remote sensing

MDPI

Article

Precise Adverse Weather Characterization by Deep-Learning-Based Noise Processing in Automotive LiDAR Sensors

Marcel Kettelgerdes [1,2,*], Nicolas Sarmiento [2], Hüseyin Erdogan [3], Bernhard Wunderle [4] and Gordon Elger [1,2]

[1] Institute of Innovative Mobility (IIMo), University of Applied Sciences Ingolstadt, Esplanade 10, 85049 Ingolstadt, Germany
[2] Institute for Transportation and Infrastructure Systems (IVI), Fraunhofer Society, Stauffenbergstraße 2, 85051 Ingolstadt, Germany
[3] Autonomous Mobility Division, Conti Temic Microelectronic GmbH, Ringlerstraße 17, 85057 Ingolstadt, Germany
[4] Faculty of Electrical Engineering and Information Technology, Chemnitz University of Technology, Reichenhainer Str. 70, 09126 Chemnitz, Germany
* Correspondence: marcel.kettelgerdes@thi.de

Abstract: With current advances in automated driving, optical sensors like cameras and LiDARs are playing an increasingly important role in modern driver assistance systems. However, these sensors face challenges from adverse weather effects like fog and precipitation, which significantly degrade the sensor performance due to scattering effects in its optical path. Consequently, major efforts are being made to understand, model, and mitigate these effects. In this work, the reverse research question is investigated, demonstrating that these measurement effects can be exploited to predict occurring weather conditions by using state-of-the-art deep learning mechanisms. In order to do so, a variety of models have been developed and trained on a recorded multiseason dataset and benchmarked with respect to performance, model size, and required computational resources, showing that especially modern vision transformers achieve remarkable results in distinguishing up to 15 precipitation classes with an accuracy of 84.41% and predicting the corresponding precipitation rate with a mean absolute error of less than 0.47 mm/h, solely based on measurement noise. Therefore, this research may contribute to a cost-effective solution for characterizing precipitation with a commercial Flash LiDAR sensor, which can be implemented as a lightweight vehicle software feature to issue advanced driver warnings, adapt driving dynamics, or serve as a data quality measure for adaptive data preprocessing and fusion.

Keywords: ADAS; adverse weather; weather classification; artificial intelligence; deep learning; Vision Transformer; LSTM; automotive; LiDAR; precipitation measurement

Citation: Kettelgerdes, M.; Sarmiento, N.; Erdogan, H.; Wunderle, B.; Elger, G. Precise Adverse Weather Characterization by Deep-Learning-Based Noise Processing in Automotive LiDAR Sensors. *Remote Sens.* **2024**, *16*, 2407. https://doi.org/10.3390/rs16132407

Academic Editors: Wei Li, Haiyong Gan, Heng-Chao Li and Wenshuai Hu

Received: 24 May 2024
Revised: 24 June 2024
Accepted: 28 June 2024
Published: 30 June 2024

1. Introduction

In the rapidly growing field of automated mobility, optical sensors, particularly light detection and ranging (LiDAR), play a crucial role in advanced driving assistance systems (ADASs). Especially, solid-state LiDARs have emerged as a cost-effective and robust alternative to traditional mechanically scanning LiDARs, making them suitable for mass market adoption, not only with respect to the ADAS market, but also for smart city applications in the form of intelligent infrastructure units. However, LiDAR sensors are highly sensitive to precipitation and fog due to scattering in the near-infrared wavelength. Opposed to sparsely investigated longterm degradation over the sensor product lifecycle [1–4], previous works have extensively studied and modeled the instantaneous impact of environmental effects using lab simulations [5–12], field measurements [13,14], or both [15,16]. While lab simulations still fail to replicate natural precipitation types like snow and hail, field measurements provide more realistic data at the cost of longer measurement periods to accumulate a

balanced dataset with respect to weather condition coverage [9,16]. The primary focus of previous works has been to understand and evaluate LiDAR sensor performance and model data degradation caused by environmental effects for virtual validation of ADAS functionalities. In this contribution, a different approach is taken by investigating whether these degradation effects can be used to predict the current weather conditions.

Such an inverse model could serve as a cost-effective solution for precise precipitation measurements and as a lightweight software feature in vehicles and infrastructure units, enabling advanced driver warnings and adaptive driving dynamics. Additionally, it could be utilized as a data quality indicator for weighted heterogeneous sensor data fusion or adaptive filtering for improved object detection [17].

Hence, in a previous work by the authors [18], a physics-informed deep learning (PIDL) method was developed to predict the comprehensive precipitation spectrum, specifically the particle size and velocity distribution (PSVD) from precipitation-induced degradation effects. The authors achieved a mean absolute error (MAE) as low as 2.4 particles per size and velocity bin, considering 440 bins. Based on that, the precipitation type could be classified, and characteristic quantities like the precipitation rate, attenuation coefficient, or meteorological visibility distance could be derived using known physical relations. The model was trained using data from a one-year in-field measurement campaign conducted with a low-cost automotive Flash LiDAR and correlated, extensive weather measurements [13]. However, the model was still lacking generalizability, since the PSVD prediction was relying on information from the background scenery, which makes the deployment in the vehicle or infrastructure challenging.

Consequentially, within this work, a novel pipeline is introduced which extracts the measurement noise from the first LiDAR return pulse in a preprocessing step in order to predict the occurring weather condition solely based on measurement noise (see Figure 1). The current weather state is, for the sake of generalizability, directly predicted from the extracted noise in an end-to-end approach, consisting of

1. 15 standardized precipitation classes, corresponding to the World Meteorological Organization (WMO) Surface Synoptic Observations (SYNOP) standard [19].
2. The respective precipitation rate (in mm/h) as quantitative, interpretable metric on the respective precipitation intensity, also correlating with the meteorologic visibility as demonstrated earlier [13].

For the corresponding classification and regression task, multiple deep neural network models are developed based on varying state-of-the-art architectures as well as model sizes, and benchmarked with respect to accuracy and computational efficiency. Due to the fact that the prediction is conducted solely based on measurement noise in the form of false-positive detections in direct proximity to the sensor aperture, the models can directly be employed in an infrastructure and potentially also vehicle application without inducing concept drift.

With this in mind, this work contributes to earlier research (see Section 2) as follows:

1. Fine-granular weather classification by 15 standardized precipitation classes, corresponding to the WMO Surface SYNOP standard;
2. Intensity quantification by additional precipitation rate prediction;
3. Generalizability with respect to the background scene by noise extraction;
4. Implementation based on a series-deployed automotive solid-state LiDAR;
5. Benchmark of state-of-the-art deep learning models for image time series processing.

Since, the underlying dataset was recorded in a static infrastructure measurement unit, however, the effect of varying driving speed and corresponding headwind on the noise characteristics will need to be investigated in future works. Also, secondary effects in dynamic driving scenarios, like spray water, might have a major impact on the prediction performance.

The work is structured as follows. Following the introduction, the second section elaborates on related works regarding LiDAR- and machine learning (ML)-based weather

characterization for automotive or infrastructure applications. The third section describes the weather-induced measurement effects in LiDAR sensors which are exploited to predict the occurring weather conditions. In the next section, the in-field measurement setup for automated data acquisition and the resulting dataset are introduced. The fifth section lists and explains the architecture as well as the corresponding training and hyperparameter optimization methodology of the evaluated deep learning (DL) models. In the subsequent sixth section, the corresponding results are presented and the model performances are compared quantitatively with respect to classification accuracy and regression error, as well as model size and required computational resources. The last section concludes the work by summarizing the methodology and results.

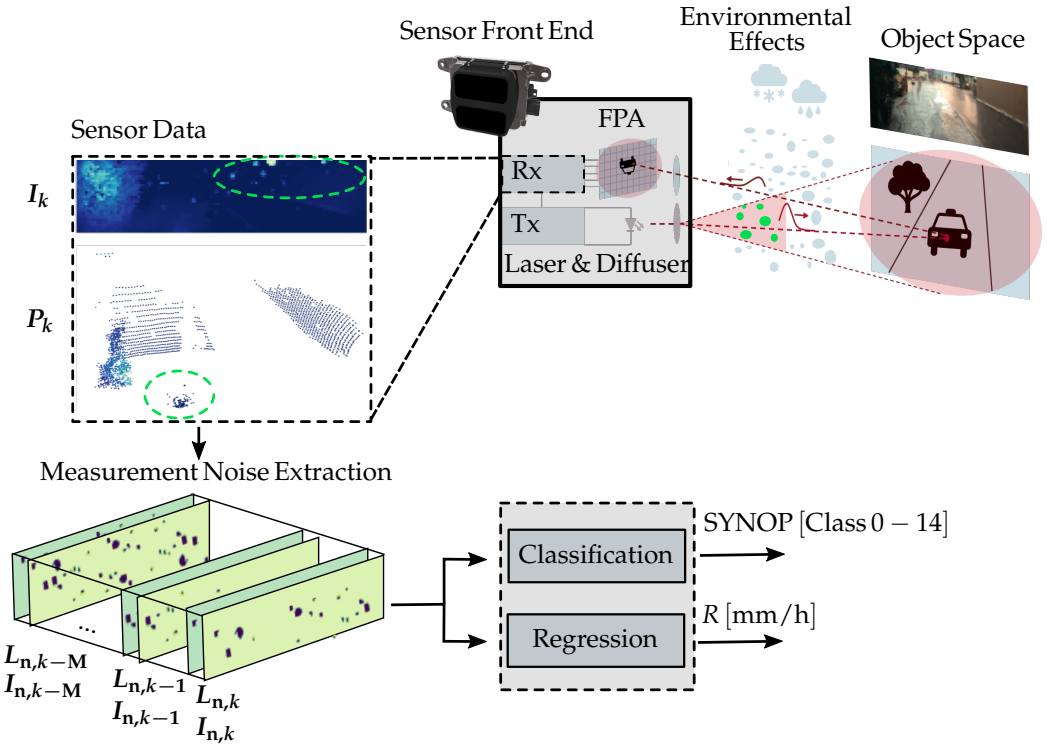


Figure 1. Illustration of precipitation effects within the optical channel of the Flash LiDAR sensor. Especially, large and sensor-close particles are detected as false-positive detections on the focal plane array (FPA) and can be clearly seen as artifacts in the corresponding backscattered intensity image $\boldsymbol{I_k}$ and the depth image $\boldsymbol{L_k}$, as well as in the corresponding point cloud $\boldsymbol{P_k}$ at time k. The reference scene is shown as an RGB image. This work proposes a pipeline that extracts the induced noise from $\boldsymbol{I_k}$ and $\boldsymbol{L_k}$ in order to be processed as image time series by a deep learning pipeline for fine-granular precipitation classification (SYNOP) and rate regression (R). For both tasks, several state-of-the-art approaches are implemented, trained, and benchmarked.

2. Related Works

Due to the high relevance for the automotive and smart infrastructure industry, various works deal with the ML-driven classification of weather conditions based on the vehicle's perception data. Especially in early works, research focused on camera-based approaches [20–23], due to the high information density of image data and the intuitive interpretability from a human perspective. As elaborated, however, especially active LiDAR sensors tend to be very sensitive regarding varying weather conditions, so more and more recent works are focusing on ML and DL methods to characterize occurring weather conditions based on LiDAR detection data (see Table 1). In general, those works can be categorized by the classified weather types, the utilized methods, the number of considered prediction classes, and the achieved performance.

Table 1. Related works in LiDAR-based weather classification, categorized by utilized methods, number of considered prediction classes (# Classes), and achieved performance.

Authors	Methods	# Classes	Performance (Metric)
Precipitation and Fog			
Heinzler et al. (2019) [15]	KNN, SVM	3	95.86–100.0% (Prec)
Rivero et al. (2020) [24]	KNN	4	98.86–99.09% (F1)
Sebastian et al. (2021) [25]	DL(Conv)	3	46.51–100% (Acc)
		5	2.44–92.15% (Acc)
Wu et al. (2022) [26]	DL (sparse Conv)	4	94.83–99.09% (Prec)
Da Silva et al. (2023) [27]	DL (Conv)	3	91.39–100% (F1)
Pereira et al. (2024) [28]	DL (Transf)	6	91.88–99.38% (Acc)
PSVD			
Kettelgerdes et al. (2023) [18]	PIDL	–	2.4 particles (MAE)

Following this, early research by Heinzler et al. [15] addressed this question by employing classic machine learning (ML) methods in the form of k-nearest neighbor (KNN) and support vector machine (SVM) classifiers. In order to train the algorithms, the authors utilized two separate datasets from varying driving scenarios and the CEREMA test chamber in France [29], simulating fog and rain conditions. As input features, they exploited characteristics of the LiDAR point cloud, such as the Cartesian point position and its return pulse width and index, as well as its intensity. Similar to this work, the point cloud information is spatially filtered in a preprocessing step with a maximum considered distance of 20 m. By that, the authors achieved a precision between 95.86% and 100.0% for the tested Velodyne LiDAR and the SVM classifier on the static climate chamber dataset. Even though they distinguished overall four classes of fog with respect to the corresponding visibility in their investigations, only one fog class could be considered in the classifier due to feature invariance, resulting in three overall classes (clear/sun, fog, rain).

A subsequent work by Rivero et al. [24] implemented the KNN classifier to categorize weather conditions based on the point cloud information of a static parking lot scene, using similar features. They achieved high accuracy in detecting rain and snow under daylight conditions, with a F1-scores of up to 99.09% and 98.86%, respectively.

Wu et al. [26] utilized and compared several state-of-the-art DL approaches to an own, voxelization-, and sparse-convolution-based architecture, to classify the weather conditions on data from a static infrastructure setup, reaching a precision between 99.09% for fog and 94.83% for sun.

However, the results of the latter works appear to be highly dependent on the scenery and measurement setup. Additionally, they differentiate rather coarsely between three to four general weather classes (clear/sun, fog, rain, snow), while quantitative metrics on the intensity of the respective effects (e.g., visibility distance or precipitation rate) are neglected.

In a similar manner to [26], Da Silva et al. [27] projected the LiDAR point cloud into the birds eye view perspective in a preprocessing step in order to train and evaluate several convolution-based DL architectures based on a driving dataset, which distinguishes three weather classes (clear, fog, and rain). The authors achieved a high F1-score between 91.39% and 100% on the test dataset and, opposing previous works, proved generalizability between different driving scenarios.

Sebastian et al. [25] utilized the open accessible DENSE dataset of Bijelic et al. [5,30], which consisted of a driving dataset as well as a static dataset, again, from the CEREMA fog and rain chamber [29], in which several LiDAR sensors and a reference camera were tested under varying fog densities and rain intensities. Based on those datasets, they developed a DL architecture that projects the point cloud to an imaginary image plane in order to be further processed by a convolutional encoder structure, finally classifying the occurring weather conditions in a granularity of up to five classes (clear, snow, light fog, dense fog, rain) on the driving dataset and three classes (clear, fog, rain) in case of the static dataset.

By that, they achieved a prediction accuracy between 2.44% for dense fog and 92.15% for clear conditions on the driving dataset and between 46.51% for rain conditions and up to 100% for clear and fog conditions on the static dataset. Similar to Heinzler et al. [15], they explained the poor results for fog and rain prediction by the invariance of the utilized feature space.

In a recent work, Pereira et al. [28] incorporated fused LiDAR and camera data in order to classify the weather conditions by two general weather classes (rain and fog), but further included three classes of meteorological visibility to give a coarse indication on the respective rain or fog intensity, effectively resulting in six distinct weather classes. For that purpose, they trained and evaluated several modern transformer-based [31] architectures on the DENSE dataset, already mentioned above. For that, they partly used earlier proposed, convolution-based models [25,27] as backbone, finally achieving an accuracy of up to 99.38% for weather classification and 91.88% for visibility classification. However, since the models are trained on a static dataset from an indoor test chamber, it is yet to be investigated whether the models could be deployed in an infrastructure or driving application without inducing concept drift.

As Heinzler et al. [15] pointed out as possible follow-up research, this work aims for a finer class division by utilizing advanced DL-based classification approaches as well as data accumulation over time to exploit inherent time series information. Furthermore, predictions are conducted solely on extracted noise, since the models are intended to generalize with respect to the background scenery and application. Lastly, regression models are investigated in addition to the classifiers in order to enhance the predicted precipitation class with a corresponding precipitation rate as intensity metric.

3. Mathematical Modeling of Precipitation Effects

Since the authors thoroughly discussed the physical and empirical modeling of precipitation influences on Flash LiDAR sensors in a previous work [13], following [18], the key points are only briefly summarized to provide an understanding of the measurement effects that allow for inverse precipitation state predictions.

3.1. Link-Budget Equation

Accordingly, the optical power $P_{r,u,v}$ received on a pixel (u, v) in the Flash LiDAR's FPA can be calculated based on a number of elements: a system element, a geometric beam propagation element, and an attenuation element that is influenced by both the target and the atmospheric conditions in the optical channel. Given certain assumptions, these elements can be considered as multiplicative functions [13,16]:

$$P_{r,u,v} = \underbrace{C_\mathrm{A} P_{0,u,v} \tau_\mathrm{h}}_{system} \underbrace{\frac{1}{L_{u,v}^2}}_{geom} \underbrace{H_{\mathrm{T},u,v}(L_{u,v}, \beta) T^2(L_{u,v}, \alpha)}_{attenuation} \tag{1}$$

The system component (see also Figure 1, sensor front end) represents the design parameters of the optical front end, such as the optical output power $P_{0,u,v}$ in the instantaneous field of view of a pixel (u, v) (refer to Figure 2), the pulse width τ_h, and the so-called system constant C_A, which, among other things, includes the optical efficiency and aperture of the receiver optic.

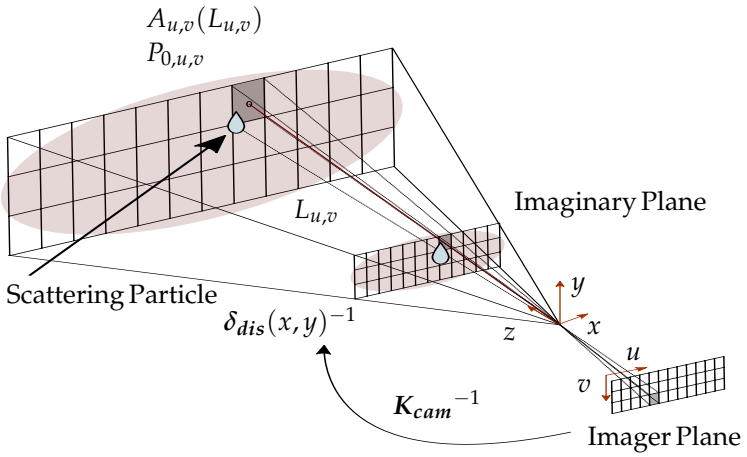


Figure 2. Projective Flash LiDAR model, adapted with permission from Ref. [13], 2024, IEEE. Assuming full transmitter beam and receiver frustum crossover, the illuminated section $A_{t,u,v}(L_{u,v})$ per detector element (u, v) can be assumed as the respective observed area $A_{u,v}(L_{u,v})$ and, hence, calculated as a function of the inverse intrinsic camera model $\{\boldsymbol{K_{cam}}^{-1}, \boldsymbol{\delta_{dis}}^{-1}\}$ and the target distance $L_{u,v}$. In general, the size ratio of an occurring particle and $A_{u,v}$ decreases significantly with $L_{u,v}$, which makes Flash LiDAR sensors rather sensitive to aperture-close false-positive detections.

While the geometrical term describes the measurement distance $L_{u,v}$-dependent quadratic power density decrease, the attenuation term divides into the target response function $H_{T,u,v}(L_{u,v}, \beta)$ (see also Figure 1, object space) and the response function of the optical channel $T^2(L_{u,v}, \alpha)$ (see also Figure 1, measurement noise).

The latter describes the two-way signal attenuation $T(L_{u,v}, \alpha)$ by particle scattering and absorption effects. Both are considered by the attenuation coefficient α, integrated over $L_{u,v}$:

$$T(L_{u,v}, \alpha) = e^{-\int_{l=0}^{L_{u,v}} \alpha(l)dl} \tag{2}$$

α can again be calculated based on the extinction efficiency Q_{ext}, representing the particle diameter D-dependent scattering mechanism [13,16,32]. Hence, Q_{ext} is integrated over the particle size distribution in a unit volume $N(D)$:

$$\alpha = \frac{\pi}{4} \int_{D=0}^{\infty} D^2 Q_{\text{ext}}(D) N(D) dD \tag{3}$$

$H_{T,u,v}(L_{u,v}, \beta)$, on the other hand, depends on the target nature which is described by the backscattering coefficient β. Depending on the reflection mechanism, this might simply be expressed by a reflection coefficient in terms of an ideally diffuse target surface, or a bidirectional reflectance distribution function (BRDF) in case a surface with a more complex reflection behavior is considered [16]. The overall power, backscattered by a particle distribution $N(D)$, can be calculated by integrating over the respective backscattering efficiency $Q_{\text{b}}(D)$ instead of Q_{ext}, analogous to α [13,16,32].

3.2. Stochastic Measurement Noise

However, with the underlying assumptions, this term does not cover the backscattering of large, sensor-close particles, which lead to stochastically distributed false-positive detections in the sensor frustum (see Figure 1, measurement effects). Due to their operating principle, Flash LiDAR systems are particularly sensitive to scattering particles that are close to the sensor aperture. Since Flash LIDAR systems rely on a diffuse laser beam with comparatively strong beam divergence (see Figure 2), the ratio between their cross-section and the size of an occurring scattering particle becomes larger in direct proximity to the laser aperture. Consequentially, the irradiance on that particle is significantly higher than that of a more distant particle. Accordingly, the proportion of backscattered power is high

enough to trigger a false-positive detection in one or more pixels of the FPA, whereas more distant particles are not exposed to sufficient irradiance to induce a false-positive detection over the backscattered optical power. This is a major difference to scanning LiDAR systems, which work with a collimated laser beam with comparatively very small divergence. Hence, scattering particles can potentially induce false-positive detections over the whole measurement range. Accordingly, this effect may be considered with an additional probabilistic modeling approach, as suggested by Kilic et al. [33], for example, by incorporating it as probabilistic target response $H^{X}_{T,u,v}(L^{X})$ in a random distance L^{X} [13].

Both of the described effects—The overall precipitation induced signal attenuation, and more significantly, the stochastic false-positive detections, due to large, sensor-close particles, can be observed as measurement effects in the dataset and were in magnitude even found to be correlated with the precipitation rate and type [13]. This finally motivates the exploitation of these undesired effects by employing data-driven models which are capable of reconstructing the present precipitation condition from it.

4. Measurement Setup and Dataset

In order to understand the underlying training data for the implemented model, the measurement setup and campaign, as well as the resulting dataset, are described in further detail.

4.1. Data Acquisition

The basic configuration [13] comprises two major components—a perception sensor unit (PSU) equipped with a reference RGB camera and the sensors under test (SUT), along with a weather sensor unit (WSU) that captures detailed weather information every 60 s (see Figure 3). Both units are remotely linked via a cloud-based platform which allows them to function independently from each other. Moreover, this platform also enables the data to be stored and displayed online.

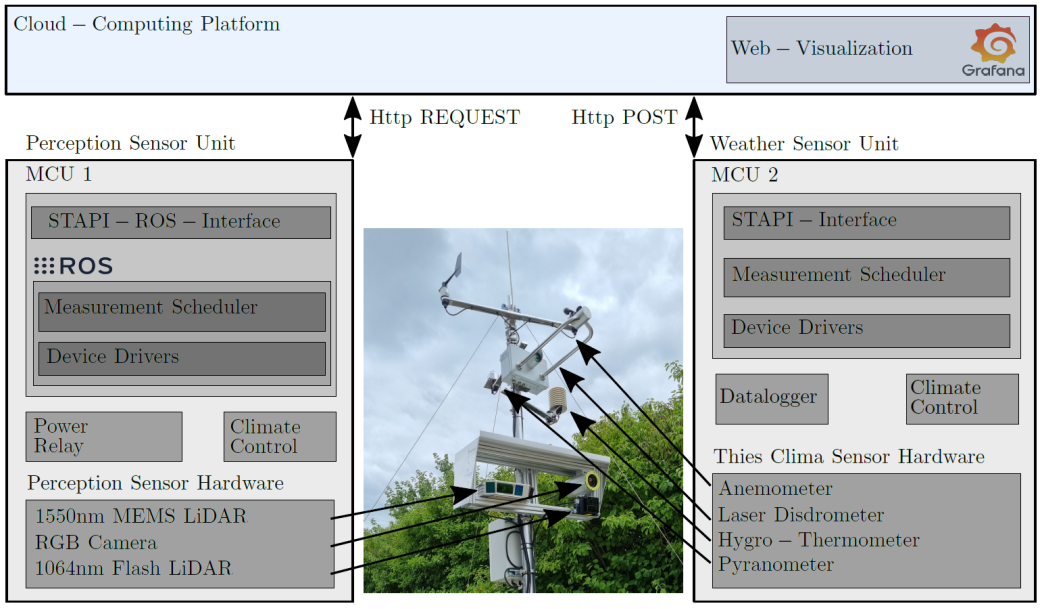


Figure 3. Outdoor measurement setup consisting of a perception sensor unit with the devices under test and a weather sensor unit for continuous acquisition of comprehensive weather data. Both systems are connected to a cloud-computing platform in order to exchange, store, and visualize the respective data online. Adapted with permission from Ref. [13], 2024, IEEE.

4.1.1. Weather Sensor Unit

Primarily, the WSU is equipped with measurement tools from the manufacturer, *Thies Clima*. These devices are used to quantify wind, temperature, air humidity, sunshine expo-

sure, and rain conditions. Out of these, the primary focus of this study is on precipitation, which is gauged via a laser disdrometer [34]. The disdrometer can accurately capture existing precipitation by registering particles based on their specific velocity v and diameter D. The data on these particles are preserved in a discrete 2D histogram, denoted as $n(\mathrm{I} \times \mathrm{J})$, which comes with velocity bins j and diameter bins i, each with a width Δv_j, ΔD_i and a nominal value v_j, D_i. The particle counting is performed over an integration time t_{int}, which is typically around 60 s, as in our case, and within a certain detector cross-section A_{det} (in our setup, 0.00456 m^2).

An exemplary PSVD for a precipitation rate of $R = 21\,\mathrm{mm/h}$ is shown in Figure 4. The patterns of these distributions may vary greatly based on location and type of rainfall [35,36]. In addition to identifying the type of precipitation from the PSVD, it is also possible to calculate characteristic parameters like the wavelength-dependent signal attenuation α, the backscattering coefficient β, and the rainfall rate R [32,37]. For events of solely liquid rain, R can be calculated as shown below:

$$R = \frac{6\pi}{10^4} \int_{D=0}^{\infty} D^3 v(D) N(D) dD \tag{4}$$

By incorporating the density of the corresponding particles, this relation can also be applied to solid precipitation like snow, expressing the liquid water equivalent precipitation rate [38]. Therefore, the measured n has to be normalized to the average number of particles per unit volume $N(D_i)$ by conducting the following transformation [34,39]:

$$N(D_i) = \frac{n_i^{\Sigma j}}{\bar{v}_i^{j} \cdot t_{\mathrm{int}} \cdot A_{\mathrm{det}} \cdot \Delta D_i} \tag{5}$$

with $n_i^{\Sigma j}$ being the sum of all particles in diameter bin i and $\bar{v}_i^{j}$ being the average particle velocity in diameter bin i.

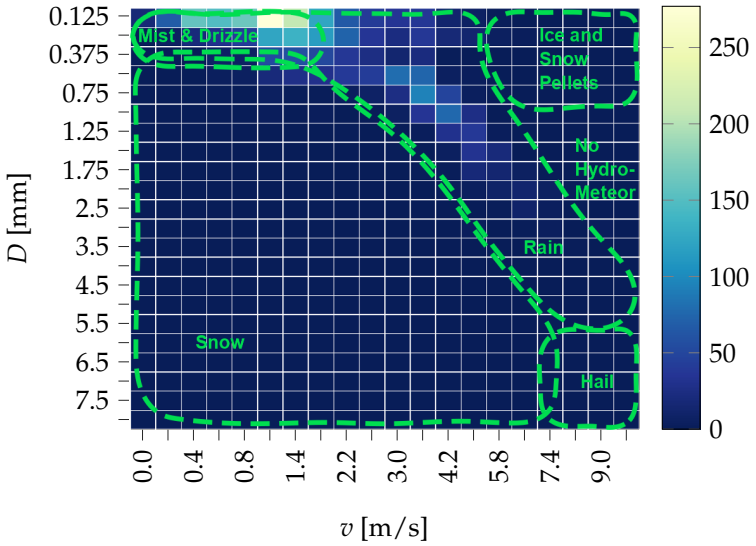


Figure 4. Exemplary PSVD for a measured precipitation rate of $R = 21\,\frac{\mathrm{mm}}{\mathrm{h}}$, as reprinted with permission from Ref. [18], 2024, IEEE, describing the number of particles n with a diameter D and a velocity v as a 2D histogram. In addition to the precipitation rate, calculated with (6), the precipitation class can be identified by the particle characteristics, as indicated.

By that, the corresponding key parameters, including the attenuation and backscattering coefficient, can be calculated by discrete integration over the particle size bins:

$$R \approx \widehat{R} = \frac{6\pi}{10^4} \cdot \frac{1}{t_{\text{int}} \cdot A_{\text{det}}} \cdot \sum_{i=1}^{I} D_i^3 n_i^{\Sigma j} \tag{6}$$

and

$$\alpha \approx \widehat{\alpha} = \frac{\pi}{4 \cdot t_{\text{int}} \cdot A_{\text{det}}} \sum_{i=1}^{I} D_i^2 Q_{\text{ext}}(D_i) \frac{n_i^{\Sigma j}}{v_i^{\bar{j}}} \tag{7}$$

with β accordingly by replacing $Q_{\text{ext}}(D_i)$ with $Q_{\text{b}}(D_i)$.

Opposed to the wavelength-dependent backscattering and extinction coefficient, the disdrometer is able to derive precipitation rate and sum directly with an error of less than 15% in case of rain. The WSU dataset relevant for this work, however, consists of 15 recorded weather classes with the corresponding precipitation rate, as further elaborated in Section 4.2.

4.1.2. Perception Sensor Unit

On the other hand, the PSU regularly fetches the current weather state from the server to examine for unseen severe weather situations, such as a particular precipitation rate. When this happens, the MCU initializes all SUTs to collect a data sample of approximately 15 s, which is then stored locally before being transferred to the cloud. This process aims to maintain a balanced dataset while ensuring that the total volume of data remains manageable. To automate and standardize the handling of perception sensor data, the MCU, along with all sensor and device drivers, utilizes the well-known open Robot Operating System (ROS).

Delving into the SUT hardware, in general, two LiDARs were installed: a micro-electromechanical system (MEMS)-based scanning LiDAR and an automotive Flash LiDAR from Continental, whereby this study lays emphasis on the latter (refer to Figures 1 and 3, center). The Flash LiDAR qualifies as a direct time-of-flight device, furnishing 16 bit depth ($\boldsymbol{L}$) and intensity ($\boldsymbol{I}$) images in addition to a corresponding point cloud $\boldsymbol{P}$ by ascertaining the pixel-wise pulse return time along with its intensity within a resolution of W $\times$ H $= 128 \times 32$ pixels and a frame rate of up to $f = 25$ Hz. For that, it operates on a transmitter wavelength of $\lambda = 1064$ nm and features a receiver field of view of FOV $= 120° \times 27.5°$, as well as a measurement range of $L_{\text{min}} = 0.5$ m to $L_{\text{max}} = 25$ m.

In general, the sensor is able to detect up to two return pulses per pixel, which enhances the sensor performance, especially with respect to scattering particles in case of adverse weather conditions or dust. However, since exactly those scattering effects are exploited, only the first return pulse with its corresponding depth and intensity image is taken into account for further processing, as discussed in the following.

4.2. Dataset and Preprocessing

4.2.1. Dataset

The recorded LiDAR data are stored in form of a serialized rosbag with an attached weather message, containing, among others, comprehensive precipitation data. The data were acquired in the field over a duration of one year, covering all seasons. This is illustrated by the precipitation class distribution (see Figure 5a), which shows quite a balanced data distribution for slight drizzle (1), rain (4–6), snow (9–10), or mixed precipitation, including small hail (13). All class IDs, including the respective SYNOP-standard ID and descriptions, are listed in Table 2.

Some classes, like slight drizzle and rain (3), moderate drizzle (2), and slight snow pellets or small hail shower (12), were highly underrepresented due to their less-likely occurrence, which again results from them being either a transition class between two major categories (2), or containing some ambiguity like class 3 or 12. Consequently, the

datasets from the corresponding classes are majorly augmented by oversampling, similarly to Pereira et al. [28]. The class-specific oversampling magnitude is illustrated in Figure 5a, in which the original data share of underrepresented classes is marked in black, whereas the final test/train split is visualized by the corresponding colors.

Table 2. Precipitation class IDs and descriptions in accordance with the SYNOP standard [19], using the same indexing with permission from Ref. [18], 2024, IEEE.

ID	SYNOP	Description
0	0	No Precipitation
1	51	Slight Drizzle
2	53	Moderate Drizzle
3	58	Slight Drizzle and Rain
4	61	Slight Rain
5	63	Moderate Rain
6	65	Heavy Rain
7	68	Slight Rain or Drizzle
8	69	Moderate or heavy Rain or Drizzle
9	71	Slight Snowfall
10	73	Moderate Snowfall
11	77	Snow Grains
12	87	Slight Snow Pellet or Small Hail Shower
13	88	Slight Snow Pellet or Small Hail Shower, mixed
14	90	Hail Shower, mixed

For model training, the data are split into 80% training-, 15% validation-, and 5% test-data. Since the number of data samples after augmentation is still not fully uniformly distributed among SYNOP-classes, the training–validation–test split is applied class-specific in order to ensure a fair representation of generally underrepresented classes in the validation and test evaluations (see Figure 5a).

Further, the dataset was found to cover a wide range of precipitation intensities, covering precipitation rates up to more than $R = 50\,\mathrm{mm/h}$ [13]. In order to illustrate this, Figure 5b shows the precipitation rate spread of the corresponding classes for all contained samples.

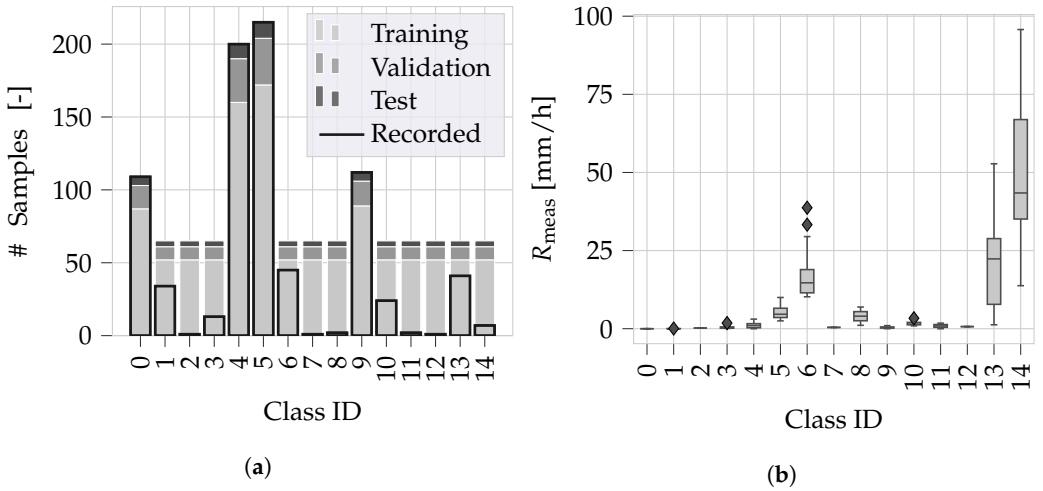


Figure 5. (**a**) Distribution of the training data with respect to the precipitation class (see Table 2). While the proportion of the originally recorded samples is highlighted by a black frame, the train, test, and validation split is highlighted by color. Highly underrepresented classes were augmented by oversampling. (**b**) Distribution of the measured precipitation rates R_{meas} with respect to the precipitation class, showing typically occurring Middle European R spans for the corresponding classes and precipitation rates of up to $R \geq 90\,\mathrm{mm/h}$.

4.2.2. Input Data Format

For model training, an image time series TS, consisting of C = 2 channel images $\mathbf{x}$, each with intensity $\boldsymbol{I}$ and depth $\boldsymbol{L}$ channel, is utilized as model input:

$$\mathrm{TS} = \{\boldsymbol{x_k}, ..., \boldsymbol{x_{k-\mathbf{M}}}\} \tag{8}$$

$$= \{(\boldsymbol{I_k}, \boldsymbol{L_k}), ..., (\boldsymbol{I_{k-\mathbf{M}}}, \boldsymbol{L_{k-\mathbf{M}}})\} \tag{9}$$

Opposed to that, the corresponding SYNOP precipitation class and rate R are used as target values, for which they were de-serialized and converted to tensors to allow for fast data loading during training. All models are trained on a time series length of M = 200 images, which is equivalent to a window length of:

$$t_{\mathrm{int,LiD}} = \frac{\mathrm{M}}{f} = 8\,\mathrm{s} \tag{10}$$

Time-series-based inferencing instead of single image processing is absolutely essential for the noise-based precipitation state prediction, since single noise frames are highly ambiguous with respect to weather influence. Consequentially, the input tensor for all investigated architectures has the shape $(\mathrm{B}, \mathrm{M}, \mathrm{C}, \mathrm{H}, \mathrm{W})$ with B describing the batch size as relevant training hyperparameter and H × W as the dimensions of the corresponding 16 bit monochromatic images, as described earlier.

4.2.3. Measurement Noise Extraction

As mentioned before, this work claims for generalizability with respect to the background scene and application. In order to fulfill this requirement, only precipitation-induced noise is processed. As elaborated in Section 3.2, it is generally occurring rather close to the sensor aperture in the case of Flash LiDAR sensors. Following that, the major part of the noise was found to occur at a distance of $L_{n,k} \leq L_{\mathrm{crop}} = 1\,\mathrm{m}$. Accordingly, only false-positive detections at a distance smaller than—and just above—the sensor's specified minimum working distance of $L_{\mathrm{min}} = 0.5\,\mathrm{m}$ are considered; hence, data are mostly discarded on the application side. In case of the SUT, L_{min} is mainly dictated by the collinear orientation of transmitter and receiver and the resulting distance in which a full crossover between transmitter beam and receiver frustum is guaranteed [40,41].

Due to design-related close-distance backscattering sensitivity, it was found that the information content of close-approximity noise is, opposed to full PSVD prediction [18], sufficient to characterize the precipitation precisely by its respective SYNOP class and rate.

This does not only allow for an application-agnostic deployment, but also for performant noise extraction through simple distance-based cropping of $\boldsymbol{I_k}$ and $\boldsymbol{L_k}$. Consequentially, for each pixel (u, v), the noise frames $\boldsymbol{I_{\mathbf{n},k}}$ and $\boldsymbol{L_{\mathbf{n},k}}$ can be found by (see also Figure 1):

$$I_{\mathrm{n},k}(u,v) = \begin{cases} I_{\mathrm{n},k}(u,v), & \text{for } L_{\mathrm{n},k} \leq L_{\mathrm{crop}} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$L_{\mathrm{n},k}(u,v) = \begin{cases} L_{\mathrm{n},k}(u,v), & \text{for } L_{\mathrm{n},k} \leq L_{\mathrm{crop}} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

5. Model Architectures and Training

In this section, the implemented DL architectures are introduced and the training process, including the hyperparameter optimization, is described.

5.1. Model Architectures

Modern DL concepts, like convolutional- and transformer-based architectures, demonstrate outstanding performance in classic computer vision tasks like object detection, action recognition, and image segmentation [42,43]. However, these models typically require a

vast amount of computational resources. Therefore, as mentioned earlier, the DL models utilized in this work were not only selected with respect to the problem setting of an image time-series-based classification and regression problem, but also with focus on the in-field applicability and therefore fast inferencing on respective edge devices. Following this, all models were optimized and evaluated with respect to their performance, but also to their size and number of required multiply–accumulate (MAC) operations, which again dictate memory requirements and inference time. In the following, the investigated models are introduced, with clear emphasis on their main features and differences. For a comprehensive in-depth description, the reader is referred to the corresponding works.

5.1.1. Baseline Convolution-LSTM

As a simple baseline architecture, a naive convolution–long short-term memory (Conv-LSTM) model was implemented (see Figure 6). It consists of three subsequent convolutional units, each with a 2D convolution, maxpooling, and an ReLu layer, followed by three LSTM [44] cells with varying hidden state and input size, processing temporal context over the image time series. Following this, the Conv-LSTM implemented here has to be distinguished from a convolutional LSTM, in which the convolutions replace specific gate functions in the LSTM cell itself [45].

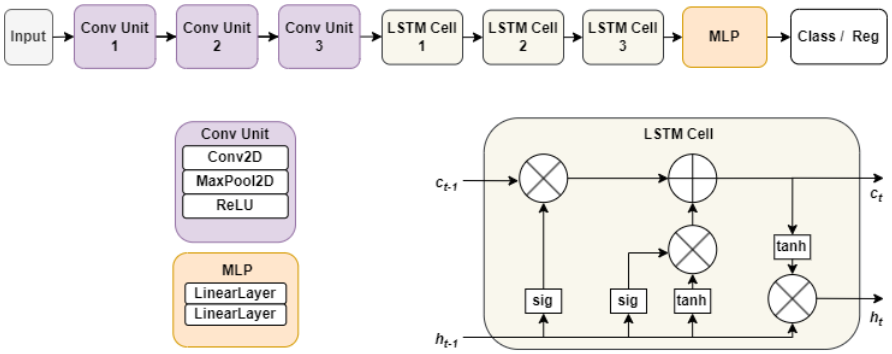


Figure 6. General architecture of the simple, implemented convolutional LSTM.

In order to do so, the fundamental core of the LSTM cell consists of a memory cell $\mathbf{c_t}$, accumulating temporal information and being subject to manipulation—including access, addition, and deletion of data (see Figure 6, LSTM cell). This is implemented by several self-parameterized controlling gates, such as an input gate $\mathbf{i_t}$, which controls new information flow $\mathbf{x_t}$ to the memory, and the forgot gate $\mathbf{f_t}$, in order to delete past information from $\mathbf{c_{t-1}}$. Finally, the output gate $\mathbf{o_t}$ controls which way information propagates from the last cell state $\mathbf{c_t}$ to the cell's final hidden state $\mathbf{h_t}$. Each gate function itself is realized by a respective nonlinear activation function in the form of either a sigmoid function σ or the hyperbolic tangent. The mathematical model can be described as follows, where $\circ$ denotes the Hadamard product, the matrices $\mathbf{W}$ contain the respective, learnable edge weights and $\mathbf{b}$ the according bias [44,45]:

$$\begin{aligned}
\mathbf{i_t} &= \sigma(\mathbf{W_{xi}x_t} + \mathbf{W_{hi}h_{t-1}} + \mathbf{W_{ci}} \circ \mathbf{c_{t-1}} + \mathbf{b_i}) \\
\mathbf{f_t} &= \sigma(\mathbf{W_{xf}x_t} + \mathbf{W_{hf}h_{t-1}} + \mathbf{W_{cf}} \circ \mathbf{c_{t-1}} + \mathbf{b_f}) \\
\mathbf{c_t} &= \mathbf{f_t} \circ \mathbf{c_{t-1}} + \mathbf{i_t} \circ \tanh(\mathbf{W_{xc}x_t} + \mathbf{W_{hc}h_{t-1}} + \mathbf{b_c}) \\
\mathbf{o_t} &= \sigma(\mathbf{W_{xo}x_t} + \mathbf{W_{ho}h_{t-1}} + \mathbf{W_{co}} \circ \mathbf{c_t} + \mathbf{b_o}) \\
\mathbf{h_t} &= \mathbf{o_t} \circ \tanh(\mathbf{c_t})
\end{aligned} \tag{13}$$

For the final classification (Class) or regression (Reg) task, a multilayer perceptron (MLP) with two linear layers is utilized.

5.1.2. 3D Convolutional Network

In order to consider temporal context within the convolution operations, 3D spatiotemporal convolutions are evaluated. For that purpose, the ResNet3D [46] architecture is utilized. It is an adaptation of the renowned Residual Network (ResNet) architecture [47], designed to process three-dimensional data by replacing classic 2D convolutions with 3D convolutions to extract spatiotemporal features. By that, it extends the foundational principles of ResNet into the 3D domain, leveraging volumetric data across various applications, particularly in video analysis [48]. Like its 2D counterpart, ResNet3D is structured in subsequent layers, each with 1...N blocks of convolution, ReLu, and normalization layers. Each layer is connected to its subsequent neighbor by additional residual connections, either directly or by a residual unit with an additional convolution and normalization layer (see Figure 7).

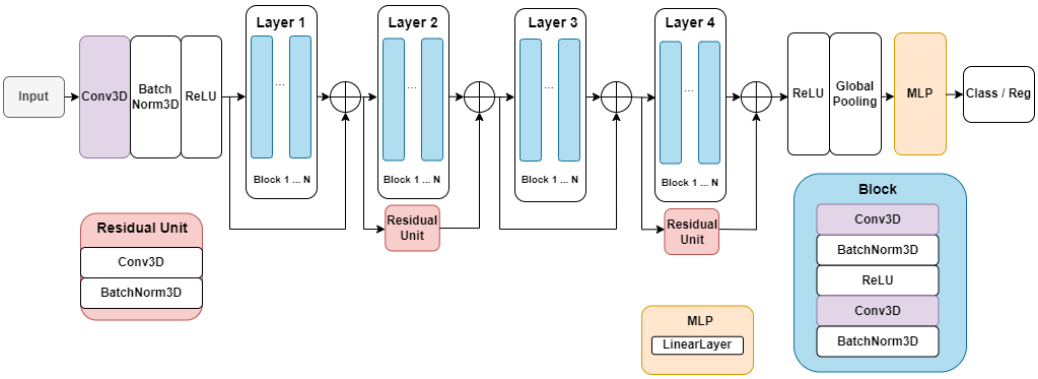


Figure 7. General architecture of the implemented ResNet3D, based on the work of Tran et al. [46].

The residual connections are critical in mitigating the vanishing gradient problem, allowing for deeper networks by enabling a direct flow of gradients. Further, the inclusion of residual connections ensures that information is preserved across the network during inferencing, enhancing the model's ability to capture and utilize deep, complex patterns in 3D data. Similar to the Conv-LSTM, the classification and regression task is accomplished by a respective MLP, here with one linear layer.

5.1.3. Vision Transformers

In addition to the more classic approaches above, modern transformer-based models for image processing, following the fundamental work of Vaswani et al. [31], were implemented. Vision Transformers (ViTs), as introduced by Dosovitskiy et al. [49], basically build up on the original transformer concept, applying the concept of multihead self-attention to image data.

In accordance with the application of this work, single noise images with their corresponding $C = 2$ intensity and depth channel $\mathbf{x}_{n,k} = \{\mathbf{I}_{n,k}, \mathbf{L}_{n,k}\}$ of size $H \times W$ are therefore divided into a sequence of $S = HW/P^2$ flattened 2D patches with patch size P. In order to map the patch sequence to a latent vector of length D, a trainable linear projection $\mathbf{E}$ is used. Its output is, in the ViT context, referred to as patch embedding (see Figure 8). In order to provide the model a capability of encoding spatial context, a learnable 1D position embedding $\mathbf{E}_{pos}$ is added to each projected and flattened image patch. Hence, the input of the first transformer encoder can be described as follows [49]:

$$\mathbf{z_0} = \left[\mathbf{x}_{class}; \mathbf{x_p}^1\mathbf{E}; \mathbf{x_p}^2\mathbf{E}; \cdots; \mathbf{x_p}^N\mathbf{E}\right] + \mathbf{E}_{pos}, \quad \mathbf{E} \in \mathbb{R}^{(P^2 \cdot C) \times D}, \mathbf{E}_{pos} \in \mathbb{R}^{(S+1) \times D} \tag{14}$$

After passing through multiple transformer encoders (referred to as depth), information flows through an MLP head for the classification or regression, incorporating features such as layer normalization, linear transformation, GELU activation, and dropout.

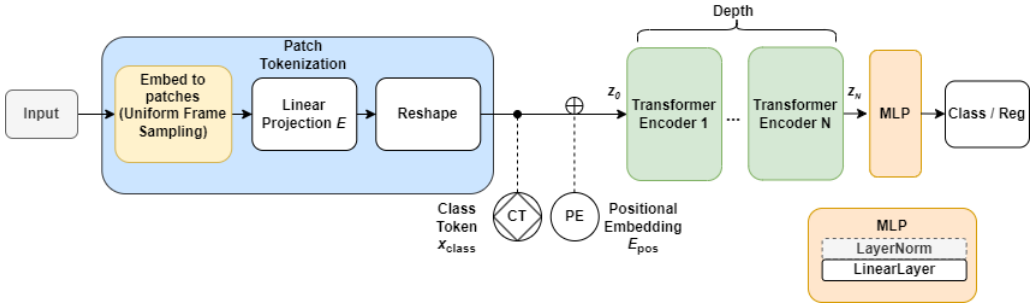


Figure 8. General architecture of the baseline Vision Transformer (ViT-B), as well as its simplified version (ViT-S) with missing normalization layer in the MLP, based on the work of Dosovitskiy et al. [49] and Beyer et al. [50].

The transformer encoder blocks are in accordance with the architecture proposed by Vaswani et al. [31] (see Figure 9). Thus, the patch embeddings are normalized and forwarded to a multihead attention block. The output is added to a residual connection from the patch embedding. The result is normalized and fed through an MLP to be finally added with the residual bypassing the MLP. The classic **QKV** self-attention (SA) [31] generally computes a weighted sum over all values **V** in an input sequence $\mathbf{z} \in \mathbb{R}^{N \times D}$. The respective attention weights $A_{i,j}$ are based on the similarity between two elements of the sequence and their corresponding query $\mathbf{Q^i}$ and key $\mathbf{K^i}$ representation [31]:

$$\begin{aligned} [\mathbf{Q}, \mathbf{K}, \mathbf{V}] &= \mathbf{z}\mathbf{U}_{\mathbf{QKV}} & \mathbf{U}_{\mathbf{QKV}} &\in \mathbb{R}^{D \times 3D_h}, \\ \mathbf{A} &= \text{softmax}\left(\mathbf{Q}\mathbf{K}^\top / \sqrt{D_h}\right) & \mathbf{A} &\in \mathbb{R}^{N \times N}, \\ \text{SA}(\mathbf{z}) &= \mathbf{A}\mathbf{V}. \end{aligned} \tag{15}$$

The multihead self-attention (MSA) is, finally, an extension of SA in which several self-attention operations (heads) are run in parallel, while projecting their concatenated outputs [31].

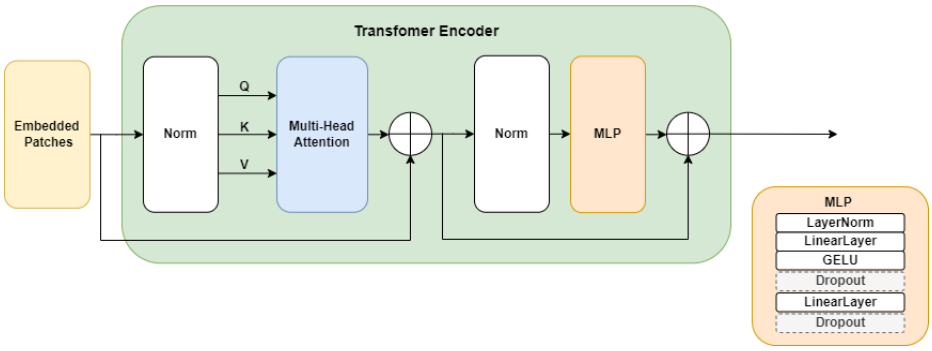


Figure 9. General architecture of the transformer encoder of the baseline Vision Transformer (ViT-B), as well as its simplified version (ViT-S) with missing dropout layers in the MLP, based on the works of Dosovitskiy et al. [49] and Beyer et al. [50].

Now, as discussed in Section 4.2.2, the input consists of an image time series TS rather than single images. Therefore, in order to consider temporal context, the original work of Dosovitskiy et al. [49] is extended such that a stack of $M = 200$ images is processed by the ViT instead of a single frame. In order to do so, uniform frame sampling in accordance with Wang [51] and Arnab et al. [52] is utilized to simply extract the patches over the whole TS instead of a single frame to forward $N \times M$ patch embeddings to the subsequent linear projection **E**. In the following, the original baseline Vit [49] with the abovementioned adaptation is referred to as ViT-B.

Following that, SimpleViT (ViT-S) is a simplification of the ViT-B model [50], which was originally designed for the well-known ImageNet dataset [53]. Following the embedding

process of ViT-B to adeptly process image sequences, ViT-S addresses the data efficiency challenge of its predecessor through optimized training strategies and architectural tweaks (see Figures 8 and 9), making it more suitable for scenarios where 3D data are scarce or expensive to procure, which aligns well with the challenges faced in this work.

5.1.4. Convolutional Transformer

The Compact Convolutional Transformer (CCT), as proposed by Hassani et al. [54], is designed to deviate from the traditional Vision Transformers above by substituting patch embeddings with convolutional embeddings (see Figure 10). However, in order to apply the CCT architecture to image time series, the convolution is again applied to the whole uniformly sampled image stack [51,52]. The alteration from patch to convolutional embedding enhances the inductive bias, making positional embeddings potentially redundant and, therefore, optional [54]. Additionally, CCT utilizes sequence pooling after the transformer encoder, which applies a linear projection and a subsequent softmax function to the transformer encoder output sequence before forwarding it to the MLP head for classification or regression. As a result of these modifications, CCTs demonstrate superior accuracy compared to smaller Vision Transformers, such as ViT-Lite, on benchmark datasets like ImageNet [53,54]. Unlike conventional Vision Transformers, which typically require extensive datasets for training, CCTs are capable of rapid training on smaller datasets while achieving satisfying accuracy levels—again, a feature that aligns with the tight training budget constraints in this work.

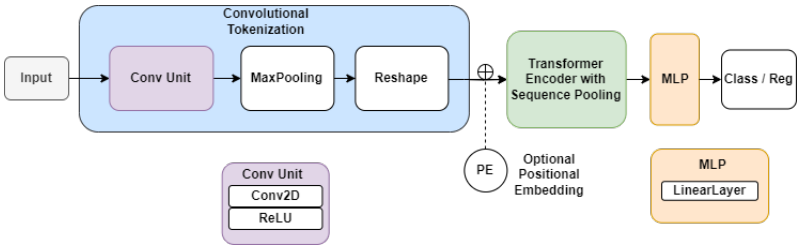


Figure 10. General architecture of Compact Convolutional Transformers (CCTs) as proposed by Hassani et al. [54].

5.1.5. Video Vision Transformer

Opposed to the direct patch embedding of the entire TS, Video Vision Transformer (ViViT) approaches, introduced by Arnab et al. [52], separate the processing of temporal and spatial features. In order to do so, they proposed four different models, in which the fusion of temporal and spatial features is realized on different architectural depths. These reach from assigning a specific MSA in the transformation encoder to respective spatial and temporal feature processing, to assigning a whole transformer encoder for spatial and temporal feature processing. The latter is implemented in this work (see Figure 11), which is referred to as late spatiotemporal fusion by the authors [52].

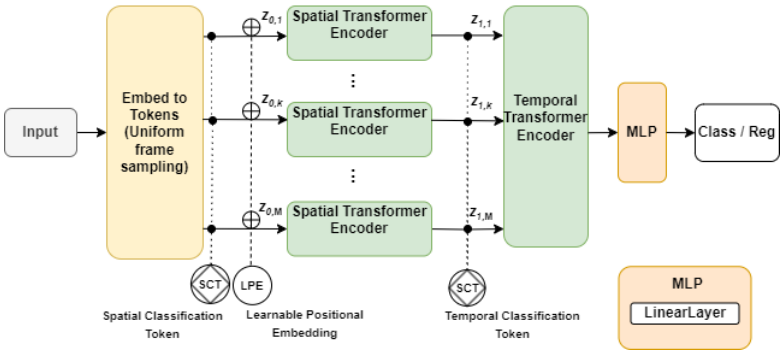


Figure 11. General architecture of the Video Vision Transformer (ViViT) as introduced by Arnab et al. [52].

Instead of projecting all image patches in the TS to one transformer encoder, the patches of each two-channel image per time step are separately projected to a respective spatial transformer encoder, whose outputs are then processed by a mutual temporal transformer encoder. Similar to the previous models, the output is finally fed to aN MLP head, conducting the classification or regression task.

5.2. Model Training and Optimization

In this section, the model training, including hyperparameter and structural optimizations with respect to the problem setting, is described.

5.2.1. Model Training

As elaborated earlier, the investigated models typically require a major amount of computational resources. Considering a future in-field application on a low-power edge device, the authors place focus on the conflicting goals of achieving maximum predictive performance, while reducing model size and the number of MAC operations to a minimum.

For training in general, the well-known ADAptive Moment estimation (ADAM) optimizer [55] is used, since it is proven to be effective in training models that utilize convolutional and attention-based mechanisms [49,55,56]. In terms of the learning rate, an initial value of $\eta = 3e-4$ is set. Further, cosine annealing learning rate scheduling, as proposed by Loshchilov and Hutter [57], is implemented, since it showed satisfying performance within previous studies in the field [54]. All models are trained for a maximum of 200 epochs; however, in order to prevent overfitting, early stopping is applied as soon as the validation error is converging, even though the training error is further decreasing.

Opposed to the generally improved convergence behavior of large batch sizes during stochastic gradient descent (SGD) optimization [58], initial experiments indicated that an increase in the batch size did not lead to significant improvements using the ADAM optimizer, except in the case of the ResNet3D model. Consequentially, the batch size is only fine-tuned for this model, whereas it is kept to a constant value of $N_{batch} = 1$ for all other architectures. This increases training time; however, considering limited computational resources (see Appendix C), it allows for exploration of larger model architectures during structural optimization, while complying with GPU memory restrictions.

Additionally, it was found that the performance of the models improves with longer image time series M, which aligns with the intuition that longer observation periods of the measurement noise lead to improved model performance. Since the model performance showed no significant improvement for longer sequences, a constant input number of $M = 200$ images is utilized, as described earlier.

As loss metric for the respective regression heads, the common mean squared error (MSE) between predicted precipitation rate $R_{pred,i}$ and the respective measured value $R_{meas,i}$ is used:

$$\mathcal{L}_{MSE} = \frac{1}{N_{batch}} \sum_{i=1}^{N_{batch}} \left(R_{pred,i} - R_{meas,i} \right)^2. \tag{16}$$

whereas for the classification heads, the widely used cross-entropy loss function [59] is implemented:

$$\mathcal{L}_{CE} = - \sum_{c=1}^{N_{SYNOP}=15} y_{o,c} \log(p_{o,c}) \tag{17}$$

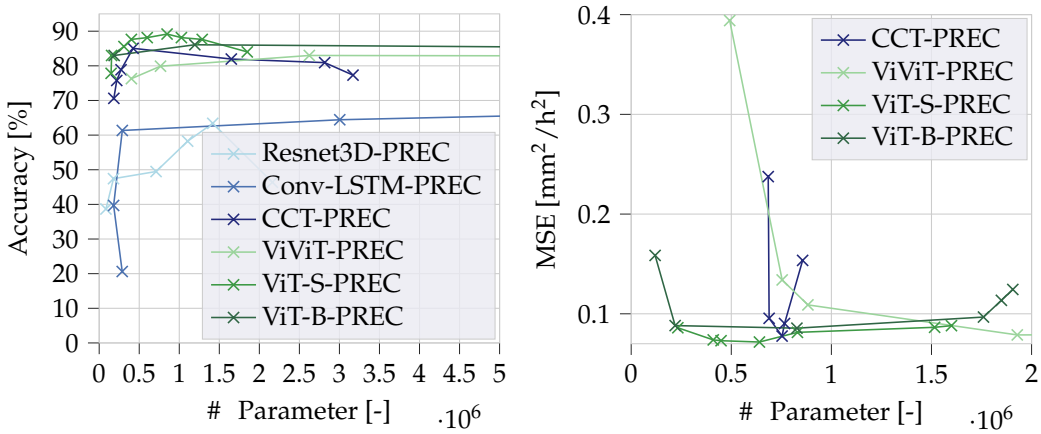
with y as binary indicator (0 or 1) if class label c is the correct classification for observation o, and p is the predicted probability that observation o is of class c.

5.2.2. Structural Optimization

An extensive ablation study of the proposed architectures is performed with the intention to adapt the architectures to the problem setting and therefore optimizing performance and model size with respect to the task. For this, the Bayesian-based Tree-structured Parzen Estimator (TPE) [60] is utilized, in order to search for each model's optimal configuration

in a corresponding hyperparameter space (see Tables A1 and A2 in Appendix A). The configuration of the original model is listed next to the partly discrete and continuous search ranges and acts as upward estimation in terms of complexity, since the the input data are of lower dimension and the prediction task is assumed to be less complex compared to benchmark applications. The experimental setup aims to conduct approximately 100 trials per model, irrespective of the corresponding size of its hyperparameter space. Opposed to the final training, the models are trained for 30 epochs per trial and evaluated on the validation data. This configuration strikes a balance between computational resources and the number of trials, minimizing memory usage while providing a sufficient learning window to assess model performance effectively.

The circa 100 resulting model configurations per architecture are evaluated with respect to their classification and regression accuracy versus their model size (see Figure 12). Since model size and accuracy are typically conflicting optimization goals, a Pareto analysis is conducted. In order to do so, a scatter plot is generated, in which each point represents one resulting model configuration, marking the accuracy and MSE over the according number of model parameters. For the sake of clarity, instead of all trials, only the resulting Pareto front in the form of the convex hull over the best model configurations is visualized for each architecture. The closer the classification accuracy curves are to the upper left corner, the better the model performance is with respect to size and accuracy. In case of regression performance (in terms of the MSE metric), the same applies to the lower left corner. To emphasize optimizations and adaptations with respect to the problem setting, the original names of the model architectures are extended with the suffix "Precipitation Rate Estimation and Classification" (PREC).



(**a**) Classification accuracy versus model size. (**b**) Regression accuracy versus model size.

Figure 12. Pareto study on the classification and regression accuracy with respect to the model size for each investigated DL architecture during hyperparameter and network structure optimization, based on the validation data. The ViT models especially stand out by reaching a comparatively high accuracy with a fairly small model size of fewer than 500 k parameters.

Looking at the classification performance, one can clearly see that the ViT models are outperforming the convolution-based ResNet as well as the Conv-LSTM model. Especially, the ViT-S architecture is showing excellent performance, with up to over 89 % classification accuracy on the validation dataset with a fairly small model size of fewer than 1 M parameters. Opposed to that, the Conv-LSTM was only able to increase performance with an unacceptable size of greater than 5 M parameters.

A similar result can be observed for the regression performance. Here, the Conv-LSTM-PREC and ResNet3D-PREC model, however, showed massive problems in learning a meaningful correlation at all. Due to this, neither of the models are illustrated and were not further considered for the regression task.

Based on this study, two parameterizations per model are selected for a final training with a maximum of 200 epochs and early stopping as described above: the best performing model within a size of fewer than 5 M parameters and a small model version with significantly smaller number of parameters but only slight performance loss compared to the best in class model. The small model version is marked with a "small" (s) suffix in the following.

6. Discussion of the Results

6.1. Classification

The final classification results on the test dataset are listed in Table 3. As expected, the ViT models show, again, superior performance compared to the ResNet3D-PREC and Conv-LSTM-PREC, with the ViT-B-PREC and ViT-S-PREC models both achieving up to 84.41% top 1 accuracy and up to 96.10% top 2 accuracy (meaning that the target class is within the two most probable predicted classes). Interestingly enough, however, ViT-S-PREC is significantly smaller, with circa 29% fewer parameters, but on the other hand conducting more MAC operations due to the missing dropout (see Section 5.1), which leads to a design trade-off between memory requirements and computing power.

Table 3. Comparison of model sizes and respective classification performance on the test dataset. The best performance in terms of top 1 and top 2 accuracy is marked in bold, whereas the correspondingly required computational resources are indicated by the number of model parameters (# Params) and MAC operations.

Model	Top 1	Top 2	F1 Score	# Params	MACs
3D Convolutional Networks					
ResNet3D-PREC	74.02%	88.31%	0.69	1.41 M	368.50 G
ResNet3D-PREC-s	77.92%	**96.10**%	0.73	0.17 M	37.71 G
Convolution-LSTMs					
Conv-LSTM-PREC	64.93%	92.21%	0.67	0.73 M	3.02 G
Conv-LSTM-PREC-s	74.04%	85.71%	0.74	0.21 M	1.32 G
Vision Transformers					
ViT-B-PREC	**84.41**%	**96.10**%	0.85	1.19 M	0.22 G
ViT-B-PREC-s	79.22%	93.51%	0.78	0.19 M	1.50 G
ViT-S-PREC	**84.41**%	**96.10**%	0.85	0.85 M	1.86 G
ViT-S-PREC-s	79.22%	**96.10**%	0.83	0.32 M	0.63 G
Convolutional Transformers					
CCT-PREC	83.12%	**96.10**%	0.84	0.42 M	16.63 G
CCT-PREC-s	75.32%	93.51%	0.76	0.27 M	1.97 G
Video Vision Transformers					
ViViT-PREC	79.22%	93.51%	0.80	2.62 M	0.46 G
ViViT-PREC-s	75.32%	92.21%	0.75	0.81 M	0.25 G

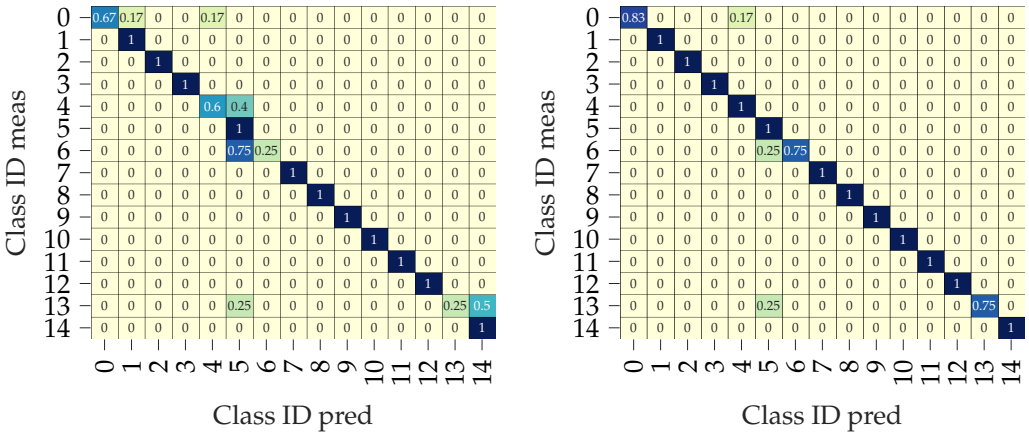
It is worth mentioning that the corresponding small model versions show quite a similar performance with a model size of just 190 k and 320 k parameters. Opposed to the ViT models, however, the ResNet-PREC and Conv-LSTM-PREC show better performance on the test data compared to the initial validation results, which is due to the fact that the models improved quite remarkably over the 200 training epochs compared to the 30-epoch training during model validation. Also, the small model versions are performing significantly better, with Conv-LSTM-PREC-s reaching up to 74.04% top 1 accuracy and ResNet-PREC-s achieving 77.92% top 1 accuracy, which might indicate some remaining degree of overfitting in the case of the larger model versions.

The general superiority of the transformer-based model is not surprising and was also demonstrated and explained by earlier works [43]. Thus, by incorporating the self-

attention mechanism (see Section 5.1), Vision Transformers have the ability to learn a global, spatiotemporal context over the whole time series, whereas 3D convolutions and LSTMs rely on a rather local, temporal context. As mentioned earlier, the intensity of the detected LiDAR noise can, however, fluctuate quite strongly in between single frames, which makes a broad temporal context crucial.

This observation is further backed by evaluating the discriminatory power of the models with respect to specific precipitation classes. For this purpose, the normalized top 1 and top 2 confusion matrices of each model were consulted and, in case of the best performing ViT models, are illustrated in Figures 13 and 14. Similar to the discussed related works, the predicted precipitation classes (Class ID pred) are evaluated with respect to the actual, measured value (Class ID meas). The relative share of correct classifications (true positives) is shown on the diagonal, whereas class-specific misclassifications are represented by the upper and lower column values (false positives) as well as the the left and right row values (false negatives).

Thus, the classification performance appears quite remarkable for most of the 15 classes, especially considering the top 2 confusion matrices. Hence, all transformer-based models achieved a test accuracy of mostly 100% when distinguishing different precipitation types like hail, snow, drizzle, and rain, and, therefore, performed comparably to or better than ML models in previous works, but solely based on noise.



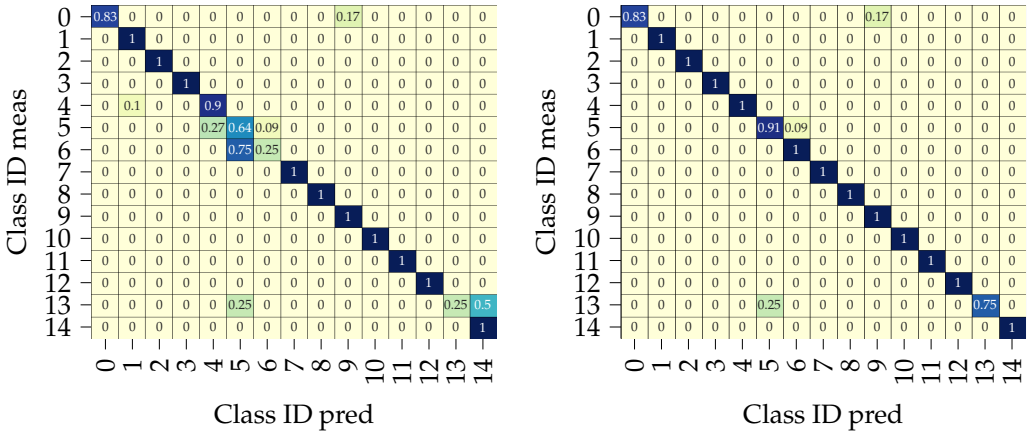
(**a**) ViT-S top 1 accuracy (overall 84.41 %). (**b**) ViT-S top 2 accuracy (overall 96.10 %).

Figure 13. Test data classification results of the ViT-S-PREC model in the form of normalized confusion matrices, evaluating the predicted precipitation classes (Class ID pred) with respect to the actual, measured value (Class ID meas).

However, the models still have difficulties in distinguishing closely related classes, like 4, 5, 6—meaning slight, medium, and heavy rain—or hail shower intensity (13,14). In these classes, all models lack discriminatory power (see also the normalized confusion matrices for the ViViT model Figure A2 and the CCT model Figure A1, as well as the baseline models Figures A3 and A4 in Appendix B). This might occur due to a strong similarity of the corresponding noise characteristics, especially in edge cases between two intensity classes. Moreover, the precipitation intensity can fluctuate quite strongly in short time periods. Since the measurement data are acquired over $t_{\mathrm{int}} = 60\,\mathrm{s}$ intervals, while the targeted time window for prediction is comparatively small with $t_{\mathrm{int,LiD}} = 8\,\mathrm{s}$, misclassification can likely occur close to the decision boundaries. Furthermore, synchronization issues between PSU and WSU, and therefore sporadic faulty labels, cannot always be avoided.

In addition to that, CCT-PREC and ViViT-PREC appear to show difficulties in distinguishing slight drizzle from clear weather (0,1). The same applies to the baseline models, with additional difficulties in discriminating slight rain and drizzle (3,4) as well as slight and moderate snowfall (9,10).

In summary, the transformer-based architectures in particular proved to be highly accurate in precipitation classification. The ViT-B-PREC and ViT-S-PREC models especially stand out by solely showing a lack of discriminatory power for 3 out of 15 classes, while being comparatively small in size.



(**a**) ViT-B top 1 accuracy (overall 84.41 %). (**b**) ViT-B top 2 accuracy (overall 96.10 %).

Figure 14. Test data classification results of the ViT-B-PREC model in the form of normalized confusion matrices, evaluating the predicted precipitation classes (Class ID pred) with respect to the actual, measured value (Class ID meas).

6.2. Regression

Regarding the regression, the results are similar. The ViT models perform superior, with the ViT-S-PREC reaching a coefficient of determination between measured and predicted precipitation rate of $r^2 = 85.63\%$ with a size of just 640 k parameters (see Table 4), which corresponds to an MSE of 0.22 mm^2/h^2. Again, the small model version ViT-S-PREC-s with just 400 k parameters achieved a comparative performance. The other transformer-based architectures, however, only achieved coefficients of determination of $\leq$80%, whereas the baseline models were not able to conduct any meaningful prediction at all.

Table 4. Comparison of model sizes and respective regression performance on the test dataset. The best performance in terms of MSE and r^2 is marked in bold, whereas the correspondingly required computational resources are indicated by the number of model parameters (# Params) and MAC operations.

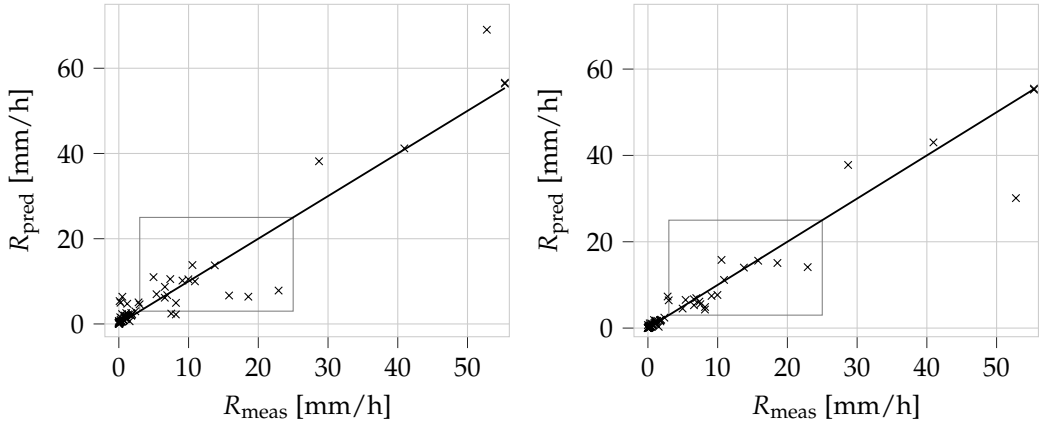
Model	MSE [mm^2/h^2]	r^2	# Params	MACs
Vision Transformers				
ViT -B-PREC	0.32	78.80%	0.83 M	1.90 G
ViT-B-PREC-s	0.43	69.43%	0.31 M	0.70 G
ViT-S-PREC	**0.22**	**85.63**%	0.64 M	2.91 G
ViT-S-PREC-s	0.24	84.00%	0.40 M	1.48 G
Convolutional Transformers				
CCT-PREC	0.33	77.69%	0.75 M	1.15 G
Video Vision Transformers				
ViViT-PREC	0.53	64.48%	1.23 M	0.25 G

In a final experiment, the authors evaluated whether an embedded SYNOP class could potentially enhance the prediction of the respective precipitation rate by following the assumption that a correctly predicted class prior could support the network in deriving the precipitation rate, especially in the described edge cases. In order to do so, the ground truth

SYNOP classes were embedded into the input of the first transformer encoder as a class token. For that, two methods are tested: Firstly, a trainable linear layer (linear embedding) is implemented in order to transform the classification output as one-hot encoded vector to a latent vector which matches the transformer input tensor shape, such that both can be concatenated. Secondly, the one-hot encoded classifier input is simply concatenated with zeros (zero padding) in order to match the input tensor shape of the transformer. As illustrated in Figure 15 and Table 5, performance is enhanced notably in both cases, with a coefficient of determination of up to $r^2 = 92.73\%$. Comparing the regression curves, one can see a significant variance decrease in the critical region between $R = 5\,\mathrm{mm/h}$ and $R = 25\,\mathrm{mm/h}$, which indicates a clearly improved regression in this region. However, since the classification performance was weakest in exactly the two covered transition regions between classes 4, 5, and 6, the application of an SYNOP class embedding from a prior class prediction did, opposed to the ground truth embedding, not further improve regression performance.

Table 5. Comparison of precipitation rate regression performance for different classification prior embedding methods. The best performance in terms of MSE and r^2 is marked in bold, whereas the correspondingly required computational resources are indicated by the number of model parameters (# Params) and MAC operations.

Model	MSE [mm²/h²]	r²	# Params	MACs
Vision Transformers				
ViT-S-PREC Synop Linear Embedded	**0.11**	92.61%	1.48 M	0.322 G
ViT-S-PREC Synop Zero Padded	**0.11**	**92.73**%	1.47 M	0.322 G



(**a**) ViT-S-PREC ($r^2 = 85.63\,\%$). (**b**) ViT-S-PREC-SynopEmbed ($r^2 = 92.73\,\%$).

Figure 15. Predicted precipitation rate R_{pred} versus measurement value R_{meas} for ViT-S based on the test data via direct prediction (**a**) and with an embedded SYNOP class (**b**). A prior precipitation classification increases regression accuracy if used as class embedding, as shown by the respective coefficient of determination r^2. Especially, close to the classification boundaries of slight, medium, and heavy rain (gray box), the regression clearly improved.

6.3. Implications for Future Use Cases

The results show that the transformer-based models especially have great potential to be deployed in static applications like roadside units (RSUs) within smart infrastructure traffic environments, in which LiDAR sensors are more and more used for 3D tracking and counting of traffic participants [61–63]. Since those sensors are typically mounted at a height of several meters to ensure that sufficient perceptual area coverage and only detections within a distance of maximum $L_{crop} = 1\,\mathrm{m}$ are processed by the models, they would not be impacted by close traffic objects. In this case, only the influence of the

mounting angle, more precisely the sensor pitch, would have to be evaluated with respect to the precipitation angle and respective wind conditions.

In case of an in-vehicle deployment, however, the impact of driving speed and, hence, strong headwind and spray, have to be investigated, since these might have a major impact on the prediction performance of the models. In this case, weather characterization could possibly be conducted in predefined speed ranges. In addition to strong headwind, objects are also more likely to occur within a detection range of less than 1 m. However, objects between the minimum detection range and filter distance $L_{\min} \leq L_{u,v} \leq L_{\text{crop}}$ can easily be filtered out, either based on object detection, or by additional temporal filtering, since precipitation particles are only visible for one frame, whereas objects typically occur for several frames. The same generally applies for objects located at a distance smaller than the minimum detection range $L_{u,v} \leq L_{\min}$, which occurs rather rarely and is typically considered as sensor blockage.

In general, blockage in the form of dirt could also have a significant influence in both use cases. However, it can already be detected in many LiDARs, like the SUT, such that cleaning can either be advised or automatically conducted.

In the case of a widespread deployment in vehicles and infrastructure, the comprehensive local weather data could not only be directly used to issue driver warnings or adapt driving dynamics and ADAS functions of the ego vehicle—it could additionally be shared via Car2X communication in order to generate dense traffic weather maps. These would not only offer information on local driving conditions, but might also generate enormous potential in meteorological data assimilation to improve numerical models for weather prediction.

7. Conclusions

In this study, the authors implemented and evaluated cutting edge deep learning architectures for image time series processing to characterize occurring adverse weather conditions based on precipitation-induced noise in LiDAR measurements. It was shown that the weather condition could be precisely classified by 15 WMO-standardized SYNOP precipitation classes and an additional precipitation rate regression. Hence, the implemented Vision Transformers, derived from the ViT-S and ViT-B architectures, could classify the precipitation type with a top 1 test accuracy of 84.41% and a top 2 accuracy of up to 96.10%. In addition, the ViT-S model was able to predict the precipitation rate with a coefficient of determination of up to r^2 = 85.63% with a fairly small size of 640 k parameters. Since, predictions are solely conducted based on filtered noise in direct proximity to the sensor aperture, the lightweight models can potentially be deployed in varying automotive and smart infrastructure applications without inducing concept drift. Considering the strongly increasing deployment of LiDAR sensors in these domains, such comprehensive local weather state data might be transmitted over Car2X communication to create dense traffic weather maps, which offer enormous potential for driver warnings and dynamic adaptation of ADAS functions, but also for meteorological data assimilation in order to improve numerical weather prediction models. Classification errors mostly appeared in edge cases between varying precipitation intensities (“slight”, “medium”, “heavy”), where the models lacked discriminatory power. This might result from a rather sparse amount of data in edge cases, but also from limitations of the test setup for data acquisition, which need to be further investigated. However, all transformer-based models generally achieved a remarkable classification performance on distinguishing different precipitation types like hail, snow, drizzle, and rain by achieving a test accuracy of over 90% for almost all classes.

Author Contributions: Conceptualization, M.K. and G.E.; methodology, M.K.; software, N.S. and M.K.; validation, N.S. and M.K.; formal analysis, M.K., N.S. and G.E.; investigation, M.K.; resources, H.E.; data curation, M.K. and N.S.; writing—original draft preparation, M.K. and N.S.; writing—review and editing, G.E. and B.W.; visualization, M.K. and N.S.; supervision, G.E. and B.W.; project administration, H.E.; funding acquisition, G.E., B.W. and H.E. All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the German Federal Ministry of Digital and Transport (BMDV) within the Automated and Connected Driving funding program under Grant No. 45AVF2010G 10-01284 (SAVeNoW) and the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within the Gaya-X 4 Future Mobility project family under Grant No. 19S21006O (GAIA-X 4 PLC-AAD).

Data Availability Statement: Data is contained within the article: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Acknowledgments: The financial support of the BMDV and BMWK is greatly acknowledged. Furthermore, we appreciate the valuable contributions of our students Philipp Opheys, Taylan Volkan, Thomas Hirmer, Laurenz Nitsche, Robert Eggl, Vanessa Braun, Lukas Petz, Neziha Tavsan, and Ghazi Maamar participating in the joint THI/Continental Artificial Intelligence student project 2023.

Conflicts of Interest: The author Hüseyin Erdogan was employed by the company Conti Temic Microelectronic GmbH. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADAM	Adaptive Moment Estimation
ADAS	Advanced Driving Assistance Systems
BMDV	German Federal Ministry of Digital and Transport
BMWK	German Federal Ministry for Economic Affairs and Climate Action
BRDF	Bidirectional Reflectance Distribution Function
CPU	Central Processing Unit
DL	Deep Learning
GPU	Graphics Processing Unit
KNN	k-Nearest Neighbor
LiDAR	Light Detection and Ranging
LSTM	Long Short-Term Memory
MAC	Multiply-Accumulate
MAE	Mean Absolute Error
MEMS	Micro-Electromechanical System
ML	Machine Learning
MSA	Multihead Self-Attention
PIDL	Physics-Informed Deep Learning
PREC	Precipitation Rate Estimation and Classification
PSU	Perception Sensor Unit
ResNet	Residual Network
RGB	Red Green Blue
ROS	Robot Operating System
SA	Self-Attention
SGD	Stochastic Gradient Descent
SUT	Sensors Under Test
SVM	Support Vector Machine
SYNOP	Surface Synoptic Observations
ViT	Vision Transformer
ViViT	Video Vision Transformer
WMO	World Meteorological Organization
WSU	Weather Sensor Unit

Appendix A. Model Parameterization

Table A1. Hyperparameter optimization ranges and maximum number of parameters of the transformer architectures in comparison to the original models proposed by earlier works. Models were implemented as close as possible to the original architecture; however, number of parameters (# Parameters) can slightly vary.

(**a**) Baseline Visual Transformer (ViT-B).		
Parameter	**Range**	**ViT-B-16** [49,64]
Image size	(32, 128)	224
Frames	200	-
Image patch	(2–32, 4–128)	4–4
Frame Patch	2–20	-
Dim	8–128	768
Depth	1–5	12
Heads	4–64	12
Head dim	2–128	64
MLP dim	8–128	3072
Dropout	0.1–0.5	0.1
Embed dropout	0.1–0.5	0.1
Pos Embed	{learn, no}	learn
Pool	{cls, mean}	mean
# Parameters:	33.9 M max	86.6 M
(**b**) Simplified Visual Transformer (ViT-S).		
Parameter	**Range**	**ViT-S-16** [50]
Image size	(32, 128)	224
Frames	200	-
Image patch	(2–32, 4–128)	4–4
Frame patch	2–20	-
Dim	8–128	768
Depth	1–5	12
Heads	4–64	12
Head dim	64	64
MLP dim	8–128	3072
Pos Embed	{sincos3d, no}	{sincos3d,no}
# Parameters:	33.9 M max	22.1 M
(**c**) Compact Convolutional Transformer (CCT).		
Parameter	**Range**	**CCT 14/7**$\times$ **2** [54]
Image size	(32, 128)	224
Frames	200	-
Embed dim	8–128	384
Num conv layers	1–3	2
Frame kernel	{3,5,7}	3
Kernel size	3–7	7
Stride	1–3	2
Padding	0–2	3
Pool kernel	3–7	3
Pool stride	1–3	2
Pool padding	1–3	1
Num layers	1–4	14
Num heads	8–128	6
MLP ratio	1.0–4.0	3.0
Pos embed	{learn, sin, no}	L
Num out channels	2–15	64
# Parameters:	108 M max	22.36 M

Table A1. *Cont.*

(**d**) Video Vision Transformer (ViViT).		
Parameter	**Range**	**ViViT** [52]
Image size	(32, 128)	224
Frames	200	32
Image patch	(2–32, 4–128)	16–16
Frame patch	2–20	2
Dim	8–128	768
Spat depth	1–3	1
Temp depth	1–3	4
Heads	8–128	12
Head dim	64	64
MLP dim	8–128	3072
Pooling	{cls, mean}	{cls, mean}
# Parameters:	46.7 M max	55.2 M

Table A2. Hyperparameter optimization ranges and maximum number of parameters of the baseline architectures, in the case of ResNet, and also in comparison to the original model.

(**a**) Convolution-LSTM.		
Parameter	**Range**	
Image size	(32, 128)	
Frames	200	-
Kernel size	3–7	
Num channels	1–25	
Hidden state size	32–128	
MLP dim	32–128	
(**b**) 3D Residual Network (ResNet3D).		
Parameter	**Range**	**ResNet3D-18** [46,65]
Image size	(32, 128)	224
Frames	200	-
Blocks/Layer	1–3	2
Kernels size	3–7	{1, 3, 7}
Num channels	1–64	64–512
Bias	True, False	False
Batch size	4–128	-
MLP dim	16–128	512
# Parameters:	2.67 M max	33.7 M

Table A3. Best performing model parameterization for the investigated transformer architectures.

(**a**) ViT-B-PREC.		
Parameter	**Class**	**Regression**
Image size	(32, 128)	(32, 128)
Image patch	(32, 128)	(16, 64)
Frames	200	200
Frame Patch	2	2
Dim	33	34
Depth	3	2
Heads	77	38
Head dim	20	64
MLP dim	32	33
Dropout	0.18	0.15
Embed dropout	0.16	0.11
Pos Embed	learn	learn
Pooling	mean	mean

Table A3. *Cont.*

(**b**) ViT-S-PREC.		
Parameter	**Class**	**Regression**
Image size	(32, 128)	(32, 128)
Frames	200	200
Image patch	(16, 64)	(16, 32)
Frame Patch	2	2
Dim	36	66
Depth	2	1
Heads	37	29
Head dim	64	64
MLP dim	61	80
Pos Embed	sincos3d	sincos3d
(**c**) CCT-PREC.		
Parameter	**Class**	**Regression**
Image size	(32, 128)	(32, 128)
Frames	200	200
Embed dim	64	128
Num conv layers	3	3
Frame kernel	5	3
Kernel size	3	3
Stride	1	3
Padding	0	2
Pool kernel	5	3
Pool stride	3	2
Pool padding	2	1
Num layers	1	3
Num heads	64	128
MLP ratio	3.81	1.96
Pos embed	sin	sin
Num out channels	21	5
(**d**) ViViT-PREC		
Parameter	**Class**	**Regression**
Image size	(32, 128)	(32, 128)
Frames	200	200
Image patch	(32, 128)	(32, 128)
Frame patch	2	2
Dimension	46	76
Spat depth	1	3
Temp depth	1	1
Heads	77	8
Head dim	64	64
MLP dim	95	29
Pooling	cls token	cls token

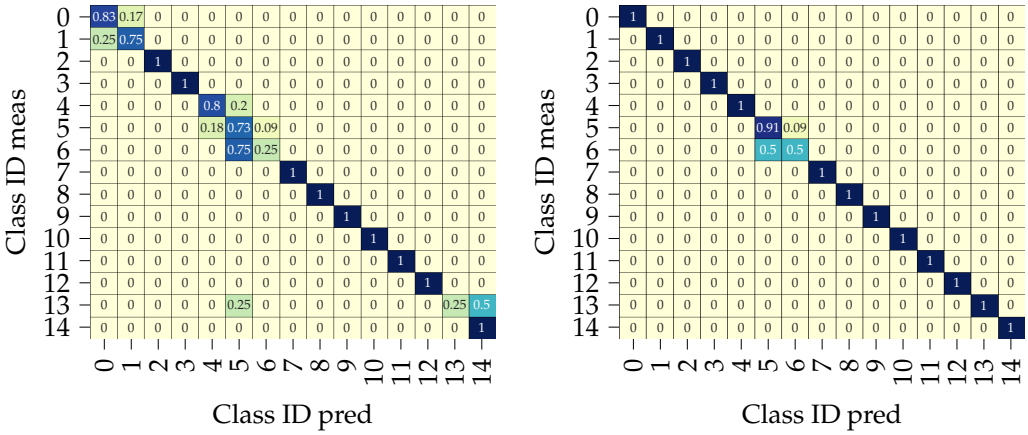
Table A4. Best performing model parameterization for the investigated baseline architectures.

(**a**) Conv-LSTM-PREC.		
Parameter	**Class**	**Regression**
Image size	(32, 128)	(32, 128)
Frames	200	200
Kernel Size	[7, 3, 5]	[3, 7, 7]
Num channels	[21, 12, 10]	[4, 24, 22]
Max Pooling	[1, 2, 2]	[2, 1, 2]
Hidden state size	[106, 104, 117]	[56, 92, 102]
MLP dim	21	109

Table A4. *Cont.*

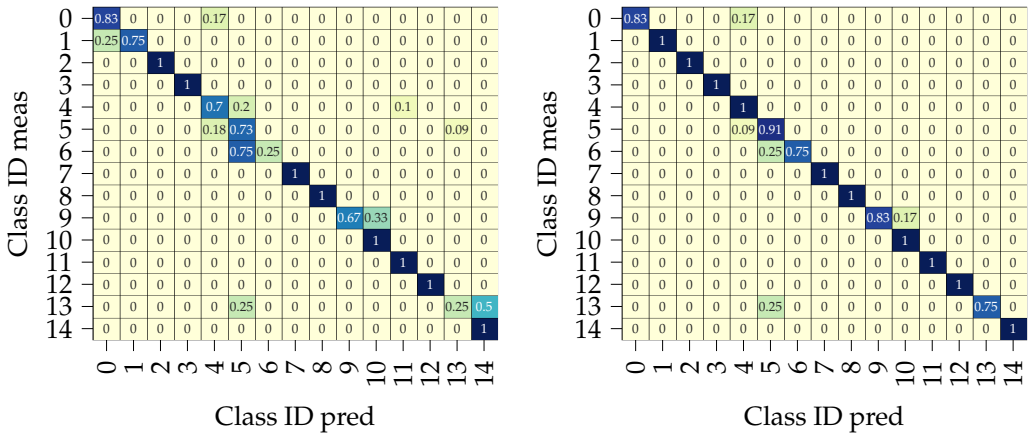
(**b**) ResNet3D-PREC.		
Parameter	**Class**	**Regression**
Image size	(32, 128)	(32, 128)
Frames	200	200
Blocks/layer	[2, 2, 3, 2]	[2, 2, 3, 2]
Kernel size	3	3
Num channels	19	19
Bias	True	True
Batch	8	8
MLP dim	19	19

Appendix B. Model Test Results



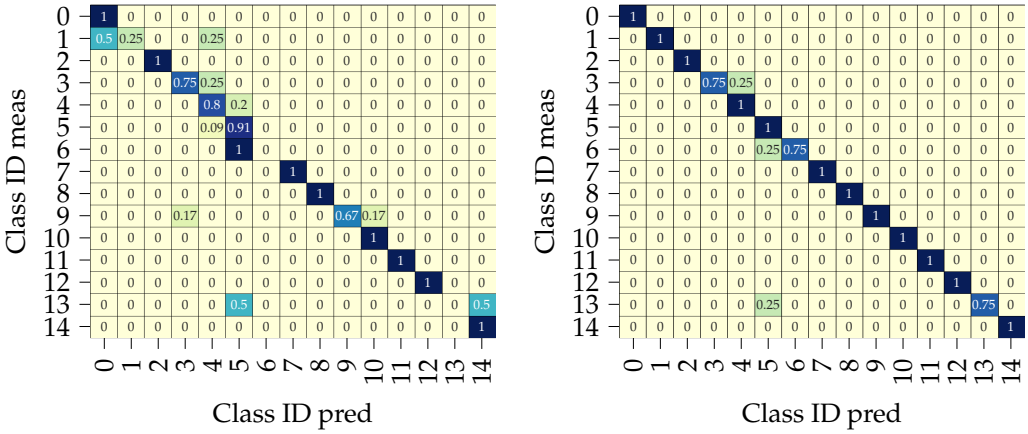
(**a**) CCT top 1 accuracy (overall 83.12 %). (**b**) CCT top 2 accuracy (overall 96.10 %).

Figure A1. Test data classification results of the CCT-PREC model in the form of normalized confusion matrices, evaluating the predicted precipitation classes (Class ID pred) with respect to the actual, measured value (Class ID meas).



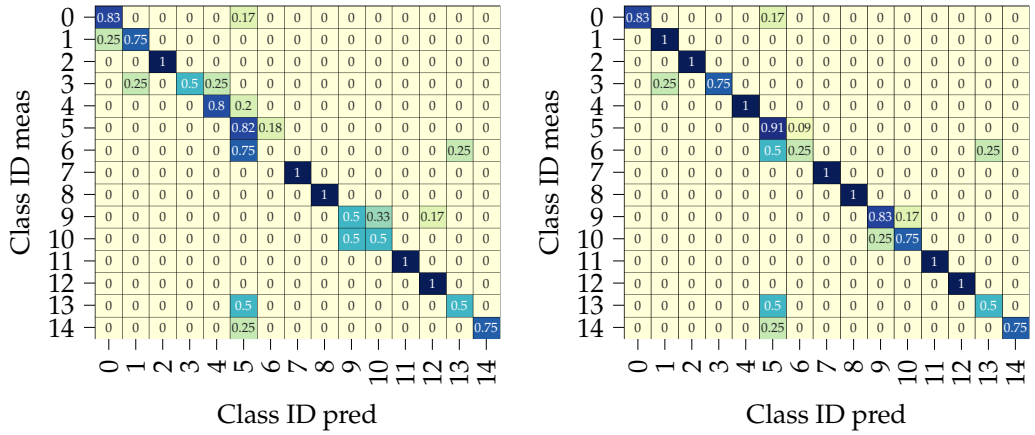
(**a**) ViViT top 1 accuracy (overall 79.22 %). (**b**) ViViT top 2 accuracy (overall 93.51 %).

Figure A2. Test data classification results of the ViViT-PREC model in the form of normalized confusion matrices, evaluating the predicted precipitation classes (Class ID pred) with respect to the actual, measured value (Class ID meas).



(**a**) ResNet3D top1 accuracy (overall 74.02 %). (**b**) ResNet3D top 2 accuracy (overall 88.31 %).

Figure A3. Test data classification results of the ResNet3D-PREC model in the form of normalized confusion matrices, evaluating the predicted precipitation classes (Class ID pred) with respect to the actual, measured value (Class ID meas).



(**a**) Conv-LSTM top1 accuracy (overall 64.93 %). (**b**) Conv-LSTM top 2 accuracy (overall 92.21 %).

Figure A4. Test data classification results of the Conv-LSTM-PREC model in the form of normalized confusion matrices, evaluating the predicted precipitation classes (Class ID pred) with respect to the actual, measured value (Class ID meas).

Appendix C. Computational Resources

The training and corresponding experiments were conducted on a Linux OS machine with a AMD Ryzen Threadripper 3970X 32-Core CPU @ 3.7 GHz and two NVIDIA GeForce RTX 2080 Ti (11 GB).

References

1. Goelles, T.; Schlager, B.; Muckenhuber, S. Fault Detection, Isolation, Identification and Recovery (FDIIR) Methods for Automotive Perception Sensors Including a Detailed Literature Survey for Lidar. *Sensors* **2020**, *20*, 3662. [CrossRef] [PubMed]
2. Kettelgerdes, M.; Hirmer, T.; Hillmann, T.; Erdogan, H.; Wunderle, E.; Elger, G. Accelerated Real-Life Testing of Automotive LiDAR Sensors as Enabler for In-Field Condition Monitoring. In Proceedings of the Symposium Elektronik und Systemintegration. Hochschule Landshut/Cluster Mikrosystemtechnik, Landshut, Germany, 17 April 2024; Volume 4, pp. 87–98.
3. Kettelgerdes, M.; Hillmann, T.; Hirmer, T.; Erdogan, H.; Wunderle, B.; Elger, G. Accelerated Real-Life (ARL) Testing and Characterization of Automotive LiDAR Sensors to facilitate the Development and Validation of Enhanced Sensor Models. *arXiv* **2023**, arXiv:2312.04229.
4. Strasser, A.; Stelzer, P.; Steger, C.; Druml, N. Enabling Live State-of-Health Monitoring for a Safety-Critical Automotive LiDAR System. In Proceedings of the SAS—2020 IEEE Sensors Applications Symposium, Piscataway, NJ, USA, 9–11 March 2020; pp. 1–6. [CrossRef]

5. Bijelic, M.; Gruber, T.; Ritter, W. A Benchmark for Lidar Sensors in Fog: Is Detection Breaking Down? In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 760–767. [CrossRef]
6. Hasirlioglu, S.; Riener, A.; Huber, W.; Wintersberger, P. Effects of exhaust gases on laser scanner data quality at low ambient temperatures. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 1708–1713. [CrossRef]
7. Jokela, M.; Kutila, M.; Pyykönen, P. Testing and Validation of Automotive Point-Cloud Sensors in Adverse Weather Conditions. *Appl. Sci.* **2019**, *9*, 2341. [CrossRef]
8. Kutila, M.; Pyykonen, P.; Holzhuter, H.; Colomb, M.; Duthon, P. Automotive LiDAR performance verification in fog and rain. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 1695–1701. [CrossRef]
9. Montalban, K.; Reymann, C.; Atchuthan, D.; Dupouy, P.E.; Riviere, N.; Lacroix, S. A Quantitative Analysis of Point Clouds from Automotive Lidars Exposed to Artificial Rain and Fog. *Atmosphere* **2021**, *12*, 738. [CrossRef]
10. Mayra, A.; Hietala, E.; Kutila, M.; Pyykonen, P. Spectral attenuation in low visibility artificial fog: Experimental study and comparison to literature models. In Proceedings of the 2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 7–9 September 2017; pp. 303–308. [CrossRef]
11. Daniel, L.; Phippen, D.; Hoare, E.; Stove, A.; Cherniakov, M.; Gashinova, M. Low-THz Radar, Lidar and Optical Imaging through Artificially Generated Fog. In Proceedings of the International Conference on Radar Systems (Radar 2017), Murrayfield Stadium, Edinburgh, 24–27 October 2022; Institution of Engineering and Technology: Stevenage, UK, 2017. [CrossRef]
12. Hasirlioglu, S.; Kamann, A.; Doric, I.; Brandmeier, T. Test methodology for rain influence on automotive surround sensors. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 2242–2247. [CrossRef]
13. Kettelgerdes, M.; Elger, G. In-Field Measurement and Methodology for Modeling and Validation of Precipitation Effects on Solid-State LiDAR Sensors. *IEEE J. Radio Freq. Identif.* **2023**, *7*, 192–202. [CrossRef]
14. Linnhoff, C.; Hofrichter, K.; Elster, L.; Rosenberger, P.; Winner, H. Measuring the Influence of Environmental Conditions on Automotive Lidar Sensors. *Sensors* **2022**, *22*, 5266. [CrossRef] [PubMed]
15. Heinzler, R.; Schindler, P.; Seekircher, J.; Ritter, W.; Stork, W. *Weather Influence and Classification with Automotive Lidar Sensors*; IEEE: Piscataway, NJ, USA, 2019; pp. 1527–1534. [CrossRef]
16. Rasshofer, R.H.; Spies, M.; Spies, H. Influences of weather phenomena on automotive laser radar systems. *Adv. Radio Sci.* **2011**, *9*, 49–60. [CrossRef]
17. Kashinath, S.A.; Mostafa, S.A.; Mustapha, A.; Mahdin, H.; Lim, D.; Mahmoud, M.A.; Mohammed, M.A.; Al-Rimy, B.A.S.; Fudzee, M.F.M.; Yang, T.J. Review of Data Fusion Methods for Real-Time and Multi-Sensor Traffic Flow Analysis. *IEEE Access* **2021**, *9*, 51258–51276. [CrossRef]
18. Kettelgerdes, M.; Pandey, A.; Unruh, D.; Erdogan, H.; Wunderle, B.; Elger, G. Automotive LiDAR Based Precipitation State Estimation Using Physics Informed Spatio-Temporal 3D Convolutional Neural Networks (PIST-CNN). In Proceedings of the 2023 29th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Queenstown, New Zealand, 21–24 November 2023; pp. 1–6. [CrossRef]
19. Carnahan, R.L. *Federal Meteorological Handbook No. 2: Surface Synoptic Codes: FCM-H2_1988*; US Department of Commerce, Office of the Federal Coordinator for Meteorological Services and Supporting Research: Washington, DC, USA, 1988.
20. Chaabani, H.; Werghi, N.; Kamoun, F.; Taha, B.; Outay, F.; Yasar, A.U.H. Estimating meteorological visibility range under foggy weather conditions: A deep learning approach. *Procedia Comput. Sci.* **2018**, *141*, 478–483. [CrossRef]
21. Vaibhav, V.; Konda, K.R.; Kondapalli, C.; Praveen, K.; Kondoju, B. Real-time fog visibility range estimation for autonomous driving applications. In Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 20–23 September 2020; pp. 1–6. [CrossRef]
22. Dhananjaya, M.M.; Kumar, V.R.; Yogamani, S. Weather and Light Level Classification for Autonomous Driving: Dataset, Baseline and Active Learning. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; pp. 2816–2821. [CrossRef]
23. Abu-Alrub, N.; Abu-Shaqra, A.; Rawashdeh, N.A. Compact CNN-based road weather condition detection by grayscale image band for ADAS. In Proceedings of the Autonomous Systems: Sensors, Processing and Security for Ground, Air, Sea and Space Vehicles and Infrastructure 2022, Bellingham, DC, USA, 6 June 2022; Dudzik, M.C., Jameson, S.M., Axenson, T.J., Eds.; Proceedings of SPIE; p. 20. [CrossRef]
24. Vargas Rivero, J.R.; Gerbich, T.; Teiluf, V.; Buschardt, B.; Chen, J. Weather Classification Using an Automotive LIDAR Sensor Based on Detections on Asphalt and Atmosphere. *Sensors* **2020**, *20*, 4306. [CrossRef]
25. Sebastian, G.; Vattem, T.; Lukic, L.; Burgy, C.; Schumann, T. RangeWeatherNet for LiDAR-only weather and road condition classification. In Proceedings of the 2021 IEEE Intelligent Vehicles Symposium (IV), Nagoya, Japan, 11–17 July 2021; pp. 777–784. [CrossRef]
26. Wu, J.; Ma, B.; Wang, D.; Zhang, Q.; Liu, J.; Wang, Y.; Ma, G. Weather Classification for Lidar based on Deep Learning. In Proceedings of the SAE International400 Commonwealth Drive, Shanghai, China, 22 December 2022; SAE Technical Paper Series. [CrossRef]

27. Da Silva, M.P.; Carneiro, D.; Fernandes, J.; Texeira, L.F. MobileWeatherNet for LiDAR-Only Weather Estimation. In Proceedings of the IJCNN 2023 Conference Proceedings, Gold Coast, Australia, 18–23 June 2023; pp. 1–8. [CrossRef]
28. Pereira, C.; Cruz, R.P.M.; Fernandes, J.N.D.; Pinto, J.R.; Cardoso, J.S. Weather and Meteorological Optical Range Classification for Autonomous Driving. *IEEE Trans. Intell. Veh.* 2024, *early access*. [CrossRef]
29. Colomb, M.; Hirech, K.; André, P.; Boreux, J.J.; Lacôte, P.; Dufour, J. An innovative artificial fog production device improved in the European project "FOG". *Atmos. Res.* **2008**, *87*, 242–251. [CrossRef]
30. Bijelic, M.; Gruber, T.; Mannan, F.; Kraus, F.; Ritter, W.; Dietmayer, K.; Heide, F. Seeing through Fog without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020.
31. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *arxiv* **2017**, arXiv:1706.03762.
32. Hasirlioglu, S.; Riener, A. Introduction to rain and fog attenuation on automotive surround sensors. In Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 16–19 October 2017; pp. 1–7. [CrossRef]
33. Kilic, V.; Hegde, D.; Sindagi, V.; Cooper, A.B.; Foster, M.A.; Patel, V.M. Lidar Light Scattering Augmentation (LISA): Physics-based Simulation of Adverse Weather Conditions for 3D Object Detection. *arXiv* **2021**, arXiv:2107.07004.
34. *THIES LNM 5.4110.xx.x00 Manual*; THies Clima: Göttingen, Germany, 2021.
35. Ryu, J.; Song, H.J.; Sohn, B.J.; Liu, C. Global Distribution of Three Types of Drop Size Distribution Representing Heavy Rainfall From GPM/DPR Measurements. *Geophys. Res. Lett.* **2021**, *48*, e2020GL090871. [CrossRef]
36. *Handbook Radiowave Propagation Information for Desining Terrestrial Point-to-Point Links*, 2008th ed.; ITU: Geneva, Switzerland, 2009.
37. Smith, J.A.; Hui, E.; Steiner, M.; Baeck, M.L.; Krajewski, W.F.; Ntelekos, A.A. Variability of rainfall rate and raindrop size distributions in heavy rain. *Water Resour. Res.* **2009**, *45*, W04430. [CrossRef]
38. Yu, T.; Joshil, S.S.; Chandrasekar, V.; Xiau, H. Snowfall Rate Estimation Based On Disdrometer During ICE-POP. In Proceedings of the 33rd International Union of Radio Science General Assembly and Scientific Symposium (URSI GASS), Rome, Italy, 29 August–5 September 2020.
39. Acharya, R. Tropospheric impairments: Measurements and mitigation. In *Satellite Signal Propagation, Impairments and Mitigation*; Elsevier: Amsterdam, The Netherlands, 2017; pp. 195–245. [CrossRef]
40. Halldórsson, T.; Langerholc, J. Geometrical form factors for the lidar function. *Appl. Opt.* **1978**, *17*, 240–244. [CrossRef] [PubMed]
41. Sassen, K.; Dodd, G.C. Lidar crossover function and misalignment effects. *Appl. Opt.* **1982**, *21*, 3162–3165. [CrossRef]
42. Chai, J.; Zeng, H.; Li, A.; Ngai, E.W.T. Deep learning in computer vision: A critical review of emerging techniques and application scenarios. *Mach. Learn. Appl.* **2021**, *6*, 100134. [CrossRef]
43. Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. A Survey on Vision Transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 87–110. [CrossRef]
44. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
45. Shi, X.; Chen, Z.; Wang, H.; Yeung, D.Y.; Wong, W.K.; Woo, W.C. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. *arXiv* **2015**, arXiv:1506.04214v2.
46. Tran, D.; Wang, H.; Torresani, L.; Ray, J.; LeCun, Y.; Paluri, M. A Closer Look at Spatiotemporal Convolutions for Action Recognition. *arXiv* **2018**, arXiv:1711.11248.
47. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Imagerivero Recognition. *arXiv* **2015**, arXiv:1512.03385
48. Hara, K.; Kataoka, H.; Satoh, Y. Towards Good Practice for Action Recognition with Spatiotemporal 3D Convolutions. In Proceedings of the 2018 24th International Conference on Pattern Recognition (ICPR), Beijing, China, 20–24 August 2018; pp. 2516–2521. [CrossRef]
49. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16 x 16 Words: Transformers for Image Recognition at Scale. *arXiv* **2021**, arXiv:2010.11929.
50. Beyer, L.; Zhai, X.; Kolesnikov, A. Better plain ViT baselines for ImageNet-1k. *arXiv* **2022**, arXiv:2205.01580.
51. Wang, P. Visual Transformer Pytorch Model. Available online: https://github.com/lucidrains/vit-pytorch (accessed on 2 May 2024).
52. Arnab, A.; Dehghani, M.; Heigold, G.; Sun, C.; Lučić, M.; Schmid, C. ViViT: A Video Vision Transformer. *arxiv* **2021**, arXiv:2103.15691.
53. Deng, J.; Dong, W.; Socher, R.; Li, L.J.; Li, K.; Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
54. Hassani, A.; Walton, S.; Shah, N.; Abuduweili, A.; Li, J.; Shi, H. Escaping the Big Data Paradigm with Compact Transformers. *arXiv* **2022**, arXiv:2104.05704.
55. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.
56. Zhang, J.; Karimireddy, S.P.; Veit, A.; Kim, S.; Reddi, S.J.; Kumar, S.; Sra, S. Why are Adaptive Methods Good for Attention Models? *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 15383–15393
57. Loshchilov, I.; Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv* **2016**, arXiv:1608.03983.
58. Zhuang, B.; Liu, J.; Pan, Z.; He, H.; Weng, Y.; Shen, C. A Survey on Efficient Training of Transformers. *arXiv* **2023**, arXiv:2302.01107.
59. Cox, D.R. The Regression Analysis of Binary Sequences. *J. R. Stat. Soc. Ser. Stat. Methodol.* **1958**, *20*, 215–232. [CrossRef]

60. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for Hyper-Parameter Optimization. In Proceedings of the 25th Annual Conference on Neural Information Processing Systems 2011, Granada, Spain, 12–15 December 2011; Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2011; Volume 24.
61. Cui, Y.; Xu, H.; Wu, J.; Sun, Y.; Zhao, J. Automatic Vehicle Tracking With Roadside LiDAR Data for the Connected-Vehicles System. *IEEE Intell. Syst.* **2019**, *34*, 44–51. [CrossRef]
62. D'Arco, M.; Fratelli, L.; Graber, G.; Guerritore, M. Detection and Tracking of Moving Objects Using a Roadside LiDAR System. *IEEE Instrum. Meas. Mag.* **2024**, *27*, 49–56. [CrossRef]
63. Lin, C.; Wang, Y.; Gong, B.; Liu, H. Vehicle detection and tracking using low-channel roadside LiDAR. *Measurement* **2023**, *218*, 113159. [CrossRef]
64. Research, G. ViTs Models Configuration. 2024. Available online: https://github.com/SHI-Labs/Compact-Transformers (accessed on 1 May 2024).
65. 3D ResNet PyTorch Model. Available online: https://github.com/kenshohara/3D-ResNets-PyTorch/blob/master/models/resnet.py (accessed on 2 May 2024).