



## Article

# Uncertainty Estimation in Deep Neural Networks for Point Cloud Segmentation in Factory Planning

Christina Petschnigg <sup>1,\*</sup>  and Jürgen Pilz <sup>2</sup> <sup>1</sup> Department of Factory Planning, BMW Group, Knorrstraße 147, 80788 Munich, Germany<sup>2</sup> Department of Statistics, Alpen-Adria-University Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt, Austria; Juergen.Pilz@aau.at

\* Correspondence: Christina.Petschnigg@bmw.de

**Abstract:** The digital factory provides undoubtedly great potential for future production systems in terms of efficiency and effectivity. A key aspect on the way to realize the digital copy of a real factory is the understanding of complex indoor environments on the basis of three-dimensional (3D) data. In order to generate an accurate factory model including the major components, i.e., building parts, product assets, and process details, the 3D data that are collected during digitalization can be processed with advanced methods of deep learning. For instance, the semantic segmentation of a point cloud enables the identification of relevant objects within the environment. In this work, we propose a fully Bayesian and an approximate Bayesian neural network for point cloud segmentation. Both of the networks are used within a workflow in order to generate an environment model on the basis of raw point clouds. The Bayesian and approximate Bayesian networks allow us to analyse how different ways of estimating uncertainty in these networks improve segmentation results on raw point clouds. We achieve superior model performance for both, the Bayesian and the approximate Bayesian model compared to the frequentist one. This performance difference becomes even more striking when incorporating the networks' uncertainty in their predictions. For evaluation, we use the scientific data set S3DIS as well as a data set, which was collected by the authors at a German automotive production plant. The methods proposed in this work lead to more accurate segmentation results and the incorporation of uncertainty information also makes this approach especially applicable to safety critical applications aside from our factory planning use case.

**Keywords:** point clouds; 3D segmentation; Bayesian deep learning; dropout training; uncertainty estimation; digital factory; factory planning



**Citation:** Petschnigg, C.; Pilz, J. Uncertainty Estimation in Deep Neural Networks for Point Cloud Segmentation in Factory Planning. *Modelling* **2021**, *2*, 1–17. <https://doi.org/10.3390/modelling2010001>

Received: 12 December 2020

Accepted: 29 December 2020

Published: 4 January 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

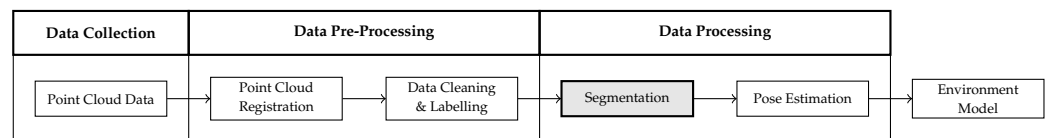


**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

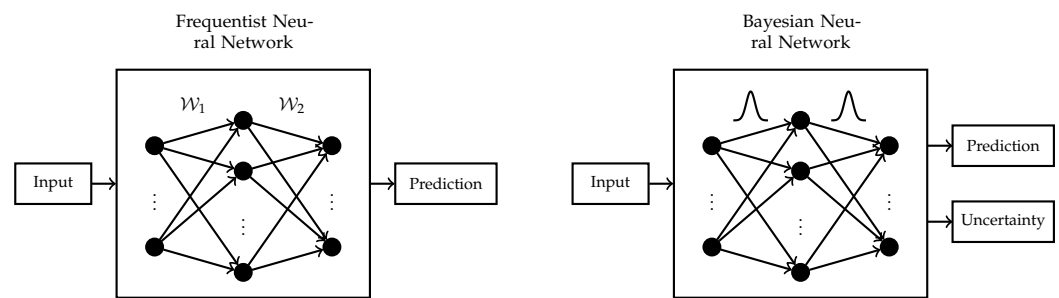
A three-dimensional (3D) model of factory buildings and inventory, as well as the simulation of process steps, play a major role in different planning domains. In general, virtual planning has many advantages when compared to analogue planning. The most stringent benefit is the detection of planning mistakes early on in the planning process. This is favourable, as planning mistakes are detected well before implementation [1], i.e., before factory ramp-up, before new machinery is ordered, before construction is under way, or before the production process is detailed. This is due to the fact that the structure and layout of the building influence several other domains. A changing building model can entail changes in spatial availability for production or logistics assets. Thus, the layout of production lines or the concept of machines may have to be adapted accordingly. Further, virtual planning reduces travel efforts, as planners do not have to meet on-site to discuss modifications or reorganizations. They can rather meet in a multi-user simulation model or a virtual reality supported 3D environment, which saves a substantial amount of travel time and cost. Digital 3D models are the basis for building reorganizations as well as the introduction of completely new or modified manufacturing process steps.

There are several challenges to tackle in order to determine the as-is state of a production plant. First of all, the current data in the respective plant have to be collected. In order to acquire 3D information, laser scanning and photogrammetry are useful digitalization techniques. After plant digitalization, the collected data have to be pre-processed, including data fusion of inputs from different sources, as well as subsequent data cleaning and labelling. The data fusion includes a registration step, where the point clouds from different scans or different sources are fused in a common global coordinate system. However, simulating the factory or the assembly process on the basis of the pre-processed point clouds alone is not possible. Generally, the point clouds generated by laser scanners and photogrammetry techniques suffer from occlusions, i.e., a set of objects blocks the sight to other objects, which results in holes within the point cloud. For instance, the outcomes of collision checking are not reliable when the point cloud is not complete. Additionally, a point cloud does not contain any information regarding how to separate different objects. Therefore, the introduction of new or the displacement of existing objects is time consuming, as the respective set of points has to be selected manually. A segmentation step can be introduced in order to separate different objects from one another automatically. Finally, to generate an environment model, the poses of the relevant objects need to be estimated and the point sets have to be replaced by computer-aided design (CAD) models. Figure 1 summarizes the process for generating an environment model on the basis of a point cloud, which was introduced in our previous work [2]. Each of the process steps can be subdivided into further building blocks. This paper discusses the segmentation step in more detail. The remaining process steps are described in our next contribution.



**Figure 1.** Process diagram of generating an environment model on the basis of a point cloud. The segmentation step is highlighted as this is the focus of the remaining paper.

The semantic segmentation of a point cloud is widely solved using deep neural networks. Most of the existing deep learning architectures make use of the frequentist notion of probability. However, these so-called frequentist neural networks suffer from two major drawbacks. They do not quantify uncertainty in their predictions. Often, the softmax output of frequentist neural networks is interpreted as network uncertainty, which is, however, not a good measure. The softmax function only normalizes an input vector but cannot as such be interpreted as network (un)certainty [3]. Especially for out of distribution samples, the softmax output can give rise to misleading interpretations [4]. In the case of deep learning frameworks being integrated into safety critical applications, like autonomous driving, it is important to know what the network is uncertain about. There was one infamous accident that was caused by a partly autonomous driving car that confused the white trailer of a lorry with the sunlit sky or a bright overhead sign [5]. By considering network uncertainties, similar scenarios could be mitigated. In frequentist neural networks, the network parameters  $\mathcal{W}$  are point estimates, whereas, in Bayesian neural networks, a distribution is placed over all the network parameters. Conceptually, the network optimization is more complex for Bayesian neural networks; however, they allow for additional quantification of network uncertainty. Figure 2 summarizes the differences between a single hidden layer frequentist and Bayesian neural network. Another shortcoming of frequentist neural networks is their tendency to overfit on small data sets with a high number of features. However, in this work, we focus on uncertainty estimation rather than the challenge of feature selection.



**Figure 2.** Frequentist vs. Bayesian neural network. In the frequentist neural network, the parameters  $W_1$  and  $W_2$  correspond to point estimates. In the Bayesian neural network, a probability distribution is placed over the parameters. While the frequentist network only outputs a prediction, the Bayesian neural network additionally estimates the uncertainty in this prediction.

We present a novel Bayesian 3D point cloud segmentation framework that is based on PointNet [6], which is able to capture uncertainty in network predictions. The network is trained using variational inference with multivariate Gaussians with a diagonal covariance matrix as variational distribution. This approach hardly adds any additional parameters to be optimized during each backward pass [7]. Further, we formulate an approximate Bayesian neural network by applying dropout training, as suggested in [3]. We use an entropy based interpretation of uncertainty in the network outputs and distinguish between overall, data related, and model related uncertainty. These types of uncertainty are called predictive, aleatoric, and epistemic uncertainty, respectively [8]. It makes sense to consider this differentiation, as it shows which predictions are uncertain and to what extent this uncertainty can be reduced by further model refinement. The remaining uncertainty after model optimization and training is then inherent to the underlying data set. Other notions of uncertainty that are based on the variance or credible intervals of the predictive network outputs are discussed and evaluated. To the best of our knowledge, no other work has treated the topic of uncertainty estimation and Bayesian training of 3D segmentation networks that operate on raw and unordered point clouds without a previous transformation into a regular format. We embed all of the proposed networks in an industrial prototype for environment modelling in an automotive assembly plant. Aside from an automotive data set that is collected by the authors at a German manufacturing plant, the proposed networks are evaluated on a scientific data set in order to ensure the comparability with other state-of-the-art frameworks. Summing up, the contributions of this paper are:

- Workflow: we describe how to quantify uncertainty in segmentation frameworks that operate on raw and unstructured point clouds. Further, it is discussed how to use this information for improving the generation of factory models.
- Framework: we formulate a 3D segmentation model that is trained in a fully Bayesian way using variational inference and an approximate Bayesian model, which is derived by the application of dropout training.
- Experiment: we evaluate how the different sources of uncertainty affect the neural networks' segmentation performance in terms of accuracy. Further, we outline how the factory model can be improved by considering uncertainty information.

The remainder of this paper is organized in the following way. Section 2 conducts a thorough literature review on 3D point cloud processing frameworks, including deep neural networks, Bayesian neural networks, and uncertainty quantification. In the subsequent Section 3, the frequentist, the approximate Bayesian, and the fully Bayesian models are described in more detail. Section 4 discusses the scientific and industrial data sets that are used for the evaluation of our models and elaborates their characteristics. The models are evaluated with respect to their performance in Section 5. Finally, Section 6 provides a discussion, which describes the bigger scope of this work and concludes the paper.

## 2. Literature Review

The following paragraphs cast light upon prior research in the areas of 3D point cloud processing as well as Bayesian neural networks and uncertainty estimation. The neural networks that are discussed in the first section are all based on the classical or frequentist interpretation of probability. Bayesian neural networks, rather, take on the Bayesian interpretation of probability, which views probability as a personal degree of belief.

### 2.1. 3D Point Cloud Processing

In contrast to images that have a regular pixel structure, point clouds are irregular and unordered. Further, they do not have a homogeneous point density, due to occlusions and reflections. Neural networks that process 3D point clouds have to tackle all of these challenges. Most of the networks are based on the frequentist interpretation of probability and they are divided into three classes based on the format of their input data. There are deep learning frameworks that consume voxelized point clouds [9–12], collections of two-dimensional (2D) images that are derived by transforming 3D point clouds to the 2D space from different views [8,13,14], and raw unordered point clouds [6,15,16]. On the one hand, the voxelization of point clouds has the advantage of providing a regular structure apt for the application of 3D convolutions. On the other hand, it renders the data unnecessarily big, as unoccupied areas of the point cloud are still represented by voxels. Generally, this format conversion introduces truncation errors [6]. Further, voxelization reduces the resolution of the point cloud in dense areas, which leads to a loss of information [17]. Transforming 3D point clouds to 2D images from different views allows for the application of standard 2D convolutions having the advantage of elaborate kernel optimizations. Yet, the transformation to a lower space can cause the loss of structural information embedded in the higher dimensional space. Additionally, in complex scenes, a high number of viewports have to be taken into account in order to describe the details of the environment [17]. For this reason, the following work focuses on the segmentation of raw point clouds.

In order to generate a factory model out of raw point clouds, the objects of interest have to be detected and their pose needs to be estimated. One approach that extracts six degrees-of-freedom (DoF) object poses, i.e., the translation and orientation with respect to a predefined zero point, in order to generate a simulation scene is presented in [18]. The framework is called Scan2CAD, and it describes a frequentist deep neural network that consumes voxelized point clouds as well as CAD models of eight household objects and directly learns the 6DoF CAD model alignment within the point cloud. The system that is presented in [19] has similar input data and estimates the 9DoF pose, i.e., translation, rotation and scale, of the same household objects. A framework for the alignment of CAD models, which is based on global descriptors computed by using the Viewpoint Feature Histogram approach [20] rather than neural networks, is discussed in [21]. Generally, direct 6DoF or 9DoF pose estimation on the basis of point clouds and CAD models can be used to set up environment models and simulation scenes. However, these approaches always require the availability of CAD models, which is not the case for many building and inventory objects in real-world factories. Thus, we follow the approach of semantic segmentation, instead of direct pose estimation. Semantic segmentation enables us to extract reference point clouds of objects, for which no CAD model is available. These objects can either be modelled in CAD automatically by using meshing techniques or by hand if the geometry is too difficult to capture realistically. Further, the segmentation approach enables us to part the point cloud into bigger contexts, i.e., subsets of points belonging to the construction, assembly, or logistics domain. These smaller subsets of points can be sent to the respective departments for further processing, which reduces the computational burden of the point cloud to be processed. Mere pose estimation is not sufficient for fulfilling this task. Aside from the semantic segmentation of point clouds, this work focuses on the formulation of Bayesian neural networks and how to leverage the uncertainty information that can be calculated in order to increase the models' accuracy.

## 2.2. Bayesian Deep Learning and Uncertainty Quantification

In contrast to frequentist neural networks, where the network parameters are point estimates, Bayesian neural networks (BNNs) place a distribution over each of the network parameters. For this reason, a prior distribution is defined over the parameters. After observing the training data, the aim is to calculate the respective posterior distribution, which is difficult, as it requires the solution of a generally intractable integral. There exist several approximation approaches, including variational inference (VI) [22,23], Markov Chain Monte Carlo (MCMC) methods [24–26], Hamiltonian Monte Carlo (HMC) algorithms [27], and Integrated Nested Laplace approximations (INLA) [28]. VI provides a fast approximation to the posterior distribution. However, it comes without any guaranteed quality of approximation. MCMC methods in contrast are asymptotically correct, but they are computationally much more expensive than VI. Even the generally faster HMC methods are clearly more time consuming than VI [29]. We decide to apply VI due to efficiency reasons, as the data sets used for evaluating this work are huge in size.

In the literature, there are several ways of how uncertainty can be quantified in BNNs. It is possible to distinguish between data and model related uncertainty, which are referred to as aleatoric and epistemic uncertainty, respectively [30]. The overall uncertainty inherent to a prediction can be computed as the sum of aleatoric and epistemic uncertainty, and it is called predictive uncertainty. Such a distinction is beneficial for practical applications in order to determine to what extent model refinement can reduce predictive uncertainty and to what extent uncertainty stems from the data set itself. One possibility of describing predictive uncertainty  $U_{pred}$  in a label  $y^*$  belonging to an input  $\underline{x}^*$  with weights  $\underline{w}$  is based on entropy, i.e.,  $U_{pred} = \mathbb{H}[y^* | \underline{w}, \underline{x}^*]$  [31]. Another way of quantifying uncertainty in the network parameters of BNNs is presented in [7]. This approach only introduces two uncertainty parameters per network layer, which allows us to grasp uncertainty layer-wise, but it does not impair network convergence. The overall model uncertainty is measured by estimating credible intervals of the predictive network outputs. This is based on the notion that higher uncertainty in the network parameters results in higher uncertainty in the network outputs. Further, the predictive variance can also be used for uncertainty estimation.

## 3. Model Descriptions

In the following the frequentist, the approximate Bayesian and the fully Bayesian models are explained. In order to formulate these models, let  $X = \{\underline{x}_1, \dots, \underline{x}_n\}$  be the input data and  $Y = \{y_1, \dots, y_n\}$  the corresponding labels, where  $y_i \in \{1, \dots, m\}$ ,  $m \in \mathbb{N}$ ,  $i \in \{1, \dots, n\}$ ,  $n \in \mathbb{N}$ . Further, let  $\mathcal{W}$  and  $\mathcal{B}$  denote all of the network parameters, including the weights and biases, respectively. The network weights and biases of the  $i$ -th network layer are denoted by  $\mathcal{W}_i$  and  $\mathcal{B}_i$ ,  $i \in \{1, \dots, d\}$ , where  $d \in \mathbb{N}$  is the network depth. The described network architectures mainly apply convolutional layers, thus, we write  $conv(i, j)$  for a convolutional layer with input dimension  $i \in \mathbb{N}$  and output dimension  $j \in \mathbb{N}$ . In the following,  $\sigma(\cdot)$  denotes a non-linear function. Note that the introduced variables and parameters are used throughout the remaining work. In the sequel, uncertainty estimation is explained in more detail and its practical implementation is discussed.

### 3.1. Frequentist PointNet

The baseline for the following derivations and evaluations is the PointNet segmentation architecture [6]. This framework consumes raw and unordered point sets in a block structure. The number of points in each of the blocks is exactly 4096—either due to random down-sampling or due to up-sampling by repeated drawing of points. Each input point is represented by a vector  $\underline{x}$  containing xyz-coordinates that are centred about the origin and RGB values. For later illustration purposes, we add another three dimensions, which hold the original point coordinates, i.e.,  $\dim(\underline{x}) = 9$ . The actual network input is a tensor of dimension  $bs \times 4096 \times 6$ , where  $bs \in \mathbb{N}$  represents the batch size. The batch size corresponds to the number of input blocks being treated at a time. Each of these blocks



consists of exactly 4096 points. Further, the centred point coordinates are rather used for network training than the original ones, thus the last dimension is 6 instead of 9.

In this architecture, a symmetric input transformation network is applied first. It is followed by a convolutional layer  $conv(6, 64)$  and a feature transformation network. After the feature transformation, another two convolutional layers  $conv(64, 128, 1024)$  are applied before extracting global point cloud features using a max pooling layer. These global features are concatenated to the local features, which correspond to the direct output of the feature transformation network. The resulting network scores are generated by four convolutional layers  $conv(1088, 512, 256, 128, m)$ , where  $m \in \mathbb{N}$  is the number of classes. The rectified linear unit (ReLU) is used as a non-linear activation function in this network.

### 3.2. Approximate Bayesian PointNet

For the approximate Bayesian PointNet segmentation network, we use the notion that dropout training in neural networks corresponds to approximate Bayesian inference [3]. In the following, this network will be referred to as dropout PointNet. Dropout in a single hidden layer neural network can be defined by sampling binary vectors  $\underline{c}_1 \in \{0, 1\}^{d_1}$  and  $\underline{c}_2 \in \{0, 1\}^{d_2}$  from a Bernoulli distribution, such that  $c_{1,q} \sim Be(p_1)$  and  $c_{2,k} \sim Be(p_2)$ , where  $q = 1, \dots, d_1$  and  $k = 1, \dots, d_2$ . The variables  $d_1$  and  $d_2$  corresponds to the number of weights in the respective layer and  $p_1, p_2 \in [0, 1]$ . Subsequently, the network prediction  $\underline{\hat{y}}$  reads

$$\underline{\hat{y}} = \sigma(\underline{x}(\underline{c}_1 \mathcal{W}_1) + \mathcal{B}_1)(\underline{c}_2 \mathcal{W}_2). \quad (1)$$

The bias in the second layer is omitted, which corresponds to centring the output. For  $n \in \mathbb{N}$  network inputs and  $m \in \mathbb{N}$  classes, the network output  $\underline{\hat{y}}$  is normalized to obtain  $\underline{\hat{p}}$  using the softmax function

$$\hat{p}_{ij} = \frac{\exp(\hat{y}_{ij})}{\sum_{j'=1}^m \exp(\hat{y}_{ij'})}, \quad i = 1, \dots, n, \quad j = 1, \dots, m. \quad (2)$$

The log of this function results in the log-softmax loss. In order to improve the generalization ability of the network,  $\mathcal{L}_2$  regularization terms for the network weights and biases can be added to the loss function. The optimization of such a neural network acts as approximate Bayesian inference in deep Gaussian process models [3]. This approach neither changes the model nor the optimization procedure, i.e., the computational complexity during network training does not increase. It is suggested to apply dropout before every weight layer in the network; however, empirical results with respect to convolutional neural networks show inferior performance when doing so. Thus, we place dropout before the last and the last three layers in the PointNet model for S3DIS and the automotive factory data set, respectively. Other than that, the frequentist model is left unchanged. Placing dropout within the input or feature transform network results in considerably lower performance.

### 3.3. Bayesian PointNet

BNNs place a distribution over each of the network parameters, as already mentioned. In Bayesian deep learning, all of the network parameters, including weights and biases, are expressed as one single random vector  $\underline{w}$ . The prior knowledge regarding the parameters  $\underline{w}$  is captured by the a priori distribution  $p(\underline{w})$ . After observing some data  $(X, Y)$ , the a posteriori distribution can be derived. Using Bayes' Theorem, the posterior density reads

$$p(\underline{w}|Y, X) = \frac{p(Y|\underline{w}, X)p(\underline{w})}{\int p(Y|\underline{w}, X)p(\underline{w})d\underline{w}}. \quad (3)$$

The likelihood  $p(Y|\underline{w}, X)$  is given by  $\prod_{i=1}^n BNN(\underline{x}_i; \underline{w})_{y_i}$ , which corresponds to the product of the BNN outputs for all of the training inputs under the assumption of stochastic independence. However, the integral in the denominator is usually intractable, which

makes the direct computation of the posterior difficult. In Section 2.2, different methods for posterior approximation are discussed. We use VI, as it is most efficient in the case of a huge amount of training data, as already mentioned. The idea of VI is to approximate the posterior  $p(\underline{w}|Y, X)$  by a parametric distribution  $q_{\underline{\varphi}}(\underline{w})$  and  $\underline{\varphi}$  represents the so-called variational parameters. To this end, the Kullback–Leibler divergence (KL-divergence) between the variational and posterior density is minimized, i.e.,

$$KL(q_{\underline{\varphi}}(\underline{w})||p(\underline{w}|Y, X)) := \mathbb{E}_{q_{\underline{\varphi}}(\underline{w})} \left( \ln \frac{q_{\underline{\varphi}}(\underline{w})}{p(\underline{w}|Y, X)} \right) = \int q_{\underline{\varphi}}(\underline{w}) \ln \frac{q_{\underline{\varphi}}(\underline{w})}{p(\underline{w}|Y, X)} d\underline{w}. \quad (4)$$

The KL-divergence does not describe a real distance metric, as the triangle inequality and the property of symmetry are not fulfilled. Nevertheless, it is frequently used in BNN literature in order to measure the distance between two distributions. Because of the unknown posterior in the denominator of the KL-divergence, it cannot be optimized directly. According to [32], the minimization of the KL-divergence is equivalent to the minimization of the negative log evidence lower bound (ELBO), which reads

$$ELBO = - \int q_{\underline{\varphi}}(\underline{w}) \ln p(Y|\underline{w}, X) d\underline{w} + KL(q_{\underline{\varphi}}(\underline{w})||p(\underline{w})). \quad (5)$$

After the optimization of the variational distribution, it can be used to approximate the posterior predictive distribution for unseen data. Let  $\underline{x}^*$  be an unseen input with corresponding label  $y^*$ . The posterior predictive distribution represents the belief in a label  $y^*$  for an input  $\underline{x}^*$ , and it is given by

$$p(y^*|\underline{x}^*, Y, X) = \int p(y^*|\underline{w}, \underline{x}^*) p(\underline{w}|Y, X) d\underline{w}. \quad (6)$$

The two factors under the integral correspond to the (future) likelihood and the posterior. The intractable integral can be approximated by Monte Carlo integration with  $K \in \mathbb{N}$  terms and the posterior distribution is replaced by the variational distribution, i.e.,

$$p(y^*|\underline{x}^*, Y, X) \approx \frac{1}{K} \sum_{k=1}^K BNN(\underline{x}^*; \hat{\underline{w}}_k)_{y^*} \quad \text{with } \hat{\underline{w}}_k \underset{i.i.d.}{\sim} q_{\underline{\varphi}}(\underline{w}), \quad (7)$$

with  $BNN$  denoting a forward pass through the network and  $\hat{\underline{w}}_k$  being the  $k$ -th weight sample drawn from the variational distribution. Finally, the prediction  $\hat{y}^*$  is given by the index of the largest element in the mean of the posterior predictive distribution and, thus, reads

$$\hat{y}^* = \arg \max_{j \in \{1, \dots, m\}} \frac{1}{K} \sum_{k=1}^K BNN(\underline{x}^*; \hat{\underline{w}}_k)_j. \quad (8)$$

After having discussed the theoretical background, we describe our Bayesian model and the corresponding variational distribution. The model that we suggest has a similar structure to the framework in [7]. In the remaining section, the subscript indices  $w$  and  $b$  represent that a quantity is related to the network weights and biases, respectively. The weights  $\mathcal{W}_i$  and biases  $\mathcal{B}_i$  of the  $i$ -th network layer  $i \in \{1, \dots, d\}$  are defined, as follows

$$\tau_{wi} := \log(1 + \exp(\delta_{wi})) \quad (9)$$

$$\tau_{bi} := \log(1 + \exp(\delta_{bi})) \quad (10)$$

$$\mathcal{W}_i := \underline{\mu}_{wi} \odot (\mathbf{1}_{d_i} + \tau_{wi} \underline{\varepsilon}_{wi}) \quad (11)$$

$$\mathcal{B}_i := \underline{\mu}_{bi} \odot (\mathbf{1}_{d'_i} + \tau_{bi} \underline{\varepsilon}_{bi}), \quad (12)$$

where  $\delta_{wi} \in \mathbb{R}$ ,  $\delta_{bi} \in \mathbb{R}$ ,  $\underline{\mu}_{wi} \in \mathbb{R}^{d_i}$  and  $\underline{\mu}_{bi} \in \mathbb{R}^{d'_i}$  are the variational parameters. Further,  $\mathbf{1}_d$  denotes the  $d$ -dimensional vector that consists of all ones,  $\underline{\varepsilon}_{wi} \in \mathbb{R}^{d_i}$  as well as  $\underline{\varepsilon}_{bi} \in \mathbb{R}^{d'_i}$

are multivariate standard normally distributed and  $\odot$  represents the Hadamard product. Thus, the weights and biases follow a multivariate normal distribution with a diagonal covariance matrix, i.e.,

$$\mathcal{W}_i \sim \mathcal{N}(\underline{\mu}_{wi}, \tau_{wi} \text{diag}(\underline{\mu}_{wi})^2) \quad (13)$$

$$\mathcal{B}_i \sim \mathcal{N}(\underline{\mu}_{bi}, \tau_{bi} \text{diag}(\underline{\mu}_{bi})^2). \quad (14)$$

For more detailed insights on the respective gradient updates, see [7]. We use the leaky ReLU activation function in the Bayesian model with a negative slope of 0.01 due to the dying ReLU problem. The mean of the weights is initialized using the Kaiming normal initialization with the same negative slope as for the leaky ReLU activation.

### 3.4. Uncertainty Estimation

As already described, the estimated uncertainty can be split into predictive, aleatoric, and epistemic uncertainty. In practice, predictive uncertainty  $U_{pred}$  is approximated by marginalization over the weights,

$$U_{pred} \approx - \sum_{y^* \in \{1, \dots, m\}} \left( \frac{1}{K} \sum_{k=1}^K p(y^* | \hat{w}^k, \underline{x}^*) \right) \cdot \log \left( \frac{1}{K} \sum_{k=1}^K p(y^* | \hat{w}^k, \underline{x}^*) \right). \quad (15)$$

In Equation (15)  $p(y^* | \hat{w}^k, \underline{x}^*)$  corresponds to the predictive network output of label  $y^*$  for an input data point  $\underline{x}^*$  and the  $k$ -th weight sample  $\hat{w}^k$  of the variational distribution. The total number of Monte Carlo samples is given by  $K \in \mathbb{N}$ . Aleatoric uncertainty  $U_{alea}$  is interpreted as the average entropy  $\mathbb{H}$  over all of the weight samples,

$$U_{alea} = \mathbb{E}_{q_{\varphi}(\underline{w})} [\mathbb{H}[y^* | \hat{w}^k, \underline{x}^*]] \approx - \frac{1}{K} \sum_{k=1}^K \sum_{y^* \in \{1, \dots, m\}} p(y^* | \hat{w}^k, \underline{x}^*) \cdot \log(p(y^* | \hat{w}^k, \underline{x}^*)). \quad (16)$$

Finally, epistemic uncertainty  $U_{ep}$  is the difference between predictive uncertainty and aleatoric uncertainty, i.e.,  $U_{ep} = U_{pred} - U_{alea}$ . Further, uncertainty in network predictions can be quantified by calculating the variance of the predictive network outputs. Another way is to calculate a credible interval on the network outputs for each class. For instance, the 95%-credible interval can be calculated for each class. The prediction is considered to be uncertain in the case that the 95%-credible interval of the predicted class overlaps with the 95%-credible interval of any other class.

## 4. Data Sets

Two different data sets are used in order to evaluate our Bayesian and the approximate Bayesian segmentation approach, which forms the core contribution of this work. The first one is the Stanford large-scale 3D indoor spaces data set that is open to scientific use and, thus, ensures the comparability of our approach to other methods. The second data set is a large-scale point cloud data set collected and pre-processed by the authors at a German automotive OEM.

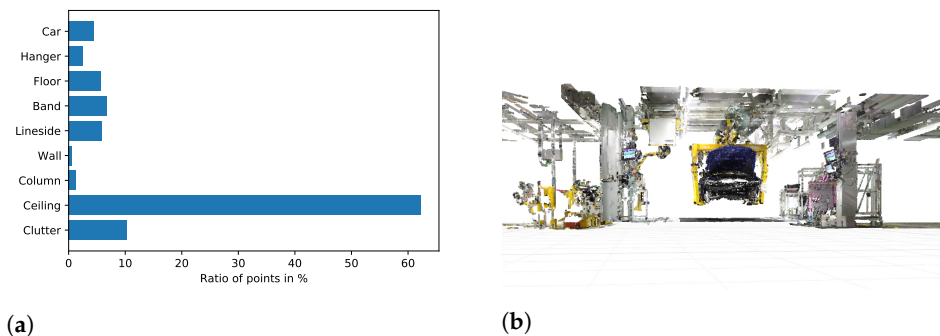
### 4.1. Stanford Large-Scale 3D Indoor Spaces Data Set

The Stanford large-scale 3D indoor spaces (S3DIS) data set [33] is an RGB-D data set of six indoor areas. It features more than 215 million points that are collected over an area totalling more than 6000 m<sup>2</sup>. The areas are spread across three buildings, including educational facilities, offices, sanitary facilities, and hallways. The annotations are provided on the instance level and they distinguish six structural elements from seven furniture elements. This totals the 13 classes, including the building structures of ceiling, floor, wall, beam, column, window, and door, as well as the furniture elements of table, chair, sofa, bookcase, board, and clutter. The data set can be downloaded from <http://buildingparser.stanford.edu/dataset.html>.



#### 4.2. Automotive Factory Data Set

This data set was collected using both the static Faro Focus3D X 130HDR laser scanner [34] and two DSLR cameras. In more detail, a Nikon D5500 [35] camera with an 8 mm fish-eye lens and a Sony Alpha 7R II [36] with a 25 mm fixed focal length lens were used. We generate a global point cloud comprising 13 tacts of car body assembly by the registration of several smaller point clouds collected at each scanner position. The final point cloud comprises more than one billion points before further pre-processing. The cleaning process is achieved using noise filters for coarse cleaning and fine tuning is done by hand. The resulting point set consists of 594,147,442 points. This accounts for a reduction of approximately 40% of the points after point cloud cleaning. Most of the removed points are noise points that are caused by reflections and the blur of moving objects, like people walking by the laser scanner. The data set is divided into nine different classes, namely car, hanger, floor, band, lineside, wall, column, ceiling, and clutter. The labelling is manually done by the authors. The class clutter is a placeholder for all of the objects that cannot be assigned to one of the other classes. All of the remaining classes are either building structures or objects that can only be moved with high efforts; thus, they are essentially immovable and have to be considered during the planning tasks. The resulting data set is highly imbalanced with respect to the class distribution. Figure 3a depicts the class distribution of this data set. Clearly, there is a notable excess of points that belong to the class ceiling and relatively few points belong to the classes of wall and column. This is mainly due to the layered architecture of the ceiling that results in points that belong to the structure on various heights. Because walls and columns are mostly draped with other objects, like tools, cables, fire extinguishers, posters, and information signs, there is only a small number of points that truly belong to the classes of wall and column. This is also the reason why especially these two classes suffer from a high degree of missing data, i.e., holes in the point cloud. Any segmentation system has to cope with this class imbalance due to this inhomogeneous class distribution. Figure 3b illustrates the point cloud of one tact of car body assembly.



**Figure 3.** Overview of the collected data set. (a) Total ratio of all points in the point cloud that belong to each of the classes in %. (b) Illustration of a point cloud displaying one tact of car body assembly.

### 5. Results and Analysis

The proposed networks are evaluated on our custom automotive factory data set as well as the scientific data set S3DIS. The segmentation performance is measured with respect to their accuracy and the mean intersection over union. Further, the described ways of uncertainty quantification are evaluated in terms of accuracy after disregarding uncertain predictions. All of the considered models are implemented using Python's open source library PyTorch [37]. The input point clouds comprising rooms or assembly tacts are cut into blocks and the number of points within these blocks is sampled to 4096. These blocks serve as input for all networks. All of the models are trained using mini-batch stochastic gradient descent with a batch size of 16 on the S3DIS and the automotive factory data set for the frequentist and the proposed dropout and Bayesian networks. The momentum parameter is set to 0.9 for all the models. A decaying learning rate  $lr$  is used with an initial learning rate of  $lr = 0.001$  in the frequentist and the dropout model,

as well as  $lr = 0.01$  in the Bayesian model. The learning rate is decayed every 10 epochs by a factor of 0.7 during frequentist and dropout training, as well as 0.9 during Bayesian training. The batch size and the learning rate are optimized by using grid search and cross-validation. In the approximate Bayesian neural network, dropout is applied before the last three convolutional layers and the dropout rate is set to 0.1 for the automotive factory data set. In the case of the S3DIS data set, dropout is only applied before the last convolutional layer with a dropout rate of 0.1. Because we do not have dedicated prior information for the Bayesian model, the prior indicates that the parameter values should not diverge. Thus, we choose a prior expectation of zero for all parameters and a standard deviation of 4 and 8 for all weights and biases, respectively. In terms of approximating the posterior predictive distribution, we draw  $K = 50$  Monte Carlos samples. All of the considered models converge and training is stopped after 100 epochs.

### 5.1. Segmentation Accuracy

The three architectures, i.e., the frequentist, dropout, and Bayesian PointNet, as described in Section 3, are evaluated in the following. The accuracy and the mean Intersection over Union (IoU) are the evaluation metrics used. The accuracy is calculated by the number of correctly classified points divided by the total number of points. The IoU or Jaccard coefficient describes the similarity between two sets with finite cardinality. It is defined by the number of points of the intersection, divided by the number of points of the union of the two sets. In this case, we evaluate the overlap between the points classified as class  $i$  by the model and the points of class  $i$  in the ground truth. Thus, the IoU for class  $i$  reads

$$J_i = \frac{\# \text{ points correctly classified as } i}{\# \text{ points classified as } i + \# \text{ number points in class } i \text{ in ground truth}}, \quad (17)$$

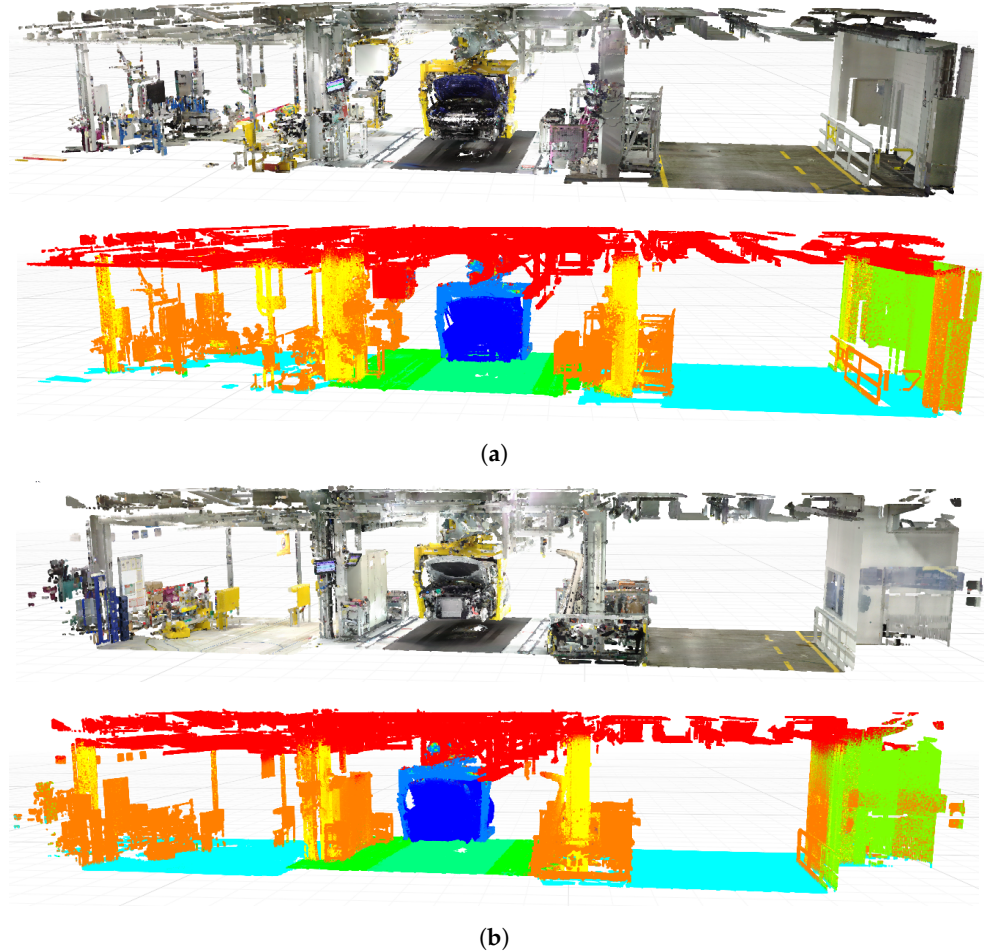
where  $i \in \{1, \dots, m\}$  indicates the class label. The mean IoU is calculated as the mean IoU value over all classes. Table 1 illustrates that Bayesian PointNet clearly surpasses the performance of frequentist and dropout PointNet with respect to accuracy as well as mean IoU on the test set of both the data sets. Bold table entries represent best model performance in the respective setting and are used throughout the remainder of this section.

**Table 1.** Segmentation results of classical, dropout, and Bayesian PointNet on S3DIS and our automotive factory data set. For every model, the training converges and it is stopped after 100 epochs.

Model	Data	Training Acc.	Test Acc.	Test mIoU
Classical PointNet	S3DIS	95.44%	87.81%	0.6977
Dropout PointNet	S3DIS	95.42%	87.71%	0.6921
Bayesian PointNet	S3DIS	<b>95.52%</b>	<b>88.57%</b>	<b>0.7042</b>
Classical PointNet	Automotive	97.66%	94.23%	0.7808
Dropout PointNet	Automotive	98.01%	94.59%	0.7972
Bayesian PointNet	Automotive	<b>98.66%</b>	<b>95.47%</b>	<b>0.8263</b>

For the S3DIS data set, we test the models on area 6 and, for the automotive factory data set, we set aside two distinct assembly tacts. It is noticeable that all of the models have a higher tendency to overfit on the S3DIS data set than on the automotive factory data set. This is due to the nature of the data itself. While there is a considerable variability between the different areas and room types in the S3DIS data set, there is less variability between the tacts of an assembly plant. They are all designed in a similar way in order to ensure the efficient execution of the assembly process. Even though distinct tacts are used for model training and testing, the variability between the training and test set is much smaller than in the case of the S3DIS data set. Figure 4 illustrates the qualitative segmentation results for the two automotive tacts in the test set. Clearly, the network generates smooth predictions, with most of the mistakes affecting the classes of column, wall, clutter, and ceiling, which meets our assumption of Section 4.2. The prior information in the Bayesian

model acts as additional observations and, thus, is able to reduce overfitting and increase model performance. An even more striking difference in performance is illustrated in the next section when the information that is provided by uncertainty estimation is considered.



**Figure 4.** Qualitative segmentation results for the two assembly tacts of the test set comprising about 94,174,898 points after down-sampling. In (a,b) the top image represents the coloured input point cloud. The bottom image illustrates the segmented output point cloud of the Bayesian segmentation network.

### 5.2. Uncertainty Estimation

We estimate uncertainty using an entropy based approach, the predictive variance, as well as an approach based on estimating credible intervals on the probabilistic network outputs, as already mentioned. Predictive and aleatoric uncertainty are calculated, as suggested in the Equations (15) and (16). Epistemic uncertainty is the difference of these two quantities. The predictive variance is determined on the basis of  $K = 50$  forward passes of the input through the network using the unbiased estimator for the variance. Based on the same sample, the 95%-confidence intervals of the network outputs for each class are calculated. Table 2 contains the results of Bayesian PointNet for one room of each room type in area 6 of S3DIS data set as well as one tact of car body assembly belonging to the test data set. The leftmost column contains the accuracy of Bayesian PointNet. The next column contains the accuracy when only considering predictions that have a predictive uncertainty smaller or equal to the mean predictive uncertainty plus two sigma of the predictive uncertainty. The same is displayed in the next columns with respect to aleatoric and epistemic uncertainty, as well as the variance of the predictive network outputs. In the last column, the predictions for which the 95%-credible interval of the predicted class overlaps with no other class' 95%-credible interval are considered certain

and retained for generating a factory model. It can be seen that the accuracy increases considerably when only looking at certain predictions with respect to any of the uncertainty measures. Generally, the results for predictive and aleatoric uncertainty as well as the credible interval based method are most promising. This confirms our notion that the predictive uncertainty value is mainly determined by aleatoric uncertainty after thorough network training. Table 3 displays the percentage of predictions, which are found to be uncertain.

**Table 2.** Evaluation of different methods for uncertainty estimation and influence on model accuracy in the Bayesian model.

	Baseline	Predictive	Aleatoric	Epistemic	Variance	Credible
Conf. Room	88.92%	<b>92.56%</b>	92.59%	91.03%	90.82%	91.01%
Copy Room	70.82%	72.56%	72.58%	72.37%	72.02%	<b>74.85%</b>
Hallway	81.13%	83.01%	82.98%	82.98%	83.20%	<b>93.06%</b>
Lounge	71.77%	73.31%	73.49%	73.09%	73.06%	<b>77.16%</b>
Office	90.77%	<b>93.02%</b>	93.01%	92.28%	92.37%	92.42%
Open Space	76.68%	79.45%	79.54%	77.96%	77.77%	<b>80.38%</b>
Pantry	76.21%	78.51%	78.58%	77.44%	77.24%	<b>79.20%</b>
Assemb. Tact	94.21%	96.63%	<b>96.64%</b>	95.54%	95.60%	94.99%

**Table 3.** Percentage of predictions dropped when excluding uncertain predictions in the Bayesian model.

	Baseline	Predictive	Aleatoric	Epistemic	Variance	Credible
Conf. Room	-	6.86%	6.90%	5.49%	5.50%	4.07%
Copy Room	-	3.25%	3.28%	4.50%	4.81%	9.53%
Hallway	-	4.64%	4.65%	5.08%	5.53%	4.54%
Lounge	-	3.37%	3.64%	4.89%	5.22%	10.96%
Office	-	5.94%	5.94%	4.70%	5.17%	3.73%
Open Space	-	6.23%	6.41%	4.97%	5.05%	8.62%
Pantry	-	4.75%	4.87%	4.58%	4.78%	7.00%
Assemb. Tact	-	6.55%	6.56%	4.09%	3.70%	1.47%

Generally, approximately 3% to 11% of the predictions are dropped using the above parameters. The number of dropped predictions decreases when predictions with a higher uncertainty value are considered, e.g., all of the predictions with uncertainty greater or equal to the mean uncertainty plus three sigma. Generally, it can be claimed that the lower the threshold for uncertain predictions, i.e., the more predictions are dropped, the higher the resulting accuracy. Thus, a trade-off between dropping uncertain predictions and segmentation accuracy needs to be found. However, this is largely dependent on the specific use case. Table 4 illustrates the results of dropout PointNet for one room of each room type in area 6 of S3DIS data set, as well as one tact of car body assembly belonging to the test data set. The accuracy of Bayesian PointNet surpasses the accuracy of dropout PointNet for most of the evaluated rooms. The results are similar in terms of the percentage of predictions dropped. Table 5 presents the percentage of disregarded predictions as compared to the baseline containing all of the predictions. Again, approximately 2% to 11% of the predictions are dropped by dropout PointNet using the same uncertainty threshold, as before.

**Table 4.** Evaluation of different methods for uncertainty estimation and influence on model accuracy in the dropout model.

	Baseline	Predictive	Aleatoric	Epistemic	Variance	Credible
<b>Conf. Room</b>	87.61%	<b>90.72%</b>	<b>90.72%</b>	89.66%	89.51%	89.58%
<b>Copy Room</b>	71.07%	72.46%	72.55%	72.54%	72.37%	<b>74.52%</b>
<b>Hallway</b>	81.94%	83.99%	83.97%	83.95%	83.70%	<b>84.87%</b>
<b>Lounge</b>	70.45%	71.87%	71.89%	72.78%	72.77%	<b>74.83%</b>
<b>Office</b>	81.06%	82.90%	82.94%	82.46%	82.57%	<b>83.71%</b>
<b>Open Space</b>	75.97%	78.45%	<b>78.47%</b>	77.68%	77.59%	77.96%
<b>Pantry</b>	74.44%	76.48%	76.52%	76.87%	76.66%	<b>77.71%</b>
<b>Assemb. Tact</b>	94.41%	96.12%	<b>96.15%</b>	94.87%	95.00%	94.62%

**Table 5.** Percentage of predictions dropped when excluding uncertain predictions in the dropout model.

	Baseline	Predictive	Aleatoric	Epistemic	Variance	Credible
<b>Conf. Room</b>	-	6.43%	6.38%	5.48%	5.67%	4.30%
<b>Copy Room</b>	-	2.96%	3.18%	4.87%	5.43%	8.67%
<b>Hallway</b>	-	4.84%	4.89%	5.54%	5.68%	6.96%
<b>Lounge</b>	-	2.94%	3.04%	5.22%	5.55%	10.19%
<b>Office</b>	-	4.37%	4.45%	4.97%	5.24%	6.71%
<b>Open Space</b>	-	6.09%	6.14%	5.92%	5.68%	5.21%
<b>Pantry</b>	-	4.37%	4.45%	5.59%	5.84%	7.06%
<b>Assemb. Tact</b>	-	8.44%	8.13%	2.01%	1.69%	9.89%

Dense point clouds are usually generated, when building up an environment model of a factory in order to capture as many details as possible. Thus, it is important to keep a high number point wise predictions after uncertainty estimation in order to generate a high quality factory model. However, a higher prediction accuracy in the segmentation step also increases the quality of the resulting environment model. A higher accuracy can be achieved by dropping a higher number of uncertain predictions, as we already discussed. In the case of environment modelling, it is vital to drop as few predictions as possible, because, otherwise, building structures and their exact location that is necessary for model generation can get lost.

Generally, we notice that the dropout model is more difficult to train than the Bayesian one, which manifests in a higher epistemic uncertainty in the dropout model. Empirically, it is shown that the application of dropout exhibits inferior performance in convolutional architectures [3], which could lead to the increased epistemic uncertainty values. Further, the impact of uncertainty on the segmentation performance is more striking in the Bayesian model. However, for applications where one or two percent of accuracy can be sacrificed, the dropout model is a good alternative to the Bayesian model, as users can take on a frequentist network and just add dropout during training and test time, without having to define and optimize a distribution over all the network parameters.

Overall, it can be concluded that Bayesian PointNet has superior performance and dropout PointNet has similar performance to the frequentist model without considering uncertainty information. When only considering certain predictions, Bayesian, as well as dropout PointNet, clearly surpass the performance of the frequentist model. In terms of uncertainty measure, the best results are achieved when using the approach using confidence intervals as well as predictive or aleatoric uncertainty. However, the confidence interval based method drops considerably more predictions in some of the examples. Figure 5 displays one tact of car body assembly, where certain predictions are displayed in black and uncertain predictions are displayed in red. The applied model corresponds to Bayesian PointNet with the credible interval based uncertainty measure. It shows that the network is certain about the majority of its predictions. Uncertain predictions are

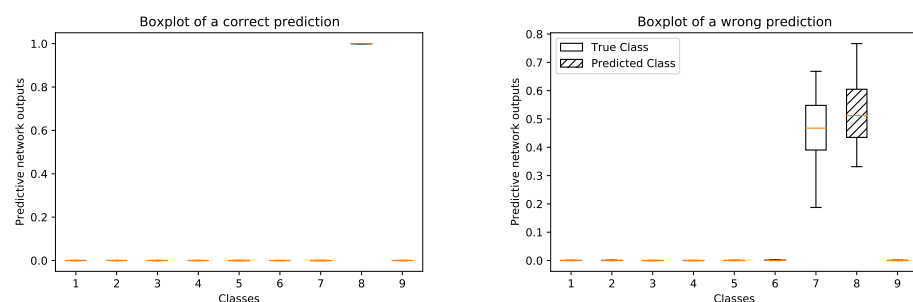


concentrated at the ceiling, wall, and columns, which is in line with our expectations of Section 4.2. Because the ceiling, but especially the walls and columns, are hung with clutter objects, this leads to uncertain predictions, as the point clouds of these classes are incomplete due to holes.



**Figure 5.** Visualization of an assembly tact of the test set, where certain predictions are displayed in black and uncertain predictions are displayed in red. In this case, the network (un)certainty is evaluated by using the method based on credible interval estimation.

Figure 6 shows the boxplots of the predictive softmax outputs of the Bayesian model for a correct and wrong prediction of two single points in the automotive factory data set. In Figure 6a, it can be seen that the network is certain about its correct prediction, i.e., all of the predictive softmax output values of the correct class are close to one, while the network outputs for all other classes are close to zero. In the case of a wrong prediction, see Figure 6b, the boxes of the true and predicted label overlap, which indicates an uncertain prediction. The white box corresponds to the correct class and the shaded box corresponds to the wrongly predicted class.



**Figure 6.** Boxplot of the predictive softmax outputs of Bayesian PointNet: (a) Boxplot of a correct prediction. The network is certain about its prediction as nearly all the probability mass is put on the correct class. (b) Boxplot of a wrong prediction. The network is uncertain about its prediction as the boxes of the true class and the predicted class overlap. The true class is represented by the white box and the wrongly predicted class is illustrated by the shaded box.

## 6. Discussion and Conclusions

We describe how Bayesian segmentation can be applied in order to facilitate the factory planning process in large-scale automotive assembly plants and how to make use of the uncertainty information that is gained by the Bayesian approach. The described use case focuses on the generation of an environment model on the basis of raw point clouds in order to determine the as-is state of a production plant. Therefore, we present a novel Bayesian neural network that is capable of 3D deep semantic segmentation of raw point clouds. This approach allows for the estimation of the uncertainty in the network predictions. Additionally, a network using dropout training to approximate Bayesian variational inference in Gaussian processes is described and compared to the Bayesian model, as well as to a frequentist baseline. In order to evaluate the models, the publicly available S3DIS data set and a manually collected data set of an automotive assembly plant are used.

Different measures of uncertainty quantification are contrasted for the Bayesian as well as the approximate Bayesian network. Three different entropy related uncertainty measures are considered, which enable us to distinguish between overall, data and model related uncertainty. Further, uncertainty quantification based on the variance and credible intervals on the network outputs are investigated. The Bayesian and the dropout model both effectively increase the network performance during test time when compared to the frequentist framework when taking the information gained by any of the uncertainty measures into account. The dropout model's performance is on-par with the frequentist baseline without taking network uncertainty into account. However, the Bayesian neural network outperforms the frequentist baseline, even without considering uncertainty. It is more robust against overfitting and allows us to work with fewer example data, due to prior information acting like additional observations, while the computational complexity basically stays the same.

The use of Bayesian neural networks instead of frequentist ones enables the quantification of network uncertainty. On the one hand, this leads to more robust and accurate models. On the other hand, in safety critical applications, uncertain predictions can be identified and treated with special care. For instance, uncertain predictions can be handed to a human operator for further treatment or trigger a pre-defined safety state. Autonomous driving, collaborative robotics, and medical diagnosis are important example applications, where uncertainty quantification plays a major role. However, many other domains without safety criticality can profit from determining network uncertainty as well. For instance, post offices sorting letters according to their ZIP code can process uncertain results with more complex statistical methods that are too time consuming for regular application. Similarly, Bayesian neural networks increase model accuracy in factory planning. However, the formulation of such a Bayesian segmentation network requires in depth mathematical knowledge and it is therefore more difficult to update. Thus, dropout training can be a good alternative, as it enables the quantification of uncertainty, while the network structure and optimization stay the same as in the frequentist model. The estimation of uncertainty information in modelling production sites increases model accuracy and it can lead to more accurate reconstructions of the real-world production system in a simulation engine or in CAD software. However, there is a trade-off between increasing the network's accuracy by discarding uncertain predictions and having a less accurate model, because too many predictions have been discarded. Therefore, we come to the conclusion that it is better to set a higher threshold for dropping uncertain predictions, i.e., fewer predictions are discarded. There is a slight loss in network accuracy; however, whole objects or building structures can get lost, when too many predictions are discarded. While all of the discussed uncertainty measures are apt for improving network performance, the most promising results are achieved using predictive and aleatoric uncertainty as well as the credible interval based method. Because the former methods allow us to set a distinct threshold for considering predictions as uncertain, which is not possible in the credible interval based method, we conclude that they are best suited for our factory planning use case.

A first methodology for systematic data collection and processing in a large-scale industrial environment was presented in [2]. On the process side, all of the non-highlighted steps that are illustrated in Figure 1 are described in more detail in future work, including technology specifications and a mathematical concept for the placement of the segmented objects in an environment model. Further, different digitalization strategies and registration approaches are discussed. Pose estimation is achieved using a clustering based routine paired with different point cloud registration strategies. Further, the economic potential of this approach will be evaluated for an exemplary assembly plant. With respect to mathematical concepts, the Bayesian neural network can be extended in a way that the parameters of the prior distribution of the network weights and biases are not treated as hyper parameters of the network. A separate prior can be placed over the prior parameters in order to estimate these quantities in a Bayesian way. Further, the application of network uncertainty for point cloud cleaning will be evaluated.

**Author Contributions:** The authors confirm contribution to the paper as follows: Conceptualization, C.P.; methodology, C.P.; software, C.P.; validation, C.P.; formal analysis, C.P. and J.P.; investigation, C.P.; resources, C.P.; data curation, C.P.; writing—original draft preparation, C.P. and J.P.; writing—review and editing, J.P.; visualization, C.P.; supervision, J.P.; project administration, C.P. and J.P. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

2D	Two-Dimensional
3D	Three-Dimensional
BNN	Bayesian Neural Network
CAD	Computer-Aided Design
DoF	Degrees-of-Freedom
ELBO	Evidence Lower Bound
HMC	Hamiltonian Monte Carlo
INLA	Integrated Nested Laplace Approximation
IoU	Intersection over Union
KL	Kullback-Leibler
MCMC	Markov Chain Monte Carlo
OEM	Original Equipment Manufacturer
ReLU	Rectified Linear Unit
RGB	Red Green Blue (colour model)
RGB-D	RGB-Depth
S3DIS	Stanford Large-Scale 3D Indoor Spaces Data Set
VI	Variational Inference

## References

1. Kuhn, W. Digital factory-simulation enhancing the product and production engineering process. In Proceedings of the 2006 Winter Simulation Conference, Monterey, CA, USA, 3–6 December 2006; pp. 1899–1906.
2. Petschnigg, C.; Bartscher, S.; Pilz, J. Point Based Deep Learning to Automate Automotive Assembly Simulation Model Generation with Respect to the Digital Factory. In Proceedings of the 2020 9th International Conference on Industrial Technology and Management (ICITM), Oxford, UK, 11–13 February 2020; pp. 96–101.
3. Gal, Y.; Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1050–1059.
4. Sensoy, M.; Kaplan, L.; Kandemir, M. Evidential deep learning to quantify classification uncertainty. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 3179–3189.
5. Banks, V.A.; Plant, K.L.; Stanton, N.A. Driver error or designer error: Using the Perceptual Cycle Model to explore the circumstances surrounding the fatal Tesla crash on 7th May 2016. *Saf. Sci.* **2018**, *108*, 278–285. [[CrossRef](#)]
6. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
7. Steinbrener, J.; Posch, K.; Pilz, J. Measuring the Uncertainty of Predictions in Deep Neural Networks with Variational Inference. *Sensors* **2020**, *20*, 6011. [[CrossRef](#)] [[PubMed](#)]
8. Chen, X.; Ma, H.; Wan, J.; Li, B.; Xia, T. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1907–1915.
9. Qi, C.R.; Su, H.; Nießner, M.; Dai, A.; Yan, M.; Guibas, L.J. Volumetric and multi-view cnns for object classification on 3d data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5648–5656.
10. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
11. Zhou, Y.; Tuzel, O. Voxnet: End-to-end learning for point cloud based 3D object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4490–4499.
12. Lei, H.; Akhtar, N.; Mian, A. SegGCN: Efficient 3D Point Cloud Segmentation With Fuzzy Spherical Kernel. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 11611–11620.

13. Feng, D.; Rosenbaum, L.; Dietmayer, K. Towards safe autonomous driving: Capture uncertainty in the deep neural network for lidar 3d vehicle detection. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 3266–3273.
14. Yang, B.; Luo, W.; Urtasun, R. Pixor: Real-time 3D object detection from point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 7652–7660.
15. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5099–5108.
16. Ravanbakhsh, S.; Schneider, J.; Poczós, B. Deep learning with sets and point clouds. *arXiv* **2016**, arXiv:1611.04500.
17. Xie, Y.; Tian, J.; Zhu, X. Linking Points With Labels in 3D: A Review of Point Cloud Semantic Segmentation. *IEEE Geosci. Remote Sens. Mag. (GRSM)* **2020**, *8*, 38–59. [[CrossRef](#)]
18. Avetisyan, A.; Dahnert, M.; Dai, A.; Savva, M.; Chang, A.X.; Nießner, M. Scan2cad: Learning cad model alignment in rgb-d scans. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 16–20 June 2019; pp. 2614–2623.
19. Avetisyan, A.; Dai, A.; Nießner, M. End-to-end cad model retrieval and 9dof alignment in 3d scans. In Proceedings of the IEEE International Conference on Computer Vision, Seoul, Korea, 27 October–2 November 2019; pp. 2551–2560.
20. Rusu, R.B.; Bradski, G.; Thibaux, R.; Hsu, J. Fast 3D recognition and pose using the viewpoint feature histogram. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 2155–2162.
21. Aldoma, A.; Vincze, M.; Blodow, N.; Gossow, D.; Gedikli, S.; Rusu, R.B.; Bradski, G. CAD-model recognition and 6DOF pose estimation using 3D cues. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011; pp. 585–592.
22. Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; Wierstra, D. Weight uncertainty in neural networks. *arXiv* **2015**, arXiv:1505.05424.
23. Graves, A. Practical variational inference for neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Granada, Spain, 12–17 December 2011; pp. 2348–2356.
24. Brooks, S.; Gelman, A.; Jones, G.; Meng, X.L. *Handbook of Markov Chain Monte Carlo*; CRC Press: Boca Raton, FL, USA, 2011.
25. Gelfand, A.E.; Smith, A.F. Sampling-based approaches to calculating marginal densities. *J. Am. Stat. Assoc.* **1990**, *85*, 398–409. [[CrossRef](#)]
26. Hastings, W.K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **1970**, *57*, 97–109. [[CrossRef](#)]
27. Duane, S.; Kennedy, A.D.; Pendleton, B.J.; Roweth, D. Hybrid monte carlo. *Phys. Lett. B* **1987**, *195*, 216–222. [[CrossRef](#)]
28. Rue, H.; Martino, S.; Chopin, N. Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2009**, *71*, 319–392. [[CrossRef](#)]
29. Blei, D.M.; Kucukelbir, A.; McAuliffe, J.D. Variational inference: A review for statisticians. *J. Am. Stat. Assoc.* **2017**, *112*, 859–877. [[CrossRef](#)]
30. Der Kiureghian, A.; Ditlevsen, O. Aleatory or epistemic? Does it matter? *Struct. Saf.* **2009**, *31*, 105–112. [[CrossRef](#)]
31. Gal, Y.; Islam, R.; Ghahramani, Z. Deep bayesian active learning with image data. *arXiv* **2017**, arXiv:1703.02910.
32. Bishop, C.M. *Pattern Recognition and Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2006.
33. Armeni, I.; Sener, O.; Zamir, A.R.; Jiang, H.; Brilakis, I.; Fischer, M.; Savarese, S. 3D semantic parsing of large-scale indoor spaces. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 1534–1543.
34. FARO Laser Scanner Focus3D X 130 HDR. The Imaging Laser Scanner. Available online: <https://faro.app.box.com/s/lz4et2dd6zxx2dwtijmxgvu7yi3m9tve/file/441635448354> (accessed on 25 December 2020).
35. Nikon D5500 Technical Specifications. Available online: [https://www.nikon.co.uk/en\\_GB/product/discontinued/digital-cameras/2018/d5500-black#tech\\_specs](https://www.nikon.co.uk/en_GB/product/discontinued/digital-cameras/2018/d5500-black#tech_specs) (accessed on 25 December 2020).
36. Sony Alpha 7R II Technical Specifications. Available online: <https://www.sony.com/electronics/interchangeable-lens-cameras/ilce-7rm2/specifications> (accessed on 25 December 2020).
37. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Wallach, H., Larochelle, H., Beygelzimer, A., d’Alché-Buc, F., Fox, E., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2019; pp. 8024–8035.