An Empirical Review of Uncertainty Estimation for Segmentation Quality Control in CAD with Point-Based Graph Neural Networks

Gerico Vidanes, David Toal, Daniel Xu Zhang, Andy Keane, Marco Nunez, Jon Gregory

Computational Engineering and Design Group
University of Southampton

December 3, 2024

Abstract

Deep neural networks are able to achieve high accuracy in automated feature recognition or semantic segmentation of geometries used in computational engineering. Being able to recognise abstract and sometimes hard to describe geometric features has applications for automated simulation, model simplification, structural failure analysis, meshing, and additive manufacturing. However, for these systems to be integrated into engineering workflows, they must provide some measures of predictive uncertainty such that engineers can reason about and trust their outputs. This work presents an empirical study of practical uncertainty estimation techniques that can be used with pre-trained neural networks for the task of boundary-representation model segmentation. A point-based graph neural network is used as a base. Monte-Carlo (MC) Dropout, Deep Ensembles, test-time input augmentation, and post-processing calibration are evaluated for segmentation quality control. The Deep Ensemble technique is found to be top performing and the error of a human-in-the-loop system across a dataset can be reduced from 3.8% to 0.7% for MFCAD++ and from 16% to 11% for Fusion360 Gallery when 10% of the most uncertain predictions are flagged for manual correction. Models trained on only 5% of the MFCAD++ dataset were also tested, with the uncertainty estimation technique reducing the error from 9.4% to 4.3% with 10% of predictions flagged. Additionally, a point-based input augmentation is presented; which, when combined with MC Dropout, is competitive with the Deep Ensemble while having lower computational requirements.

1 Introduction

Feature recognition (or semantic segmentation) of engineering geometry is a widely useful capability. One of the first applications of this was for the automated transition between computer-aided design (CAD) models and computer-aided manufacturing and process planning (Shah et al. (2000); Al-Wswasi et al. (2018)). Later, feature recognition was also used for automated analysis; where detected features are used to aid in downstream meshing, simulation, and post-processing (Zhang et al. (2014)). With the development and wider use of geometric deep learning within computational engineering (Lambourne et al. (2021); Colligan et al. (2022); Zhang et al. (2022); Vidanes et al. (2024); Cao et al. (2020); Jayaraman et al. (2021)), the recognised features could be more complex and abstract. This opens the door to future use cases like detecting structural failure or features which cause problems in meshing or additive manufacturing.

While the development of the underlying predictive models - neural networks (NN) - is proceeding rapidly in the literature, consideration for how these can be properly integrated into the engineering workflow is lacking. Despite being highly accurate and flexible, these models are not perfect since they are fundamentally statistical models (Goodfellow et al. (2016)). Coupled with their end-to-end nature, the basic system simply presents the engineer with a dense set of semantic segmentation predictions which may or may not be correct. Taking these at face value, errors in recognised features can, for example, lead to errors in the analysis models being built from these tags. In the best case this can cause simulations to fail, and in the worst case can be silent errors which give misleading simulation results. This is exacerbated when an input geometry is outside of their training distribution[1]. In contrast, traditional or algorithmic feature recognition approaches give engineers some confidence in their outputs. Unfamiliar inputs tend to produce runtime errors or simply produce blank labels which can be easily caught downstream.

To combat this and to move towards more robust and useable deep learning systems for engineering workflows, the current work studies NN uncertainty estimation techniques in so far as they can be used to make decisions about NN outputs. Essentially, an uncertainty (or confidence) value can be given to each NN prediction such that it is correlated to the likelihood of its correctness. Predictions that are very uncertain are then likely to be incorrect and thus can be discarded or flagged to the engineer for correction. Work on this area has been increasing, but as the survey from Gawlikowski et al. (2023) has identified, the literature is lacking on the validation of existing methods over real-world problems, especially for the 3D domain.

To the best of the authors' knowledge, this is the first application of uncertainty / confidence estimation techniques to NNs for 3D CAD segmentation or processing in general. Therefore, an empirical review of practical techniques is presented and the implications to engineering workflows is discussed in detail. This work is placed as an initial exploration of the space to be used as a starting point for further detailed research. Additionally, a novel test-time augmentation which involves repeated stochastic encoding of the 3D CAD model into a point cloud is presented as an uncertainty estimation technique. The paper is structured as follows. Section 2 discusses recent work that is directly related and the gaps which the current work fills. Section 3 provides necessary background on the data representation and NN inference. Section 4 discusses the uncertainty estimation techniques being reviewed and how they have been implemented in the current work. The experimental framework is discussed in Section 5, with the results being presented in Section 6. Discussions of the overall results and limitations are presented in Section 7 and conclusions are summarised in Section 8. Finally, some future work is suggested in Section 9.

The specific novel contributions of this work is as follows:

- Presenting an empirical review of uncertainty estimation techniques applied to quality control of semantic segmentation of 3D CAD geometries. Specifically using a point-based GNN.
- Point resampling is proposed as a test-time augmentation for uncertainty estimation.
- MC Dropout (Gal and Ghahramani (2016)) is combined with the point resampling test-time augmentation and is shown to be top performing across two large publicly available datasets.

[1] Whether an input is out-of-ditribution or not is also difficult to know a-priori and is a research area in itself (Yang et al. (2024)). Uncertainty estimation techniques can also be used for this.

2 Related Work

Much work has been done on uncertainty quantification for NNs in general (Gawlikowski et al. (2023)), with an important work in this space being the thesis by Gal (2016) which presents practical techniques and a rich review. Works which use convolutional / bottleneck like networks for computer vision are relevant to this work. One of the first was Kendall et al. (2015) which applied MC Dropout (Gal and Ghahramani (2016)) to a convolutional neural network (CNN) - however, only improvements in semantic segmentation accuracy were evaluated and per-pixel uncertainty masks were simply used for qualitative analysis. Filling this gap, Mukhoti and Gal (2018) present metrics for evaluating uncertainty estimation techniques in terms of how well they (inversely) correlate with semantic segmentation accuracy. They also compare MC Dropout and Concrete Dropout (Gal et al. (2017)) with standard NN inference and show improvement. Another important practical technique is the Deep Ensemble by Lakshminarayanan et al. (2017); they evaluate this technique on a regression task and an image classification task. They present the accuracy of predictions whose uncertainties pass a range of confidence thresholds, and show that the uncertainties estimated by the Ensemble technique is better than MC Dropout. However, they compare using 10 ensemble models and 10 stochastic forward passes for MC Dropout; these are arguably not comparable in terms of computation required (one can easily produce more MC Dropout samples) and more samples would produce better results as shown in Kendall et al. (2015). Gustafsson et al. (2020) also compares the Deep Ensemble technique with MC Dropout and includes semantic segmentation of a street scene as an evaluation task. They utilise a novel metric, *Area Under the Sparsification Error Curve*, which essentially measures the (reverse) correlation between uncertainty and predictive accuracy via prediction ranking. They also show that the Deep Ensemble is better, but they again compare an ensemble of 10 vs 10 MC Dropout samples. Somewhat contrary to the results just discussed is that from Hubschneider et al. (2019) which compares the MC Dropout, Deep Ensemble, and Gaussian Mixture techniques albeit only for a regression task. They show that MC Dropout is best here even with only 10 stochastic forward passes. The current work adds to this applied literature on a novel use case and input representation.

More recently, work has been done on uncertainty estimation for NNs with 3D unstructured inputs, namely point clouds. Vassilev et al. (2024) use the *KPConv* architecture (Thomas et al. (2019)) as a base and compares the standard probability output with MC Dropout and variational inference via parameter sampling. However, they only evaluate using segmentation accuracy and calibration error which is not directly relevant to this work. Petschnigg and Pilz (2021) instead use a *PointNet* architecture (Qi et al. (2017a)) and again compares standard probability with MC Dropout and variational inference. Relevant here is that they evaluate the accuracy of the predictions which pass an uncertainty threshold, similar to the filtering application in the current work. While these are important first steps into practical uncertainty estimation in the 3D domain, the range of techniques validated is lacking.

Worth mentioning at this stage is the work by Guo et al. (2017) which compares different post-processing calibration methods. These aim to transform the output of the NN such that it better reflects the confidence of the prediction. They propose the simple temperature scaling technique here and show that it is the best across many datasets, including 6 image classification and 4 document classification. Calibration and its evaluation is not directly relevant in this work as will be further discussed later but some of these methods will be tested in the current work for completeness. Another interesting work is by Laves et al. (2019) which applies temperature scaling on the individual outputs of MC Dropout inference before aggregating. They show a decrease in error rate when discarding uncertain predictions, across a range of uncertainty thresholds. The current work

performs a similar combined technique for estimating uncertainty - augmenting the input stochastically and performing dropout at test-time.

Finally, the most relevant work is the recent, large-scale, empirical review on a real use case by Ng et al. (2023). They compare Bayes by Backprop (Blundell et al. (2015)), MC Dropout, Deep Ensemble, and Stochastic Segmentation Networks (Monteiro et al. (2020)) as uncertainty estimation techniques for the quality control of NN medical image segmentation. They show that the Deep Ensemble is best. The current work aims to perform a similar empirical review on a range of techniques but for the 3D CAD application.

3 Background

3D feature recognition with deep learning is a wide field due to the different 3D representations available and the diverse applications. There exists approaches for 3D data encoded as voxels (Zhang et al. (2018); Wang et al. (2017)), triangular surface meshes (Hanocka et al. (2019)), and point clouds (Qi et al. (2017a); Thomas et al. (2019); Qi et al. (2017b); Wang et al. (2019c); Zhao et al. (2021)). On the other hand, this work is focused on CAD where geometry tends to be encoded as boundary representation (b-rep) models - a 3D shape is described by its bounding 2D surface. The surface is described by a parametric function $\mathbf{x}\colon \mathbf{U} \to \mathbb{R}^3$ - i.e. points on the 2D surface embedded in 3D space ($\mathbb{R}^3$) are given by the function $\mathbf{x}(u, v)$ defined on a rectangular parameter domain $\mathbf{U} \in \mathbb{R}^2$ (Piegl and Tiller (1996)). For non-trivial shapes, their bounding surfaces cannot be described by one parameterisation; therefore, they are composed of many patches or 'b-rep faces' (portions of the domain) with each face bounded by edges which are themselves parametric curves.

A b-rep model is a complex data structure, but approaches have been proposed for processing these with NNs. Colligan et al. (2022), Cao et al. (2020), and Jayaraman et al. (2021) treat the b-rep face topology as a graph and process these with graph convolution. Lambourne et al. (2021) also exploits the topology of the b-rep faces but presents a novel way to perform kernel convolution on this. Alternatively, Zhang et al. (2022) and Vidanes et al. (2024) encode the surfaces as point clouds while still preserving information from the b-rep model.

Regardless of the specific approach and 3D encoding used, the overall process of feature recognition when using NNs is the same. In this work, similar to those above, feature recognition is formulated as semantic segmentation of the input (rather than object detection). This process involves dense prediction where each elementary entity - voxels, pixels, mesh faces, points, or b-rep faces - is classified into a category or otherwise given a label. This prediction takes the form of a score per category - $\mathbf{z} \in \mathbb{R}^K$, where K is the number of classes - often called *logits*. Therefore, the NN forward pass or inference can be formalised as

$$f_\theta\colon \mathbf{X} \in \mathbb{R}^N \times \mathbb{R}^D \to \mathbf{Z} \in \mathbb{R}^N \times \mathbb{R}^K$$

where f_θ is the NN parameterised with weights θ, $\mathbf{X}$ is a description of the input as a set of vectors, and $\mathbf{Z}$ is the output giving each of the N entities a logit vector. As an example, $\mathbf{X}$ for a b-rep model could be the set of N b-rep faces each described by D attributes. The argmax within each logit vector then gives the index corresponding to the predicted category.

The current work builds on the NN approach from Vidanes et al. (2024) as its flexibility allows the easy integration and evaluation of a wide range of uncertainty estimation techniques. This relatively simple approach is shown to be competitive with those which directly use the b-rep data. For this, the b-rep model is first encoded into an extended point cloud representation that retains its links with

the b-rep faces - $\mathcal{P} \in \mathbb{R}^N \times \mathbb{R}^{3+D}$, where D is the extra information other than the 3D coordinates and N becomes the number of points. The NN forward pass then becomes:

$$f_{\boldsymbol{\theta}} \colon \mathcal{P} \in \mathbb{R}^N \times \mathbb{R}^{3+D} \rightarrow \mathbf{Z} \in \mathbb{R}^F \times \mathbb{R}^K,$$

where F is the number of b-rep faces. Noting that the output shape is $(F \times K)$ since the network aggregates the point features to their associated b-rep faces to produce direct and differentiable b-rep face predictions.

While label predictions can be simply obtained from the logit vector outputs of the NN, it is often useful to transform this into a vector giving the probability that the entity belongs to each category. This can be done with the softmax function that normalises the input into a vector which sums to one:

$$\sigma_{SM}(\mathbf{z})_i = \frac{e^{\mathbf{z}_i}}{\sum_{j=1}^{K} e^{\mathbf{z}_j}}.$$

The 'probability' of the entity belonging to the predicted class is then $q = \max_k \sigma_{SM}(\mathbf{z})$.

However, literature suggests that this normalised vector should not be interpreted probabilistically as it tends to be uncalibrated and overconfident (Guo et al. (2017); Gal (2016)), especially for new inputs which are not within the training distribution. Therefore, much work has been done on proper NN uncertainty estimation - for a full review see Gawlikowski et al. (2023). Put simply, these techniques aim to provide an uncertainty score for each prediction which captures the uncertainty within the input data, or the model parameters, or both. With this, one can make decisions about the quality of the NN predictions for a given input.

This work chooses to review techniques which require little to no modification of the network architecture and no changes in the training scheme. These could be applied to pre-trained models that engineers already have. Four broad categories of approaches which implement different conceptual representations of uncertainty and represent a range of computational costs are reviewed. These are approximate Bayesian inference (MC Dropout from Gal and Ghahramani (2016)), test-time input augmentation, model ensembling, and post-processing calibration.

It is also worth noting that for semantic segmentation in image or point cloud uses cases, individual pixels or points do not have much significance in themselves. In other words, picking out the most uncertain pixels or points is often not useful for users of the NN and structural metrics are often used (Ng et al. (2023)). In the current application, it is assumed (at least in the first instance) that individual b-rep faces have much more self-contained meaning. It is possible that a feature could be represented by a single b-rep face; however, very complex shapes and features can require many b-rep faces.

4 Uncertainty Estimation

4.1 Post-Processing Calibration

There are methods which aim to transform the outputs of a trained model using a known calibration dataset such that the new probability vectors are well calibrated. Perfect calibration can be defined as

$$\mathbb{P}(\hat{Y} = Y \mid \hat{P} = p) = p, \forall p \in [0, 1]$$

where $\hat{Y}$ is the predicted class, Y is the true class, $\hat{P}$ is the predicted probability or predictive confidence, and p is the true (frequentist) probability (DeGroot and Fienberg (1983); Dawid (1982)). It is uncommon for works on classification/segmentation quality control to include these approaches, but they have been represented here with the reasoning that calibrated probability outputs are more useful in picking out incorrect predictions for quality control. Two methods are used here and explained in the following.

Temperature scaling (Guo et al. (2017)) is a simplified multi-class version of the Platt scaling method (Platt et al. (1999)) for calibrating NN probability predictions that only tunes one parameter, τ[2]:

$$\hat{q} = \max_k \sigma_{SM}(\mathbf{z}/\tau)^{(k)}.$$

The logits from the calibration set geometries are used to tune the temperature scaling parameter by minimising the negative log likelihood loss between $\hat{q}$ and the 'true' probability vector (which is just the one-hot encoded class index). The L-BFGS optimiser in *PyTorch* was used with a learning rate of 0.01 following the example implementation of Guo et al. (2017)[3]. After scaling, the maximum probability, $\hat{q}$, is used as the predictive confidence.

Histogram binning (Zadrozny and Elkan (2001)) is a frequentist approach which bins the 'predicted scores' from a calibration dataset. Given a new test example, it is placed into one of the bins according to its (raw) score. The 'corrected' or calibrated probability that this new test example belongs to the predicted class is the fraction of calibration examples in the same bin of the same predicted class which were correct. Here, the maximum probability, q, was used as the 'predicted scores'[4]. 20 equal-width bins for each calibration set was used.

4.2 Monte-Carlo Dropout

The dropout technique (Srivastava et al. (2014)) randomly 'drops' neurons in a layer (i.e. zeroes out elements in the weight matrices) during training for regularisation. This is nominally not done during 'test-time' or inference and thus the resulting network is effectively averaging the weights of slightly smaller subnetworks. This prevalent technique is one of the factors which allows modern neural networks to have so many layers. Gal and Ghahramani (2016) shows that using this at test-time produces stochastic forward passes which approximates the sampling of weights for the variational inference of Bayesian NNs.

A distribution of vector outputs is obtained for a given input - $f_{\boldsymbol{\theta}_t}(\mathcal{P}) = \mathbf{Z}_t$, where t is the t-th forward pass. This can then be collapsed to a prediction vector by simply obtaining the element-wise mean after applying softmax to each. This vector is treated similarly as that above - the argmax is the predicted class index and the corresponding value is the confidence. Early works show that this aggregated vector, with enough forward passes, is more accurate than basic inference. Kendall et al. (2015) applies this to an encoder-decoder architecture for image segmentation and presents an optimal configuration of dropout layers within the convolutional NN for maximum segmentation accuracy. This was found to be placing dropout layers within the deepest layer(s) of the encoder and decoder units. This was confirmed for the point-based NNs used here (results not shown). 50 stochastic forward passes were used and aggregated here which was found to be sufficient for converged uncertainty estimates and resultant classification accuracy.

[2] τ is used here instead of T from the original work to not confuse with the use of T later in this paper.

[3] See `http://github.com/gpleiss/temperature_scaling`.

[4] This seemed to produce similar but slightly better results than using the maximums within the raw logits

4.3 Test-Time Augmentation - Point Resampling

Another way to obtain a distribution of NN outputs given the same input is to do test-time augmentation (Wang et al. (2019b,a); Gawlikowski et al. (2023)). In the current work, the 'raw' input to the system is a b-rep CAD model but the NN's observation/input is a point cloud, $\mathcal{P}$, sampled from the surface. Therefore, a natural and effective way to do data augmentation is to repeatedly sample $\mathcal{P}$ with the stratified stochastic sampling method proposed by Vidanes et al. (2024). In other words, the network input is not simply transformed to look slightly different but is actually a different instantiation of the same fundamental geometry. For each forward pass, the points seen by the network are different and have no formal correspondence, thus the per-point predictions cannot be aggregated into distributions. However, because the network has a b-rep face prediction head which aggregates relevant points into the face space, this can form a distribution of outputs for each b-rep face - $f_{\boldsymbol{\theta}}(\mathcal{P}_t) = \mathbf{Z}_t$. Similarly to the above, the distribution can be aggregated into one prediction vector per face. 50 stochastic forward passes were also used and was sufficient for convergence.

4.4 Resampling & Dropout

This work also presents a combination method with only a small computation overhead when compared to the individual components. The point resampling test-time augmentation and MC Dropout inference can be used simultaneously to produce a wider variety in the distribution of output logits given a single input and trained NN. Each logit output is produced from a different point cloud (from the same geometry) and with a different sample of network nodes being dropped - $f_{\boldsymbol{\theta}_t}(\mathcal{P}_t) = \mathbf{Z}_t$. As above, 50 stochastic forward passes are used.

4.5 Deep Ensemble

An ensemble of neural networks (Hansen and Salamon (1990)) can also be used to obtain a distribution of outputs given the same input. In this work, an ensemble of models with the same architecture and trained with the same dataset is used following Lakshminarayanan et al. (2017). The models are trained using different random initialisations (and different mini-batch sampling of the dataset) and thus take a different trajectory through weight space[5]. The outputs of each separate neural network for a given input geometry can be treated as samples from a distribution - $f_{\boldsymbol{\theta}_m}(\mathcal{P}) = \mathbf{Z}_m$ for model m - and aggregated as above. 10 models were used.

4.6 Predictive Confidence

As noted above, a predictive confidence in $[0, 1]$ can be obtained as the maximum element in the predicted vector (corresponding to the predicted class index one would obtain with argmax). To formalise this for the methods which produce a distribution of outputs, the predictive confidence is given by

$$\hat{q} = \max_k \left(\frac{1}{T} \sum_t \sigma_{SM}(\mathbf{z}_t) \right),$$

[5] Fort et al. (2019) show that this is sufficient for the models to take different modes in function space. In comparison, while MC Dropout might provide diversity in weight space, they stay in a relatively small subset of function space (i.e. $f_{\theta_1}(\mathbf{x}) = f_{\theta_2}(\mathbf{x})$ even though $\theta_1 \neq \theta_2$).

where T is the number of stochastic forward passes or the number of models in the ensemble.

Alternatively, Gal (2016) and Mukhoti and Gal (2018) use information theoretic (Shannon (1948)) scalars which summarise the uncertainty within the distributions more formally. The predictive entropy can be approximated from this distribution as:

$$\hat{\mathbb{H}} = -\sum_k \left(\frac{1}{T} \sum_t \sigma_{SM}(\mathbf{z}_t) \right) \log \left(\frac{1}{T} \sum_t \sigma_{SM}(\mathbf{z}_t) \right),$$

and represents the combined epistemic and aleatoric uncertainty. Alternatively, mutual information can be approximated as:

$$\hat{\mathbb{I}} = \hat{\mathbb{H}} + \frac{1}{T} \sum_{k,t} \sigma_{SM}(\mathbf{z}_t) \log \sigma_{SM}(\mathbf{z}_t),$$

and represents just the epistemic or model uncertainty. In other words, predictive entropy also captures the uncertainty or spread in the individual predicted vectors whereas mutual information only captures the spread across the predicted vectors. See the thesis by Gal (2016) for a deeper explanation.

The scalars obtained from these are unbounded and need to be normalised to the interval $[0, 1]$ such that they can be interpreted as a probability or confidence. This can be done with a separate calibration set to obtain the range for normalisation; alternatively, a different range can be obtained for each class since they can have separate distributions. A high uncertainty scalar value corresponds to a low predictive confidence therefore a reversed scale min-max normalisation was used.

5 Method

5.1 Base Neural Network

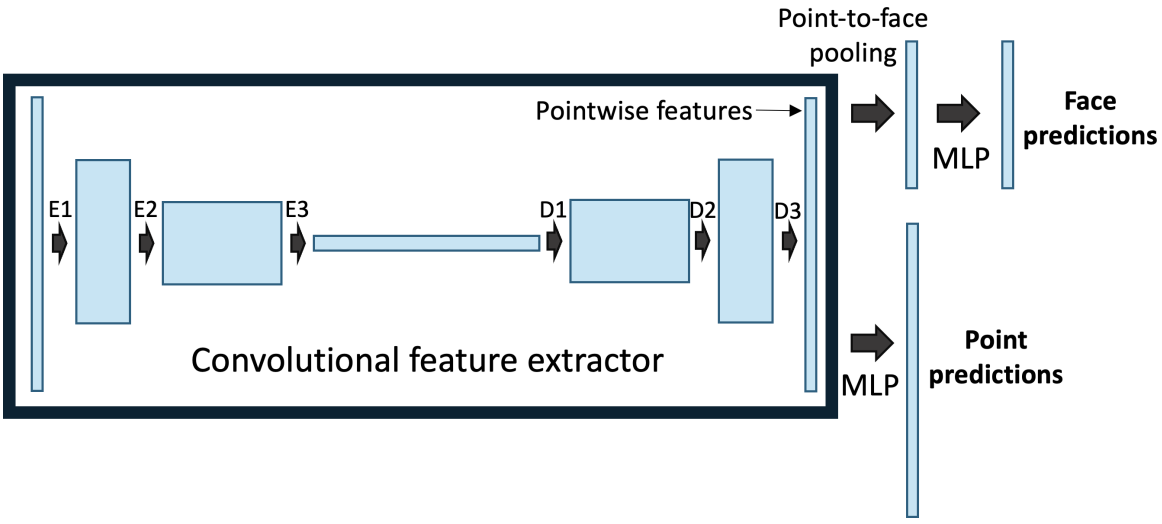


FIGURE 1: Block diagram of neural network architecture. The convolutional-type point-based feature extractor is followed by a multi-head structure. *MLP* = Multi-Layer Perceptron. *EX* = Encoder X. *DX* = Decoder X.

The underlying models used in this work are point-based graph neural networks based on the work from Vidanes et al. (2024), which are an extension on *PointNet++* (Qi et al. (2017b)). The architecture

is illustrated as a block diagram in Figure 1. The unscaled network described in their work was used for computational efficiency - i.e. default depth and width resulting in 1.4M learnable parameters. The 'facewise' prediction branch was included and the average of the point and face loss was used for training. The multi-head structure shown in Figure 1 is only to aid training as discussed in the original work; only the face predictions from the facewise branch are used in the following. To take full advantage of MC Dropout inference, extra dropout layers with a dropout probability of 0.5 were added in the final encoder layer (E3 in Figure 1) and the first decoder layer (D1 in Figure 1) - the optimal configuration suggested by Kendall et al. (2015).

The *ADAM* optimiser (Kingma and Ba (2014)) was used with the same standard training hyperparameters as Vidanes et al. (2024), except that the network weights corresponding to the minimum cross-entropy loss in the validation set were extracted, instead of those corresponding to the maximum b-rep face classification accuracy on the validation set. This was expected to give better calibrated probability outputs on the base model as suggested by Guo et al. (2017).

Following the original work, the b-rep geometries were encoded into 7D point clouds - encoding 3D coordinates, 3D surface normals, and a b-rep face index - using b-rep stratified sampling with at least 4096 points. The suggested b-rep face type feature used in Vidanes et al. (2024) and Colligan et al. (2022) was not used here since it can make the trained networks tied to a specific CAD kernel.

5.2 Experimental Framework

This empirical review includes methods with many layers of stochasticity. It is well-known that NN training is stochastic due to the random mini-batching, weight initialisation and dropout layers. Moreover, this work is interested in estimating the 'real-world' performance of the above methods. From this perspective, the evaluation of trained NNs is also stochastic since the training, validation, and testing datasets are samples from the underlying distribution or pattern being learned. On top of this, some uncertainty estimation methods being reviewed here are inherently stochastic. To maximise the reliability of the results, many repetitions and cross-validations are needed to capture the stochasticity in the performance metrics. The following details the individual layers of evaluation repetitions and what aspects of randomness they are trying to capture.

Multiple models are necessary for the Deep Ensemble inference approach, therefore 10 separate models were trained on the same training and validation data. For the other approaches, the following was also repeated for each model with results being aggregated to capture the variation caused by the random weight initialisation and the random mini-batch sampling. Resampling cross-validation (CV) was used with a separate set of 3000 unseen geometries to estimate the unbiased performance of the methods - i.e. not tied to the specific geometries in the dataset splits. For each CV run, 1000 geometries were randomly sampled from this pool and used to compute the performance metrics, with the remaining 2000 geometries being used as a calibration set where required. As an example, for the histogram binning approach, during each CV run, the bins were computed independently using the samples within the corresponding calibration set. 20 CV runs were done. Finally, the sampling of a set of stochastic forward passes and aggregation was repeated 100 times. While the estimates are converged with $T = 50$, in the sense that it remains stable with increasing T, there is still some variation in the result when sampling a different set of 50 stochastic NN outputs. As will become apparent in the next section, many repetitions here do not incur a high computational cost.

In summary, for the Deep Ensemble method there were 20 separate evaluations being aggregated (due to CV runs). For the baseline and post-processing calibration methods, 200 evaluations were

being aggregated across the different models. For the MC Dropout, resampling, and combined methods, 20000 evaluations were aggregated since each set of $T = 50$ forward passes is treated as a separate sample.

5.3 Computational Implementation

The experimental framework detailed above together with the size of the datasets being used here results in this being a significant computational task with compute, memory, and time trade-offs. This section details the data generation approach to efficiently obtain the performance metrics.

Studying the inference and uncertainty estimation techniques, one can see that there are overlaps in the required computations. For instance, a single standard NN forward pass produces a logit vector which is directly used for the baseline case, used for transformation in the post-processing calibration methods, and can be aggregated with other forward passes to obtain a prediction for the Deep Ensemble and point resampling methods. In addition, because resampling CV is being used, the same geometry could be present in multiple CV runs either in the testing or calibration set. With this in mind, the logit vectors produced from each geometry from different models could be stored and re-used to save on computation and time at the cost of memory and/or disk space.

Of course extra computation, and large memory requirements in some cases, is required for performing the 'simulation' of the different inference methods from this pool of data. But an extra advantage of this approach is that the logit pool generation step is trivially parallelisable - the generation of each logit is independent. For this work, the *IRIDIS 5* compute cluster at the University of Southampton was utilised; logit generation was parallelised on the multicore machines as well as being run across multiple nodes. In practice, inference of multiple models on one geometry was done in series within a process such that the b-rep geometry was only transformed into a point cloud once (for that process). This saves on computation and ensures that one set of logits from the 10 models for one geometry was from the same point cloud, without needing to repeatedly re-seed the random number generator.

Figure 2 illustrates the pool of logit vectors created by passing the 3000 geometries through each of the 10 models multiple times with dropout either on or off. To obtain a single prediction with accompanying confidence estimate, the appropriate subset of the data pool is sampled. For instance, to simulate MC Dropout inference, T logit vectors for a given point cloud are sampled from the subset produced by a model with dropout on. The prediction is obtained by averaging across T and the confidence can be obtained from either the mean vector or the predictive entropy or mutual information estimates. A similar process is done for simulating the combined (point resampling and MC Dropout) case but logit vectors from models with dropout on across different point encodings of a single b-rep geometry are used. To simulate inference with an ensemble of 5 models, the logit vectors produced by a random 5 models (with dropout off) for the same geometry are sampled. This can then be aggregated as before to obtain a mean prediction and confidence estimate. Not only does this allow for efficient evaluation of the different methods, the recombining of stored logit vectors allows convergence studies to be easily performed with increasing T or number of models. Many CV runs can also be trivially performed by sampling the logits of appropriate subsets of geometries. For instance, for temperature scaling, one can obtain the logit vectors (produced by a model with dropout off) from a sample of 2000 geometries and tune the τ parameter. This can then be used to transform the logits from the remaining 1000 geometries from the pool and a set of metrics can be computed. Figure 3 shows a sample of the data being stored by the parallel workers.

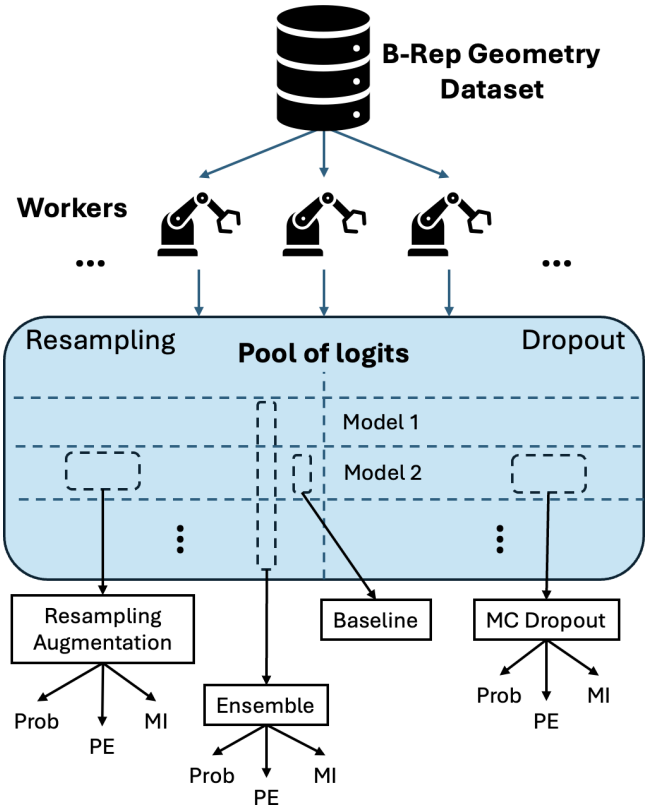


FIGURE 2: Simplified illustration of data flow from the dataset of b-rep geometries to re-usable logit vectors for simulating different inference methods.

	model	sampling_seed	dropout	geom_id	face_idx	gt	logit_0	logit_1	logit_2	...
0	2	674885	False	45559	0	24	-11.395023	-9.771173	-6.572227	...
1	2	656221	False	45559	1	3	-5.850572	-6.814668	-3.885224	...
2	2	833777	False	45559	2	0	20.654837	-3.994620	3.113753	...
3	2	747463	False	45559	3	3	-3.582532	-5.484285	-3.095505	...
4	2	464341	False	45559	4	3	-4.179116	-5.199742	-2.358987	...

FIGURE 3: Sample of logit vector dataset with metadata annotations.

5.4 3D CAD Datasets

This work uses two large and publicly available datasets of 3D CAD geometries with semantic segmentation labels. MFCAD++ from Colligan et al. (2022) is an algorithmically generated dataset where each b-rep face in the model is labelled with the manufacturing operation which created it. The geometries start as a 'stock' cuboid and manufacturing operations are simulated by applying various (random) cuts in series. There are a total of 25 classes - 24 machining features plus any remaining 'stock' faces. They provide lists of geometries for the training, validation, and test splits with 41766, 8950, and 8949 geometries respectively. The entire training and validation split was used for NN parameter tuning and early stopping. While, as mentioned previously, 3000 geometries from the test set are used for evaluation of the uncertainty estimation methods. The number of faces per geometry is approximately normally distributed with a mean of 30, ranging between 6 and 86.

The Fusion360 Gallery Segmentation Dataset from Lambourne et al. (2021) is a collection of user submitted geometries with faces labelled according to the CAD modelling operation which created it. There are 8 possible classes - 'Extrude Side', 'Extrude End', 'Cut Side', 'Cut End', 'Fillet', 'Chamfer',

'Revolve Side', and 'Revolve End'. This labelling is inherently ambiguous since the same 3D shape can be obtained with different sequences of modelling operations; this is noted in their work. The public release only provides a list of geometries for the train and test split with 30314 and 5366 geometries respectively. In the current work, the provided 'training' geometries were randomly split with a 85/15 ratio for use as a training and validation set. As before, 3000 geometries from the unseen test split are used for the evaluations in the next section. The mean number of faces per geometry is around 15, but the distribution is dramatically skewed with a range from 1 to 421.

6 Results

6.1 Predictive Accuracy Correlation

As hinted at in Section 2, the field of applying uncertainty estimation for semantic segmentation has not yet settled on standard evaluation metrics. In this section, the popular metrics from Mukhoti and Gal (2018) are used - with their relevance to this work discussed at the end of the section. As discussed in this and the referenced work, the metrics are grounded in their correlation to the semantic segmentation accuracy. The $P(\text{accurate} \mid \text{certain})$ and $P(\text{uncertain} \mid \text{inaccurate})$ conditional probabilities were proposed and can be computed respectively as: of those predictions deemed 'certain' what fraction were correct, and of those predictions which were incorrect what fraction were deemed 'uncertain'. Mukhoti and Gal (2018) also combine these metrics into the 'PAvPU'[6] metric. It can be computed as the fraction of all predictions which are either correct and 'certain', or incorrect and 'uncertain'. Therefore, it can be interpreted as the correlation between the uncertainty estimate and predictive accuracy - i.e. a value of 1.0 means that all accurate predictions are 'certain', and all incorrect predictions are 'uncertain' with no overlap. Naturally, these rely on the definition of 'certainty' which here, and in most of the literature, is provided by a confidence (or uncertainty) threshold. In the following, curves are produced to cover the range of confidence threshold values and the area-under-the-curve (AUC) is used as the summary metric for ranking methods.

Figure 4 summarises the performance of the uncertainty estimation techniques. The traditional class probability q was used as the baseline. For all inference methods, the maximum element of the (mean) probability vector has been used as the predictive confidence. For each model and CV run (and sample of stochastic forward passes) the metrics were computed across all b-rep faces in the testing set regardless of class - i.e. 'micro' averaging - giving a distribution for each uncertainty estimation technique. A line representing the random case is added - where uncertainty values for each prediction are sampled from a uniform distribution over [0,1). The middle column of Figure 4 shows results from models which were trained on 5% of the original MFCAD++ training set - otherwise, the full validation and testing set was still used.

The first thing to notice is that the $P(\text{accurate} \mid \text{certain})$ values of different methods at a confidence threshold of zero (i.e. all predictions pass the threshold) are different. This corresponds to the base accuracy of the predictions. As expected, the post-processing calibration methods have the same value as the baseline here since these do not change the prediction - the probability vector is scaled but the argmax remains the same. On the other hand, the methods which aggregate different predictions can have a different argmax value than a specific individual forward pass, and literature shows that this can improve predictive accuracy which is reflected here. Note that because of this, the

[6]The original acronym stands for "Patch Accuracy vs Patch Uncertainty" since it was computed on image patches. This is not relevant for this work, but the same acronym is used to avoid redefinition.

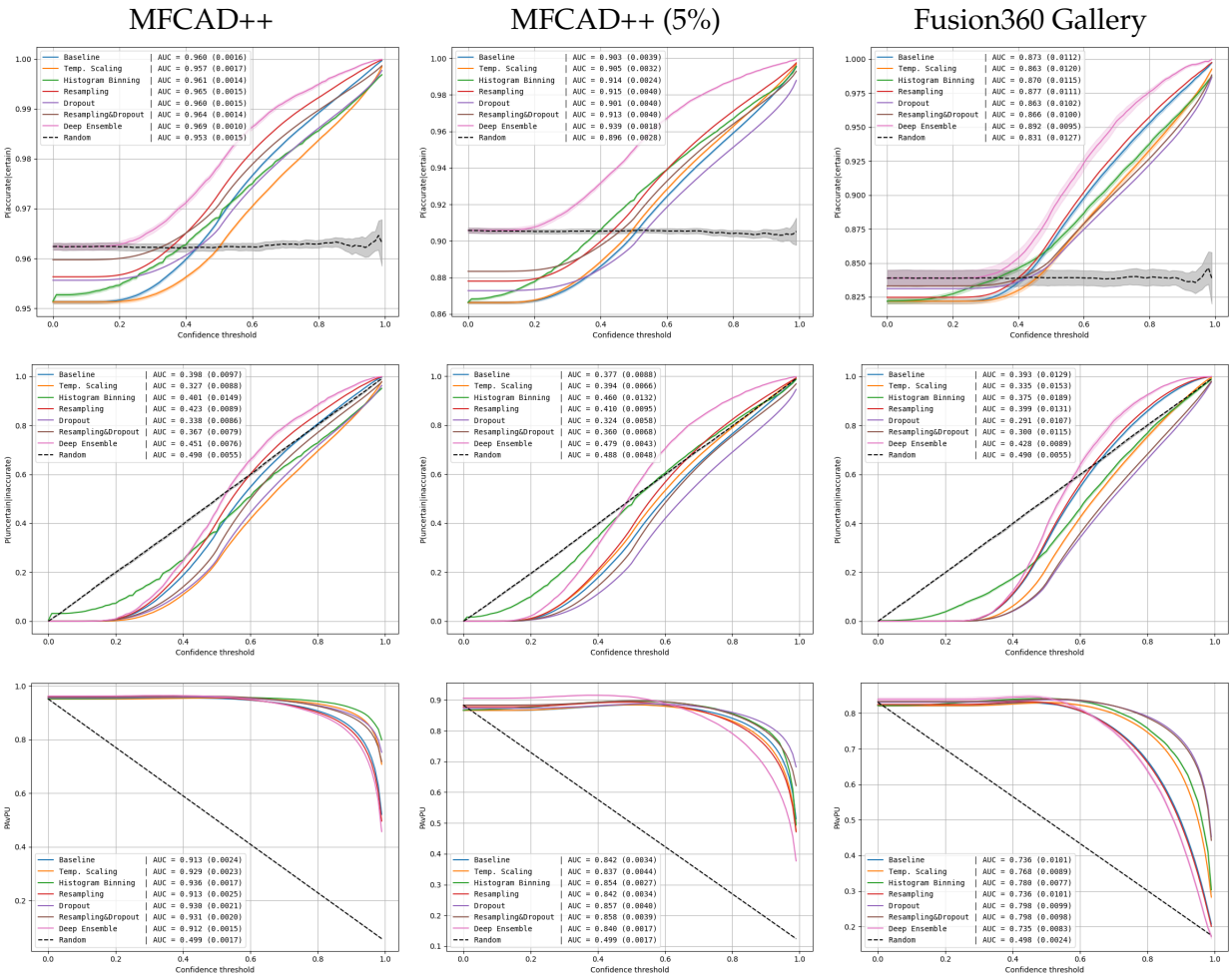


FIGURE 4: Metrics vs confidence threshold curves for different uncertainty estimation techniques evaluated on different datasets. The mean line is shown with +/- 2 standard errors as bounds. AUC format: mean (std.) - higher is better.

random case depends on the set of predictions being used; here, data corresponding to the method giving the best AUC is drawn on Figure 4.

Secondly, it is interesting that the Deep Ensemble method is the best performing for both conditional probabilities across all three test cases but is one of the worst when looking at the PAvPU metric. This is due to the number of predictions which are accurate but uncertain - not accounted for in either of the given conditional probabilities but is accounted for in the PAvPU calculation. The ensemble method was found to be the worst in this case, suggesting that the high precision 'certainty filter' is partly due to a stricter or lower recall filter.

Finally, note the flat portions of most techniques towards lower confidence thresholds; suggesting that the majority of estimated confidences produced are high. This could align with the idea in the literature that NN probabilities are generally overconfident. However, it is also worth noting that the evaluation geometries should be within the training distribution of the models - the dataset are split randomly therefore they are from the same underlying distribution. Intuitively, the minimum confidence of the models given these geometries would not be 0 because of this. Histogram binning stands apart from the rest as having continuously varying results across the thresholds - owing to its particular calibration method. Instead of using transformations or aggregations of the NN outputs,

the confidences are empirically obtained from ratios in the calibration set. It is observed from figure 4 that it performs very well because of this as measured by the PAvPU metric.

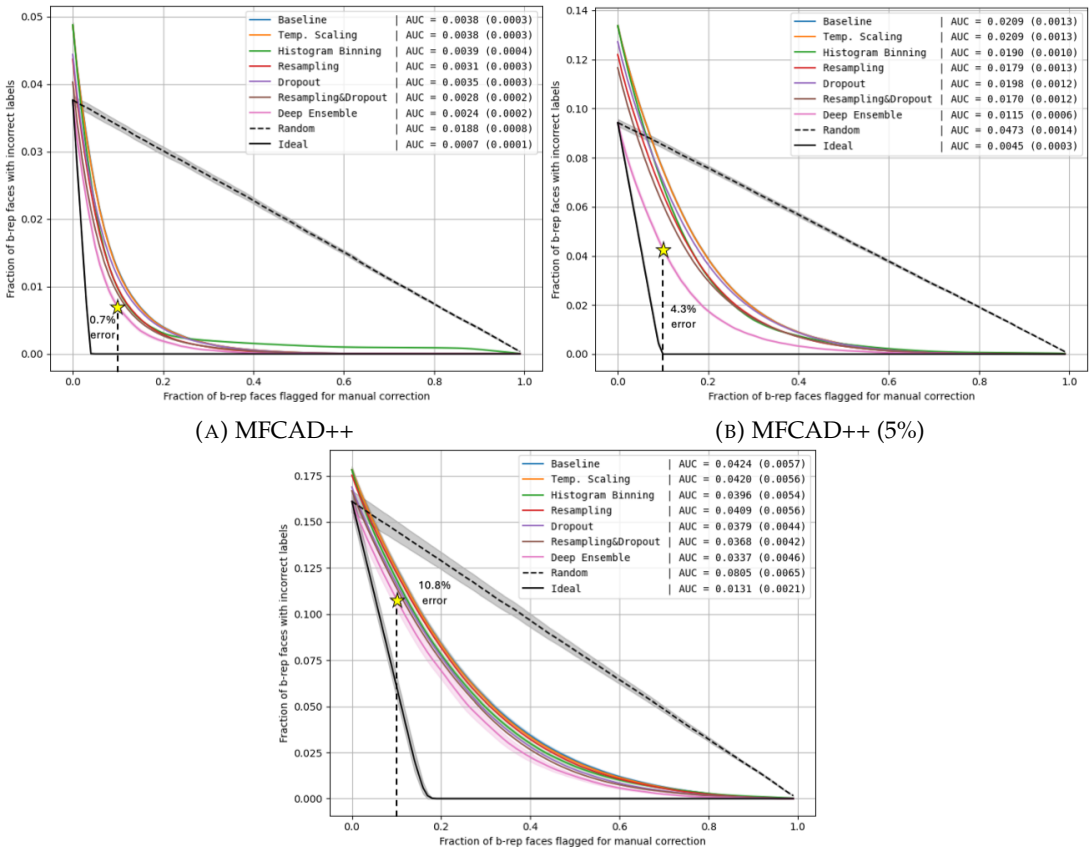
Recalling the application of interest for this work, ranking methods with these metrics alone is insufficient. As suggested by the x-axes in Figure 4, these measure the calibration of the confidence/uncertainty estimates across the whole range of probabilities. While calibration is important, it is not the focus of the application of interest for this work. More directly relevant and easier to interpret metrics are presented and discussed in the following - focusing on the use of the confidence estimate as a ranking rather than a calibrated probability.

6.2 Quality Control

The application of segmentation quality control is mainly concerned with a scalar estimate which can be used to rank predictions such that correct and incorrect ones are well separated. This is a slightly different objective to producing well-calibrated probabilities. To this end, the metrics and evaluation context introduced in Ng et al. (2023) are used in this section. Put simply, the application involves flagging the most uncertain (or least confident) outputs of the system for manual correction such that the overall output of the human-in-the-loop system is more accurate. This represents a semi-automation case where the manual effort of a human expert is ideally concentrated into the most difficult feature recognition cases whilst the system automates those which the NN is confident in. In this section, instead of comparing confidence thresholds (which produces different fractional splits of the predictions depending on the method), the predictions for the faces in the sample of 1000 test geometries are ordered in increasing confidence. A range of fractions (1 to 100 with an increment of 1 in this case) are then specified such that the least confident (or most uncertain) predictions are flagged for 'manual correction'. 'Manual correction' in this case simply means that the predictions become correct regardless of their value - emulating a perfect oracle. The error rate remaining, after correction, for the faces in the test geometries can then be calculated.

Figure 5 summarises the results of this experiment across the different datasets. As before, results are 'micro' averaged for each model, CV run, and sample of stochastic forward passes to give a distribution for each uncertainty estimation technique. Again, the maximum value within the (mean) probability vector has been used here as the confidence for all methods. Two extra cases are also shown for context. The dotted black line represents the case where predictions are flagged for manual correction randomly, regardless of their estimated confidence. The solid black line represents the ideal case where incorrect predictions are always flagged first; therefore this line always crosses the x-axis at the same value as it did for the y-axis intercept. These two cases are again dependent on the underlying predictions used; the method with the best AUC is used to draw these on the plots.

From the 'micro' averaged results, Deep ensemble is always best with the combined method following. However, when looking at the per-class results, shown in Table 1, this is not always the case. It is noted that for the full MFCAD++ and the Fusion360 Gallery cases, the combined method is best when measuring using the 'macro' average - metrics for each fraction computer per class individually then averaged. In the Fusion360 case, this is due to the combined method being better for the less common classes. In the MFCAD++ case, the combined method is also best for a number of classes however this is not as clearly separated by sample frequency.



(A) MFCAD++ (B) MFCAD++ (5%)

(C) Fusion360 Gallery

FIGURE 5: Fraction of b-rep faces with incorrect labels after flagging a fraction of predictions for 'manual correction'. The mean line is shown with +/- 2 standard errors as bounds. AUC format: mean (std.) - lower is better. (B) shows results when using models trained on 5% of the MFCAD++ training set. Best improved error rate given 20% of predictions are flagged is also shown.

6.3 Uncertainty/Confidence Estimation

As mentioned in Subsection 4.6, there are a number of ways in the literature to summarise the uncertainty from the distribution of (stochastic) predictions. Table 2 summarises the error remaining after manual correction AUC metrics when using the different uncertainty scalar estimation methods on the set of outputs produced from the different inference methods for each dataset case. It is observed that they perform more or less similarly when taking into account the standard deviations.

It seems that the uncertainty information is sufficiently captured by looking at the maximum element of the mean vector across forward passes - without explicitly incorporating the rest of the vector. However, it is noted that the probability vectors being averaged have already been normalised through the softmax function which does incorporate information from the whole logit vector.

6.4 Convergence

Figure 6 shows that the predictive accuracy and uncertainty scalar estimates are more or less stable after $T = 50$, for all three tested datasets. This is also the case for the MC Dropout and point resampling individual inference methods, not shown. Apparent from these plots is also the range of predictive performance across the trained models.

	Baseline	Temperature Scaling	Histogram Binning	Deep Ensemble	MC Dropout	Resampling	Resampling & Dropout
MFCAD++							
Chamfer	0.008 (0.003)	0.008 (0.003)	0.012 (0.004)	**0.004 (0.001)**	0.013 (0.004)	0.007 (0.003)	0.012 (0.004)
Through hole	0.009 (0.004)	0.010 (0.004)	0.005 (0.002)	0.007 (0.002)	0.003 (0.002)	0.008 (0.003)	**0.002 (0.001)**
Triangular passage	0.008 (0.002)	0.008 (0.002)	0.006 (0.002)	**0.005 (0.001)**	0.006 (0.002)	0.006 (0.002)	**0.005 (0.001)**
Rectangular passage	0.002 (0.001)	0.002 (0.001)	0.002 (0.001)	**0.000 (0.000)**	0.002 (0.001)	0.001 (0.001)	0.001 (0.001)
6-sided passage	0.001 (0.000)	0.001 (0.000)	0.001 (0.000)	0.000 (0.000)	0.001 (0.000)	0.000 (0.000)	0.000 (0.000)
Triangular through slot	0.012 (0.006)	0.012 (0.005)	0.006 (0.004)	0.008 (0.004)	**0.003 (0.002)**	0.012 (0.005)	**0.003 (0.002)**
Rectangular through slot	0.087 (0.024)	0.084 (0.024)	0.066 (0.014)	0.075 (0.014)	0.056 (0.018)	0.082 (0.024)	**0.053 (0.018)**
Circular through slot	0.042 (0.027)	0.043 (0.028)	0.027 (0.014)	0.012 (0.005)	0.003 (0.002)	0.033 (0.025)	**0.002 (0.002)**
Rectangular through step	0.051 (0.015)	0.050 (0.014)	**0.029 (0.008)**	0.036 (0.005)	0.048 (0.014)	0.048 (0.015)	0.046 (0.014)
2-sided through step	0.001 (0.001)	0.001 (0.001)	0.002 (0.001)	**0.000 (0.000)**	0.001 (0.000)	0.001 (0.000)	0.001 (0.000)
Slanted through step	0.029 (0.008)	0.029 (0.008)	**0.017 (0.005)**	0.018 (0.004)	0.021 (0.006)	0.026 (0.007)	0.019 (0.005)
O-ring	0.002 (0.001)	0.002 (0.001)	0.002 (0.001)	**0.001 (0.000)**	0.002 (0.001)	**0.001 (0.001)**	0.002 (0.001)
Blind hole	0.010 (0.003)	0.010 (0.003)	0.012 (0.003)	0.006 (0.002)	0.006 (0.002)	0.007 (0.002)	**0.005 (0.002)**
Triangular pocket	0.005 (0.001)	0.005 (0.001)	0.004 (0.002)	0.004 (0.001)	0.007 (0.002)	0.004 (0.001)	0.006 (0.002)
Rectangular pocket	0.002 (0.001)	0.002 (0.001)	0.003 (0.001)	**0.001 (0.001)**	0.003 (0.002)	**0.001 (0.001)**	0.002 (0.001)
6-sided pocket	0.001 (0.000)	0.001 (0.000)	0.001 (0.001)	**0.000 (0.000)**	0.001 (0.001)	0.001 (0.000)	0.001 (0.001)
Circular end pocket	0.006 (0.002)	0.005 (0.002)	0.006 (0.002)	**0.004 (0.001)**	0.008 (0.003)	**0.004 (0.002)**	0.006 (0.003)
Rectangular blind slot	0.077 (0.021)	0.077 (0.021)	0.056 (0.014)	0.059 (0.013)	0.037 (0.012)	0.071 (0.021)	**0.032 (0.011)**
Vertical circular end blind slot	0.034 (0.015)	0.034 (0.015)	0.020 (0.009)	0.020 (0.006)	0.016 (0.008)	0.028 (0.012)	**0.012 (0.006)**
Horizontal circular end blind slot	0.002 (0.002)	0.002 (0.002)	0.004 (0.003)	**0.001 (0.001)**	0.003 (0.003)	**0.001 (0.001)**	0.002 (0.002)
Triangular blind step	0.003 (0.002)	0.003 (0.002)	0.005 (0.002)	0.002 (0.000)	0.002 (0.001)	0.003 (0.002)	0.002 (0.001)
Circular blind step	0.001 (0.000)	0.001 (0.000)	0.001 (0.001)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Rectangular blind step	0.009 (0.003)	0.009 (0.003)	0.007 (0.002)	**0.004 (0.002)**	0.013 (0.005)	0.007 (0.003)	0.011 (0.004)
Round	0.086 (0.017)	0.085 (0.017)	0.074 (0.016)	**0.060 (0.012)**	0.098 (0.019)	0.065 (0.017)	0.071 (0.019)
Stock	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Overall (micro)	0.004 (0.000)	0.004 (0.000)	0.004 (0.000)	**0.002 (0.000)**	0.003 (0.000)	0.003 (0.000)	0.003 (0.000)
Overall (macro)	0.019 (0.002)	0.019 (0.002)	0.015 (0.002)	0.013 (0.001)	0.014 (0.002)	0.017 (0.002)	**0.012 (0.001)**
Fusion360 Gallery							
ExtrudeSide	0.016 (0.004)	0.016 (0.004)	0.011 (0.003)	**0.010 (0.002)**	0.015 (0.004)	0.015 (0.004)	0.014 (0.004)
ExtrudeEnd	0.022 (0.004)	0.022 (0.003)	0.022 (0.003)	**0.017 (0.003)**	0.024 (0.004)	0.022 (0.003)	0.023 (0.004)
CutSide	0.166 (0.039)	0.165 (0.039)	0.193 (0.037)	**0.155 (0.027)**	0.168 (0.038)	0.165 (0.040)	0.167 (0.038)
CutEnd	0.157 (0.026)	0.158 (0.026)	0.184 (0.023)	**0.141 (0.017)**	0.159 (0.027)	0.154 (0.025)	0.158 (0.027)
Fillet	0.015 (0.005)	0.015 (0.005)	0.031 (0.007)	**0.010 (0.003)**	0.028 (0.009)	0.014 (0.005)	0.027 (0.008)
Chamfer	0.203 (0.073)	0.205 (0.074)	0.134 (0.051)	0.182 (0.052)	0.096 (0.043)	0.199 (0.074)	**0.094 (0.044)**
RevolveSide	0.219 (0.046)	0.217 (0.046)	0.151 (0.029)	0.206 (0.022)	0.101 (0.025)	0.218 (0.046)	**0.100 (0.026)**
RevolveEnd	0.497 (0.002)	0.497 (0.002)	0.497 (0.002)	0.497 (0.002)	**0.484 (0.020)**	0.497 (0.003)	**0.484 (0.020)**
Overall (micro)	0.042 (0.006)	0.042 (0.006)	0.040 (0.005)	**0.034 (0.005)**	0.038 (0.004)	0.041 (0.006)	0.037 (0.004)
Overall (macro)	0.162 (0.015)	0.162 (0.015)	0.153 (0.011)	0.153 (0.010)	**0.134 (0.011)**	0.160 (0.015)	**0.134 (0.011)**

TABLE 1: Error remaining after manual correction of most uncertain predictions vs fraction AUC for each class across the evaluation datasets. Lower is better. The top performing in each row is formatted with **bold and underline**, while the second is only underlined. Ties were broken with extra decimal places not shown.

Conversely, at $M = 10$ ensemble models, the metrics have not yet fully stabilised for any of the three cases, as shown in Figure 7. This technique may benefit from an increased number of ensemble models however this has significant computational cost and is left as future work. The authors conjecture that while the numerical results may change, the overall conclusions will remain the same.

7 Discussion

The results presented in this work align with the literature while at the same time presenting a new insight. Reinforcing previous results discussed in Section 2, the Deep Ensemble technique outperforms MC Dropout when measured by $P(\text{accurate} \mid \text{certain})$, $P(\text{uncertain} \mid \text{inaccurate})$, and 'error remaining' metrics. It also produces the best base predictive accuracy, before considering confidence filtering. However, this is not the case when looking at a more calibration focused metric like PAvPU. To the best of the authors' knowledge, the calibration performance of the Deep Ensemble technique within semantic segmentation has not been previously studied. Further, the aggregated inference

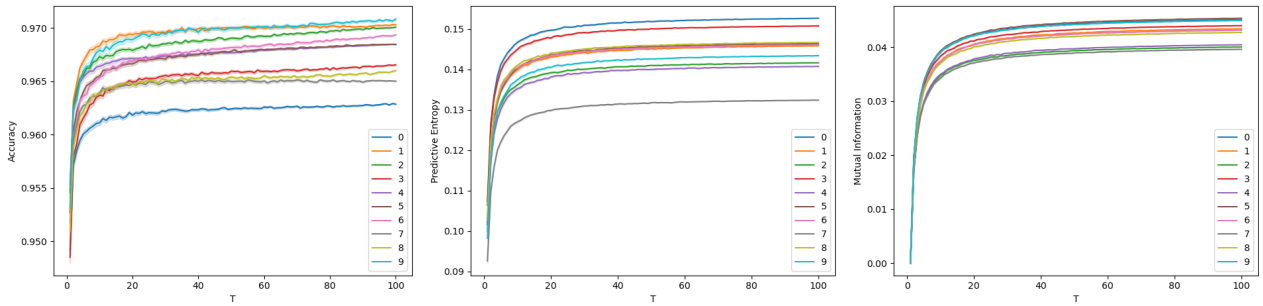
	Random	Ideal (GT)	Mean Probability	Mutual Information	MI classwise	Predictive Entropy	PE classwise
Deep Ensemble							
MFCAD++ (Micro)	0.019 (0.001)	0.001 (0.000)	**0.002 (0.000)**	0.003 (0.000)	0.003 (0.000)	0.003 (0.000)	0.003 (0.000)
MFCAD++ (Macro)	0.036 (0.001)	0.005 (0.001)	**0.013 (0.001)**	0.015 (0.001)	0.015 (0.001)	0.015 (0.001)	**0.013 (0.001)**
MFCAD++ (5%) (Micro)	0.047 (0.001)	0.004 (0.000)	**0.011 (0.001)**	0.014 (0.001)	0.014 (0.001)	0.012 (0.001)	0.012 (0.001)
MFCAD++ (5%) (Macro)	0.080 (0.002)	0.020 (0.001)	**0.042 (0.001)**	0.046 (0.001)	0.045 (0.001)	0.044 (0.002)	**0.042 (0.002)**
Fusion360 (Micro)	0.081 (0.006)	0.013 (0.002)	**0.034 (0.005)**	0.040 (0.004)	0.041 (0.004)	0.038 (0.005)	0.037 (0.005)
Fusion360 (Macro)	0.183 (0.008)	0.108 (0.007)	0.153 (0.010)	0.162 (0.008)	0.162 (0.010)	0.154 (0.010)	**0.148 (0.009)**
MC Dropout							
MFCAD++ (Micro)	0.022 (0.001)	0.001 (0.000)	**0.003 (0.000)**	0.005 (0.000)	0.004 (0.000)	0.004 (0.000)	0.004 (0.000)
MFCAD++ (Macro)	0.037 (0.002)	0.005 (0.001)	**0.014 (0.002)**	0.017 (0.002)	0.017 (0.002)	**0.014 (0.002)**	**0.014 (0.002)**
MFCAD++ (5%) (Micro)	0.064 (0.003)	0.008 (0.001)	**0.020 (0.001)**	0.022 (0.001)	0.022 (0.001)	**0.020 (0.001)**	**0.020 (0.001)**
MFCAD++ (5%) (Macro)	0.101 (0.004)	0.029 (0.003)	**0.059 (0.005)**	0.063 (0.005)	0.063 (0.005)	**0.059 (0.005)**	**0.059 (0.005)**
Fusion360 (Micro)	0.084 (0.006)	0.014 (0.002)	**0.038 (0.004)**	0.045 (0.004)	0.045 (0.004)	0.039 (0.005)	0.039 (0.004)
Fusion360 (Macro)	0.170 (0.010)	0.091 (0.010)	**0.134 (0.011)**	0.143 (0.010)	0.145 (0.010)	0.135 (0.010)	0.135 (0.010)
Point Resampling							
MFCAD++ (Micro)	0.022 (0.001)	0.001 (0.000)	**0.003 (0.000)**	0.004 (0.000)	0.004 (0.000)	**0.003 (0.000)**	**0.003 (0.000)**
MFCAD++ (Macro)	0.041 (0.003)	0.006 (0.001)	**0.017 (0.002)**	0.019 (0.002)	0.019 (0.002)	0.018 (0.002)	**0.017 (0.002)**
MFCAD++ (5%) (Micro)	0.061 (0.003)	0.007 (0.001)	**0.018 (0.001)**	0.022 (0.001)	0.023 (0.001)	0.019 (0.001)	**0.018 (0.001)**
MFCAD++ (5%) (Macro)	0.099 (0.004)	0.029 (0.003)	**0.056 (0.005)**	0.061 (0.004)	0.062 (0.004)	0.057 (0.004)	**0.056 (0.004)**
Fusion360 (Micro)	0.088 (0.006)	0.015 (0.002)	**0.041 (0.006)**	0.047 (0.004)	0.049 (0.004)	0.044 (0.006)	0.044 (0.006)
Fusion360 (Macro)	0.191 (0.011)	0.113 (0.010)	0.160 (0.015)	0.171 (0.011)	0.177 (0.012)	0.162 (0.014)	**0.159 (0.014)**
Combined							
MFCAD++ (Micro)	0.020 (0.001)	0.001 (0.000)	**0.003 (0.000)**	0.004 (0.000)	**0.003 (0.000)**	**0.003 (0.000)**	**0.003 (0.000)**
MFCAD++ (Macro)	0.034 (0.002)	0.004 (0.001)	**0.012 (0.001)**	0.014 (0.002)	0.013 (0.002)	**0.012 (0.001)**	**0.012 (0.001)**
MFCAD++ (5%) (Micro)	0.058 (0.003)	0.007 (0.001)	**0.017 (0.001)**	0.019 (0.001)	0.019 (0.001)	**0.017 (0.001)**	**0.017 (0.001)**
MFCAD++ (5%) (Macro)	0.095 (0.004)	0.027 (0.003)	**0.053 (0.004)**	0.056 (0.004)	0.055 (0.004)	**0.053 (0.004)**	**0.053 (0.004)**
Fusion360 (Micro)	0.083 (0.006)	0.014 (0.002)	**0.037 (0.004)**	0.042 (0.004)	0.042 (0.004)	0.038 (0.004)	0.038 (0.004)
Fusion360 (Macro)	0.169 (0.010)	0.090 (0.010)	**0.134 (0.011)**	0.142 (0.010)	0.144 (0.010)	**0.134 (0.011)**	**0.134 (0.010)**

TABLE 2: Error remaining after manual correction of most uncertain predictions vs fraction AUC for each inference method and uncertainty scalar estimation method across the evaluation datasets. Lower is better. The top performing in each row is formatted with **bold and underline**, while the second is only underlined. Ties were broken with extra decimal places not shown.

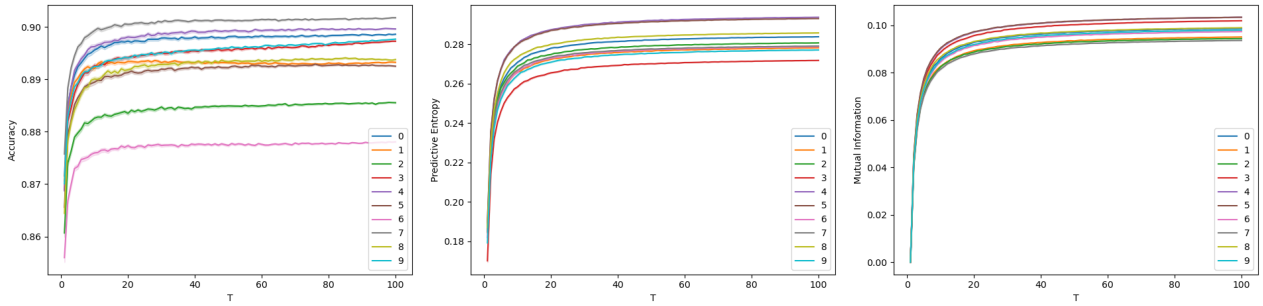
methods are often outperformed by simple histogram binning when measuring probability calibration. It is perhaps interesting future work to combine more explicit calibration approaches when estimating uncertainty - as in the work by Laves et al. (2019) previously discussed.

Considering the Deep Ensemble's significant increase in computation (and memory/storage) requirements, it is also worth noting that the proposed combined stochastic inference method (point resampling and MC Dropout) is the second best performing for the 'error remaining' metric. This technique only requires one trained model and the stochastic inferences can also be easily performed in parallel on a GPU. However, with the variance in predictive accuracy across the individual trained models shown in Figure 6, there is an argument to be made that multiple models should be trained in practice to find a 'good one'. Thus making the Deep Ensemble option more readily available and appealing.

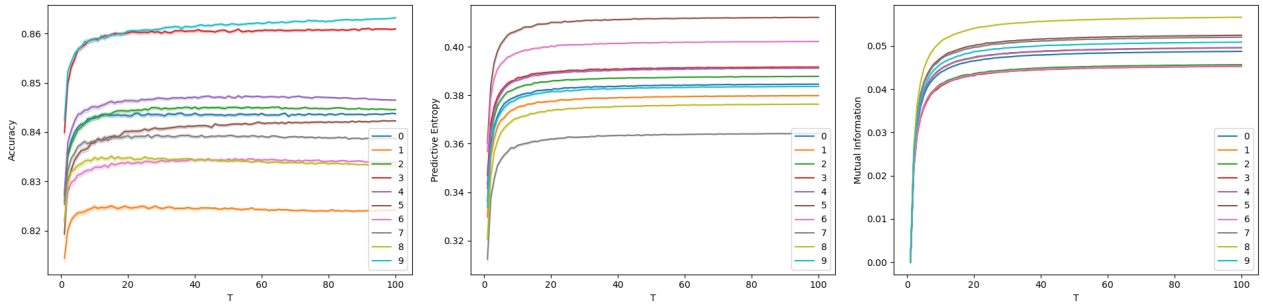
An interesting observation from these results is that the standard predictive confidence obtained from the softmax of the basic NN forward pass is not as miscalibrated as is often suggested by the literature. In the results of this work, it is not significantly worse than the extra uncertainty estimation methods. Looking at some of the factors that Guo et al. (2017) propose that cause 'modern neural networks' to be uncalibrated and overconfident - some of these do not apply to the networks used here. For instance, they observe that NNs can overfit to negative log likelihood (NLL) loss without overfitting to the 0/1 predictive accuracy loss therefore NNs with weights extracted at the minimum of the latter can have miscalibrated probability outputs. As stated in Subsection 5.1, cross



(A) MFCAD++



(B) MFCAD++ (5%)



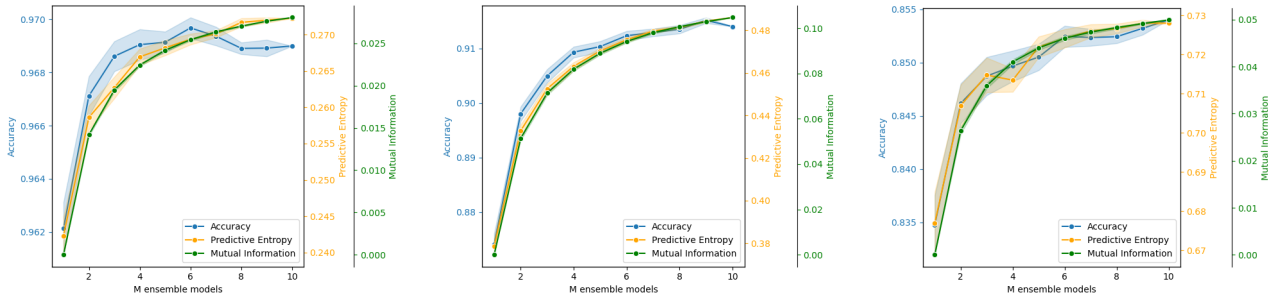
(C) Fusion360

FIGURE 6: Aggregate metrics computed from each trained model (across samples) performing combined stochastic inference, with increasing number of stochastic forward passes T.

entropy loss (directly correlated to NLL loss) is used as the early stopping criteria here instead of predictive accuracy. They also state that miscalibration grows substantially with model capacity (i.e. number of parameters); the NNs here are small compared to most used in the state-of-the-art.

8 Conclusions

The authors present this work mainly as a first of its kind exploration into the application of uncertainty estimation techniques to feature recognition in CAD, specifically using point-based graph neural networks. A number of techniques were applied and compared to two 3D CAD geometry datasets with different semantics. Reinforcing results from literature, the Deep Ensemble technique produces uncertainty estimates which can more readily be used to filter for predictions which are



(A) MFCAD++ (B) MFCAD++ (5%) (C) Fusion360

FIGURE 7: Aggregate metrics computed from increasing number of models within a Deep Ensemble.

more likely to be incorrect. It also gives the best base predictive accuracy. However, it is worth noting that the other methods are not significantly worse for the large dataset cases, and are still much better than random while having relatively moderate computation cost.

It is shown that practical and relatively simple techniques for uncertainty estimation are effective at segmentation quality control. In other words, the estimated uncertainty scalars are such that if a prediction is less uncertain than another one it is more likely to be correct. Therefore, the uncertainty estimates could be used in a human-in-the-loop approach to dramatically decrease error rates given moderate manual effort. It was also found that the mean probability vector from the distribution of forward passes was sufficient in capturing uncertainty for the purposes of segmentation quality control. It is hoped that this work can be used as a base for tackling real case studies and helps the adoption of predictive deep learning methods into the engineering workflow.

9 Future Work

As an initial exploration into the space, there is naturally much future work to be done in this area. Some suggestions for further research are presented in the following. It was suggested in Subsection 4.6 that aleatoric and epistemic uncertainty can be decomposed and estimated from the distribution of predictions obtained from some inference methods given a single b-rep geometry input. A natural extension of the current study is to investigate how decomposed uncertainty could be useful, just as Petschnigg and Pilz (2021) and Kendall and Gal (2017) do for other application areas. Going further than filtering or flagging predictions, active learning and transfer learning in the continuously changing area of CAD could benefit from accurate uncertainty estimation techniques.

The results of this study also suggest further avenues of research involving the Deep Ensemble technique. Firstly, as suggested in Subsection 6.4, a larger scale study with more trained models may yield different results. In addition, it was observed that a Deep Ensemble has a much higher base prediction accuracy than single trained network in the case where only a small amount of training data is available. There could be interesting further work here for developing data-efficient deep learning systems at the cost of increased compute and training.

Finally, this study treats the b-rep faces in the testing dataset completely independently when doing uncertainty estimation (after the NN inference step). In reality, they are embedded within geometries and have spatial and contextual relationships with the other faces within the geometry. Leveraging information from the whole geometry during uncertainty estimation could be useful. Alternatively, structural quality metrics like those used in image segmentation could be useful here for deciding whether a geometry as a whole has poor predictions rather than individual b-rep faces.

Acknowledgements

The authors acknowledge funding from Rolls-Royce plc. The authors also acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work.

References

Mazin Al-Wswasi, Atanas Ivanov, and Harris Makatsoris. A survey on smart automated computer-aided process planning (acapp) techniques. *The International Journal of Advanced Manufacturing Technology*, 97(1):809–832, 2018.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015.

Weijuan Cao, Trevor Robinson, Yang Hua, Flavien Boussuge, Andrew R Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 84003, page V11AT11A003. American Society of Mechanical Engineers, 2020.

Andrew R. Colligan, Trevor T Robinson, Declan C. Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Computer-Aided Design*, 147, February 2022. ISSN 0010-4485. doi: 10.1016/j.cad.2022.103226.

A Philip Dawid. The well-calibrated bayesian. *Journal of the American Statistical Association*, 77(379): 605–610, 1982.

Morris H. DeGroot and Stephen E. Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 32(1/2):12–22, 1983. ISSN 00390526, 14679884. URL http://www.jstor.org/stable/2987588.

Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete dropout. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/84ddfb34126fc3a48ee38d7044e87276-Paper.pdf.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 56(Suppl 1):1513–1589, 2023.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 318–319, 2020.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

Christian Hubschneider, Robin Hutmacher, and J. Marius Zöllner. Calibrating uncertainty models for steering angle estimation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1511–1518, 2019. doi: 10.1109/ITSC.2019.8917207.

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712, 2021.

Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

Alex Kendall, Vijay Badrinarayanan, and Roberto Cipolla. Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. *arXiv preprint arXiv:1511.02680*, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12773–12782, 06 2021.

Max-Heinrich Laves, Sontje Ihler, Karl-Philipp Kortmann, and Tobias Ortmaier. Well-calibrated model uncertainty with temperature scaling for dropout variational inference. *arXiv preprint arXiv:1909.13550*, 2019.

Miguel Monteiro, Loïc Le Folgoc, Daniel Coelho de Castro, Nick Pawlowski, Bernardo Marques, Konstantinos Kamnitsas, Mark van der Wilk, and Ben Glocker. Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. *Advances in neural information processing systems*, 33:12756–12767, 2020.

Jishnu Mukhoti and Yarin Gal. Evaluating bayesian deep learning methods for semantic segmentation. *arXiv preprint arXiv:1811.12709*, 2018.

Matthew Ng, Fumin Guo, Labonny Biswas, Steffen E. Petersen, Stefan K. Piechnik, Stefan Neubauer, and Graham Wright. Estimating uncertainty in neural networks for cardiac mri segmentation: A benchmark study. *IEEE Transactions on Biomedical Engineering*, 70(6):1955–1966, 2023. doi: 10.1109/TBME.2022.3232730.

Christina Petschnigg and Jürgen Pilz. Uncertainty estimation in deep neural networks for point cloud segmentation in factory planning. *Modelling*, 2(1):1–17, 2021.

L. Piegl and W. Tiller. *The NURBS Book*. Monographs in Visual Communication. Springer Berlin Heidelberg, 1996. ISBN 9783540615453. URL https://books.google.co.uk/books?id=7dqY5dyAwWkC.

John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Jami J. Shah, David Anderson, Yong Se Kim, and Sanjay Joshi. A Discourse on Geometric Feature Recognition From CAD Models . *Journal of Computing and Information Science in Engineering*, 1 (1):41–51, 11 2000. ISSN 1530-9827. doi: 10.1115/1.1345522. URL https://doi.org/10.1115/1.1345522.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

Hristo Vassilev, Marius Laska, and Jörg Blankenbach. Uncertainty-aware point cloud segmentation for infrastructure projects using bayesian deep learning. *Automation in Construction*, 164:105419, 2024.

Gerico Vidanes, David Toal, Xu Zhang, Andy Keane, Jon Gregory, and Marco Nunez. Extending point-based deep learning approaches for better semantic segmentation in cad. *Computer-Aided Design*, 166:103629, 2024.

Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019a.

Guotai Wang, Wenqi Li, Sébastien Ourselin, and Tom Vercauteren. Automatic brain tumor segmentation using convolutional neural networks with test-time augmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop, BrainLes 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16, 2018, Revised Selected Papers, Part II 4*, pages 61–72. Springer, 2019b.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019c.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *International Journal of Computer Vision*, 2024. doi: 10.1007/s11263-024-02117-4. URL https://doi.org/10.1007/s11263-024-02117-4.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616, 2001.

Hang Zhang, Shusheng Zhang, Yajun Zhang, Jiachen Liang, and Zhen Wang. Machining feature recognition based on a novel multi-task deep learning network. *Robotics and Computer-Integrated Manufacturing*, 77:102369, 2022. ISSN 0736-5845. doi: https://doi.org/10.1016/j.rcim.2022.102369.

X. Zhang, David J.J. Toal, N.W. Bressloff, A.J. Keane, F. Witham, J. Gregory, S. Stow, C. Goddard, M. Zedda, and M. Rodgers. Prometheus: a geometry-centric optimisation system for combustor design. In *ASME Turbo Expo 2014: Turbine Technical Conference and Exposition (15/06/14 - 19/06/14)*, 06 2014. URL https://eprints.soton.ac.uk/363186/.

Zhibo Zhang, Prakhar Jaiswal, and Rahul Rai. Featurenet: Machining feature recognition based on 3d convolution neural network. *Computer-Aided Design*, 101:12–22, 2018.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.