into a unified annotation pipeline enables automatic generation of accurate driving scene labels. We leveraged a pre-trained VGG19 CNN for image feature extraction, combined with customized trained classification heads for scene labeling. A lightweight PointNet was used for point-cloud interpretation, a compact multilayer perceptron for modality fusion, and sequence-level aggregation was applied to smooth noise and improve stability. With this approach, we consistently achieved F1 scores above 90 % for road-condition, above 89 % for road type, over 98 % for lighting classification, and above 78 % for visibility estimation.

These results confirm that metadata can be automatically and efficiently retrieved directly from onboard vehicle sensors without any manual intervention. This annotation pipeline can be integrated into the data ingestion stage following the collection or generation of new raw data, automatically inserting labels into the database. This enables engineers to query annotated vehicle data for specific conditions and significantly accelerates the development and verification of sensor-based vehicle functions.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org

[3] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks*. Springer, 2010, pp. 92–101.

[4] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 807–814.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015, originally released as arXiv:1409.1556.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440.

[8] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," *arXiv preprint arXiv:1909.02103*, 2019.

[9] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," *CoRR*, vol. abs/2103.00020, 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[10] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[11] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[12] C. R. N. Tassi, J. Gawlikowski, A. U. Fitri, and R. Triebel, "The impact of averaging logits over probabilities on ensembles of neural networks," in *Proceedings of the IJCAI-ECAI-22 Workshop on Artificial Intelligence Safety (AISafety 2022)*, ser. CEUR Workshop Proceedings, vol. 3215, 2022, cEUR-WS.org. [Online]. Available: https://ceur-ws.org/Vol-3215/19.pdf

[13] G. Hess, A. Tonderski, C. Petersson, K. Åström, and L. Svensson, "Lidarclip or: How i learned to talk to point clouds," 2023. [Online]. Available: https://arxiv.org/abs/2212.06858

A

Appendix: Machine Learning Models

In this appendix, additional details of the machine learning models are provided. Here you'll find more detailed evaluation tables.

Table A.1: Architecture and training settings for the Masked and Full-Frame CNNs.

Parameter	**Masked CNN**	**Full-Frame CNN**
Backbone	VGG19	VGG19
Dropout	0.30	0.30
Input size	224×224	224×224
Optimizer	Adam	Adam
LR (Heads)	1×10^{-4}	1×10^{-4}
LR (Fine-tuning)	1×10^{-6}	1×10^{-6}
Weight decay (L2 Reg)	—	1×10^{-4}
Epochs (heads)	3	3
Epochs (fine-tuning)	3	3
Batch size	128	128
Loss function	Cross-entropy	Cross-entropy

Table A.2: CLIP text prompts used for zero-shot classification. Categories: RC = Road Condition, VIS = Visibility, RT = Road Type, L = Lighting.

Category	**Class**	**Text Prompt**
RC	Wet	A photo of a road surface that is wet or has puddles.
RC	Dry	A photo of a road surface that is completely dry.
RC	Snow/Ice	A photo of a road surface covered in snow or ice.
VIS	Clear	A photo taken in clear weather with unobstructed view.
VIS	Rain	A photo showing active rainfall and wet surroundings.
VIS	Fog	A photo depicting foggy conditions with reduced visibility.
VIS	Snowfall	A photo capturing snowfall obscuring visibility.
RT	City	A photo of an urban street with buildings and traffic signals.
RT	Country	A photo of a rural country road bordered by vegetation.
RT	Highway	A photo of a multi-lane highway with guard rails.
L	Daylight	A photo captured in bright daylight with clear details.
L	Low Light	A photo captured under low-light or nighttime conditions.

Table A.3: Full-Frame CNN performance: initial vs. fine-tuning.

	Initial				**Fine-Tuned**			
Category	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Road Condition	0.8850	0.8948	0.8850	0.8807	0.8796	0.8939	0.8796	0.8726
Visibility	0.6345	0.6221	0.6345	0.6104	0.6181	0.6039	0.6181	0.6010
Road Type	0.8540	0.8639	0.8540	0.8523	0.8651	0.8731	0.8651	0.8644
Lighting	0.9910	0.9911	0.9910	0.9910	0.9846	0.9849	0.9846	0.9845

Table A.4: Masked CNN performance: inital vs. fine-tuning.

	Initial				**Fine-Tuned**			
Category	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
Road Condition	0.7530	0.7547	0.7530	0.7513	0.7906	0.7887	0.7906	0.7864
Road Type	0.8073	0.8087	0.8073	0.8056	0.8133	0.8144	0.8133	0.8117

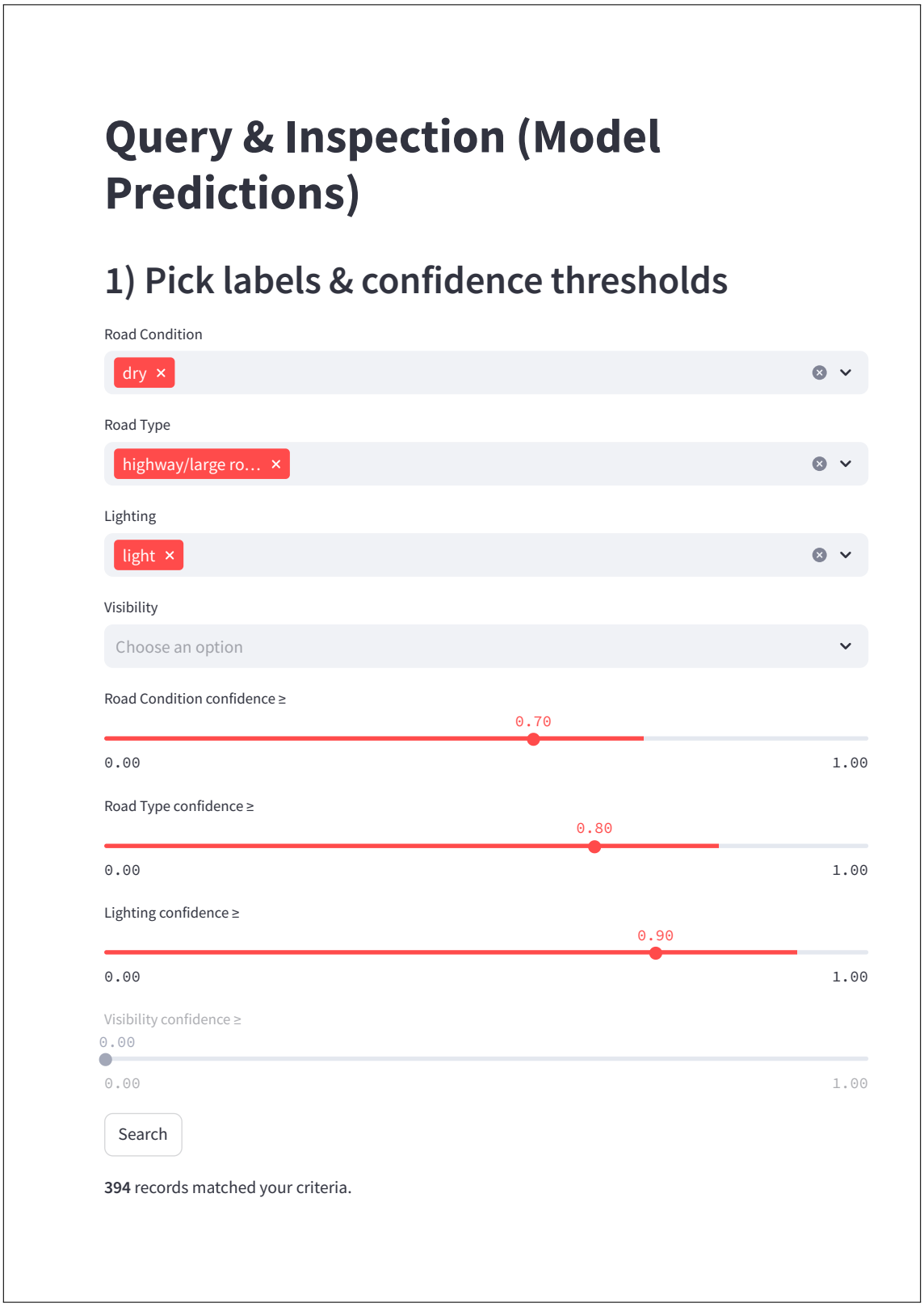
Table A.5: Zero-shot CLIP performance on each category.

Category	Accuracy	Precision	Recall	F1
Road Condition	0.8198	0.8147	0.8198	0.8140
Visibility	0.4815	0.6848	0.4815	0.4020
Road Type	0.6362	0.7461	0.6362	0.6347
Lighting	0.8686	0.9109	0.8686	0.8747

B

Appendix: User Interface

This appendix showcases the user interface built for visualizing input data streams, inspecting model prediction, and adjusting inference parameters.


Query & Inspection (Model Predictions)
1) Pick labels & confidence thresholds
Road Condition
dry ×
Road Type
highway/large ro… ×
Lighting
light ×
Visibility
Choose an option
Road Condition confidence ≥
0.70
0.00
1.00
Road Type confidence ≥
0.80
0.00
1.00
Lighting confidence ≥
0.90
0.00
1.00
Visibility confidence ≥
0.00
0.00
1.00
Search
394 records matched your criteria.

Index	↑ row_id	suit_id	vehicle_name	utc_start	utc_duration	img_parent_folder
17	66	2,767,493	MULE13	2022-10-04 04:40:59.000	06:12:55	mule13_ohs1307_202210_0
18	67	2,767,493	MULE13	2022-10-04 04:40:59.000	06:12:55	mule13_ohs1307_202210_0
19	68	2,767,493	MULE13	2022-10-04 04:40:59.000	06:12:55	mule13_ohs1307_202210_0
20	69	2,767,493	MULE13	2022-10-04 04:40:59.000	06:12:55	mule13_ohs1307_202210_0
21	70	2,767,493	MULE13	2022-10-04 04:40:59.000	06:12:55	mule13_ohs1307_202210_0
22	72	2,800,286	MULE19	2022-10-03 05:32:51.000	03:18:35	mule19_ohs1311_202210_0
23	73	2,800,286	MULE19	2022-10-03 05:32:51.000	03:18:35	mule19_ohs1311_202210_0
24	74	2,800,286	MULE19	2022-10-03 05:32:51.000	03:18:35	mule19_ohs1311_202210_0
25	75	2,800,286	MULE19	2022-10-03 05:32:51.000	03:18:35	mule19_ohs1311_202210_0
26	76	2,800,286	MULE19	2022-10-03 05:32:51.000	03:18:35	mule19_ohs1311_202210_0
27	77	2,800,286	MULE19	2022-10-03 05:32:51.000	03:18:35	mule19_ohs1311_202210_0

Download filtered CSV

Pick an index to inspect

Index 25

Model's Predicted Labels

Road Condition: dry (1.00)

Road Type: highway/large road (1.00)

Lighting: light (1.00)

Visibility: clear (0.55)

Visualize Image & 3D LiDAR Frame

Frame index

1

1 62

Figure B.1: Screenshot of the user interface.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY