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

Gerico Vidanes, David Toal, Xu Zhang, Andy Keane

Computational Engineering and Design Group
University of Southampton

September 1, 2024

1 Introduction

Contributions:

- Empirical review of uncertainty estimation methods applied to quality control of semantic segmentation of 3D CAD geometries. Specifically using a point-based GNN.
- Point resampling as a test-time augmentation for uncertainty estimation.
- Point resampling used with MC dropout for more varied distribution of predictions given the same input and trained NN. Top performing across two large datasets.

2 Related Work

Kendall et al. (2015) - probably one of the first to apply the uncertainty estimation to semantic segmentation (dense prediction) with bottleneck type architectures. But this focuses on evaluating based on segmentation accuracy, with the uncertainty masks being used for qualitative analysis of predictions.

More recent and relevant is the medical image segmentation use cases which uses uncertainty/-confidence estimation to do quality control of model predictions (Ng et al. (2023)). I.e. if a model is uncertain about a prediction, a human expert should be involved to make the patient decision. Human-in-the-loop type applications / flagging possibly incorrect or uncertain predictions. Automation quality control.

Also, some applications for point cloud domains - uncertainty for autonomous driving/navigation. And infrastructure - Vassilev et al. (2024); Petschnigg and Pilz (2021)

Quality control for semantic segmentation in image or point clouds use cases - individual pixels or points do not have much significance in themselves but structures do. Slightly different to our use case where individual b-rep faces have much more self-contained meaning (however, very complex shapes and features could be made up of many b-rep faces and so their semantic role approaches that of pixels).

Reference to Gawlikowski et al. (2023) and some of the gaps they identify and how this work fits into there.

3 Background

3.1 Feature Recognition

Feature recognition (or semantic segmentation) of engineering geometry is a widely useful capability. Automated transition between CAD and CAM was one of the first applications as well as automated transition to analysis and simulation tools where detected features are used to aid in downstream meshing and post-processing, for example Zhang et al. (2014). More recently, with development and wider use of deep learning, the recognised features could be more complex and abstract. This could enable future use cases like detecting structural failure or features which cause problems in meshing or additive manufacturing for instance. To further develop the integration of these deep learning feature recognition tools into the engineering workflow, uncertainty estimation is a crucial step. Traditional or algorithmic feature recognition approaches gave engineers some confidence on their outputs - deep learning methods should do the same. While the latter methods can be accurate and highly flexible, they are (almost notoriously) overconfident especially for inputs that are outside their training distribution (whether an input is out-of-ditribution or not is also difficult to know a-priori). For example, errors in recognised features can lead to errors in the analysis models being built from these tags which in the best case can cause simulations to fail and in the worst case can be silent errors which lead to misleading results.

3D feature recognition with deep learning is a wide field owing to the wide applications combined with the different 3D representations available **[review reference?]**. Specifically for this work, there are also a number of 3D feature recognition approaches for CAD. Lambourne et al. (2021); Colligan et al. (2022); Cao et al. (2020); Jayaraman et al. (2021) use networks which directly work with the b-rep data structure. While Zhang et al. (2022); Yao et al. (2022); Vidanes et al. (2024) utilise point clouds to encode the geometry for NN processing. This work builds on the approach from Vidanes et al. (2024) as its flexibility allows us to easily integrate and test a wide range of uncertainty estimation methods.

3.2 Uncertainty Estimation

Literature for neural network / machine learning uncertainty estimation is vast, for a full review see Gawlikowski et al. (2023). *Uncertainty Via Classification* or *Direct Uncertainty Prediction* Raghu et al. (2019). Bayesian methods where a model represents the distribution of weights (for a given NN structure) that are likely to produce the data **[get original reference for this theory]**. And approximations of this intractable posterior(?) like variational inference and bayes by backprop. A popular and practical approach which is also shown to be an approximation of a BNN is MC Dropout (Gal and Ghahramani (2016)) where dropout [reference] is used at test time and multiple stochastic forward

passes of the same input are aggregated. Alternatively, ensemble methods have been used in machine learning for a long time. Multiple models are applied to the same input to give a distribution of outputs. This has been extended to DNNs (Lakshminarayanan et al. (2017)) to estimate uncertainty. There is also test-time augmentation methods (Wang et al. (2019b,a)) which (stochastically) transform a given input such that a deterministic model provides a distribution of outputs. While not commonly in the same category, post-processing calibration methods (Guo et al. (2017); Zadrozny and Elkan (2001)) which transform the probability output of a model to reflect its confidence better can also be used for uncertainty estimation.

In this work, we perform a practical empirical review of how uncertainty estimation can be used within CAD feature recognition. To the best of our knowledge, there are so far no published works in this space and this serves as an initial exploration. To this end, we choose relatively simple uncertainty estimation methods here which require little to no modification of the network architectures from Vidanes et al. (2024) and no changes in the training scheme. Therefore, in practice these could be applied to pre-trained models that engineers already have. We choose 4 broad approaches which each implement different conceptual representations of the 'uncertainty' or predictive confidence of the system and represent a range of computational costs - MC Dropout, test-time input augmentation, ensembling, and post-processing calibration.

4 Method

4.1 Inference

Classification and segmentation (i.e. dense classification) NNs are trained to output a score per category or class - $\mathbf{z} \in \mathbb{R}^K$, where K is the number of classes - called *logits*. The networks used here from Vidanes et al. (2024) takes as input an extended point cloud representation, $\mathcal{P} \in \mathbb{R}^{3+D}$, and outputs a predicted label for each b-rep face in the input geometry, $f_{\boldsymbol{\theta}} : \mathcal{P} \in \mathbb{R}^{3+D} \rightarrow \mathbf{Z} \in \mathbb{R}^F \times \mathbb{R}^K$, where D is the extra information encoded in the point cloud and F is the number of b-rep faces in the input geometry. In other words, the output is a logit vector for each b-rep face.

Logits are nominally transformed with a softmax layer,

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

to normalise them into a vector which sums to one. This vector is commonly interpreted as the probability that the entity belongs to each class. Thus, the predicted class is the one with the maximum probability and the predictive confidence is the maximum element in the probability vector, $q = \max_k \sigma_{SM}(\mathbf{z})$. However, the literature advises that this should not be the case for modern neural networks (Guo et al. (2017); Gal (2016)) This standard inference and 'probability vector' will be treated as a baseline for comparing the following predictive confidence estimation methods.

This sentence seems break the flow of content. And it will be introduced in 4.1.1 anyway.

4.1.1 Post-Processing Calibration

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[1]:

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

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)[2]. After scaling, the maximum probability, $\hat{q}$, is used as the predictive confidence.

Histogram binning (Zadrozny and Elkan (2001)) is another calibration method. This 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 this (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 (i.e. the maximum element in the probability vector after softmax was applied to the logit vector) was used as the 'predicted scores'[3]. 20 equal-width bins for each calibration set was used.

is this the q mentioned above? if so, can refer to it to make it clear.

4.1.2 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 sub-networks. 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.

[1] τ is used here instead of T from the original work to not confuse with the use of T later in this paper.
[2] See `http://github.com/gpleiss/temperature_scaling`.
[3] This seemed to produce similar but slightly better results than using the maximums within the raw logits

4.1.3 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 CAD model but the NN's observation/input is a point cloud sampled from the surface. Therefore, a natural and effective way to do data augmentation is to repeatedly sample this point cloud 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.

4.1.4 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.

4.1.5 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[4]. The outputs of each separate neural network for a given input geometry can be treated as samples from a distribution and aggregated as above. 10 models were used.

4.2 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

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

[4] 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

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

what does the c refer to?

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 deep and more intuitive 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.

4.3 Base Neural Network

The underlying models used in this work are point-based graph neural networks based on the work from Vidanes et al. (2024). The unscaled network 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. To take full advantage of MC Dropout inference, extra dropout layers were added in the final encoder layer and the first decoder layer (the optimal configuration suggested by Kendall et al. (2015)) with a dropout probability of 0.5. The same training hyperparameters as Vidanes et al. (2024) were used 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).

For the NN input, the b-rep geometries were encoded into point clouds using b-rep stratified sampling with at least 4096 points. Each point was 7-dimensional - encoding 3D coordinates, 3D surface normals, and a b-rep face index. 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.

Is the face index used here to record the point-face relationship? If so, it seems odd to consider it as a feature to feed into the neural network, as this information is completely arbitrary. Please correct me if I am wrong.

4.4 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 and weight initialisation. Moreover, this work

and the use of drop-out layers?

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 does 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.

Finally, unless otherwise stated, metrics are calculated for each individual class separately first then averaged to obtain the evaluation metric for that model and CV run. This is often call 'macro' averaging, instead of 'micro' averaging where metrics are calculated globally regardless of class. 'Macro' averaging is useful in cases where there is significant class imbalance so that poor performance in less frequent classes is not hidden by the dominating effect of good performance on very frequent classes.

4.5 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 methods, 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 multi-core 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.

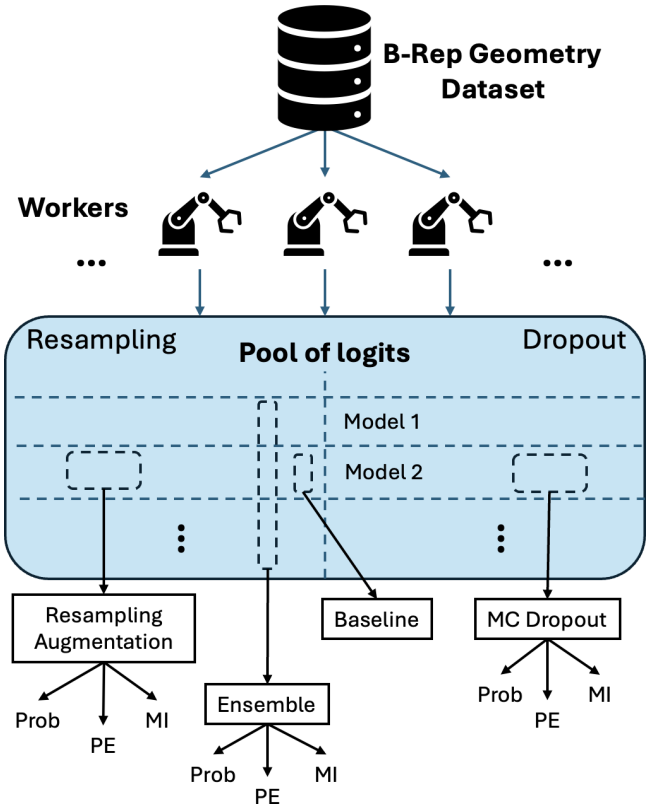


FIGURE 1: Simplified illustration of data flow from the dataset of b-rep geometries to re-usable logit vectors for simulating different inference methods.

Figure 1 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 2 shows a sample of the data being stored by the parallel workers.

	timestamp	model	sampling_seed	dropout	geom_id	face_idx	gt	logit_0	logit_1	logit_2	...
0	1.720106e+09	2	674885	False	45559	0	24	-11.395023	-9.771173	-6.572227	...
1	1.720106e+09	2	656221	False	45559	1	3	-5.850572	-6.814668	-3.885224	...
2	1.720106e+09	2	833777	False	45559	2	0	20.654837	-3.994620	3.113753	...
3	1.720106e+09	2	747463	False	45559	3	3	-3.582532	-5.484285	-3.095505	...
4	1.720106e+09	2	464341	False	45559	4	3	-4.179116	-5.199742	-2.358987	...

FIGURE 2: Sample of logit vector dataset with metadata annotations.

Consider converting the timestamp to a more meaningful format, if possible.

4.6 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. A training, validation, and test split is provided 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.

Why do we use two different writing styles to describe the train/validation/test split scheme here?

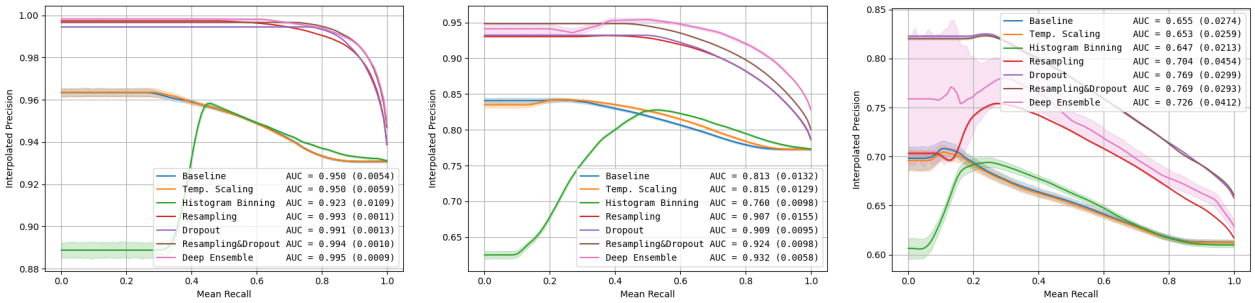
The Fusion360 Gallery Segmentation Dataset from Lambourne et al. (2021) is a collection of user submitted geometries with faces labelled according to the CAD modelling operation which created it. There are 8 possible classes - 'Extrude Side', 'Extrude End', 'Cut Side', 'Cut End', 'Fillet', 'Chamfer', '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. A train and test split is provided with 30314 and 5366 geometries respectively. In the current work, the provided 'training' geometries were randomly split with a 85/15 ratio into training and validation. 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.

Shall we explicitly exclude geometries with only a few faces (e.g., fewer than 5) from the dataset in the future? I assume this adjustment will result in extra computation time. Just some thoughts.

5 Results

5.1 Filtering Performance

The first way to evaluate the uncertainty estimation methods is how well they can be used to filter for predictions which are likely to be correct. This task can be viewed as a binary classification or detection task allowing the use of the familiar Precision-Recall (PR) curve for comparison, with area-under-the-PR-curve (AUPRC) as a summary metric. Predictions passing the confidence threshold and having a correct label are 'true positive' (TP) samples and those passing the threshold but having an incorrect label are 'false positive' (FP) samples. It also follows that correctly labelled faces which



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

FIGURE 3: Precision-Recall mean curves and 95% confidence bounds for different uncertainty estimation methods, evaluated on different datasets. Standard max probability is used as predictive confidence. Average area-under-the-curve (AUC) metric and standard deviation is shown for each method - higher is better. (B) shows results when using models trained on 5% of the MFCAD++ training set.

did not pass the confidence threshold are 'false negative' (FN) samples. This allows the calculation of the standard precision and recall metrics for a given threshold:

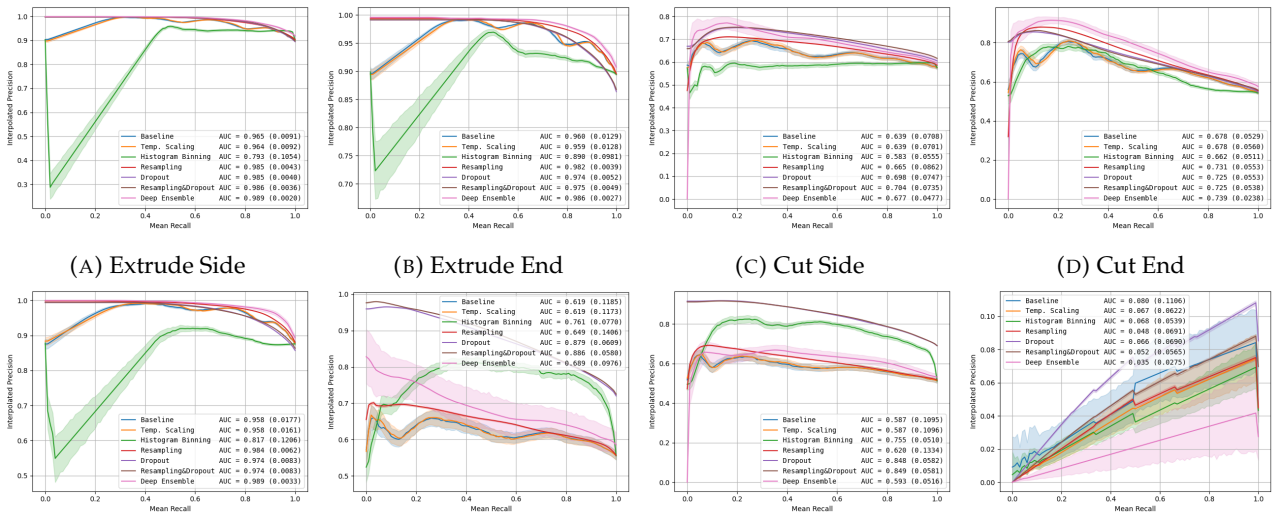
$$\text{precision} = \frac{TP}{TP + FN}, \text{recall} = \frac{TP}{TP + FN}.$$

It is worth noting that the commonly used $P(\text{accurate}|\text{certain})$ from Mukhoti and Gal (2018) maps directly to the precision metric described above. A similar probabilistic interpretation to recall would be $P(\text{certain}|\text{accurate})$. This work chooses to use the ubiquitous PR and AUPRC metrics over the conditional probabilities and *Patch Accuracy vs Patch Uncertainty* metrics from Mukhoti and Gal (2018).

Figure 3 summarises the performance of the methods across different datasets. For all inference methods, the maximum element of the (mean) probability vector is used as the predictive confidence. For each model, CV run, and class (and sample of stochastic forward passes) the precision and recall metrics were calculated for different threshold values (with macro averaging across classes). Here, an increment of 5 from 0 to 95 was used, changing to an increment of 1 up to 100. The model predictions are already quite accurate overall, so it's expected that there will be more predictions with high confidences. This gives a set of curves on the PR axis that are misaligned for each method - i.e. there is no simple correspondence of x-values. To collapse them into one mean line with error bounds, the raw PR values were interpolated onto a 'mean recall' axis from 0 to 100 with increment 1. The AUPRC metric was also computed using the interpolated values; the value shown in the plot is averaged across models and CV runs. Figure 3b shows results from models which were trained on a small subset of the MFCAD++ training set - 5% of the original. The full validation set was still used for early stopping.

The first thing to notice is that the precision values of different methods at 100% recall (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. Literature shows that MC Dropout inference and Deep Ensembles increase predictive accuracy and these results reflect that.

From figure 3a, the aggregation based methods all perform similarly well, with the Deep Ensemble being slightly better. This trend broadly remains, but the separation between methods grows in figure 3b. Looking at figure 3c, the Deep Ensemble method is now significantly worse than the combination method or even MC Dropout by itself, with a very large variance at low recall values or high confidence thresholds (i.e. the accuracy of the most confident predictions). This variance is primarily driven by the differences in performance across the CV runs for less frequent classes like 'Chamfer' and 'Revolve End' as shown in figure 4. The ambiguity of the classes together with the massive diversity of geometries in the dataset adds confounding factors when analysing results for this dataset, thus further analysis is left for future work.



(A) Extrude Side (B) Extrude End (C) Cut Side (D) Cut End

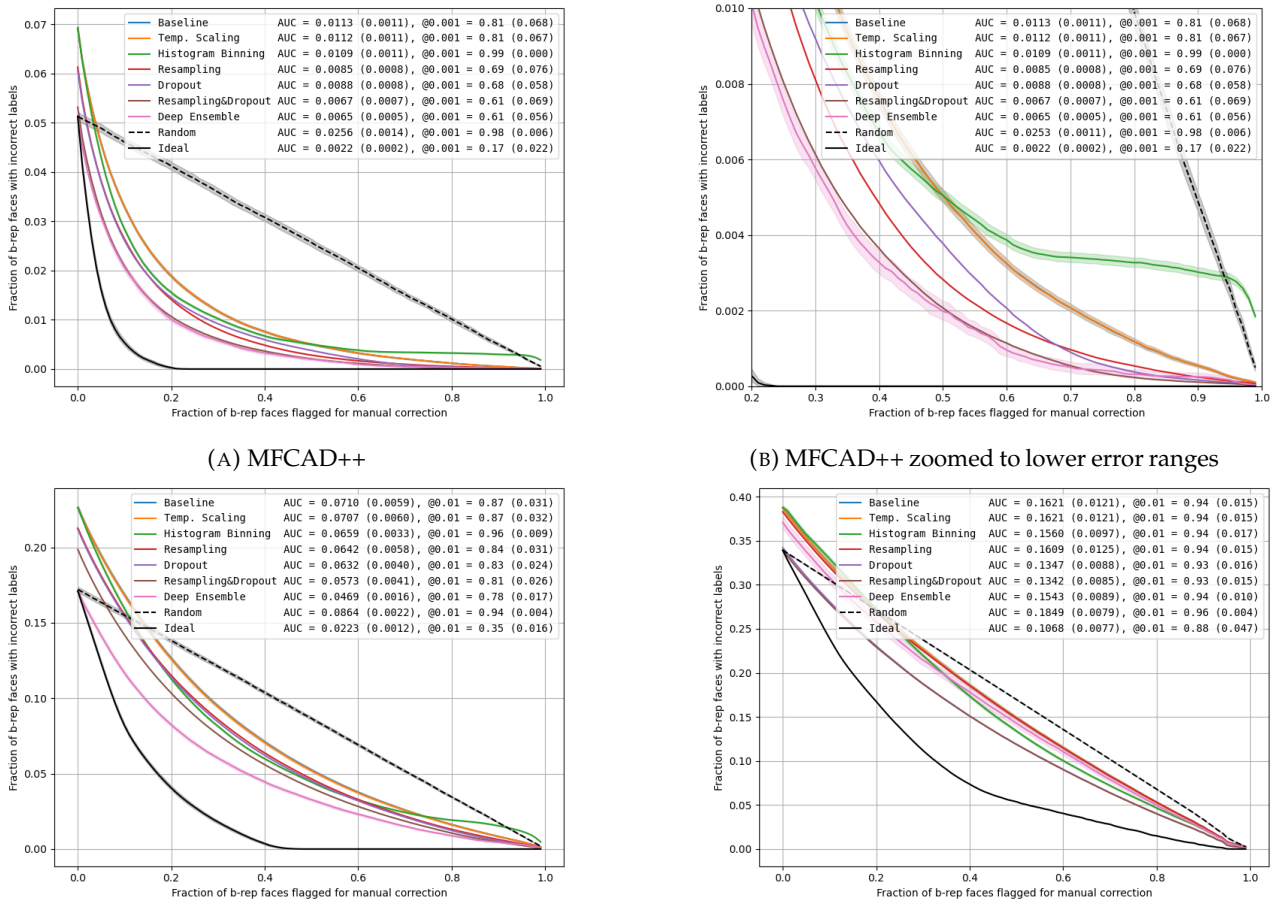
(E) Fillet (F) Chamfer (G) Revolve Side (H) Revolve End

FIGURE 4: Precision-Recall curves for individual Fusion360 Gallery classes using different uncertainty estimation methods. Average area-under-the-curve (AUC) metric is shown for each method.

5.2 Quality Control

Following Ng et al. (2023), the performance of these methods for the application of segmentation quality control is evaluated with slightly different but related metrics. Put simply, the application involves flagging the most uncertain outputs (based on some threshold) of the system for manual correction such that the overall output of the human-in-the-loop system is more accurate. This is 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 (in theory) the NN is confident in. In this section, more practical and intuitive metrics are used instead of the familiar but more conceptual precision and recall from the previous section. Instead of specifying an arbitrary range of thresholds, 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. Again, the maximum value within the (mean) probability vector is being used as the confidence for all methods. 2 extra cases are also shown for context. The dotted black line represents the case where predictions are



(A) MFCAD++

(B) MFCAD++ zoomed to lower error ranges

(C) MFCAD++ (5%)

(D) Fusion360 Gallery

FIGURE 5: Fraction of b-rep faces with incorrect labels after flagging a fraction of predictions for 'manual correction'. Average area-under-the-curve (AUC) metric is shown for each method - lower is better. The fraction of predictions required to be flagged for a given error is also shown - lower is better. (C) shows results when using models trained on 5% of the MFCAD++ training set.

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. One would expect the ideal case to be a straight line and intersect the horizontal axis at the same value as the vertical axis intersection - i.e. the fraction of b-rep faces corrected is the exact fraction which have incorrect labels originally. This is not the case in these plots since the error rate for each class is computed then averaged (macro averaging). Results for single classes, not shown, do show a straight line for the ideal case as expected. In addition, because the correctness of the predictions are different across some methods, these cases depend on which predictions are used - i.e. which curve they intersect with on the vertical axis. Here, the method with the best base prediction accuracy for each dataset is used to visualise the random and ideal cases, similar to Ng et al. (2023). The average AUC is computed as before. Finally, a specific interpolated point is shown for each method as a summary metric following Ng et al. (2023). In other words, this is the fraction of b-rep face predictions which needs to be flagged for manual correction to achieve a given error rate overall.

It can be seen that the overall results here correlate to those shown when using the PR metrics. The combination method is consistently well performing across classes. Deep Ensemble is best for the full MFCAD++ case, albeit within a standard deviation to the combination method as measured by the mean AUC. For the predictions from models trained on 5% of the MFCAD++ training set,

possible typo?

can we draw a vertical line in the plots of Figure 5 to highlight this?

Deep Ensemble is best with a significant margin. In contrast, for the Fusion360 Gallery case, the combination method is top performing by a significant margin. The point metrics shown in figure 5 are for low very low error rates since this is hard to distinguish in the plots, but this is likely not of most interest in practice as it corresponds to a large amount of manual intervention. More reasonable is perhaps for a human expert to check 20% of the most uncertain predictions. In this case, both the Deep Ensemble and the combination methods reduce the error rate for predictions on the MFCAD++ dataset from around 5% to 1%. For the models trained on the small MFCAD++ training set, the Deep Ensemble uncertainty estimation improves the error rate from around 17% to around 8% with a human in the loop. Finally, for the Fusion360 Gallery case, MC Dropout and the combination method improves error rate from around 34% to 23%.

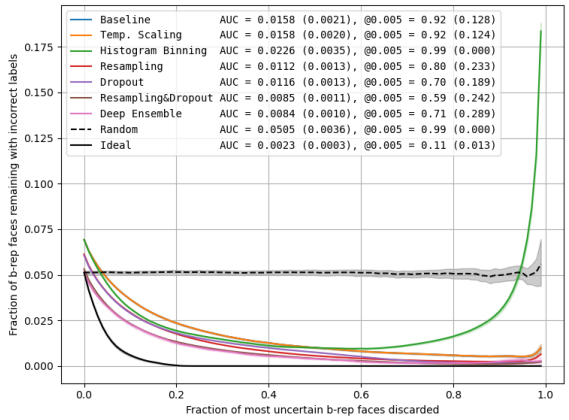
Figure 6 presents the same results in a slightly different way. Instead of considering the error rate on the entire test set after manual correction, the error rate on the filtered or 'certain' predictions is shown. Put simply, a fraction of the most uncertain b-rep face predictions are discarded and the error rate of the remaining predictions is plotted against this fraction. This better shows the performance at the strictest confidence thresholds, whereas this is diluted in the previous metric since all 'uncertain' predictions become correct after manual intervention. This is relevant to a more feature detection type application where dense prediction (a label for every b-rep face) is not needed. As mentioned before, this includes use cases like identifying portions of the geometry that might have structural failure, or be difficult to mesh, or be difficult to additively manufacture.

Essentially, this is the inverse of the PR metrics in subsection 5.1 but with a more practical interpretation. In this case, the random baseline is (approximately) a straight line since removing predictions uniformly randomly should preserve the overall distribution of correct and incorrect labels in the remaining predictions. The main trends here and the performances of methods relative to each other are of course the same as before, but the curves no longer go towards (1,0) since the perfect oracle is not correcting and increasing fraction of predictions. The increase in error rate at higher discarded fractions, most dramatic in the histogram binning approach, suggests an overconfidence in some incorrect predictions. This is also shown, but more subtly, in figure 5 by the decrease in the magnitude of the curve's gradient. The histogram approach shows the most dramatic case likely because of the equal width bins used. The top bin tends to occur from 0.95 confidence and up; therefore, predictions with a raw 'score' of ¿=0.95 will all have the same calibrated predictive confidence (for each class).

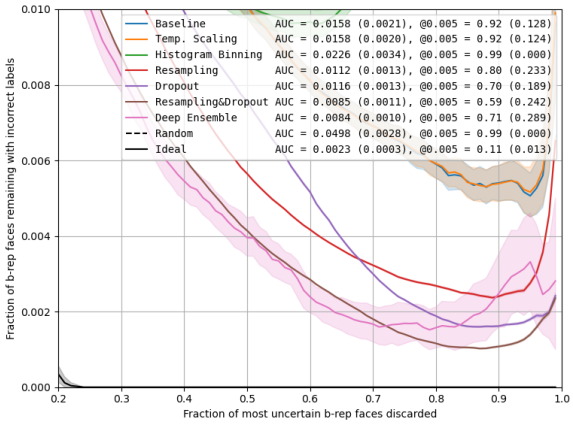
5.3 Uncertainty/Confidence Estimation

As mentioned in subsection 4.2, there are a number of ways in the literature to summarise the uncertainty from the distribution of (stochastic) predictions. Figure 7 shows the behaviour when using the different uncertainty scalar estimation methods on the set of outputs produced by combined (point resampling and MC Dropout) stochastic forward passes. They perform more or less similarly, with the simple maximum mean probability being one of the best. This is the case across the different inference methods and datasets. There is a larger spread of performance when using the Fusion360 Gallery dataset, but the mean probability is still among the best. See appendix A for full results across all inference methods and datasets.

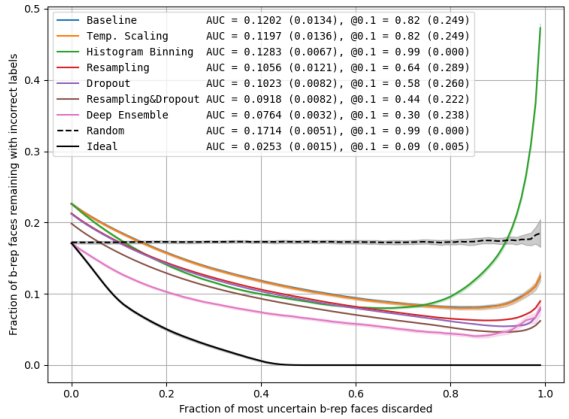
It seems that the uncertainty information is sufficiently captured by looking at the maximum element of the mean vector across forward passes - without explicitly incorporating the rest of the vector. However, it is noted that the probability vectors being averaged have already been normalised through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



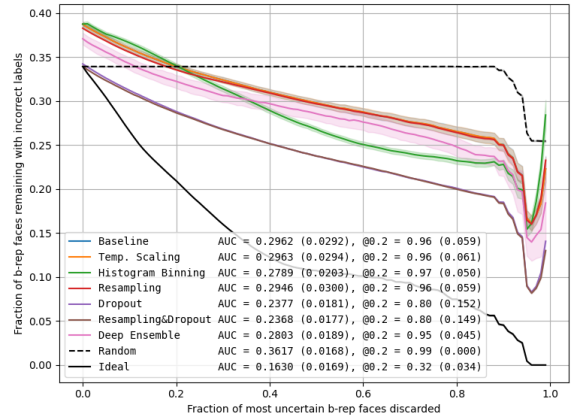
(A) MFCAD++



(B) MFCAD++ zoomed to lower error ranges



(C) MFCAD++ (5%)



(D) Fusion360 Gallery

FIGURE 6: Error rate in the remaining predictions after a fraction of uncertain b-rep faces are discarded. Average area-under-the-curve (AUC) metric is shown for each method - lower is better. The fraction of predictions required to be flagged for a given error is also shown - lower is better. (C) shows results when using models trained on 5% of the MFCAD++ training set.

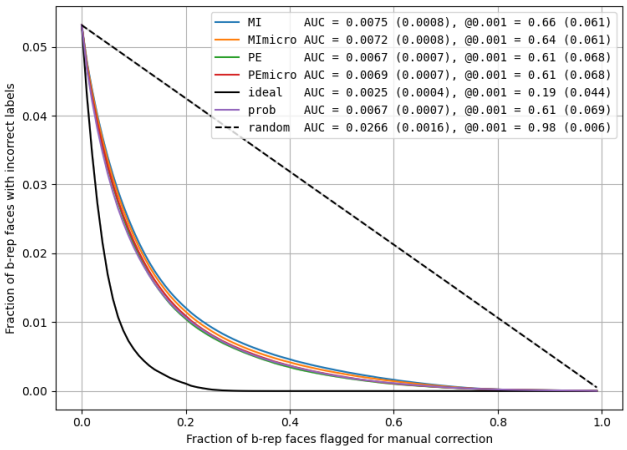


FIGURE 7: Quality control curves for different uncertainty scalar estimation methods, for the combined stochastic inference method on the MFCAD++ dataset. AUC is shown and x-axis value at given error rates - lower is better. 'Macro' refers to the scalar being normalised using the calibration set ranges of each class separately.

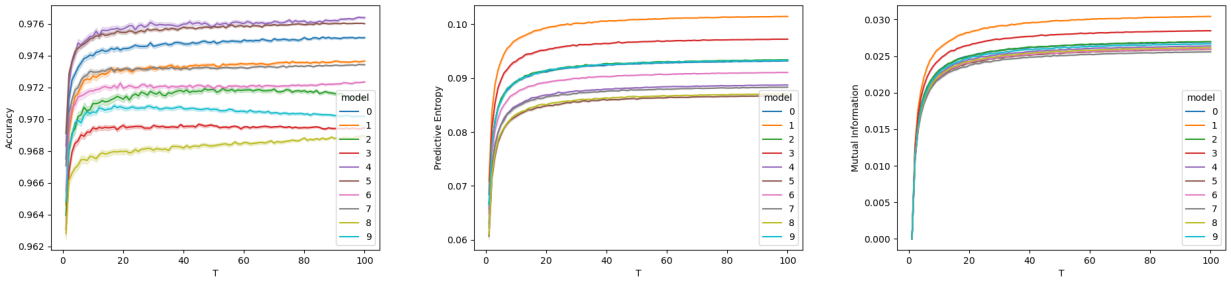


FIGURE 8: Aggregate metrics computed from each trained model performing combined stochastic inference, with increasing number of stochastic forward passes T.

I'm not sure if there's a straightforward way to plot three datasets with such drastically different ranges. Some ideas that come to mind include 3D plots, grouped bar charts, normalized radar charts, panel plots, or even the traditional approach of three separate subplots.

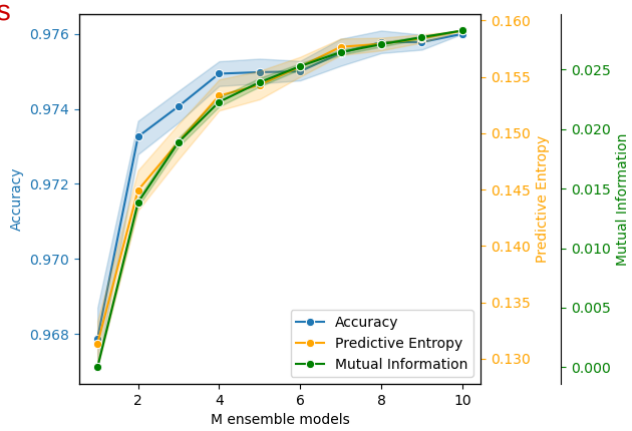


FIGURE 9: Aggregate metrics computed from increasing number of models within a Deep Ensemble.

5.4 Convergence

Figure 8 shows that the predictive accuracy and uncertainty scalar estimates are more or less stable after $T = 50$. This is also the case for the MC Dropout and point resampling individual inference methods, not shown. Conversely, at $M = 10$ ensemble models, the metrics have not yet stabilised as shown in figure 9. This method 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.

6 Discussion

Results from literature tend to suggest either MC Dropout or Deep Ensembles are best. The results in this work suggest that it is dataset (and possibly trained NN) dependent. For both the MFCAD++ cases, Deep Ensemble is significantly better than MC Dropout when measuring using the quality control metrics in subsection 5.2. However, for the Fusion360 Gallery trained models, Deep Ensemble was significantly worse than MC Dropout. The point resampling augmentation introduced in this work has mixed results. It performs similarly to MC Dropout for the MFCAD++ cases, but worse in the Fusion360 Gallery case. However, when combined with MC Dropout, this method becomes consistently good across all cases. The combined method is equally top performing for the

MFCAD++ case, although lags behind when considering models trained on the small subset of MFCAD++. For the Fusion360 Gallery case, it is a clear winner together with simple MC Dropout. This method also has the added bonus that it is more computationally efficient than a Deep Ensemble - only one model needs to be trained.

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 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 4.3, cross 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.

Related to the above, it is observed that the results for the temperature scaling method are very similar to the baseline, often having overlapping performance curves. During the tuning of the temprature scaling parameter, τ, the optimisation was often not able to improve on the already low NLL. For the MFCAD++ case, the average NLL change was 0.3% and therefore τ was close to 1 (0.96 on average). Similarly, for the Fusion360 Gallery case, the average NLL change was 0.1% with an average τ across CV runs and models of 0.98. This is unsurprising since the base NN training is already trying to minimise NLL over the large training sets; tuning an extra parameter on slightly more unseen data is unlikely to make a significant difference. For the Fusion360 Gallery case where the base accuracy of the trained models is not very high, the limiting factor does not seem to be data. In contrast, for the small MFCAD++ training set case, a 6.8% average decrease in NLL loss is observed after temperature scaling. In this case, the limited training data is the bottleneck and the supplement of 2000 geometries is helpful. However, this still only makes a very small change in the PR and quality control metrics suggesting that probabilistic calibration is perhaps not a useful metric for the application being studied in this work.

Would it be helpful to present these numbers and percentages in a table for easier comparison?

Lastly, the results for the PR curves tends to have larger variance than that of the quality control metrics - most evident in the Deep Ensemble result for Fusion360 Gallery. Both of these metrics are being computed from the same set of results and therefore the variance from the differences in trained models and CV runs should be the same. However, the PR curve computation introduces an extra source of 'variance' in that misaligned curves are being interpolated into the same intervals. This is because the 'raw' PR values are obtained by specifying a range of thresholds; the fraction of predictions which pass these thresholds is different across trained models and CV runs and therefore the obtained sets of PR points are not on the same vertical lines. For instance, the fractions of predictions from the Deep Ensemble which pass a threshold of 0.5 for the Fusion360 dataset across CV runs ranges from 0.71 to 0.92. Interpolating this onto the 'mean recall' interval introduces extra noise.

7 Conclusions

The authors present this work 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 2 3D CAD geometry datasets

with different semantics. It was found that combining MC Dropout with a point resampling test-time augmentation method outperformed all others when considering models trained on large datasets, in the context of segmentation quality control. In other words, it is good at estimating predictive uncertainties 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. However, Deep Ensemble seems to be better for both base accuracy and uncertainty estimation when considering small training dataset cases.

8 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 4.2 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 shown in subsection 5.4, the accuracy and uncertainty estimates are not quite converged yet with 10 models. A large 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 networks 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

Funding acknowledgement?

The authors 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.

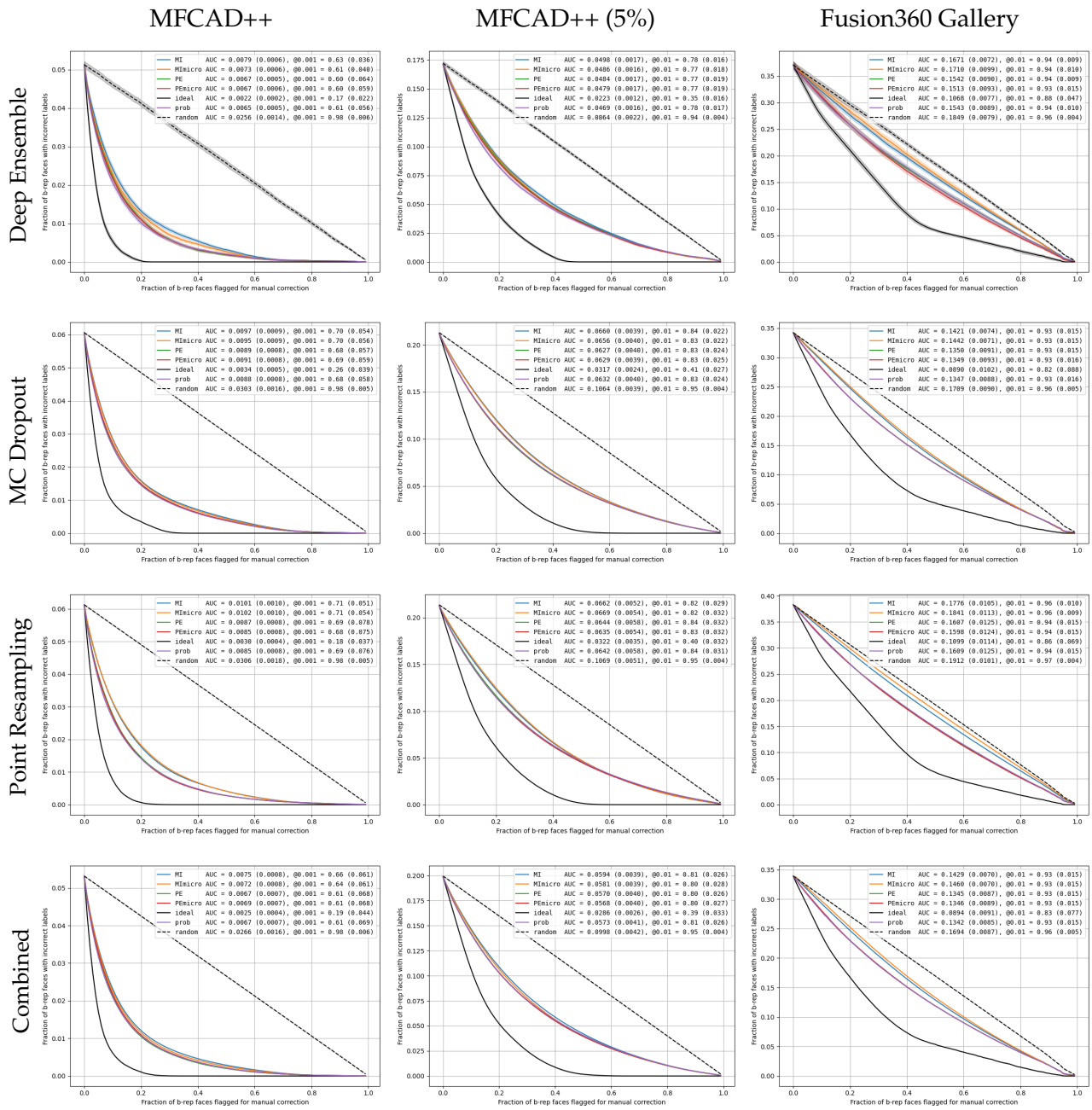


FIGURE 10: Quality control curves for different uncertainty scalar estimation methods, for each inference method and dataset. AUC is shown and x-axis value at given error rates - lower is better.

A All Uncertainty Scalar Estimation Methods

References

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.

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.

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.

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.

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

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.

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.

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.

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.

Maithra Raghu, Katy Blumer, Rory Sayres, Ziad Obermeyer, Bobby Kleinberg, Sendhil Mullainathan, and Jon Kleinberg. Direct uncertainty prediction for medical second opinions. In *International Conference on Machine Learning*, pages 5281–5290. PMLR, 2019.

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.

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.

Xinhua Yao, Di Wang, Tao Yu, Congcong Luan, and Jianzhong Fu. A machining feature recognition approach based on hierarchical neural network for multi-feature point cloud models. *Journal of Intelligent Manufacturing*, pages 1–12, 2022.

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/.