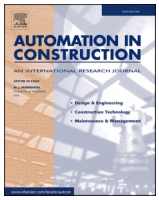
Automation in Construction 164 (2024) 105419

Contents lists available at ScienceDirect

Automation in Construction

journal homepage: www.elsevier.com/locate/autcon



Uncertainty-aware point cloud segmentation for infrastructure projects using Bayesian deep learning

Hristo Vassilev *, Marius Laska, Jörg Blankenbach

Geodetic Institute and Chair for Computing in Civil Engineering & Geo Information Systems, RWTH Aachen University, Mies-van-der-Rohe-Str. 1, Aachen 52074, North Rhine-Westphalia, Germany

ARTICLE INFO

Keywords:
Uncertainty estimation
Semantic segmentation
Point cloud
BIM
Digital twin
Bridge
Bayesian deep learning

ABSTRACT

Reliable traffic infrastructure is a key factor for any country's economy. However, aging bridges often require renovation or reconstruction. Promising approaches for enhancing asset management and cost reduction are digital twins and predictive maintenance strategies. However, the creation of geometric-semantic as-is models as a basis for digital twins currently involves labor-intensive manual data capture and modeling. Modern deep learning models, such as Kernel Point Convolution (KPConv) show promising results in reducing the time needed to create digital twins by semantically segmenting point cloud data but have so far been hindered by the lack of a reliable quality measure, which can predict when the model's prediction can be trusted. This is especially viable in the construction industry, where objects and sensors may vary widely between projects and contractors.

In this work, we present Bayesian neural networks, implemented through Variational Inference and Monte-Carlo dropout as approaches to conducting inference with KPConv, which show improvements in the confidence estimation and out-of-domain (OOD) detection in the scans of typical infrastructure point clouds. When confronted with different domain shifts to the test data, such as a change of scanning device and introduction of unseen classes the proposed model showed a 25.3% decrease in expected calibration error (ECE) and a 4.82 increase in OOD detection in terms of outlier-intersection-over-union (O-IoU) on average with respect to a deterministic baseline.

1. Introduction

Around the world, the maintenance of existing bridges has emerged as an increasing financial challenge. In Germany, the vast majority of bridges are built between the 60s and the 80s of last century and are showing more age-related deterioration than was anticipated at the time of their construction i.a. due to changes in traffic demand [1]. Planning and estimating costs for maintenance or retrofitting underdimensioned bridges to meet new building regulations requires careful surveys and inspection, which are often time- and cost-intensive [2]. In order to reduce the maintenance costs of bridges preventative maintenance strategies are becoming increasingly popular. These often make use of a Building Information Model (BIM) together with a non-destructive digital inspection system to enable the creation of a digital twin, where information throughout the life-cycle of the project can be exchanged between various stakeholders and be used to create a cost-effective maintenance strategy by means of multi-objective optimization [3].

In this context, the digital twin is composed of a geometric-semantic model (hereinafter referred to as the BIM model), which allows the linking of a variety of embedded measurement devices in a common data environment [4]. The effort inherent to the modeling of existing bridges is often the issue, which hinders the widespread adoption of digital twins in the operation and maintenance phases. This process is done either by adapting outdated, inaccurate or incomplete as-planned information or by the time-consuming process of manually processing raw survey data, such as point clouds (Scan-to-BIM) for a more accurate "as-is" BIM model [5]. Automating the process of deriving a BIM model from various sources of point cloud data is therefore currently a highly researched topic.

Most existing Scan-to-BIM methods rely on a rule-based approach and expert knowledge to directly detect common building parts with a known geometric parametrization for example - walls, floors, columns [6,7], pipes [8,9] and scaffolding [10] can be detected from raw point cloud data. Similarly, rule-based approaches have been adapted to the bridge domain, where irregular shapes and non-straight alignments

* Corresponding author.
E-mail address: hristo.vassilev@gia.rwth-aachen.de (H. Vassilev).

https://doi.org/10.1016/j.autcon.2024.105419
Received 14 July 2023; Received in revised form 12 January 2024; Accepted 5 April 2024
Available online 8 May 2024

have to be accounted for [11,12]. By knowing the possible topology of the structure these methods can achieve good performance but are usually constrained to a certain type of bridge.

A common approach is to break down the process into two steps: (1) semantic segmentation and (2) a subsequent geometry fitting algorithm, which uses the segmented point cloud generate the geometric components of a parametric BIM model [13–15]. Approaches such as deep learning have recently shown great potential in improving the generalization of semantic segmentation by replacing previously hand-crafted features with a learnable feature extractor, implemented as multiple hidden layers or operators such as convolution. This enhances the performance of machine learning models by allowing them to learn a potentially richer set of features but due to the complexity of their inner mechanisms removes the ability to interpret and control the quality of the output.

Despite their success, most current works on deep learning only focus on improving the in-distribution accuracy of the segmentation; that is, their performance on data, which is similar to the training data. In practice, however, the data acquisition step leads to an unavoidable distribution shift to the inputs of the machine learning model. The reasons for this can be a change in the point cloud acquisition method or the introduction of novel semantic classes. As this shift becomes larger, eventually leading to out-of-distribution (OOD) data, the interpretation of the output probability of a neural network has been known to become increasingly unreliable [16]. Failing to quantify the uncertainty in such cases accurately has so far been one of the major reasons hindering the widespread adoption of deep learning models due to the large possibility of heterogeneous object classes found around construction projects and the common need to change to a different surveying approach (e.g. photogrammetry, mobile or terrestrial laser scanning) depending on the specific use case.

Not being able to trust the prediction of a deep learning model hinders its potential for full automation, as it introduces an additional step of manually checking the segmentation quality. In contrast, uncertainty-aware models can guide users to select cases in which the semantic segmentation might have failed. Additionally, uncertainty estimates can be beneficial to up-stream processes, such as model selection, or down-stream processing steps such as geometry fitting by for example weighting points based on their confidence level or by reducing the level of detail. Therefore, there is a need for a framework for semantic point cloud segmentation, which can provide an accurate confidence metric, while maintaining similar overall accuracy.

Although probabilistic machine learning models can provide a solution to this problem up until recently their formulation for deep neural networks was considered too computationally expensive. However, with some simplifications recent approaches are able to produce scalable implementations of Bayesian neural networks (BNNs), such as Monte-Carlo Dropout [17], Variational inference [18,19] and normalizing flows [20] which show promising improvements in the uncertainty estimation capabilities of different deep learning methods. Already there have been improvements for tasks such as image classification and semantic segmentation [21], however, there have been so far very limited works, which study uncertainty in the domain of semantic point cloud segmentation. Additionally, probabilistic models can potentially deliver an estimate about the source of their uncertainty - whether it stems from the bad quality of the data (aleatoric uncertainty) or whether it comes from shortcomings of the model (epistemic uncertainty). This quality can be crucial for decision-making processes, such as investing resources in improving the machine learning model or cleaning and augmenting the dataset, depending on which type of uncertainty is predominant.

Our research aims to establish a joint quality measure for the gathered point cloud data and the resulting semantic segmentation by providing per-point semantic uncertainty. The novelty of our work lies in our approach to capturing model uncertainty, by means of applying BNNs to point cloud segmentation in BIM, which as we show improves the calibration of the uncertainty metric. Specifically, we achieve this by defining the neural network's weights as stochastic parameters and training them using Bayesian inference. We compare the performance of these trained models with a deterministic baseline on in-domain and data under covariate shift by measuring the expectation of the calibration error (ECE). Additionally, their ability to detect out-of-distribution (OOD) data, is assessed by observing the predictive entropy on novel semantic classes. While our focus remains on improving model calibration and OOD detection we provide some insight into the ability of the Bayesian models to separate model (epistemic) and data (aleatoric) uncertainty.

In summary, this contribution aims to answer the following research questions in the context of point cloud semantic segmentation:

- Does a BNN improve uncertainty estimation on data with a covariate shift?
- Does a BNN improve uncertainty estimation on novel classes?
- Can a BNN separate epistemic from aleatoric uncertainty?

In Section 2 we discuss prior work in the field of uncertainty estimation in neural networks and subsequently the state-of-the-art deep learning methods for point cloud segmentation. In Section 3 we then introduce our methodology. In Section 4 we elaborate on our experiments. Finally, in Section 5 we interpret the results and draw our conclusions.

2. Related work

2.1. Semantic modeling

In the following, a summary is provided of recent developments in automated bridge modeling from point clouds as well as a broad overview of deep learning models with a focus on their application in the infrastructure domain. Based on this we deduce an optimal deep learning architecture, which we later upgrade to a Bayesian neural network and compare our results.

There have already been various approaches, which aim to automatically extract bridge BIM models from point clouds. Currently, the majority rely on assumptions about the data, which allow the extraction of low-level features, such as fitted planes [11], geometric primitives [15] and surfaces [22] etc. (*bottom-up approaches*) or instead break down a complex segmentation problem into a series of sequential segmentation tasks, which are by themselves simpler and can be solved by heuristic-based algorithms (*top-down approaches*). Some examples include the slice-based classification [13] which focuses on RC concrete slab and beam-slab bridges and [12] which detect structural components in steel girder bridges with a curved shape based on their relative orientation to an extracted alignment. Such rule-based methods are strongly bound to the type of bridge for which they are designed, which makes their extension to additional semantic classes and geometric shapes difficult.

Contrary to the aforementioned approaches, the application domain of learning-based methods is theoretically only bound by the amount of provided training data and model expressiveness, which has allowed especially deep learning to achieve very promising results in 3D computer vision and recently in the field of geometric digital twins. However, unlike more common data formats, such as image data, point clouds lack some essential characteristics for deep learning, such as 1) a specific order of the points in memory and 2) a spatial structure to define interactions between neighbouring points. In the following we briefly go through some of the models, relevant for our study based on their solutions to these issues. A more comprehensive review of further neural network architectures and their applications is provided in [23].

To resolve the ordering issue point clouds are sometimes transformed into an intermediate representation such as images through projection [24] or voxels through discretization [25,26]. However, both methods lead to the loss of information, which can be crucial in detailed or

complex scans. Other methods operate directly on the points - PointNet [27] was the first network to achieve this by using max pooling as a permutation invariant method for feature aggregation. This method alone only extracts information about the global shape of the point cloud, while lacking the ability to derive the local features of each point. This was shown in [2], where a multi-scale local descriptor followed by a classical artificial neural network was able to out-preform PointNet in the semantic segmentation of RC bridges. Alternatively, to alleviate this problem, PointNet can be applied recursively in a progressively larger neighborhoods around each point, leading to hierarchical features, that capture both local and global geometry [28]. Furthermore, Hu et al. [29] show that a simplified PointNet backbone coupled with a multi-view convolutional neural network (CNN) operating on drone images can produce a rich embedding of a bridge, which is then decoded recursively by binary tree network, resulting in a reconstructed 3D-Model of a cable-stayed bridge. Other architectures, such as DGCNN [30] address the problem of local feature extraction by defining a graph with edges connecting each point to its k-nearest neighbors, allowing graph convolution methods. The graph is recomputed dynamically after each layer, allowing for a global receptive field, which can be beneficial for large structures such as bridges: [31] extend the DGCNN architecture, by introducing a hierarchical feature extraction at different ranges from the query point, thereby improving the segmentation performance of fine-grained but tall structures, such as electric poles around railway bridges.

Point convolution is another family of recent architectures, which apply continuous convolutions directly to a 3D point set. Inspired by image convolutions, KPConv [32] defines the kernel explicitly as a set of kernel points with either a flexible or rigid configuration. The kernel points are analogous to the well-known filters of image CNNs and carry the weights of the network. This method leads to strong local features, due to the ability of KPConv to better encode local surface deformation and details, which is supported by results on different indoor and outdoor datasets. Although recent developments have pushed the performance boundary even further, these often rely on strategies for increasing the amount of model parameters of existing models [33] or applying memory-intensive operations, such as global self-attention [34–36], which require expensive hardware. Therefore, the focus of this work has remained on instead improving the quality of uncertainty estimation of the KPConv architecture, which strikes a balance between performance and computational demand.

2.2. Uncertainty estimation in deep learning

In the context of machine learning, uncertainty refers to a lack of confidence in the model's prediction. It can have different causes, such as limited or insufficient data, exaggerated model complexity and inherent randomness in the underlying data distribution. Therefore, uncertainty can be categorized into two main types:

Aleatoric uncertainty: Also known as data uncertainty or irreducible uncertainty. It is caused by the presence of inherent randomness in the observed data. Since it captures the noise and variability in the data itself, it cannot be reduced even after acquiring more data or using a better model. For example, in image classification, aleatoric uncertainty can arise due to variations in lighting conditions, occlusions, or difference in viewpoints.

Epistemic uncertainty: Also known as model uncertainty or reducible uncertainty, epistemic uncertainty arises from uncertainty in the model's parameters and structure. It reflects the lack of knowledge or uncertainty in the model's representation of the underlying data distribution. Epistemic uncertainty can be reduced with more data, model improvements, or better training.

Quantifying uncertainty in machine learning models is important, as it allows for better decision-making and risk assessment. Gawlikowski et al. [37] provide an extensive review of methods for estimating the uncertainty in deep neural networks, which can be divided into deterministic methods, Bayesian neural networks (BNNs), Ensamble methods and test time augmentation. Different implementation details and theoretical background of BNNs are further elaborated in [38]. To aid the understanding of the goals and challenges of implementing BNNs in the point cloud domain the following section will provide a compact version of the theoretical background.

2.3. Bayesian methods for neural network uncertainty

Bayesian neural networks (BNNs) are adapted machine learning models, which allow capturing the uncertainty in their parameters. This is achieved by putting a prior distribution over the weights $\mathbf{W}$ instead of using fixed deterministic values. Then given a training set of features $\mathbf{X} = \{x_1, \ldots, x_N\}$, and labels $\mathbf{Y} = \{y_1, \ldots, y_N\}$ one can use Bayesian inference [39] to calculate the posterior distribution of the weights $P(\mathbf{W} \mid \mathbf{X}, \mathbf{Y})$:

$$P(\mathbf{W}|\mathbf{X}, \mathbf{Y}) = \frac{P(\mathbf{X}, \mathbf{Y})P(\mathbf{W})}{\int_{\mathbf{W}} P(\mathbf{X}, \mathbf{Y})P(\mathbf{W})\, d\mathbf{W}} \tag{1}$$

With a learned posterior distribution the model can be used to make predictions $\widehat{y}$ from a given test item $\widehat{x}$ by calculating the expectation of that posterior $P(\widehat{y}|\widehat{x}) = \mathbb{E}_{P(\mathbf{W}|\mathbf{X},\mathbf{Y})}[P(\widehat{y}|\widehat{x}, \mathbf{W})]$. However the problem lies in the denominator of Eq. 1, which can be though of as integrating over all possible configurations of weights $\mathbf{W}$, that explain the data $(\mathbf{X}, \mathbf{Y})$. In the case of a deep neural network this is a computationally intractable problem due to the presence of multiple hidden layers and the large dimensionality of the inputs. For this reason different approaches have been developed, which reduce the computation overhead but still provide an approximation to the posterior distribution of $\mathbf{W}$. Amongst others these include MC-Dropout, Variational inference and ensembles. The following section will briefly discuss the most relevant approaches for implementing a BNN.

2.3.1. Approximation with MC-dropout

Dropout is a well-known regularization technique in deep learning, which involves using a fixed Bernoulli distribution to randomly switch off the neurons in an artificial neural network (ANN) during training. Gal et al. [17] show that without changes in the training or optimisation stage, the simple action of applying the same dropout procedure during the inference stage can be interpreted mathematically as a well known probabilistic model, a Gaussian Process, when multiple passes of the same inputs are averaged out. This mechanism has so far been very influential as already many modern deep neural networks make use of dropout, which makes the extension to a BNN comparatively simple as it involves no additional trainable parameters. Some applicataions involve the uncertainty estimation in computer vision tasks such as image segmentation [40] and stereo depth regression [41].

2.3.2. Approximation through a gaussian distribution

Another class of methods attempts to model the distribution of the weights as a multivariate Gaussian (MVG) either by Laplace Approximation [19] or Variational Inference [18].

Variational Inference (VI) [42] models the parameters of the MVG $\theta = \{\mu; \Sigma\}$ over the network weights explicitly and trains them by minimizing the Kullback-Leibler divergence (KL-divergence) between a parametric approximation (surrogate posterior) and the true Bayesian posterior. Often a mean-field approximation is used [18], which for simplicity models the MVG with a diagonal covariance, merely doubling the number of trainable parameters. This way local uncertainty surrounding the mean of each weight can be captured, while at the same time ensuring an efficient computation. This technique has also seen applications in uncertainty estimation in computer vision, some of which include [43], where VI was used to reduce the overconfidence of the GoogLeNet architecture for image classification.

Nevertheless, mean-field VI is sometimes critiqued to be an over-simplification, as it fails to model depth-wise and layer-wise

correlations. Partial solutions sometimes come in the form of Laplace Approximations, which use a deterministic approach to calculate the mean of the weights and use the second-order derivative (Hessian matrix) of the weights with respect to the loss function to derive the covariance matrix of the MVG. However, since calculating the Hessian of the weights in an ANN is also an intractable problem, some simplifications are often used, such as the Kronecker-Factored Approximate Curvature (KFAC) [44], which allows capturing some of the uncertainty correlations within the weights of a single layer. Implementations of KFAC can be found in libraries such as Backpack-for-Pytorch [45]. Furthermore, Kristiadi et al. have shown that applying Bayesian inference to only the last layer of an ANN already provides a good uncertainty estimation [19], which further simplifies the optimization problem.

Another argument against the MVG is that it isn't flexible enough to capture the true distribution, which may be multi-modal, as is the case with ANNs [46]. Even though the different modes all lead to approximately similar accuracy on the validation data, Fort et al. [46] have shown that by capturing the modes of the posterior with an ensemble of randomly initialized deterministic networks, different solutions emerge at each mode, which disagree with each other on the wrong predictions. This mechanism has been successfully used for uncertainty estimation in [47] and has been shown to provide a robust confidence interval. However, scalability is often an issue with ensemble models, as they require training and copying multiple copies of the ANN, which can prove cumbersome in large models such as KPConv.

Instead, the limited flexibility of the MVG is often addressed by transforming its probability density function through normalizing flows (NFs) [20]. NFs are a series of bijective transformations applied to the outputs of a sample produced with variational Bayes. Bijectors are invertible functions for which a tractable solution for the Jacobian exists in order to appropriately scale the posterior density, allowing them to be trained through backpropagation therefore potentially learning a multimodal distribution on the weights of the Network. Although due to their implementation complexity in point cloud models, NFs are out of scope for this study, they show a scalable alternative for increasing the expressiveness of the MVG and are a promising direction for improving the discussed method in the feature.

In summary, modeling and training a BNN is mainly a trade-off between the runtime efficiency of the network and the expressiveness of the weights' posterior distribution. Increasing the efficiency of a BNN is therefore often achieved at the expense of simplified uncertainty modeling. MC-dropout and mean-field VI are simple and scalable methods, which have shown great potential for improving the uncertainty estimation of deep learning methods.

2.4. Research gaps

So far the described methods demonstrate uncertainty estimation improvements in the image domain, while applications on point clouds are still relatively new: the work by Cortinhal et al. [24] introduces a projection-based convolutional BNN for increasing the robustness of real-time point cloud applications such as autonomous driving. Wang et al. [48] use mean-field VI to train their own CNN, which shows improvements such as the decrease in calibration error relative to a different reference architecture. Petschnigg et al. [49] train PointNet with mean-field VI and dropout to increase the accuracy of the prediction after filtering out uncertain results in the factory planning domain. To the best of our knowledge uncertainty estimation using the KPConv backbone has so far remained unexplored. Additionally, a more thorough study of different uncertainty sources, such as covariate shift and novel classes will benefit the Scan-to-BIM workflow as these remain underinvestigated in the domain of RC bridges.

3. Method

In this work, our main focus is improving the uncertainty prediction of the well-known Kernel Point Convolution architecture (KPConv) [32], while keeping the segmentation quality of the original model. We compare three variants - a deterministic network as a baseline, a VI-model, and an MC-dropout model. We perform two types of tests - 1) we test the calibration of the models on test data under different covariate shifts and 2) we use the entropy of the predictive distribution as a measure for detecting novel classes. In the following Section 3.1 we first discuss the implementations of the three models. We then go over to Section 3.2 and Section 3.3 where we discuss the inference process and the interpretation of the result with different metrics and uncertainty types.

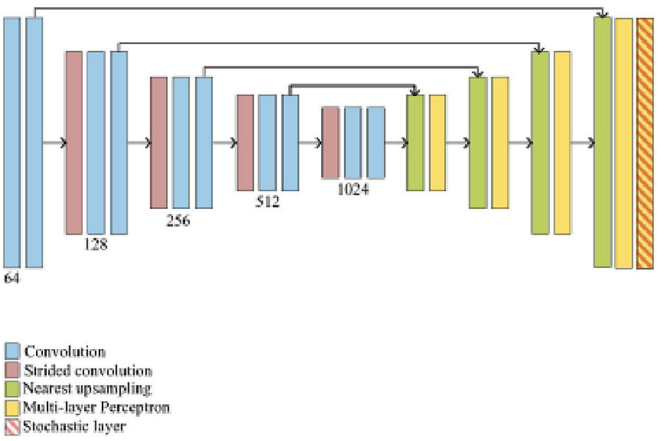


Fig. 1. KPConv architecture with stochastic weights in the final layer.

3.1. KPConv variants

3.1.1. Deterministic KPConv

A starting point for our work is the KPConv architecture [32], which will be briefly summarized in the following section.

Kernel point convolution is a method for extracting patterns in point clouds. These are then often used for classification and in the fully convolutional setting for semantic segmentation. Specifically, its convolutional operator can be interpreted as adapted image convolution, with the difference that it uses a kernel that directly operates with the Cartesian coordinates of the neighborhood surrounding a specific point of a point cloud. The kernel consists of a number of kernel points, each of which is associated with a weight matrix. The feature of each point in the neighborhood is multiplied by a weighted average of the kernel matrices, with the weight depending on the distance to the kernel point.

In order to extract high-level features and enlarge the network's receptive field a strided convolution layer is applied at the beginning of each block. This layer reduces the spatial resolution of the query points, while at the same time increasing the number of dimensions of the weight matrix (Fig. 1). Repeating the process multiple times results in the extraction of progressively higher-level features and the reduction of the amount of overall points. In order to recover the lost point cloud resolution a series of decoder blocks is then used, which concatenate high-resolution information from the encoder to high-level features in the decoder and use multi-layer perceptron (MLP) on this information to determine the features of the high-resolution point cloud. Finally, the last MLP layer operates on a cloud with the original resolution and performs point-wise classification resulting in the segmented point cloud.

A practical limitation of KPConv is that it requires the loading of the entire scene and its derived features into the GPU memory in order to execute the computation. This effectively limits the size of the scene, which can be processed at a given time depending on the available hardware. Therefore, for large scenes instead of processing the entire scene, multiple spherical sub-regions are sampled and processed in

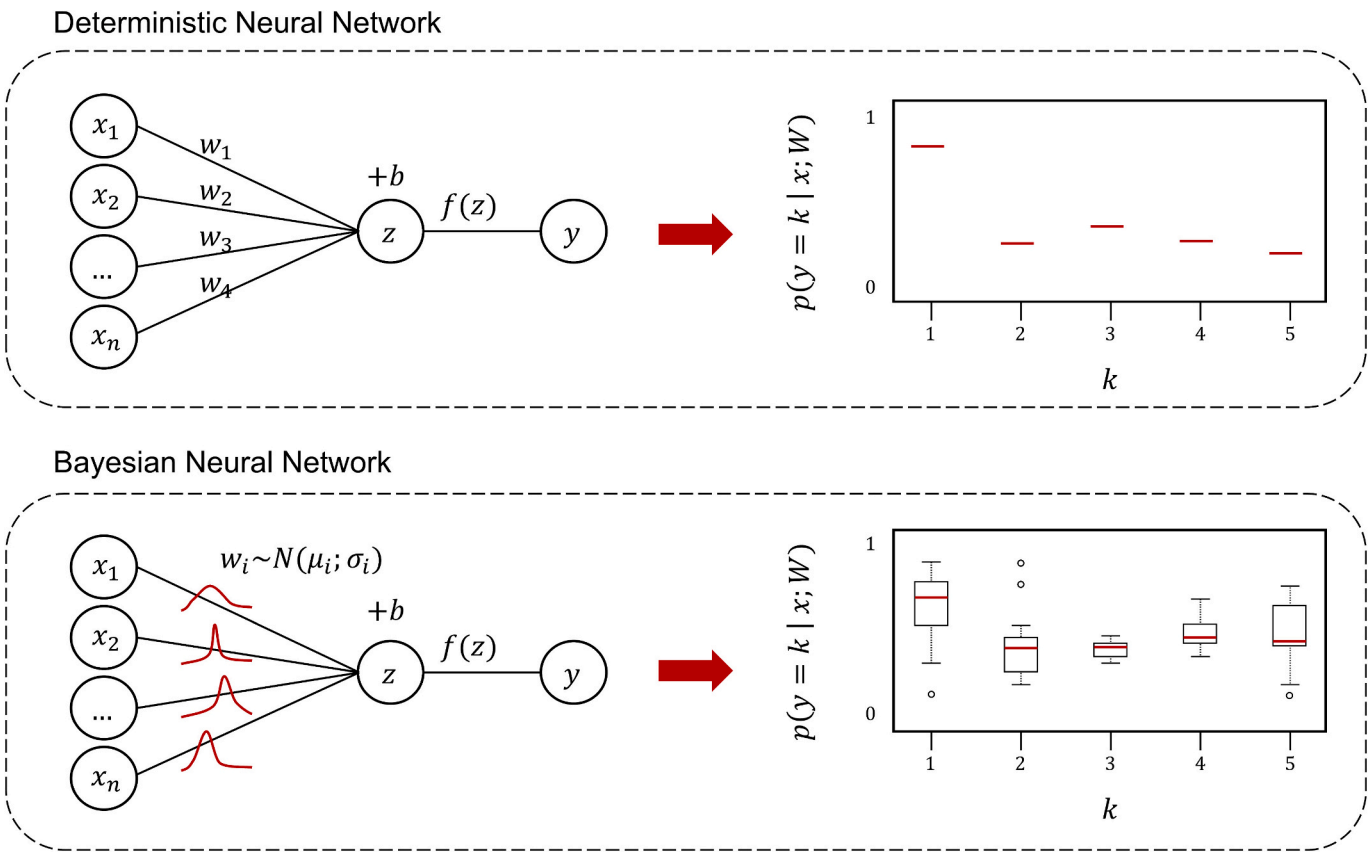


Fig. 2. Simplified principle of Bayesian neural networks in comparison to conventional deterministic DNNs. Here k stands for the k-th class in the dataset, while $P(y = k|\mathbf{x}, \mathbf{W})$ corresponds to the probability of the class being the correct one. The figure on the right represents a box plot with some arbitrary percentile.

isolation. The results are then combined by averaging the predictions of overlapping spheres. The sampling can be executed in a semi-random fashion during training, such that points from less-represented classes are sampled more often than highly-represented classes in order to reduce any inherent bias in the dataset.

3.1.2. *KPConv with variational inference*

Given the existing KPConv architecture we replace the last two MLP blocks, which we call head, with a Bayesian layer. The rest of the network is kept in its original deterministic form and trained using L2 regularization. The choice to limit the inclusion of the Bayesian layers to the classifier only is motivated empirically, as this configuration yielded the best uncertainty results. Further inclusion of Bayesian layers led to diminishing returns in terms of uncertainty estimation while leading to a loss in accuracy.

Rather than fixed values, the parameters of the head are sampled from a distribution. The training procedure, which estimates the parameters of this distribution is described in the following. Firstly, we put a prior distribution $P(\mathbf{W})$ over the weights. Similarly to [18] we choose a scale mixture of two Gaussian distributions with mean zero and standard deviations of $\sigma_1 = e^2$ and $\sigma_2 = e^{-2}$ and mixture probability of $P(\sigma = \sigma_1) = 0.75$. As discussed, given a dataset $\mathscr{D} = \{\mathbf{X}, \mathbf{Y}\}$ no tractable solution for the true posterior distribution $P(\mathbf{W}|\mathscr{D})$ exists we instead optimize the parameters $\theta = \{\mu; \sum\}$ of a variational posterior $q(\mathbf{W}|\theta)$ by minimizing the Kullback-Leibler divergence between the variational posterior and the true posterior:

$$\theta^{\star} = \underset{\theta}{argmin} = \mathrm{KL}[q(\mathbf{W}|\theta)\|P(\mathbf{W})\,] \tag{2}$$

However, this objective still requires the computation of the true posterior. Instead, from Eq. 2 an equivalent optimization objective, called the evidence lower bound (ELBO) can be derived as in [37]:

$$\mathrm{ELBO} = \mathbb{E}_{q(\mathbf{W}|\theta)}(logP(\mathscr{D}|\mathbf{W})\,) - \mathrm{KL}[q(w|\theta)\|P(\mathbf{W})\,] \tag{3}$$

Here the ELBO can be decomposed into the negative log-likelihood of the dataset and the KL-divergence between the variational posterior and the prior. The second term can be interpreted as a complexity cost, which penalizes the network if the weight distribution diverges from the prior. Since we train the network using mini-batches instead of the entire dataset the likelihood is expressed as an average of all data points in the mini-batch with size B. In order to keep the proportion between the negative log-likelihood and the complexity equivalent to the derived loss in Eq. 3 the complexity cost is scaled by the number of batches in the dataset N. It should be noted that in the case of KPConv, the batches are not created by splitting the point clouds into equal batches but instead by randomly sampling multiple sub-regions from the point cloud as discussed previously. This makes the choice of N rather arbitrary, as the amount of sampled batches before an epoch is considered "completed" is not clearly defined. Nevertheless, we keep the number of batches constant throughout our experiments and introduce an additional scaling coefficient λ, which we discuss later in our Experiments (Section 4.3).

$$L(\theta|D) = \lambda \cdot \frac{1}{N}\mathrm{KL}[q(\mathbf{W}|\theta)\|P(\mathbf{W})\,] - \frac{1}{B}\sum_{j=1}^{B} logP\left(y_j, x_j|\mathbf{W}\right) \tag{4}$$

With the rescaled loss in Eq. 4 we now need a way to propagate gradients to the parameters θ of the variational posterior, which is usually not possible from stochastic samples. Instead we use the reparametrization trick [18], which splits the weight matrix into a non-trainable noise source $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$ and a deterministic shift $\boldsymbol{\mu}$ and scale $\boldsymbol{\sigma}$, where the scale is parametrized as $\boldsymbol{\sigma} = log(1 + exp(\boldsymbol{\rho})\,)$ to avoid negative values. A sample i from the posterior can be drawn as

$$\mathbf{w}^{(i)} = \boldsymbol{\mu} + log(1 + exp(\boldsymbol{\rho})\,) \circ \boldsymbol{\epsilon}^{(i)} \tag{5}$$

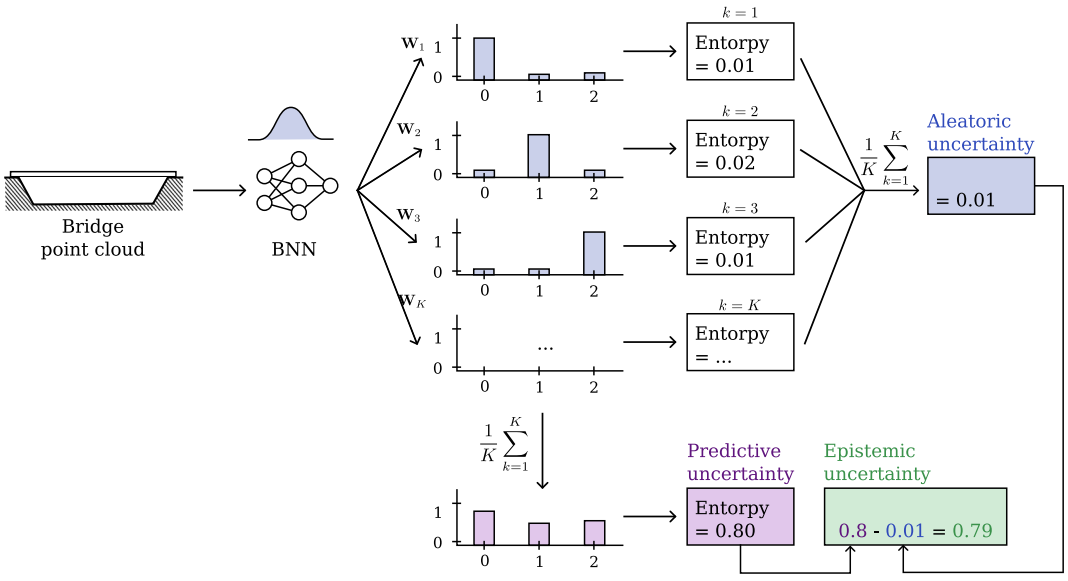


Fig. 3. Overview of the computation process of different uncertainty types.

where ∘ stands for element-wise multiplication.

From here we train the network through back-propagation using stochastic gradient descent with momentum. Although for the deterministic network we use a learning rate of 1×10^{-2}, we find that reducing the initial learning rate for the BNN to 5×10^{-5} was necessary in order to avoid unstable training and reach convergence. We then furthermore apply a decay factor of 0.98 on the learning rate at each epoch.

3.1.3. KPConv with MC-dropout

As discussed, a simple alternative method for introducing stochasticity into the predictions of a neural network is by extending the use of dropout layers to the inference phase. This is normally convenient for networks, which already use dropout during training since it avoids the use of parameterized distributions. For the purpose of enabling the comparison to our VI-Model we therefore create an alternative KPConv model, in which dropout with a 0.5 probability is implemented in the last two fully connected layers of the network. During testing this block is sampled multiple times for a given input to acquire the BNN prediction. The following section describes in detail how this result is then evaluated.

3.2. Inference with Bayesian neural networks

In order to calculate the distribution over the predictions of a BNN from an unseen sample $\boldsymbol{x}^*$ we need to marginalize out the weights. This can be achieved by averaging the predictions of multiple passes through the Bayesian layers. In order to calculate the distribution over the predictions, we perform K number of forward passes through the network for a given test input $\boldsymbol{x}^*$. Therefore, K results at the end of the network are aggregated and saved for further analysis, such as calculation of the epistemic (Eq. 12), the aleatoric (Eq. 11) and predictive uncertainty (Eq. 10). The actual prediction of the network is the mean prediction [37]:

$$P(\hat{\boldsymbol{y}}^*|\boldsymbol{x}^*\mathscr{D}) = \frac{1}{K}\sum_{k=1}^{K}\text{softmax}(f(\boldsymbol{x}^*, \boldsymbol{w}_k)) \qquad \text{with} \qquad \boldsymbol{w} \underset{i.d.d.}{\sim} q(\boldsymbol{w}|\theta) \tag{6}$$

and $f(\cdot)$ being the Bayesian neural network. Because the output is passed through a softmax function the resulting vector $P(\hat{\boldsymbol{y}}^*|\boldsymbol{x}^*\mathscr{D})$ can be considered a categorical distribution over the number of classes in the dataset from which the chosen class $\boldsymbol{y}^*$ is the largest component and the size of the component is interpreted as the predictive confidence (See Eq. 8). Additionally, the posterior distribution gives access to further information about the prediction, such as the decomposition of the predictive uncertainty into epistemic and aleatoric (See Fig. 2).

3.3. Metrics

In the following we discuss the metrics used in our experiments. We start with conventional metrics for semantic segmentation quality and then explain the measures of uncertainty. Finally, the aggregated result of the BNN is used to separate the overall predictive uncertainty into epistemic and aleatoric.

3.3.1. Mean intersection over union

The mean intersection over union (mIoU), also known as the Jaccard index is a common indicator for the performance of segmentation tasks. The calculation for the IoU for one class is defined as:

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{7}$$

where TP stands for True Positive, FP stands for false positive and FN stands for false negative predictions. The IoU is calculated for each class separately and finally averaged to provide the mIoU.

The main benefit of the mIoU metric in comparison to metrics such as the mean categorical accuracy (mAcc) is that it compensates for class disbalance in a scene. For example a segmentation result of a scene with few points belonging to class “traffic sign” in comparison to many points belonging to class “road” can display good mAcc even if all points are classified as road. In contrast the Jaccard coefficient of class “traffic sign” will be zero, indicating a faulty segmentation.

3.3.2. Calibration error

The predictions of a machine learning model are often filtered by setting a threshold on the confidence of the chosen class as discussed above. In this context the calibration error is a useful metric, since it allows us to see, whether a model can accurately predict its confidence on test data, which might be different from the training data. This is accomplished by performing inference on the test set and binning the predictions in terms of confidence. In addition the the accuracy of the predictions in each bin is calculated. The model is said to be well calibrated if the average bin confidence $\text{conf}(b_m)$ is equal to the average accuracy $\text{acc}(b_m)$ in the bin m. A weighted average of the difference in all bins gives us the Expected Calibration Error (ECE):

$$\mathrm{conf}(\hat{y}^*) = max\left(\hat{y}^*_m\right)_{m=1}^{M} \qquad \text{for } M \in \mathbb{N} \text{ classes} \tag{8}$$

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|b_m|}{N} |\mathrm{acc}(b_m) - \mathrm{conf}(b_m)| \tag{9}$$

Additionally, we can visualize the calibration of a model by plotting the confidence of the predictions in each bin against their accuracy. This plot is often supplemented with a hypothetical perfectly calibrated model for reference, that is a model for which $\mathrm{acc}(b_m) = \mathrm{conf}(b_m)$ for each bin, resulting in a diagonal line. Models for which the calibration curve is below this diagonal can be described as under-confident, while plots above the diagonal represent overconfident models (See e.g. Fig. 6).

3.3.3. Uncertainty types

The uncertainty associated with a prediction y^* from a given samples x^* is referred to as predictive uncertainty. However, this uncertainty may have different sources, as described in 2.2, and can therefore be further broken down into aleatoric (data) and epistemic (model) uncertainty. Whereas the data uncertainty informs us about noise or ambiguity in the data, the model uncertainty describes uncertainty in the parameters of the model, due to e.g. lack of data. In the following, we describe our method of estimating the different types of uncertainty based on multiple passes through the BNN. An overview is of the process is provided in Fig. 3.

Predictive uncertainty A metric often used to describe the predictive uncertainty is the Entropy over the marginalized categorical distribution [49]:

$$U_{\mathrm{pred}} \approx \sum_{\hat{y}^* \in \{1,\ldots,m\}} \left(\frac{1}{K} \sum_{k=1}^{K} p\left(\hat{y}^*_m | w_k, x^*\right) \right) \cdot log \left(\frac{1}{K} \sum_{k=1}^{K} p\left(\hat{y}^*_m | w_k, x^*\right) \right) \tag{10}$$

In Eq. 10 $\hat{y}^*_m$ is the m-th class from the categorical distribution, while w_k is the k-th sample from the BNNs posterior. Note that this metric can still be applied to both the prediction from a deterministic network, as well as the marginalized prediction of the BNN. Compared to the confidence discussed above, which simply takes the maximum probability over all classes, the predictive distribution incorporates information about the entire categorical distribution. Since the entropy is defined in the range $\mathrm{H} \in \{0; inf\}$ we can optionally normalize it for the range $\{0; 1\}$ using $\mathrm{H}_{\mathrm{norm}} = 1 - \frac{\mathrm{H}}{log(N)}$ where N stands for the number of classes in the dataset. This normalization step makes models comparable regardless of N.

Aleatoric uncertainty The aleatoric uncertainty can be calculated by finding the average entropy of each pass through the BNN [49]:

$$U_{\mathrm{alea}} \approx \mathbb{E}_w[\mathbb{H}[\hat{y}|w_k, x]] \approx$$
$$\approx -\frac{1}{K} \sum_{k=1}^{K} \sum_{\hat{y} \in \{1,\ldots,m\}} p(\hat{y}_m | w_k, x) \cdot log(p(\hat{y}_m | w_k, x)) \tag{11}$$

Epistemic uncertainty Finally, by finding the difference between the predictive and aleatoric uncertainties, we get the epistemic uncertainty [49]:

$$U_{\mathrm{ep}} = U_{\mathrm{pred}} - U_{\mathrm{alea}} \tag{12}$$

In general the epistemic uncertainty is captured by high variation in the passes of the BNN, which can be interpreted as the activation of neurons in the network with a high standard deviation.

4. Experiments

So far we have discussed methods of constructing and evaluating a BNN. In the following, we elaborate on our experiments, which have the goal of showing whether a BNN can provide a more accurate uncertainty estimate than the deterministic baseline. It is essential, that this is performed on test data, which are different from the training data, as this is where normally the softmax estimate of a regular ANN degrades. We first focus our experiments on test data under covariate shift (Section 4.3) and then more specifically as the introduction of novel classes (Section 4.4).

Table 1
Details about the data used in this study.

Dataset	Sensor type	Has colors	Has intensities	#Semantic classes	#Points
S3DIS	Depth Camera	True	False	13	273.54 Mil
Karman	TLS	True	False	13	546.46 Mil
Infrastructure-MLS	Vehicle-mounted MLS	False	False	18	125.92 Mil
Bridge-TLS	TLS	False	False	18	259.96 Mil

4.1. Datasets

In this work we mainly on data in the domain of bridges and traffic infrastructure. Specifically, we concentrate on reinforced concrete (RC) bridges, with them being the most widely built type in Germany [1]. However, to validate and compare our method to feature works, we additionally provide results on the common indoor benchmark dataset S3DIS as well as an alternative indoor dataset captured with a terrestrial laser scanner (See Table 1).

4.1.1. Stanford 3D indoor scene (S3DIS) dataset

The Stanford 3D - Indoor Scene Dataset (S3DIS)(Armeni et al., 2016) is the first work to tackle the challenge of semantic parsing of an entire building at once. The data is hierarchically structured into floors, rooms and finally - object instances with the corresponding labels of one of 13 semantic classes: ceiling, floor, wall, beam, column, window, door, table, chair, sofa, bookcase, board or clutter. The rooms come from 3 different buildings, each with a different architectural style and appearance. The room types consist mainly of office areas, educational and exhibition spaces, conference rooms restrooms, lobbies, and small hallways. The data is captured by a stationary Matterport 3D-camera [50].

4.1.2. Kármán-indoor dataset

The Kármán indoor dataset was created after scanning 66 rooms over 3 floors of the Kármán-Auditorium at the RWTH Aachen with a geodetic terrestrial laser scanner (Riegl VZ-400), capturing color and intensity information in addition to the geometry. In the context of this paper the dataset was made compatible with the S3DIS dataset by using the same semantic classes as mentioned above. The room uses include library, lecture room, office, storage, study room, staircase and hallway. Although the building typology is mostly similar to the one used in 3DIS there are some relevant content differences in this dataset worth mentioning: 1) the library is a large area containing free-standing book shelves not normally found in the S3DIS dataset, 2) the hallways are wider and are furnished with benches and desks to serve as lounges. Most importantly, the terrestrial laser scanner, while providing a much more accurate geometrical representation, suffers from many occlusions generated by furniture, such as tables.

4.1.3. Infrastructure MLS dataset

The Infrastructure MLS dataset was captured with a vehicle-mounted mobile laser scanner and consists of 70 highway scenes in the German state of North Rhein-Westphalia, from which 10 include a concrete bridge and 60 represent sign bridges. We manually segment the point clouds into 18 semantic classes, with the choice of classes following two

Table 2
Classes from the infrastructure domain. Entries marked with * are artificially removed for the OOD detection experiment.

Index	Short name	IFC class	IFC predefined type
0	Unlabeled	None	
1	assembly_abutment	IfcElementAssembly	ABUTMENT
2	**assembly_signbridge***	IfcElementAssembly	SIGNALASSEMBLY skip
3	**barrier_flexible***	IfcRailing	GUARDRAIL
4	**barrier_railing***	IfcRailing	_
5	**barrier_rigid***	IfcRailing	GUARDRAIL
6	beam_girder	IfcBeam	GIRDER_SEGMENT
7	beam_piercap	IfcBeam	PIERRCAP
8	column_pier	IfcColumn	PIERSTEM
9	footing	IfcFooting	_
10	natural_ground	_	_
11	**pipe***	IfcPipe	_
12	slab_deck	IfcSlab	FLOOR
13	slab_paving	IfcSlab	PAVING
14	slab_sidewalk	IfcSlab	SIDEWALK
15	stair	IfcStair	_
16	**traffic_*sign****	_	_
17	wall_noisebarrier	IfcWall	_

goals: firstly, the classes had to represent objects, which are recognizable from the point cloud geometry alone, since other features such as colors or intensities weren't available. On the other hand the choice of classes had to conform where possible to the Industry Foundation Standard (IFC) in order to enable industry-conform downstream processes such as geometry reconstruction in the feature (See Table 2).

The entire dataset was divided into 10 sets, each consisting of one bridge scene and 6 sign bridge scenes, therefore retaining the scene contents similar throughout the sets. The full training split for the experiments consists of 8 such sets, while the validation split contains the remaining 2 sets.

4.1.4. Bridge TLS dataset

In addition to the MLS data we present a set of three bridges captured with a TLS, each containing increasing amount of difficulties with regards to noise and novel classes. The main purpose of testing on this dataset is to represent a series of possible, although knowingly not optimal domain shifts to the training data (See Section 4.3), which may be present in surveying practice. An overview of the point clouds can be seen in Fig. 5.

Bridge 1 - The first bridge contains a common RC-bridge with a beam cross-section over a divided highway in Germany. The bridge is mostly similar to bridges contained in the MLS-dataset, with the exception of the sensor now being the Riegl VZ-400 Terrestrial laser scanner.

Bridge 2 - The second scene contains a bridge with a similar topology but a longer, curved span. Additionally, both sides include a shoulder lane. The scene is filled with many artifacts caused by moving vehicles, as the scanning was performed, while the bridge was in operation.

Bridge 3 - The third scene contains an RC bridge with a slab cross-section, which is novel with respect to the training data. The bridge spans over multiple railway tracks. The scan contains many novelties, such as the train tracks, overhead catenary fixed to the bridge as well as multiple railway specific assets, such as signage, train wagons and various technical equipment.

Since no additional training is performed when transferring between the MLS and TLS datasets, measures had to be taken to ensure sufficient consistency. Firstly, the same labeling scheme was used as described in Table 2. Any novel classes were kept under 'unlabeled'. Additional features, which have now became available, such as intensity and color were hidde n from the model. Finally, the scans were filtered to only include those from below the bridge, as this is the case with the MLS dataset.

4.2. Run time

All of our experiments were conducted on a Desktop machine, equipped with NVIDIA RTX A5000 with 24GB of VRAM, utilizing an 11th Gen Intel Core i9–11,900, clocked at 2.50 GHz. As described in 3.1.1 our mini batch consists of a randomly sampled spherical sub-region of the point cloud with a radius 0f 1.2 m for indoor and 12 m for outdoor respectively, which ensured a large receptive field. To avoid memory issues with a large receptive field the point clouds needed to be sub-sampled with a grid of size 3 cm for indoor and 12 cm for outdoor. These hyper-parameters were determined empirically in a previous study with the goal of reaching optimal mIoU [51, p. 90]. The configuration resulted in a testing run time of 26.5 s for the MLS-Infrastructure dataset with the deterministic network. A naive implementation of the BNN resulted in a roughly linear run time increase relative to the number of samples made from the posterior distribution. Since we use 50 samples this led to an overall run time of 21.84 min or 1310 s. However, a simple optimization was made possible by caching the result of the deterministic part of the network and only performing the forward pass through the stochastic layers at the end. This reduced the computational overhead by roughly 5 times compared to the naive approach, making the optimized BNN approximately 10 times slower than the deterministic ANN when testing, with a run time of 4.24 min.

4.3. Uncertainty under covariate shift

As already mentioned, a classifier trained on a finite set of real data may fail to provide an accurate uncertainty estimate when the test data are sampled from a different distribution than the training data. However, BNNs have been proposed as a possible mitigation of this problem. To test this hypothesis we design our first test, in which the test data are subject to covariate shift, meaning a change to the distribution of the input data, without changing the composition of the output classes. As we do not directly influence the contents of the dataset, we instead sort

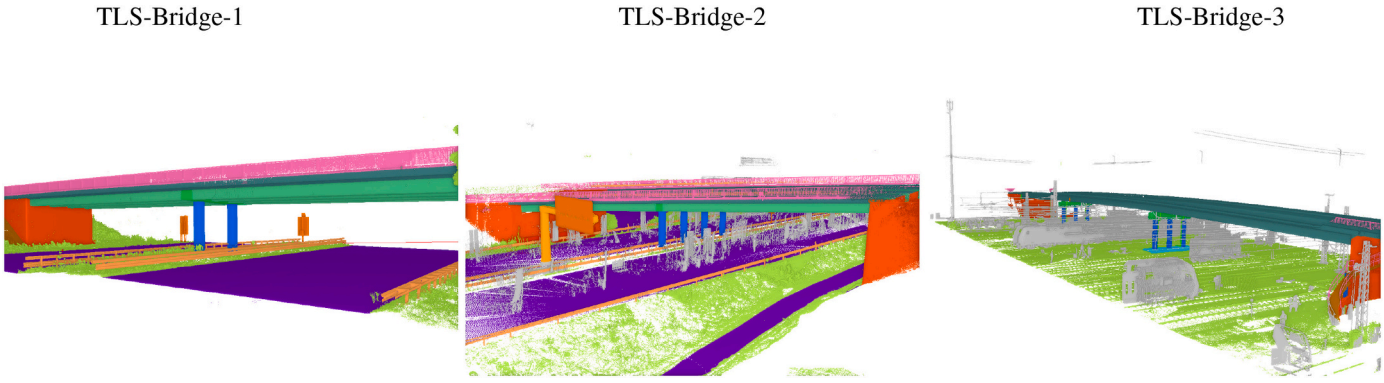


Fig. 5. The three TLS bridges. The colors represent semantic classes with gray being the laser-scanner artifacts.

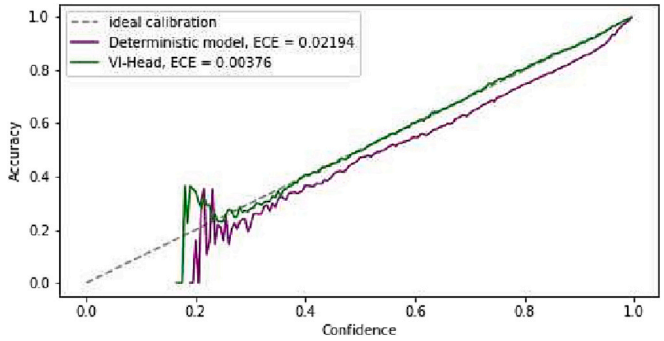


Fig. 6. Calibration plot on the S3DIS benchmark dataset, when testing on the Area-5 test split.

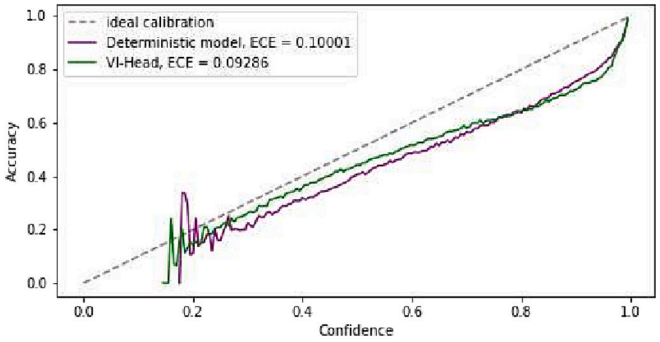


Fig. 7. Calibration plot on the Karman indoor dataset, with training done on the S3DIS dataset.

Table 3
Results of the models' performance with varying test data, while using the same training sets of S3DIS (top) and our Infrastructure-MLS (bottom) respectively.

Test set	Method	mAcc	mIoU	ECE	[a]ECE ± [%]
Training set: **S3DIS**					
S3DIS (Area 5)	Deterministic	89.3516	**65.8231**	0.02194	±0.00
	VI-Head	88.8985	65.5627	**0.00376**	-82.86
Kármán dataset	Deterministic	**73.3060**	**52.8520**	0.10001	±0.00
	VI-Head	69.6002	52.1709	**0.09286**	-7.15
Training set: **MLS bridge dataset**					
MLS: Validation set	Deterministic	**97.7979**	**78.7287**	0.01440	±0.00
	VI-Head	97.7856	77.5159	**0.00871**	-39.51
	MC-Dropout	97.6197	77.3454	0.01363	-5.35
TLS: Bridge 1	Deterministic	86.1599	66.9395	0.05437	±0.00
	VI-Head	**89.4061**	**67.5779**	0.02567	-52.79
	VI-Head (n=40)	86.3048	62.3455	**0.02129**	-60.84
	MC-Dropout	84.7915	61.0532	0.04368	-19.66
TLS: Bridge 2	Deterministic	**83.8583**	**63.7883**	0.03584	±0.00
	VI-Head	81.8590	59.2747	**0.03537**	-1.31
	MC-Dropout	83.0308	59.4402	0.06103	+70.28
TLS: Bridge 3	Deterministic	**70.8667**	**39.5248**	0.18887	±0.00
	VI-Head	69.0182	37.1133	**0.17383**	-7.96
	MC-Dropout	69.4275	38.4769	0.19281	+2.09

[a] The change in ECE is shown relative to a deterministic baseline for each test dataset.

the test scenes, described in the last section, by their mIoU and analyze the calibration error (ECE).

We start with the least significant change to the contents, which are the validation data of the S3DIS and MLS-Infrastructure datasets respectively. In this case, the sensors and semantic classes remain the same, meaning that this test should be the least challenging. We use this as a baseline for the comparison with further experiments, since in both cases the training and validation splits are composed of semantically similar objects and have been collected using the same sensor or device. We report the calibration of the model (ECE - EqEq. 9), calculated with a bin size of 0.005, which we choose to be small in order to provide an accurate metric. Additionally, we provide the mIoU (Eq. 7) in order to track any change in segmentation performance after applying the stochastic layers.

On the S3DIS dataset, we perform the usual procedure of testing on the Area-5 subset, while training on the remaining rooms. In the results, the deterministic KPConv showed signs of underconfidence and was outperformed by the VI model, which was able to output an almost perfect confidence score resulting in a 5.83-fold decrease in ECE (See Fig. 6 and Table 3) without any significant impact on the mIoU with respect to the deterministic baseline on our machine (See Table 3).

We observe a similar result on our MLS-Infrastructure dataset, where the VI-Head is able to mitigate some of the underconfidence, which is present with the deterministic model. This effect can be observed

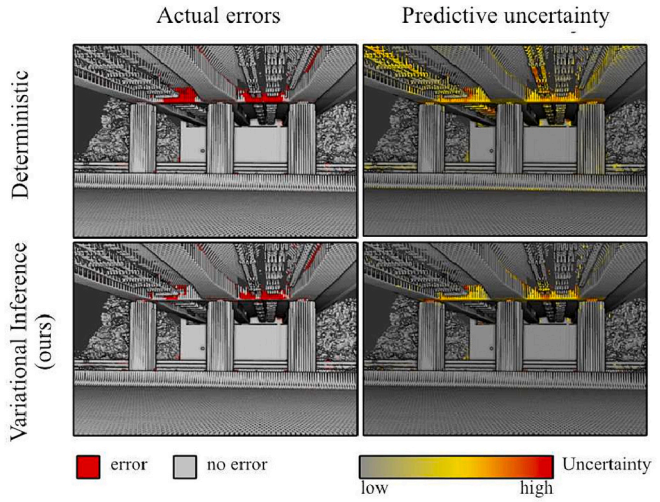


Fig. 8. Qualitative comparison of the segmentation done with the deterministic and Bayesian models.)

qualitatively in Fig. 8, in which the confidence (Eq. 8) is color coded and compared with the actual error.

Additionally, we can visualize the confidence of the model by color coding each point with the score resulting form Eq. 8 and comparing that to a visualization, in which only the errors in the segmentation are displayed in red. A well calibrated model should then be able to predict the error from the confidence. As can be seen in our example (Fig. 8) the under-confidence of the deterministic model marks the pipes underneath the bridge as uncertain even though the segmentation is correct. In this sense the VI-Head was able to recover some of the correctly labeled building parts, which would have otherwise been discarded as uncertain by the deterministic model.

In summary, it can be seen that the ECE on in-domain data is improved significantly by the VI-Head model. However, a drop of 1.21% of mIoU should be noted with respect to the deterministic model. Model, which can be attributed to changes in the training procedure.

In the next experiment we test the performance of the models, which we trained on the datasets captured by a mobile device (S3DIS, Infrastructure-MLS) on datasets captured by a TLS (Kármán-indoor and Bridge-TLS) in order to test their generalization capabilities of the models to other sensor types and data capturing conditions. The Kármán-indoor dataset proves challenging for both the deterministic and VI-Head models with the latter achieving a slight calibration improvement (Fig. 7). On Infrastructure data the VI-Head model achieves a slight increase of 0.63% mIoU and 2.11-fold decrease in ECE on the TLS-Bridge-1 (Table 3). Increasing the λ term in Eq. 4 by a factor of 40 can further improve calibration at the cost of a drop in segmentation

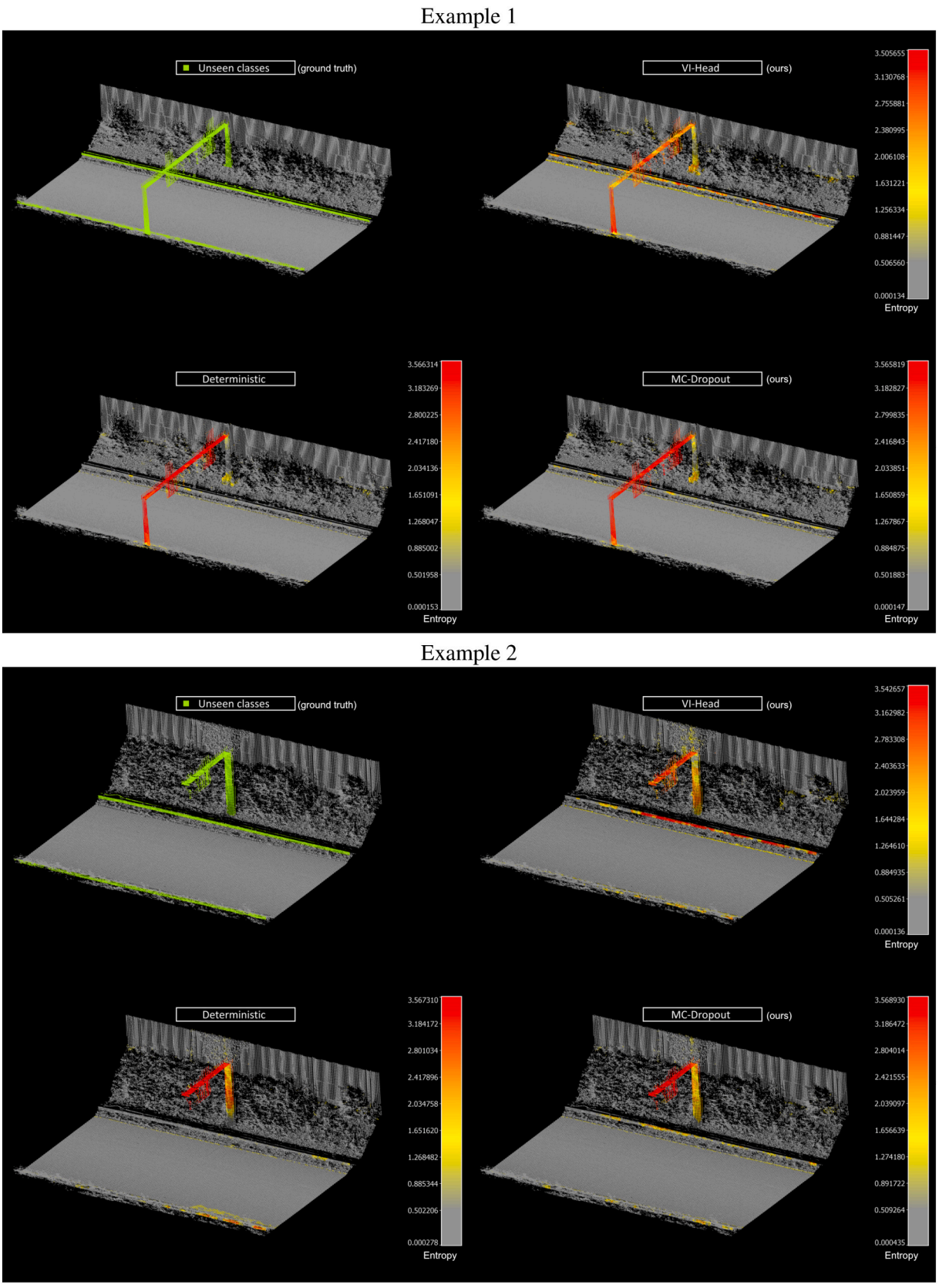


Fig. 9. Two sign bridge examples for OOD detection with training performed after hiding some of the classes (marked green in the first graphic). The remaining follow the color scheme: Red = high predictive entropy, Gray = low predictive entropy. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

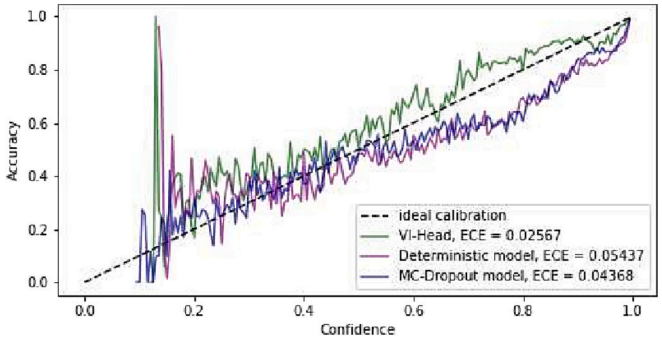


Fig. 10. Calibration plot while testing on TLS-Bridge-1 with the VI-Head model.

accuracy. By observing the calibration plot (Fig. 10) which visualizes the calibration error in terms of the distance from the ideal calibration line, we conclude that the deterministic model is under-confident, while the VI-Head most closely follows the ideal calibration. However, an improvement with the VI-Head is less apparent in the presence of strong laser scanning artifacts as is the case of the TLS-Bridge-2, where we see a decrease of 4,51% in mIoU and no difference in terms of ECE. With the MC-dropout model we observe small improvements in the ECE on the TLS-Bridge-1, which do not carry over to the TLS-Bridge-2. An overview of these results is provided in Fig. 11.

4.4. Out-of-distribution detection

Up until now we have discussed changes in the data distribution due to the acquisition process. However, a more extreme example is that of a

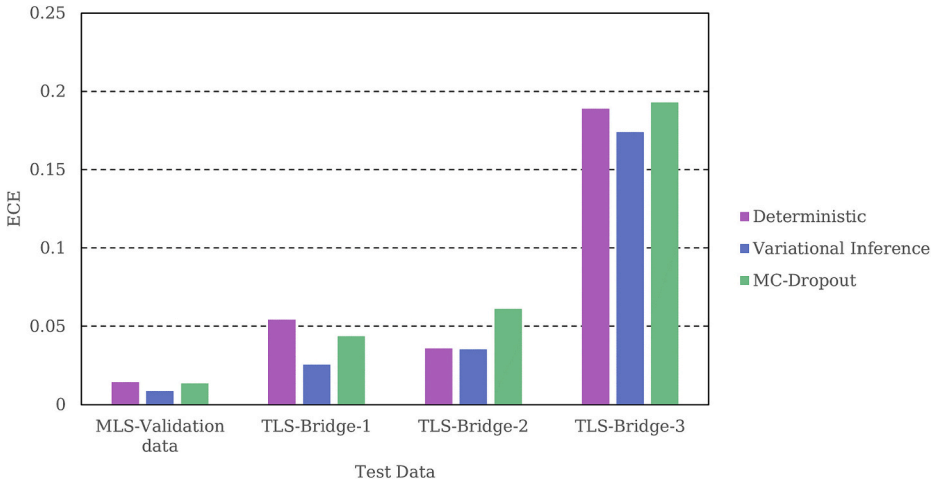


Fig. 11. Overview of calibration results with covariate shift.

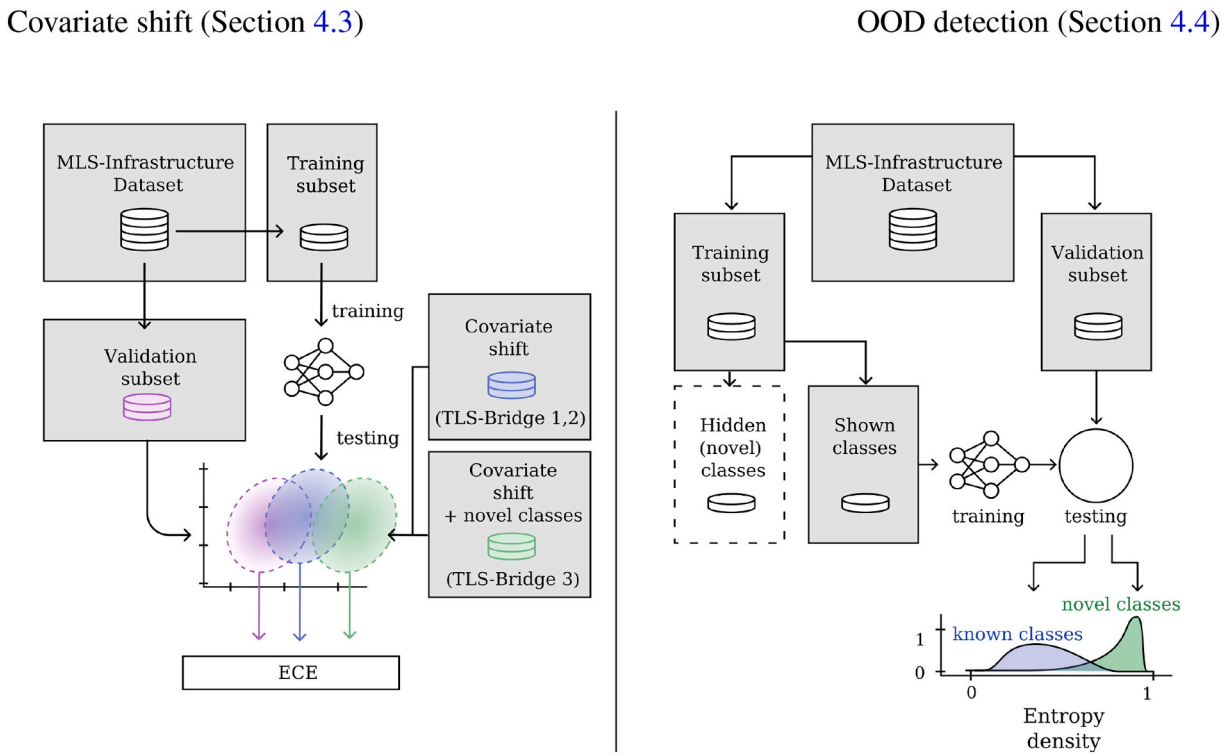


Fig. 4. Overview of our experiments workflow in Section 4.

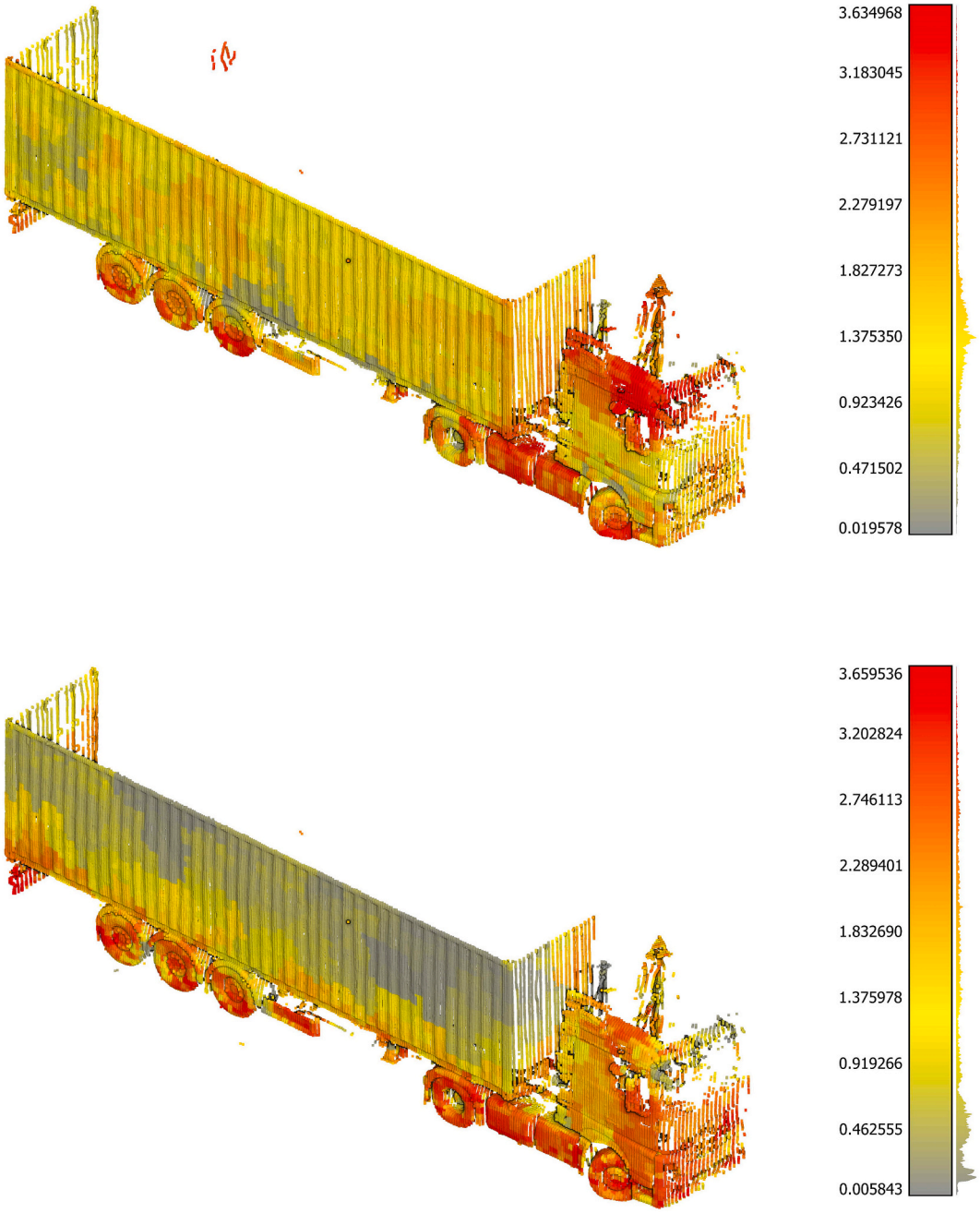


Fig. 12. Rare objects such as parked trucks were absent in the training data. Inference with VI-Head (top) and deterministic (bottom). The predictive entropy is used to color code the point clouds.

complete change in data content, such as the case of presenting semantic classes unseen during training. A desired quality of an uncertainty metric is the detection of such Out-of-Distribution (OOD) examples for the purpose of improving the robustness of the classifier and for active learning, that is the case in which the learning algorithm queries the user to selectively label the detected OOD examples to improve segmentation performance. In order to simulate such unseen OOD examples in our tests we artificially remove examples of chosen semantic classes (Table 2) in the scenes from the training split our MLS dataset but keep the scenes of the validation split unchanged. This workflow is illustrated in Fig. 4. By examining the predictive entropy 10 on predictions of the hidden class we observe that the MC-dropout results in the overall highest entropy (Fig. 13)

Although the overall uncertainty of the VI-Head model is lower on novel classes than the one provided by MC-Dropout, our qualitative study shows that it sometimes provides better coverage (Fig. 9) on rare objects such as parked trucks, which naturally don't occur in our training data (Fig. 12.). Additionally, to analyze the effectivness of the OOD-detection we perform the following test: we interpret the OOD detection as classification of OOD examples, where an example is classified as OOD, if its normalized predictive uncertainty (Eq. 10) is above 0.95. By comparing the result with the knowledge of which classes were hidden we can compute the outlier IoU (O-IoU). Over the entire dataset with hidden classes we observe the following results O-IoU results: Variational inference - 56.54%, Deterministic - 56.37% and MC-Dropout 61.19%. In summary, the MC-dropout provides the highest uncertainty on novel classes, which in turn results in the best OOD detection.

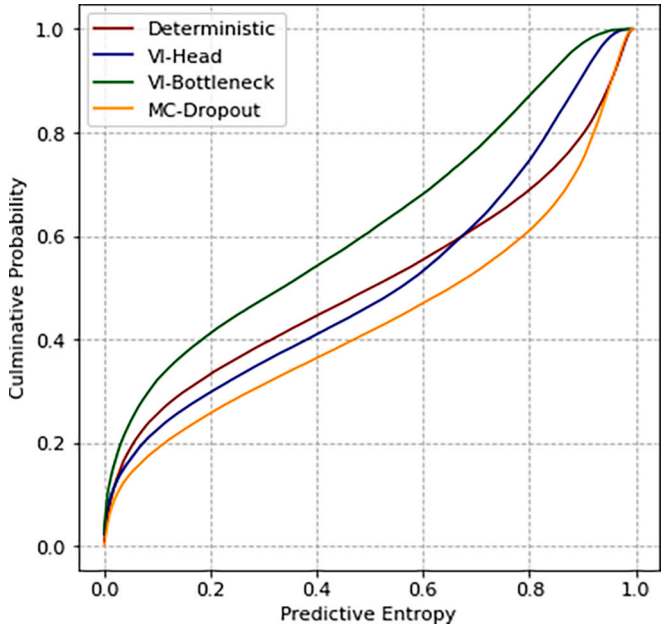


Fig. 13. Cumulative distribution of the normalized Entropy, when conducting inference on the hidden classes, as described in 4.4. Curves which lie lower indicate overall higher entropy.

4.5. Uncertainty classification

Determining whether the uncertainty of the prediction comes from the model or the quality of the input data is a useful feature, which can guide a practitioner in finding ways to improve the result. We explore the ability of the model to give an estimate for these two quantites in the following:

First, we focus on the validation data of the MLS-Infrastructure dataset (Fig. 15). We look at a bridge in particular and compare qualitatively the resulting uncertainties. The quantities are computed using the mechanisms described in Section 3.3.3. Specifically, we notice, that the aleatoric uncertainty is often concentrated on the pier caps, while the epistemic uncertainty spreads to the columns and pipes. Because of the concentration of the aleatoric uncertainty, we can conclude, that there there is noise present in the features produced by KPConv in that part of the bridge. Similarly, we analyze the Bridges from the TLS Dataset and notice an increase in aleatoric uncertainty with Bridge-2 (Fig. 14), which we attribute to the presence of strong artifacts, caused by moving cars.

However, we should be cautious to assume, that these cases are entirely due to inherent noise in the data since the Bayesian part of the network is only observing the already extracted features, which by itself happens in a deterministic way. This is one of the bigger limitations of our approach and will therefore be addressed in feature research.

5. Discussion

In this section we discuss the results of our experiments and point out the strengths and weaknesses of the presented methods.

We observe that the original deterministic implementation of KPConv suffers from underconfidence on various datasets. This is an undesirable quality if the aim is to detect errors in the segmentation, since too many correctly classified examples may be filtered out along with the errors.

Our experiments indicate that this problem of underconfidence is highly dependent on the similarity of the training and test sets: already on the validation sets, which are presumed to come from the same distribution as the training data, the Bayesian models show improvements in terms of ECE score with the VI-Head model performing best and the MC-Dropout showing marginal improvements with respect to the deterministic model. Testing on TLS data further increases the calibration error of the deterministic model, while the VI-Head again shows the best performance, although also being affected slightly negatively. Overall the VI-Head achieves a 25.3% improvement in ECE with respect to the deterministic model on bridges and a 45% improvement on the indoor experiment, which makes it superior to the deterministic and MC-dropout models, albeit at a loss of 1.4% mIoU on average. Notably, the change in IoU can be attributed to the additional KL-term in Eq. 3, which serves the function of model regularization.

We showed by hiding specific classes in our training data, that the VI-Head model is a good indicator of OOD-examples through its ability to cover most such objects with a moderate degree of uncertainty. The MC-dropout model provided even higher uncertainty values for hidden classes, which on average resulted in an increase of 4.82% O-IoU compared to the deterministic model.

Together these two experiments show an interesting relationship - that is, VI performed better on slight to moderate covariate shifts, however MC-dropout had the advantage when dealing exclusively with novel classes.

The separation of aleatoric and epistemic uncertainty was successfully validated qualitatively in some examples, as displayed in Fig. 15, which allowed the differentiation between elements that could be described as less common: a specific pipe configuration or column shape (epistemic uncertainty) and elements, which are generally hard to differentiate: pier caps from concrete beams (aleatoric uncertainty). However, we conclude that also here a potential for improvement in further works remains, since we did not observe the quality described in [40], where the aleatoric uncertainty was modelled explicitly and could be seen to remain constant when presented with varying amounts of training data (See Fig. 16).

Even though real time inference is often not seen as a valued quality in

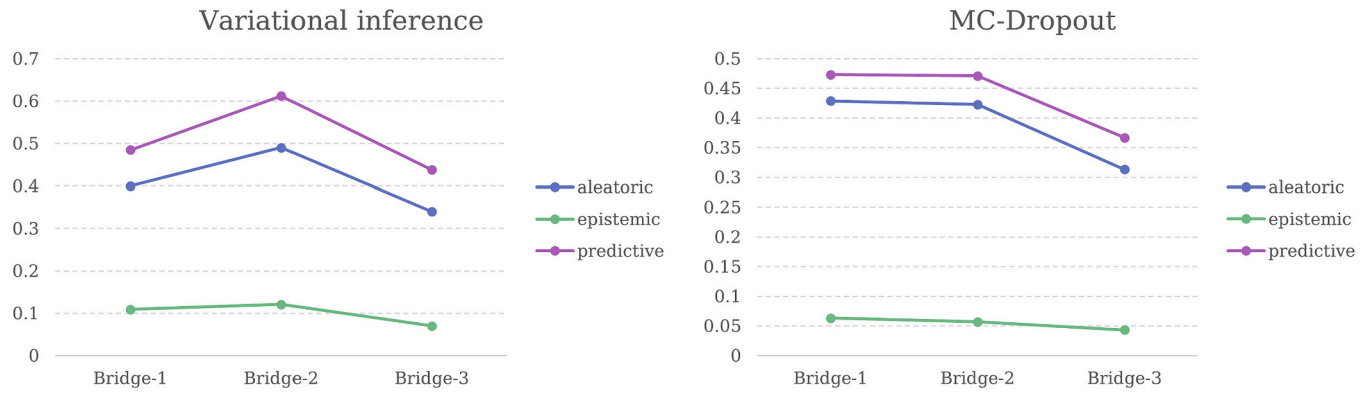


Fig. 14. Mean uncertainty in the point clouds with covariate shift.

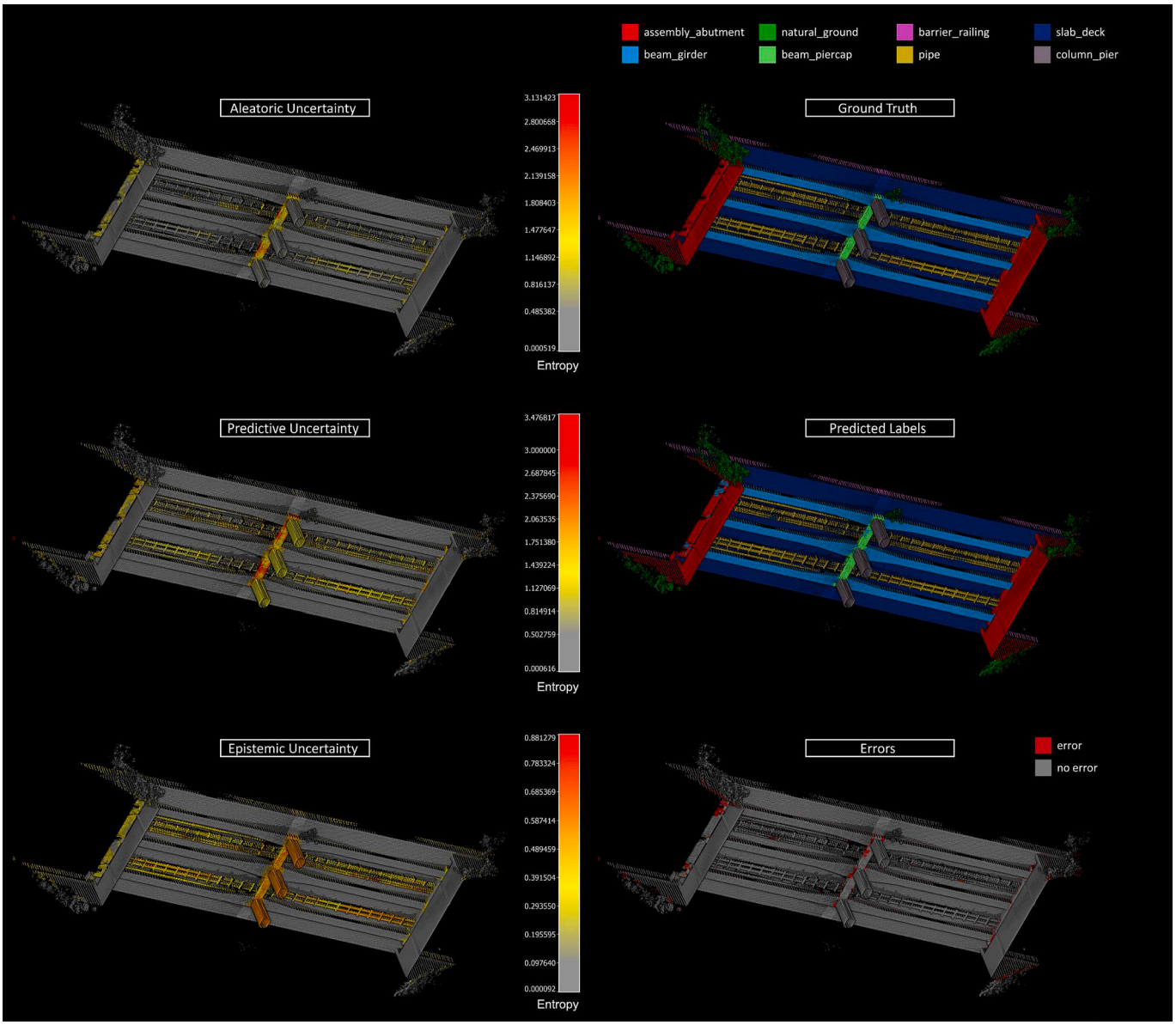


Fig. 15. Segmentation results and uncertainty types from the VI-Head model of KPConv on one of the bridges in the introduced Infrastructure-MLS dataset. The view is oriented bottom-up at the deck of the bridge.

the Scan-to-BIM workflow it should be noted, that the runtime of the two proposed Bayesian models is significantly increased: since we do 50 samples from our posterior we notice an approximately 50-fold increase in runtime for ANNs with Bayesian layers in the middle of the network, such as our VI-Bottleneck model. This can be mitigated to some extend with our models which only implement one Bayesian layer at the end of the network such as the proposed VI-Head model because the majority of the ANN is sampled once, while the stochastic classifier head can be used for the MC-integration. With this optimization we reduce the runtime to a factor of 10 with respect to the deterministic network. Nevertheless, with a moderate dataset size such as the validation split of our MLS dataset we still observed a runtime of 2 h and 10 min when conducting inference with a minimum of 20 votes per point. More information on the voting mechanism of KPConv can be found in the original paper [32].

6. Conclusions

In summary, this work presents MC-Dropout and Variational Inference as two ways of improving the uncertainty estimation in the outputs of the popular KPConv architecture for point cloud semantic segmentation, with VI showing a reduction of 25.3% in calbration error, while MC-dropout leads to a better detection of OOD examples, indicated by an increase of f 4.82 O-IoU. We note some disadvantages, such as the lack of a model, that performs good on both covariate shifts and novel classes. Besides filtering out uncertain classes model uncertainty can be used for active machine learning. Additionally, practitioners can use the type of uncertainty to gain insights about which data source and model type works best for the current task. Our work with point clouds in the infrastructure domain showed that the KPConv architecture is powerful in detecting the structural members of typical RC bridges even when the data lacks photometric information and is provided from the limited field of view of a mobile laser scanner, which can however be used to gather large quantities of data quickly to improve the performance of deep learning models on other types of survey methods such a terrestrial laser scanners. We see this work as a step towards a more robust semantic segmentation in the infrastructure domain with better generalization capabilities, which are beneficial when data acquisition is expensive.

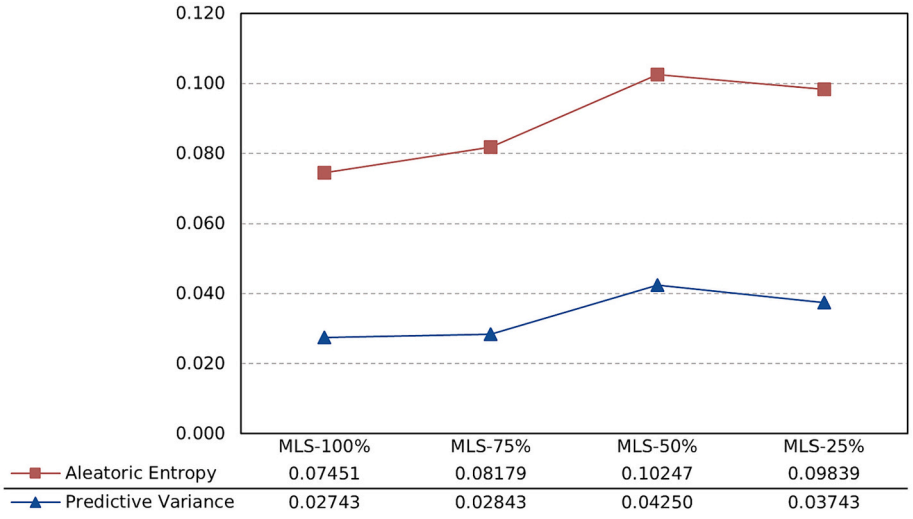


Fig. 16. Average Aleatoric Entropy and Predictive Variance as training data volume decreases.

CRediT authorship contribution statement

Hristo Vassilev: Conceptualization, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing. **Marius Laska:** Supervision, Writing – review & editing. **Jörg Blankenbach:** Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Hristo Vassilev reports financial support was provided by German Research Foundation. Hristo Vassilev reports a relationship with German Research Foundation that includes: funding grants.

Data availability

Data will be made available on request.

Acknowledgments

This project has received funding by (DFG, German Research Foundation) within the framework of "SPP 2388: Hundert plus - Verlängerung der Lebensdauer komplexer Baustrukturen durch intelligente Digitalisierung" with project number 501834640.

References

[1] P. Haardt, R. Holst, The German approach to bridge management: From reactive to predictive management procedures, in: Tenth International Conference on Bridge and Structure Management, 2008, pp. 3–15, https://doi.org/10.17226/17628.

[2] T. Xia, J. Yang, L. Chen, Automated semantic segmentation of bridge point cloud based on local descriptor and machine learning, Automat. Construct. 133 (2022) 103992, https://doi.org/10.1016/j.autcon.2021.103992.

[3] C.-S. Shim, N.-S. Dang, S. Lon, C.-H. Jeon, Development of a bridge maintenance system for prestressed concrete bridges using 3D digital twin model, Struct. Infrastruct. Eng. 15 (10) (2019) 1319–1332, https://doi.org/10.1080/15732479.2019.1620789.

[4] S. Vilgertshofer, M.S. Mafipour, A. Borrmann, J. Martens, T. Blut, R. Becker, J. Blankenbach, A. Gobels, J. Beetz, F. Celik, B. Faltin, M. Konig, TwinGen: Advanced Technologies to Automatically Generate Digital Twins for Operation and Maintenance of Existing Bridges, 1st edition, CRC Press, London, 2023, pp. 213–220, https://doi.org/10.1201/9781003354222-27.

[5] S. Alizadehsalehi, I. Yitmen, Digital twin-based progress monitoring management model through reality capture to extended reality technologies (DRX), Smart Sustain. Built Environ. (2021), https://doi.org/10.1108/SASBE-01-2021-0016.

[6] L. Li, F. Su, F. Yang, H. Zhu, D. Li, X. Zuo, F. Li, Y. Liu, S. Ying, Reconstruction of three-dimensional (3D) indoor interiors with multiple stories via comprehensive segmentation, Remote Sens. (Basel) 10 (8) (2018) 10–12, https://doi.org/10.3390/rs10081281.

[7] R. Maalek, D.D. Lichti, J.Y. Ruwanpura, Automatic recognition of common structural elements from point clouds for automated progress monitoring and dimensional quality control in reinforced concrete construction, Remote Sens. (Basel) 11 (9) (2019), https://doi.org/10.3390/rs11091102.

[8] K. Kawashima, S. Kanai, H. Date, Automatic recognition of piping system from large-scale terrestrial laser scanned point cloud, Seimitsu Kogaku Kaishi/J. Jpn. Soc. Precision Eng. 78 (8) (2012) 722–729, https://doi.org/10.2493/jjspe.78.722.

[9] Y. Perez-Perez, M. Golparvar-Fard, K. El-Rayes, Segmentation of point clouds via joint semantic and geometric features for 3D modeling of the built environment, Automat. Construct. 125 (March 2020) (2021) 103584, https://doi.org/10.1016/j.autcon.2021.103584.

[10] Y. Xu, S. Tuttas, L. Hoegner, U. Stilla, Reconstruction of scaffolds from a photogrammetric point cloud of construction sites using a novel 3D local feature descriptor, Automat. Construct. 85 (February 2017) (2018) 76–95, https://doi.org/10.1016/j.autcon.2017.09.014.

[11] J.H. Lee, J.J. Park, H. Yoon, Automatic bridge design parameter extraction for scan-to-BIM, Appl. Sci. 10 (20) (2020) 7346, https://doi.org/10.3390/app10207346.

[12] Y. Yan, J.F. Hajjar, Automated extraction of structural elements in steel girder bridges from laser point clouds, Automat. Construct. 125 (2021) 103582, https://doi.org/10.1016/j.autcon.2021.103582.

[13] R. Lu, I. Brilakis, C.R. Middleton, Detection of structural components in point clouds of existing RC bridges: detection of bridge components in point clouds, Comput. Aided Civ. Inf. Eng. 34 (3) (2019) 191–212, https://doi.org/10.1111/mice.12407.

[14] M.S. Mafipour, S. Vilgertshofer, A. Borrmann, Heuristic optimization for digital twin modeling of existing bridges from point cloud data by parametric prototype models, in: Proc. of the ASCE International Conference on Computing in Civil Engineering, 2023, https://doi.org/10.1061/9780784483893.

[15] G. Qin, Y. Zhou, K. Hu, D. Han, C. Ying, Automated reconstruction of parametric BIM for bridge based on terrestrial laser scanning data, Adv. Civil Eng. 2021 (2021) 1–17, https://doi.org/10.1155/2021/8899323.

[16] M. Hein, M. Andriushchenko, J. Bitterwolf, Why ReLU networks yield high-confidence predictions far away from the training data and how to mitigate the problem, in: 2019 Conference on Computer Vision and Pattern Recognition (arXiv: 1812.05720), May 2019, https://doi.org/10.1109/CVPR.2019.00013 arXiv:1812.05720.

[17] Y. Gal, Z. Ghahramani, Dropout as a Bayesian approximation: Representing model uncertainty in deep learning, in: 33rd International Conference on Machine Learning, ICML 2016 3, 2016, pp. 1651–1660, arXiv:pp.1506.02142, 10.48550/arXiv.1506.02142.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, in: 32nd International Conference on Machine Learning, ICML 2015 2, 2015, pp. 1613–1622, arXiv:1505.05424.

[19] A. Kristiadi, M. Hein, P. Hennig, Being Bayesian, even just a bit, fixes overconfidence in ReLU networks, in: 37th International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020, 2020, p. 11, https://doi.org/10.48550/arXiv.2002.10118.

[20] C. Louizos, M. Welling, Multiplicative normalizing flows for variational Bayesian neural networks, in: 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017, Jun. 2017, https://doi.org/10.48550/arXiv.1703.01961 arXiv:1703.01961.

[21] A.G. Kendall, Geometry and Uncertaintyin Deep Learning for Computer Vision, Ph. D. thesis, 2019, https://doi.org/10.17863/CAM.35260.

[22] L. Truong-Hong, R. Lindenbergh, Automatically extracting surfaces of reinforced concrete bridges from terrestrial laser scanning point clouds, Automat. Construct. 135 (2022) 104127, https://doi.org/10.1016/j.autcon.2021.104127.
[23] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, M. Bennamoun, Deep learning for 3D point clouds: a survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (12) (2021) 4338–4364, https://doi.org/10.1109/TPAMI.2020.3005434.
[24] T. Cortinhal, G. Tzelepis, E. Erdal Aksoy, SalsaNext: fast, uncertainty-aware semantic segmentation of LiDAR point clouds, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12510 LNCS, 2020, pp. 207–222, https://doi.org/10.48550/arXiv.2003.03653.
[25] B. Graham, M. Engelcke, L. Van Der Maaten, 3D semantic segmentation with submanifold sparse convolutional networks, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2018, pp. 9224–9232, arXiv:1711.10275, https://doi.org/10.1109/CVPR.2018.00961.
[26] G. Riegler, A.O. Ulusoy, A. Geiger, OctNet: learning deep 3D representations at high resolutions, in: Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017 2017-Janua, 2017, pp. 6620–6629, arXiv:1611.05009, https://doi.org/10.1109/CVPR.2017.701.
[27] C.R. Qi, H. Su, K. Mo, L.J. Guibas, PointNet: deep learning on point sets for 3D classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, https://doi.org/10.1109/CVPR.2017.16 arXiv:1612.00593v2.
[28] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: deep hierarchical feature learning on point sets in a metric space, in: Advances in Neural Information Processing Systems 2017-Decem, 2017, pp. 5100–5109, arXiv:1706.02413, 10.48550/arXiv.1706.02413.
[29] F. Hu, J. Zhao, Y. Huang, H. Li, Structure-aware 3D reconstruction for cable-stayed bridges: a learning-based method, Comput. Aided Civ. Inf. Eng. 36 (1) (2021) 89–108, https://doi.org/10.1111/mice.12568.
[30] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein, J.M. Solomon, Dynamic graph CNN for learning on point clouds, in: ACM Transactions on Graphics, Jun. 2019, https://doi.org/10.1145/3341165 arXiv:1801.07829.
[31] J.S. Lee, J. Park, Y.-M. Ryu, Semantic segmentation of bridge components based on hierarchical point cloud model, Autom. Constr. 130 (2021) 103847, https://doi.org/10.1016/j.autcon.2021.103847.
[32] H. Thomas, C.R. Qi, J.E. Deschaud, B. Marcotegui, F. Goulette, L. Guibas, KPConv: flexible and deformable convolution for point clouds, in: Proceedings of the IEEE International Conference on Computer Vision 2019-Octob, 2019, pp. 6410–6419, arXiv:1904.08889, https://doi.org/10.1109/ICCV.2019.00651.
[33] G. Qian, Y. Li, H. Peng, J. Mai, H.A.A.K. Hammoud, M. Elhoseiny, B. Ghanem, PointNeXt: revisiting PointNet++ with improved training and scaling strategies, in: 36th Conference on Neural Information Processing Systems (NeurIPS 2022), Oct. 2022, https://doi.org/10.48550/arXiv.2206.04670 arXiv:2206.04670.
[34] N. Engel, V. Belagiannis, K. Dietmayer, Point transformer, IEEE, Access 9 (2021) 134826–134840, arXiv:2012.09164, https://doi.org/10.1109/ACCESS.2021.3116304.
[35] X. Lai, L. Jiang, L. Wang, H. Zhao, X. Qi, Stratified transformer for 3D point cloud segmentation, in: 2022 IEEE CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8500–8509, https://doi.org/10.48550/arXiv.2203.14508.
[36] C. Park, Y. Jeong, M. Cho, J. Park, Fast point transformer, in: In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, New Orleans, LA, USA, 2022, pp. 16928–16937, https://doi.org/10.1109/CVPR52688.2022.01644.
[37] J. Gawlikowski, C.R.N. Tassi, M. Ali, J. Lee, M. Humt, J. Feng, A. Kruspe, R. Triebel, P. Jung, R. Roscher, M. Shahzad, W. Yang, R. Bamler, X.X. Zhu, A survey of uncertainty in deep neural networks, Artif. Intell. Rev. (2022) 1513–1589, arXiv:2107.03342, 10.48550/arXiv.2107.03342.
[38] L.V. Jospin, W. Buntine, F. Boussaid, H. Laga, M. Bennamoun, Hands-on Bayesian neural networks – a tutorial for deep learning users, IEEE Comput. Intell. Mag. 17 (2) (2022) 29–48, arXiv:2007.06823, https://doi.org/10.1109/MCI.2022.3155327.
[39] J. Joyce, Bayes' theorem, in: E.N. Zalta (Ed.), The Stanford Encyclopedia of Philosophy, Fall, 2021 edition, Stanford University, Metaphysics Research Lab, 2021. URL, https://plato.stanford.edu/archives/fall2021/entries/bayes-theorem/.
[40] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision?, in: 31st Conference on Neural Information Processing Systems (NIPS 2017), Oct. 2017, https://doi.org/10.48550/arXiv.1703.04977 arXiv:1703.04977.
[41] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, A. Bry, End-to-end learning of geometry and context for deep stereo regression, in: 2017 - IEEE International Conference on Computer Vision, 2017, https://doi.org/10.48550/arXiv.1703.04309 arXiv:1703.04309.
[42] E. Hinton, Keeping neural networks simple by minimizing the description length of the weights, in: 1993 - Proceedings of the sixth annual conference on Computational learning theory, 1993, https://doi.org/10.1145/168304.168306.
[43] J. Steinbrener, K. Posch, J. Pilz, Measuring the uncertainty of predictions in deep neural networks with variational inference, Sensors 20 (21) (2020) 6011, https://doi.org/10.3390/s20216011.
[44] H. Ritter, A. Botev, D. Barber, Online structured laplace approximations for overcoming catastrophic forgetting, in: 32nd Conference on Neural Information Processing Systems (NeurIPS 2018), May 2018, https://doi.org/10.48550/arXiv.1805.07810 (arXiv:1805.07810).
[45] F. Dangel, F. Kunstner, P. Hennig, BackPACK: packing more into Backprop, in: 2020 - International Conference on Learning Representations, 2020, https://doi.org/10.48550/arXiv.1912.10985.
[46] S. Fort, H. Hu, B. Lakshminarayanan, Deep Ensembles: A Loss Landscape Perspective, arXiv:1912.02757, Jun. 2020, https://doi.org/10.48550/arXiv.1912.02757. arXiv:1912.02757.
[47] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: 31st Conference on Neural Information Processing Systems, Nov. 2017, https://doi.org/10.48550/arXiv.1612.01474 arXiv:1612.01474.
[48] X. Wang, K. Zhang, H. Li, H. Luo, J. Wang, ProbNet: Bayesian deep neural network for point cloud analysis, Comput. Graph. 104 (2022) 106–115, https://doi.org/10.1016/j.cag.2022.04.004.
[49] C. Petschnigg, J. Pilz, Uncertainty estimation in deep neural networks for point cloud segmentation in factory planning, Modelling 2 (1) (2021) 1–17, https://doi.org/10.3390/modelling2010001.
[50] I. Armeni, O. Sener, A.R. Zamir, H. Jiang, I. Brilakis, M. Fischer, S. Savarese, 3D semantic parsing of large-scale indoor spaces, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-Decem, 2016, pp. 1534–1543, https://doi.org/10.1109/CVPR.2016.170.
[51] T. Weinold, 22. Internationale Geodätische Woche Obergurgl 2023, ISBN 978-3-87907-738-0, 2023.