An Empirical Review of Uncertainty Estimation for Quality Control in CAD Model Segmentation

⋆

Gerico Vidanes[1], David Toal[1], Andy Keane[1], Daniel Xu Zhang[2], Marco Nunez[3], and Jon Gregory[3]

[1] University of Southampton, UK
[2] Falmouth University, UK
[3] Rolls Royce plc., UK

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.

Keywords: Neural Networks · Uncertainty · Point-Cloud · Computer-Aided Design · Semantic Segmentation · Feature Recognition.

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 auto-

⋆ Supported by Rolls-Royce plc.

mated transition between computer-aided design (CAD) models and computer-aided manufacturing and process planning[27,1]. Later, feature recognition was also used for automated analysis; where detected features are used to aid in downstream meshing, simulation, and post-processing[38]. With the development and wider use of geometric deep learning within computational engineering[17,4,3,13] [37,31], 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. Coupled with their end-to-end nature, the basic system simply presents the engineer with a dense set of semantic segmentation predictions with varying correctness. 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. 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.[9] 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.

2 Related Work

Much work has been done on uncertainty quantification for NNs in general[9,7]. Works which use convolutional / bottleneck like networks for computer vision

are relevant to this work. One of the first was Kendall et al.[14] which applied MC Dropout[8] to a convolutional neural network (CNN) - however, only improvements in semantic segmentation accuracy were evaluated and per-pixel uncertainty masks were only for qualitative analysis. Filling this gap, Mukhoti and Gal[20] present metrics for evaluating uncertainty estimation techniques in terms of how well they (inversely) correlate with semantic segmentation accuracy. Another important practical technique is the Deep Ensemble[16]. There referenced work 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.

More recently, work has been done on uncertainty estimation for NNs with 3D unstructured inputs, namely point clouds. Vassilev et al.[30] use the *KPConv* architecture[29] 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[22] instead use a *PointNet* architecture[25] 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.

The work by Gue et al.[10] is also worth mentioning, 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 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 but some of these methods will be tested for completeness.

Finally, the most relevant work is the recent, large-scale, empirical review on a real use case by [21]. They compare Bayes by Backprop[2], MC Dropout, Deep Ensemble, and Stochastic Segmentation Networks[19] 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[39,34], triangular surface meshes[11], and point clouds[25,29,26,35,40]. 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 $\mathbf{x}\colon \mathbf{U} \rightarrow \mathbb{R}^3$ - 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 $\mathbf{U} \in \mathbb{R}^2$ [23]. 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. The b-rep face topology can be treated as a graph and processed with graph convolution[4,3,13]. Lambourne et al.[17] also exploits the topology of the b-rep faces but presents a novel way to perform kernel convolution on this. Alternatively, the surfaces can be encoded as point clouds while still preserving information from the b-rep model[37,31].

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. 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 $f_\theta \colon \mathbf{X} \in \mathbb{R}^N \times \mathbb{R}^D \rightarrow \mathbf{Z} \in \mathbb{R}^N \times \mathbb{R}^K$; 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.[31]. 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[10,7],

especially for new inputs which are not within the training distribution. Therefore, much work has been done on proper NN uncertainty estimation[9]. 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[8]), 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[21]. 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[6,5]. 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 and explained in the following.

Temperature scaling[10] is a simplified multi-class version of the Platt scaling method[24] for calibrating NN probability predictions that only tunes one parameter, τ[4]:

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

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

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

Histogram binning ([36]) 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'. 20 equal-width bins for each calibration set was used.

4.2 Monte-Carlo Dropout

The dropout technique[28] randomly 'drops' neurons in a layer 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. Additionally, it has been shown that using this at test-time produces stochastic forward passes which approximates the sampling of weights for the variational inference of Bayesian NNs[8].

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. 50 stochastic forward passes were used and aggregated here which was found to be sufficient for converged uncertainty estimates and resultant classification accuracy.

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[33,32,9]. 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 in [31]. 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 used here 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 combined 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[12] can also be used to obtain a distribution of outputs given the same input. An ensemble of models with the same architecture and trained with the same dataset is used[16]. 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. 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.

5 Method

5.1 Base Neural Network

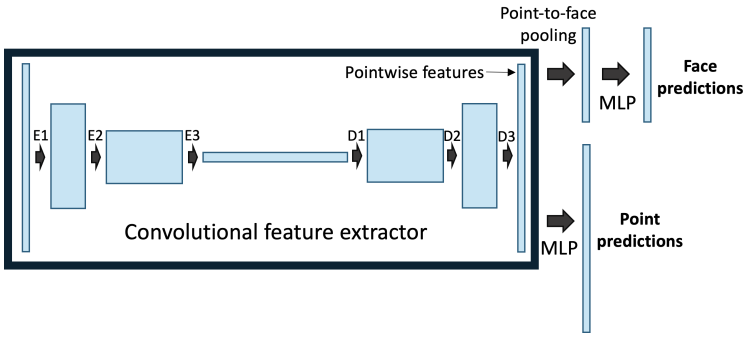


Fig. 1: Block diagram of neural network architecture. The convolutional-type point-based feature extractor is followed by a multi-head structure. *MLP* = Multi-Layer Perceptron. *EX* = Encoder X. *DX* = Decoder X.

The underlying models used in this work are point-based networks based on the work from Vidanes et al.[31], which are an extension on *PointNet++*[26]. 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.[14].

The *ADAM* optimiser ([15]) was used with a learning rate of 0.001 and the network weights corresponding to the minimum cross-entropy loss in the validation set were extracted.

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[31] with at least 4096 points.

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 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.

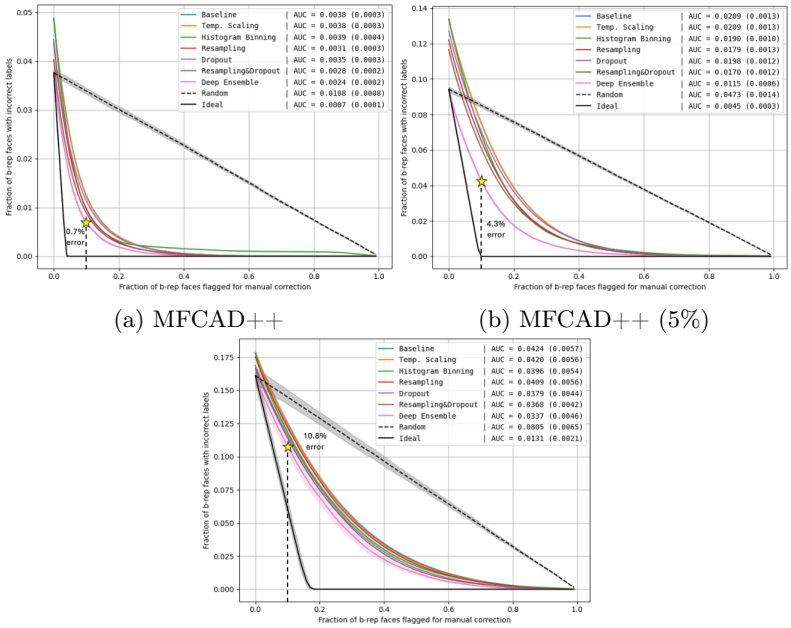
Multiple models are necessary for the Deep Ensemble inference approach, therefore 10 separate models were trained on the same training and validation data. For the other approaches, the following was also repeated for each model with results being aggregated to capture the variation caused by the random 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. 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.

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 and CV runs. 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.

This work uses two large and publicly available datasets of 3D CAD geometries with semantic segmentation labels for evaluation. First is MFCAD++[4], an algorithmically generated dataset where each b-rep face in the model is labelled with the manufacturing operation which created it. There are a total of 25 classes. 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. The number of faces per geometry is approximately normally distributed with a mean of 30, ranging between 6 and 86. Second is the Fusion360 Gallery Segmentation Dataset [17] - a collection of user submitted geometries with faces labelled according to the CAD modelling operation which created it. 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. The mean number of faces per geometry is around 15, but the distribution is dramatically skewed with a range from 1 to 421. For both datasets, 3000 geometries from each test set are used for evaluation of the uncertainty estimation methods.

6 Results

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. To this end, the metrics and evaluation context introduced in [21] 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. 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 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. The area-under-the-curve (AUC) is used as a summary metric.



(a) MFCAD++ (b) MFCAD++ (5%)

(c) Fusion360 Gallery

Fig. 2: 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.

Figure 2 summarises the results of this experiment across the different cases. The class-agnostic corrected error has been computed for each model, CV run, and sample of stochastic forward passes to give a distribution for each uncertainty estimation technique. The maximum value within the (mean) probability vector was used 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.

The differences in y-intercept of different methods corresponds to the differences in base accuracy of the predictions. As expected, the post-processing calibration methods have the same value as the baseline since these do not change the class prediction. 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 random case depends on the set of

predictions being used; here, data corresponding to the method giving the best AUC is drawn on Figure 2.

7 Discussion

The results presented in this work reinforce previous results discussed in Section 2 - the Deep Ensemble technique outperforms MC Dropout and all other methods. It also produces the best base predictive accuracy, before considering confidence filtering. Interestingly, the histogram binning approach is sometimes competitive with the aggregated inference methods. It is perhaps interesting future work to combine calibration approaches when estimating uncertainty - as in the work by Laves et al.[18].

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, it was found that there was significant variance in the predictive accuracy across individual trained models. Therefore, 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. Additionally, Figure 2b suggests that the Deep Ensemble technique is significantly better than all other methods in the case where only a small amount of data is used/available for training.

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.[10] 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 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 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 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 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.

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

1. Al-Wswasi, M., Ivanov, A., Makatsoris, H.: A survey on smart automated computer-aided process planning (acapp) techniques. The International Journal of Advanced Manufacturing Technology **97**(1), 809–832 (2018)
2. Blundell, C., Cornebise, J., Kavukcuoglu, K., Wierstra, D.: Weight uncertainty in neural network. In: International conference on machine learning. pp. 1613–1622. PMLR (2015)
3. Cao, W., Robinson, T., Hua, Y., Boussuge, F., Colligan, A.R., Pan, W.: 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. vol. 84003, p. V11AT11A003. American Society of Mechanical Engineers (2020)
4. Colligan, A., Robinson, T., Nolan, D., Hua, Y., Cao, W.: Hierarchical cadnet: Learning from b-reps for machining feature recognition. Computer-Aided Design **147** (Feb 2022). https://doi.org/10.1016/j.cad.2022.103226
5. Dawid, A.P.: The well-calibrated bayesian. Journal of the American Statistical Association **77**(379), 605–610 (1982)
6. DeGroot, M.H., Fienberg, S.E.: The comparison and evaluation of forecasters. Journal of the Royal Statistical Society. Series D (The Statistician) **32**(1/2), 12–22 (1983), http://www.jstor.org/stable/2987588
7. Gal, Y.: Uncertainty in Deep Learning. Ph.D. thesis, University of Cambridge (2016)
8. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: international conference on machine learning. pp. 1050–1059. PMLR (2016)

9. Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al.: A survey of uncertainty in deep neural networks. Artificial Intelligence Review **56**(Suppl 1), 1513–1589 (2023)
10. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International conference on machine learning. pp. 1321–1330. PMLR (2017)
11. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D.: Meshcnn: a network with an edge. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019)
12. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence **12**(10), 993–1001 (1990)
13. Jayaraman, P.K., Sanghi, A., Lambourne, J.G., Willis, K.D., Davies, T., Shayani, H., Morris, N.: Uv-net: Learning from boundary representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11703–11712 (2021)
14. Kendall, A., Badrinarayanan, V., Cipolla, R.: Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding. arXiv preprint arXiv:1511.02680 (2015)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Lakshminarayanan, B., Pritzel, A., Blundell, C.: Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in neural information processing systems **30** (2017)
17. Lambourne, J.G., Willis, K.D., Jayaraman, P.K., Sanghi, A., Meltzer, P., Shayani, H.: Brepnet: A topological message passing system for solid models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 12773–12782 (06 2021)
18. Laves, M.H., Ihler, S., Kortmann, K.P., Ortmaier, T.: Well-calibrated model uncertainty with temperature scaling for dropout variational inference. arXiv preprint arXiv:1909.13550 (2019)
19. Monteiro, M., Le Folgoc, L., Coelho de Castro, D., Pawlowski, N., Marques, B., Kamnitsas, K., van der Wilk, M., Glocker, B.: Stochastic segmentation networks: Modelling spatially correlated aleatoric uncertainty. Advances in neural information processing systems **33**, 12756–12767 (2020)
20. Mukhoti, J., Gal, Y.: Evaluating bayesian deep learning methods for semantic segmentation. arXiv preprint arXiv:1811.12709 (2018)
21. Ng, M., Guo, F., Biswas, L., Petersen, S.E., Piechnik, S.K., Neubauer, S., Wright, G.: Estimating uncertainty in neural networks for cardiac mri segmentation: A benchmark study. IEEE Transactions on Biomedical Engineering **70**(6), 1955–1966 (2023). https://doi.org/10.1109/TBME.2022.3232730
22. Petschnigg, C., Pilz, J.: Uncertainty estimation in deep neural networks for point cloud segmentation in factory planning. Modelling **2**(1), 1–17 (2021)
23. Piegl, L., Tiller, W.: The NURBS Book. Monographs in Visual Communication, Springer Berlin Heidelberg (1996), https://books.google.co.uk/books?id=7dqY5dyAwWkC
24. Platt, J., et al.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. Advances in large margin classifiers **10**(3), 61–74 (1999)
25. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)

26. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
27. Shah, J.J., Anderson, D., Kim, Y.S., Joshi, S.: A Discourse on Geometric Feature Recognition From CAD Models . Journal of Computing and Information Science in Engineering **1**(1), 41–51 (11 2000). https://doi.org/10.1115/1.1345522, https://doi.org/10.1115/1.1345522
28. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)
29. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6411–6420 (2019)
30. Vassilev, H., Laska, M., Blankenbach, J.: Uncertainty-aware point cloud segmentation for infrastructure projects using bayesian deep learning. Automation in Construction **164**, 105419 (2024)
31. Vidanes, G., Toal, D., Zhang, X., Keane, A., Gregory, J., Nunez, M.: Extending point-based deep learning approaches for better semantic segmentation in cad. Computer-Aided Design **166**, 103629 (2024)
32. Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., Vercauteren, T.: Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. Neurocomputing **338**, 34–45 (2019)
33. Wang, G., Li, W., Ourselin, S., Vercauteren, T.: 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. pp. 61–72. Springer (2019)
34. Wang, P.S., Liu, Y., Guo, Y.X., Sun, C.Y., Tong, X.: O-cnn: Octree-based convolutional neural networks for 3d shape analysis. ACM Transactions On Graphics (TOG) **36**(4), 1–11 (2017)
35. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)
36. Zadrozny, B., Elkan, C.: Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In: Icml. vol. 1, pp. 609–616 (2001)
37. Zhang, H., Zhang, S., Zhang, Y., Liang, J., Wang, Z.: Machining feature recognition based on a novel multi-task deep learning network. Robotics and Computer-Integrated Manufacturing **77**, 102369 (2022). https://doi.org/https://doi.org/10.1016/j.rcim.2022.102369
38. Zhang, X., Toal, D.J., Bressloff, N., Keane, A., Witham, F., Gregory, J., Stow, S., Goddard, C., Zedda, M., Rodgers, M.: 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), https://eprints.soton.ac.uk/363186/
39. Zhang, Z., Jaiswal, P., Rai, R.: Featurenet: Machining feature recognition based on 3d convolution neural network. Computer-Aided Design **101**, 12–22 (2018)
40. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021)