TÉCNICO LISBOA

Next Best View for Object Model Acquisition

Madalena Pombinho Caeiro Caldeira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Prof. Plínio Moreno López
Prof. José Alberto Rosado dos Santos Victor

Examination Committee

Chairperson: Prof. João Manuel de Freitas Xavier
Supervisor: Prof. Plínio Moreno López
Member of the Committee: Prof. José António da Cruz Pinto Gaspar

June 2024

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

This work was created using LaTeX typesetting language
in the Overleaf environment (www.overleaf.com).

Acknowledgments

I would like to reserve my first words to thank Professor Plínio for the guidance, patience, and incredible availability during this time.

To Mom, Dad, Teresa and Tottas, Luisinha and Carminho, and Alda. For their constant support and affection throughout this journey. A special thanks to Alda, who so dearly taught me how to deal with the common frustrations of being an engineering student at IST.

To all my colleagues, for all their companionship throughout the years and their readiness to help me understand from the simplest to the most complex concepts.

To my friends, for always being present, whether to give me advice or to help me unwind and have a good laugh. A special thanks to my dearest friend Kika for being my number one companion and motivator through long days of study and for always being such a joyful presence.

To Luís, who experienced with me the adventures and misadventures of writing a thesis and who helped me so much in the most stressful times. For being the real MVP, always reminding me of what is truly important.

And to keep a family tradition, a last thanks to JP.

Abstract

The Next Best View problem is a computer vision problem widely studied in robotics. Over the years, several deep learning models that successfully solve this problem have been proposed. Predictions obtained with the help of deep learning models naturally have some uncertainty associated with them. However, the standard models do not allow for the quantification of the uncertainty. Bayesian estimation theory contributed to demonstrate that Dropout layers can be used to estimate prediction uncertainty in neural networks. This thesis adapts the point-net based neural network architecture for Next-Best-View, PC-NBV, incorporating Dropout layers, thus allowing the computation of the uncertainty estimate associated with the predictions.

The thesis aims to improve the network's accuracy in correctly predicting the next best viewpoint, proposing a way to make the 3D reconstruction process more efficient. Various accuracy uncertainty metrics capable of modeling prediction error and accuracy are studied. Two measures of uncertainty are obtained, these enable the reduction of the model's error and the increase in its accuracy from 30% to 80%, by identifying and disregarding predictions with high uncertainty. We also propose a method that directly uses these uncertainty metrics to improve the final prediction; however this method presents very residual improvements.

Keywords

Next Best View; Uncertainty Quantification; Robotics; Deep Learning Networks

Resumo

O problema da próxima melhor vista (Next Best View) é um problema de visão computacional bastante estudado em robótica. Vários modelos de aprendizagem profunda têm sido propostos ao longo dos anos para resolver este problema com sucesso. As previsões obtidas com o apoio de modelos de aprendizagem profunda têm, como é natural, alguma incerteza associada, no entanto os modelos padrão não permitem o cálculo desta incerteza. A teoria da estimação bayesiana contribuiu para demonstrar que as camadas de Dropout podem ser usadas para estimar a incerteza da previsão em redes neuronais. Esta tese adapta a arquitetura da rede neuronal baseada na rede point-net para previsão da próxima melhor vista, PC-NBV, incorporando camadas de Dropout, permitindo assim o cálculo da estimativa da incerteza associada às previsões.

A tese tem como objetivo melhorar a precisão da rede no que toca à previsão correta do próximo melhor ponto de vista, propondo assim uma forma de tornar o processo de reconstrução 3D mais eficiente. Estudam-se várias métricas de incerteza das precisões que sejam capazes de modelar o erro e a precisão da previsão. Conseguem-se obter duas medidas de incerteza que possibilitam a redução do erro do modelo e o aumento da precisão do modelo de 30% para 80% através da identificação e desconcideração de previões com incerteza alta. Propõe-se ainda um método para utilização direta das métricas de incerteza para melhoria da previsão final, este método apresenta no entando melhorias muito residuais.

Palavras Chave

Próxima Melhor Vista; Quantificação da Incerteza; Robótica; Redes de Aprendizagem Profunda

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

2D	two-dimensional
3D	theree-dimensional
DB-NBV	Double Branch Next Best View Network
KL	Kullback-Leibler
MC	Monte Carlo
MCDM	Monte Carlo Dropout Method
NBV	Next Best View
PC-NBV	Point Cloud - Next Best View Network
Voxel	Volumetric Pixel
Voxels	Volumetric Pixels

1

Introduction

Contents

1.1 Motivation

Developments in the fields of robotics and artificial intelligence have been allowing several improvements in other scientific and non-scientific areas, revolutionizing everyday tasks in the process. Areas like healthcare, agriculture and manufacturing undergone great evolution with the integration of intelligent robotics in their processes. This influence can also be visible in the domains of archaeology, conservation and restoration, where their strategic integration has become an invaluable asset, aiding men in tasks that require lots of precision and time investment.

The work to be developed in this dissertation is inspired in a project [2] that focuses on the use of intelligent robotics in this area, with the goal of making the reconstruction process of Pompeii frescoes more efficient.

Several frescoes in poor conditions are painted on broken stone surfaces, and there is a strong interest in their reconstruction. However, their manual reconstruction requires intricate work and is very time-consuming. As a result, some methodologies for reconstruction rely on computational help. It is possible to use computer algorithms that, just like solving a jigsaw puzzle, are able to perform the reconstructions. For this they require the theree-dimensional (3D) models of the pieces, which is the task category on which this dissertation falls onto.

Making 3D models of objects requires lots of information, usually a large amount of images of the object. So a plausible option to acquire all of the images is to use robotic hands to ease the process. For the task of the model acquisition of the pieces [2] foresees the use of two robotic arms that will manipulate both the object to be scanned and a camera. The end goal would be for these robotic arms to perform the scans of all the fresco pieces as efficiently as possible.

1.2 Definition of the Problem

The work developed in this dissertation focuses on the algorithm to perform the 3D model acquisition of the fresco pieces. As previously mentioned, the data collection method must be as efficient as possible, which in this case means that the algorithm needs to find a way of getting the most information of the piece's surface with as few scans as possible. To achieve this goal the algorithm has to find the camera viewpoints (i.e. camera pose with respect to the object being scanned) that will get the most new information on the object. This is known as the Next Best View (NBV) problem, which the algorithm must adeptly solve to achieve its intended outcome.

This dissertation will only focus on making the NBV search process more efficient. The use of the robotic arm that manipulates the object is overlooked, shifting the thesis focus to simply improving the efficiency of the NBV search.

1.3 Adopted Approach

There are several already existing algorithms that use artificial intelligence to solve the NBV problem. Some, such as [3], [4], [1] and [5], showed very good results in performing 3D model reconstructions in an efficient manner. These either predict the NBV through classification or regression tasks. The classification networks sample a number of candidate viewpoints, attributing a label to each one. The network receives the object's partial reconstruction model and returns the label that corresponds to the next best viewpoint. Regression networks, on the other hand, can either regress the exact location of the NBV on the continuous space or regress a coverage score for the viewpoints in a set of candidate viewpoints, defining the NBV as the one that maximizes this score.

These networks, however, do not account for the uncertainty of their predictions, which in real world scenarios, can be vital to guarantee a good reconstruction performance, since a prediction with high uncertainty will most likely be a wrong prediction that will potentially jeopardize the reconstruction efficiency. Therefore, in this dissertation a NBV regression algorithm that regresses the views coverage scores - Point Cloud - Next Best View Network (PC-NBV) - is analyzed in a more detailed way, modifications are made to its architecture to make it account for uncertainty.

We aim to use the uncertainty of the estimated point cloud coverage using the idea of Monte Carlo sampling to estimate the uncertainty of the output of a Neural Network. To do this, the deterministic output becomes probabilistic through the application of Dropout layers. The goal is for the model to output a prediction along with an uncertainty measurement that reflects how sure the model is of its prediction, in an attempt of enhancing precision with the information given by this measurement, and subsequently improving the efficiency of the model acquisition process. We use two types of metrics that give us two different kinds of uncertainty measurements. One is error related, as it estimates the prediction uncertainty related to the closeness of the prediction to the groundtruth. Another is accuracy related, as it estimates the prediction uncertainty related to the ability of the network in choosing the right NBV.

1.4 Structure of the Dissertation

This work is divided in 4 more chapters.

Chapter 2 starts by defining the NBV problem and by reviewing several NBV networks, to make an informed decision on the one that is going to be selected to use in this work. Later it explains the problem of uncertainty in deep learning networks and reviews a strategy to transform networks allowing them to measure their prediction's uncertainty.

Chapter 3 provides an overview of the work developed in this dissertation. Describes the used dataset and how it was created, the model training process, the implemented PC-NBV model modifica-

tions, the followed steps to obtain and evaluate uncertainty, and the metrics used to quantify the model performance.

By turn, Chapter 4 presents the results of all the tests and analyses made during the dissertation, interpreting all the results and proposing ways in which the uncertainty measurements can be used to improve the models performance.

Lastly Chapter 5 summarizes all the learnings achieved with this dissertation and leaves suggestions as for what could be done in future works.

2

State of the Art

Contents

2.1 The Next Best View Problem

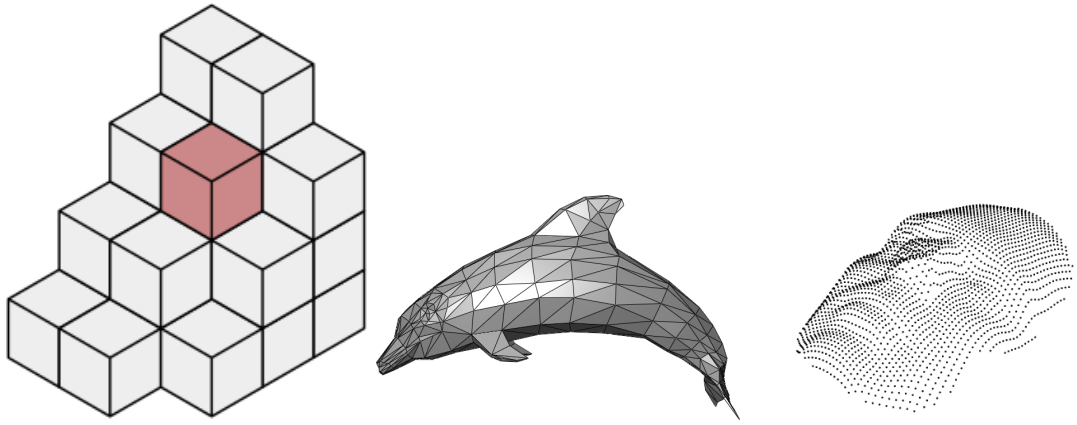
The NBV problem is an important research topic in robotic 3D reconstruction. In its essence, it aims to find the optimal viewpoint for a camera (or sensor) for it to acquire data of an object in observation, in order to maximize the information gained and enhance its overall understanding. This is useful for object 3D model reconstruction since it enables to build the object's 3D model using a minimal number of camera scans.

The common methods for the resolution of this problem can be broadly divided into two general groups: synthesis methods and generate-and-test methods.

Synthesis methods directly calculate the pose of the next best view by means of combining several pieces of heuristic information such as visibility, information gain, uncertainty reduction and others. This type of methods are mainly designed for a very specific task with specific constraints, as they form a solution using logical reasoning and domain-specific knowledge rather than exhaustive exploration. As such, although having low computational cost, they are not robust and lack flexibility - being only able to excel in situations where clear rules or constraints are available.

On the other hand generate-and-test methods, as the name suggests, form a solution by creating a set of several possible solutions for the problem and then evaluating each one to determine the best one based on specific evaluation criteria. Since finding the next best camera position in the continuous space by evaluating each possible position is very costly these methods reduce the search space by sampling some candidate views - therefore defining a discrete search space as an approximation of the continuous one. They then look for the next best position in that discrete space, turning the optimization problem into an evaluation problem - a set of evaluation criteria is defined, all the viewpoints in the search space are evaluated and ranked, and the best performing one is selected as the NBV. Although being more computationally demanding, these types of methods are very adaptable and explore a wider range of possibilities, finding solutions that might not be immediately apparent from domain knowledge alone and showing good results in situations where heuristic rules are not well defined.

Before presenting the different approaches for the NBV problem resolution it is first important to point out that there are different ways of representing 3D objects in a suitable way for computational processing. The most common ones being: Triangular Meshes, Volumetric Pixels (Voxels) and Point Clouds. Triangular Meshes are a set of connected triangles used to approximate the surface of an object. Each triangle is defined by its vertices, with (x,y,z) coordinates, and edges that connect two vertices. The accuracy of a representation with Triangular Meshes highly depends first on the complexity of the object to represent, and then on the density of Triangles in the Mesh. Voxels are another from of representation, they can be seen as the 3D equivalent of a pixel in a two-dimensional (2D) representation. In this type of representation the object is represented in a 3D grid, and a Volumetric Pixel (Voxel) is a point in that grid that represents a specific point in the 3D space. It contains information associated with that point,



(a) (b) (c)

Figure 2.1: Different 3D Model Representation: (a) Voxel (b) Triangular Mesh (c) Point Cloud

like density, color, or any other relevant characteristic needed. Just like Triangular Meshes, the accuracy of the representation of the object in this form really depends on the complexity of the object and on the size of the Voxel, smaller Voxels can represent more delicate details. Both Triangular Meshes and Voxels are structured ways of representing data - the triangles are all connected, and the Voxels have a specific place on a grid. However, the same does not apply in the case of Point Cloud representation. A Point Cloud is a set of points is the 3D space that do not follow a specific structure, they represent the actual geometry of an object only with a set of discrete points. Each point is defined by a (x,y,z) coordinate and can also contain some color or intensity information. Here once again, an higher density of points is translated in a more accurate representation of an object. Naturally, NBV solutions need to have a way of representing the objects, so they usually pick one of these representations to do so.

With these concepts clarified we will now go deeper into the different approaches of solving the NBV problem. Early works started to propose surface-based approaches, which consists of finding the set of positions that maximize the coverage of the object's surface. These approaches usually use Triangular Meshes for the model representation [6] [7]. However, the computation and storage costs of meshes is really high, making these methods challenging to work with. So then works began to develop algorithms using a volumetric approach [8], where the point cloud obtained from the sensor is transformed into Voxels, and the chosen NBV is the one that minimizes the occluded volumes. With this approach some works like [9–14] performed ray projection - a simulation of the path of light rays as they interact with surfaces - for the estimation of the information gain on each candidate view. This process also requires high computational costs as well as it is very time consuming.

More recently, works began to propose solutions using deep learning to predict the NBV. These methods learn from data on past experiences and can predict which viewpoints yield the most informative observations.

2.2 Deep Learning for NBV Problem Resolution

Deep learning is a class of machine learning algorithms, it is an artificial intelligence method that teaches computers to process data in a way that is inspired by the human brain, through the means of multiple layered neural networks. These multiple layers allow the networks to learn complex patterns and representations from large volumes of data, being able to extract relevant features.

The vast majority of the methods that use deep learning to find the NBV are generate-and-test methods. So, as explained above, they approximate the continuous space by creating a discrete set of possible camera viewpoints, referred as candidate viewpoints. Each method has its own set of evaluation criteria to measure the quality of each candidate view, that is usually translated in an evaluation score - a score that calculates the information gain that each view point allows. After ranking all the candidate viewpoints according to their score, the best classified one is selected as the NBV. Deep learning enters this process by learning how to approximate the function of the evaluation score for each candidate viewpoint, being then able to predict it.

For the networks to be able to make these predictions they need to be fed with training data containing the groundtruth information. To acquire this data, a simulated reconstruction process can be performed with objects whose complete 3D models are previously known. Firstly, typically there is a need to create the set of candidate viewpoints. This is usually done applying some common sampling methods, like uniform sampling for instance, on the area around the object to be scanned. When the set of candidate views is defined a view of the object from each viewpoint is captured, after this the simulated search for the NBV can begin. A random viewpoint is defined as the starting viewpoint, and a partial model of the object starts being built. For each candidate view an evaluation score is calculated, different works have different definitions for this score. After evaluating all of the candidate views the one with the highest score is defined as the NBV and the view from that viewpoint is added to the partial model. This process is then repeated several times until a stopping condition is met. During the iterations data like the partial model, the view scores and the selected NBV is saved, forming the training pairs. Later a network can be trained on these data points and learn to predict the evaluation scores.

Deep learning algorithms have shown to improve the efficiency and efficacy of NBV planning, since they achieve higher object coverage scores with less inference time. In this section several deep learning networks for NBV are reviewed.

2.2.1 NBV-net

NBV-net [3] is one of the first works that proposes deep learning as a way to solve the NBV problem. It approaches the problem as a classification exercise - for each partial object model inputted in the network a camera pose for the NBV is assigned, being each candidate view a class. The object's 3D

model is represented in Voxels. In this way, the network receives a Voxel grid of the object's partial model as input and, by learning an utility function, retrieves the candidate view selected as the NBV.

In this work the candidate view set is created by doing a sphere tessellation. The object to be reconstructed is placed in the center of the reference frame, and then a polar coordinate's point with fixed magnitude is rotated by uniformly moving the angles.

The utility function learned by the network defines the NBV as the view whose perception lies in a free space, has an overlap with the object's partial point cloud higher than a set threshold, and whose overlap region must have at least three 3D features. The computation of the coverage score is made in the same way as in [12], where it is calculated as

$$coverage(A) = pos(A) \cdot reg(A) \cdot sur(A) \cdot dist(A) \tag{2.1}$$

where A is the point cloud of a selected point of view; $pos(A)$ accesses if the robot is in a free collision path, it can either be zero or one; $reg(A)$ checks if there is a minimal overlap between A and the object model, if there is then it is 1, otherwise is 0; $sur(A)$ measures how much surface of unknown volume is seen, it can take any value in the interval $[0, 1]$, is calculated as the ratio between the unknown volume seen in the point view of A, and the unknown total volume until that point; $dist(A)$ evaluates the distance to the robot and it can also take any value between $[0, 1]$, in the NBV-net case the distance does not change.

NBV-net uses 3D convolutional layers, and its architecture is presented in the Figure 2.2. The network is composed by three 3D convolutional layers, each followed by a max pooling layer, and five fully connected layers with ReLU activation, with exception of the output layer that has a softmax activation.

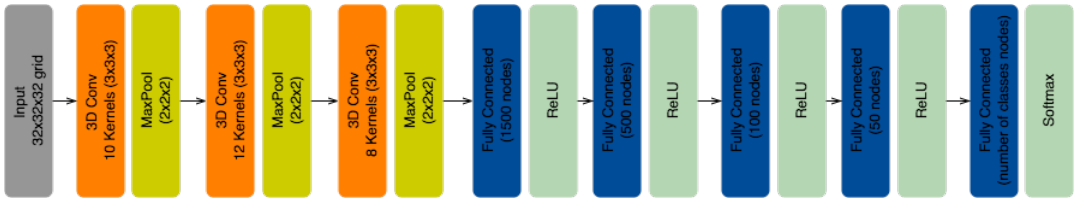


Figure 2.2: NBV-net Architecture. Input: Gray; Convolutional Layers: Orange; Pooling Layers: Yellow; Fully Connected Layers: Blue; Activation Layers: Green.

The authors compared the performance of NBV-net against other non machine learning approaches, the results showed that NBV-net reached very competitive coverage scores in the firsts iterations (it achieved high coverage scores with only the first few scans) and it also showed to have a significantly lower inference time.

2.2.2 NBV-net Regression

NBV-net Regression [4] adapts the NBV-net network [3] and solves the NBV problem as a regression. It is treated as a regression problem and now the network is capable of looking for a position in a continuous space, not being limited to just some discrete set of views. Similarly to NBV-net it also uses Voxel representation. This network receives as input the reconstruction state of the object and predicts the position coordinates for the sensor, and then calculates its orientation. The used training set was defined in the same way as it was in NBV-net, including the learned utility function, however without restricting the number of viewpoints.

In this work the authors tested the performance of different network architectures, being the one in Figure 2.3 the one that presented better results. The network is composed by four 3D convolutional layers, three of them followed by a max pooling layer, and 5 fully connected layers with ReLU activation, since this is a regression problem the last layer has instead Tanh activation.

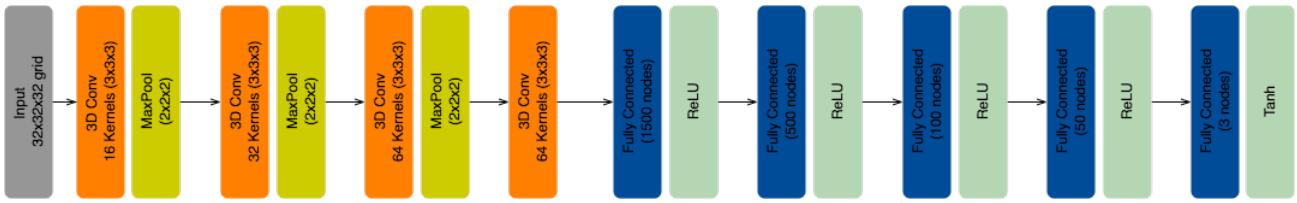


Figure 2.3: NBV-net Regression Architecture. Input: Gray; Convolutional Layers: Orange; Pooling Layers: Yellow; Fully Connected Layers: Blue; Activation Layers: Green.

The experimental results showed that this regression network was able to achieve slightly higher coverage scores than the classification NBV-net. However, the classification method still had a slightly lower inference time.

2.2.3 Point Cloud Based Next-Best-View Network: PC-NBV

Another recently proposed NBV network that has shown to have good performance is PC-NBV [1]. Unlike the other networks already mentioned above, PC-NBV uses point cloud for the model representation. Point cloud representation is less computationally costly compared to Voxelization, which will improve the inference time of the network. In addition, inspired by works such as [15–18] this network uses deep learning not only to predict the NBV but also for point cloud processing, since deep learning has proved to be quite successful in this area [19], [20]. The network receives a raw point cloud and predicts what is the NBV out of a set of 33 candidate views. The candidate views are defined by uniformly sampling view points on a sphere around the object. The dataset used by PC-NBV [1] to generate the training pairs was created with the synthetic dataset ShapeNet [21]. Each training pair is composed by the object's full point cloud, the object's partial point cloud (that represents the object's reconstruction state), the view

scores and information of which views were already selected so far.

The network learns the utility function that measures the improvement of the object's surface coverage, defined as

$$C(P) = \frac{1}{|P_o|} \sum_{p \in P} U(min_{p_0 \in P_0} ||p - p_0||_2 + \varepsilon) \tag{2.2}$$

where U is the Heaviside step function, ϵ is a distance threshold, P_0 is the complete point cloud and P is the partial point cloud. Essentially this score (from now on referred to as coverage score) quantifies the amount of new points a viewpoint captures in relation to the object partial point cloud (points of the object's total point cloud that have not yet been included in the partial point cloud), and presents the value as a ratio between the number of new points and the total number of points of the object's full point cloud.

The network's architecture is represented in Figure 2.4. It consists firstly of a feature extraction unit proposed in [22] - this unit is composed by a sequence of dense blocks, and it is used to extract structure-aware features. [22]'s ablation study showed that this unit improved the network's capacity, contributed for a significant reduction of the number of parameters, and showed that it extracted better features from the point cloud. The feature extraction unit is then followed by a max pooling layer and a self attention unit proposed in [18]. According to this study, attention mechanisms have become an integral part of models that have to capture global dependencies. This unit enables the network to learn how to allocate attention according with similarity of color and texture, it helps to model multi-level dependencies across image regions. This unit is then followed by a shared multilayer perceptron, a max pooling layer and another multilayer perceptron.

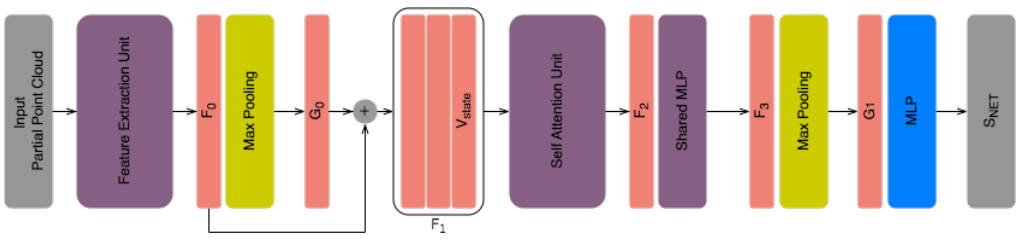


Figure 2.4: PC-NBV Architecture. Input and Output: Gray; Larger Units: Purple; Pooling Layers: Yellow; Fully Connected Layers: Blue; Features: Pink.

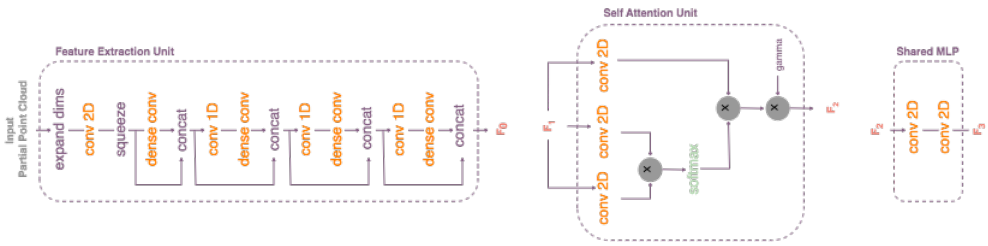


Figure 2.5: PC-NBV Units Architecture. Convolutional Layers: Orange; Activation Layers: Green; Inputs: Gray; Features: Pink

The total loss of the model is given by

$$\mathcal{L} = \frac{1}{n_s}\sum_{i=1}^{n_s}(y_i - \hat{y}_i)^2 + \lambda\sum_{j=1}^{M} w_j^2 \tag{2.3}$$

that is the sum of the MSE between the coverage score prediction ($\hat{Y}$) and the groundtruth (Y) with the L2 loss of the weights (W). The MSE between the coverage score prediction and the groundtruth measures how accurate the model is. The L2 regularization loss measures how well the model adheres to the regularization constraints.

This network shows great results compared with other NBV methods, including NBV-net. It quickly reaches high surface coverage levels both with objects similar to the ones analyzed in the training data, as with objects that the network has never seen before. Also, the inference time of PC-NBV is smaller than the other methods, because the point cloud representation does not require any complex data transformations like ray projection operations that the other methods require.

2.2.4 Double Branch Next-Best-View Network: DB-NBV

In the networks reviewed so far the choice of the NBV is made through the evaluation score predicted by the networks: the candidate view with the best score is chosen as the NBV. In [5] the authors introduce an innovation - they propose a network, Double Branch Next Best View Network (DB-NBV), that not only evaluates each candidate views in this way, but also incorporates a ranking of the candidate views. The rational behind this resides on the fact that ranking all the candidate views is computationally easier than regressing the exact coverage score of each candidate view, since there is only a need to know the relative rough scores between views. Moreover, the authors introduced a new weight feature, W, that accounts for the importance of each point in the point cloud. W is defined as

$$W = \frac{p_{xy} \cdot p_{robot}}{2} \tag{2.4}$$

where p_{xy} is the vector from the point to the object centroid and p_{robot} is the vector from the robot's

position to the object centroid. Essentially, W being a dot product measures how aligned or correlated the vectors are, determining how well-aligned the direction form the point to the object centroid is with the direction from the robot to the object centroid. In this way points that are more aligned with the robot's position relative to the object will be considered more important. W is also saved as information for the training pairs.

DB-NBV derives from PC-NBV, both the architecture and the way of generating the training pairs are quite similar (with the only difference of the generation of the W weight). The coverage score used in this work is the same as the one used before in PC-NBV, (2.2). The network's architecture is illustrated in Figure 2.6. The DB-NBV network like the name suggests is divided into two branches, the rank branch and the evaluation branch. In both the network inputs enter an extraction block that is composed by the feature extraction unit referred in [22], that is then followed by a max pooling layer and a self attention unit that returns a global feature F_3 which in turn enters another extraction block. This last one is composed by a self attention unit followed by a shared multilayer perceptron and a max pooling layer followed by another multilayer perceptron. The output of this second extraction block is either, depending on the branch, the rank or the predicted coverage scores of each view.

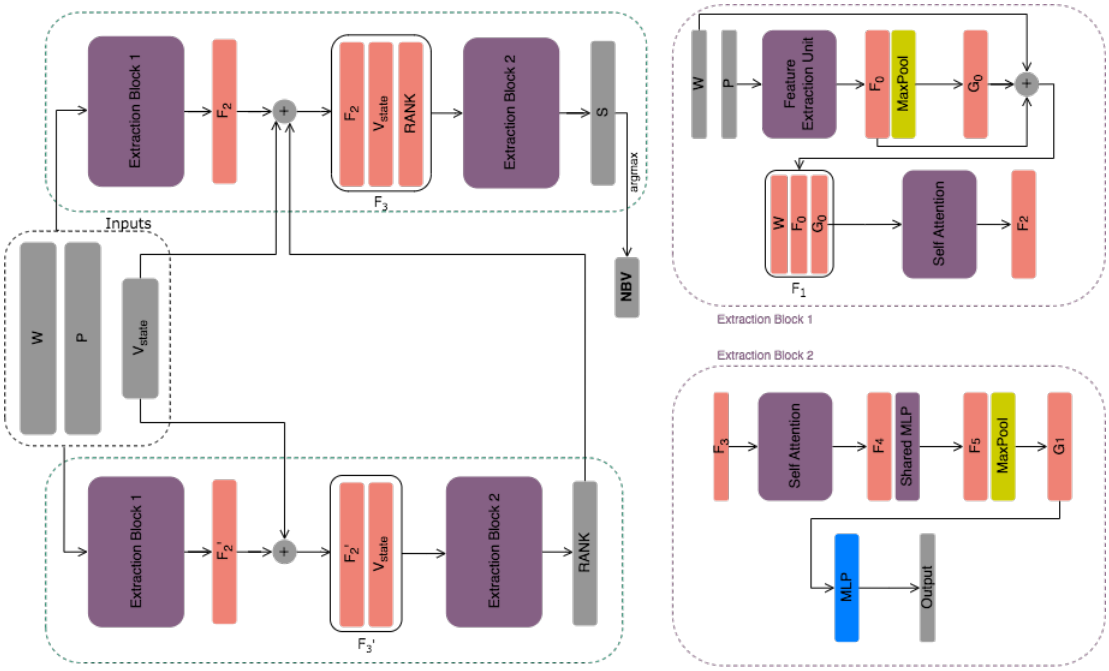


Figure 2.6: DB-NBV network architecture. Input and Output: Gray; Larger Units: Purple; Pooling Layers: Yellow; Fully Connected Layers: Blue; Features: Pink.

The authors compared the performance of this network against a volumetric NBV method, the Voxel based method NBV-net, and the point cloud method PC-NBV. The experimental results retrieved that DB-NBV achieved overall higher coverage scores than the other methods and in a short inference time.

In the first iterations other methods had a higher coverage score. However, from the third iteration forward DB-NBV starts achieving higher scores.

2.3 Uncertainty in Deep Learning

Deep learning models are not always certain about their predictions. Taking for example a model that is fed with a test point that is out of the distribution of the set it was trained on: it is forced to output a prediction for this point. However, it does not have enough training data to support the decision. The probabilities of it resulting in a wrong prediction are high, the model makes a prediction with high uncertainty. Hence, in some applications it can be important to know the model's prediction with an extra value that measures how certain the model is of its output.

In literature, uncertainty is usually split into two types: aleatoric and epistemic [23]. Aleatoric uncertainty is related to the notion of randomness, is introduced by the noise inherent in the observations. This type of uncertainty is also sometimes referred to as irreducible uncertainty because it usually cannot be removed. Epistemic uncertainty on the other hand is related to the model uncertainty - uncertainties in the model parameters or on the models structure - it represents the ignorance of the model. Due to its nature this type of uncertainty can often be reduced by increasing the size of the training data, for example.

The standard tools used to create deep learning models, like feedforward neural networks and convolutional neural networks, cannot usually access the models confidence in its predictions. However, in Bayesian networks [24] this becomes possible - this type of networks represent probabilistic dependencies between variables, here the weights of the networks are defined as probabilistic distributions, which offers a mathematically grounded framework to reason about model uncertainty. [25] explains that Bayesian networks are able to reason under uncertainty and introduces a method that uses dropout layers [26] to approximate neural networks to Bayesian networks and the Monte Carlo sampling method to enable the network to infer the uncertainty of the predictions.

In a standard neural network the network's goal is to learn a function f with parameters θ (the weights of the network) that maps the inputs x to the observed outputs y such that $|f(x, \theta) - y| \approx 0$ (minimizing the discrepancy between the predicted values $f(x, \theta)$ and the actual values y). So, in reality the goal of the training process is to learn the fixed values of the weights that minimize this discrepancy in order to make predictions (y') close to reality. In Bayesian neural networks the situation is slightly different. The weights are no longer fixed valued but are instead random variables, therefore represented as probabilistic distributions. Here the goal of the training process is to learn the posterior distributions of the weights, given the observed data (x, y). The posterior distribution can be analytically obtained with

the Bayes Theorem (2.5)

$$P(\mathbf{W}|\mathbf{X},\mathbf{Y}) = \frac{P(\mathbf{Y}|\mathbf{W},\mathbf{X}) \times P(\mathbf{W})}{P(\mathbf{X},\mathbf{Y})} \tag{2.5}$$

However, this is computationally challenging due to high-dimensional parameter space and complex network architectures.

So a way to approximate this computation when it is computationally infeasible, is through the use of variational inference [27]. Variational inference frames the problem of finding the posterior over the weights as an optimization problem. It attributes a variational distribution over the network's weights ($q(\mathbf{W})$):

$$b_{i,j} \approx Bernoulli(p_i) for j = 1, ..., k_i, \mathbf{W}_i = M_i diag(b_i) \tag{2.6}$$

and minimizing the Kullback-Leibler (KL) divergence between $q(\mathbf{W})$ and the true posterior it is possible approximate the posterior.

[25] states that this minimization is equivalent to maximizing the log evidence lower bound, optimizing the objective function (2.7)

$$\mathcal{L}_{GP-MC} \propto \frac{1}{N}\sum_{n=1}^{N}\left(-\frac{\log p(\mathbf{y}_n|\mathbf{x}_n,\hat{\omega_\mathbf{n}})}{\tau}\right) + \sum_{i=1}^{L}\left(\frac{p_i l^2}{2\tau N}||\mathbf{M}_i||_2^2 + \frac{l^2}{2\tau N}||\mathbf{m}_i||_2^2\right) \tag{2.7}$$

The work shows that the objective function optimized by dropout layers (2.8):

$$\mathcal{L}_{dropout} = \frac{1}{N}\sum_{i=1}^{N}(E(\mathbf{y}_i,\hat{\mathbf{y}}_i)) + \lambda\sum_{i=1}^{L}\left(||\mathbf{W}_i||_2^2 + ||\mathbf{b}_i||_2^2\right) \tag{2.8}$$

is equivalent to the objective function obtained from the KL divergence (2.7).

Thus, [25] shows that a neural network with dropout applied before every weight layer is mathematically equivalent to a Bayesian Network.

However, the question still holds - how can the uncertainty be measured in these type of networks? In Bayesian networks, in each forward pass the network assigns a different value for the weights, sampled from the weight's distribution. So in practice, in this type of network, in each forward pass there is a different learned function f. Hence, with the same data a different prediction $y' = f(x,\theta)$ is made. So, if several forward passes are made with the same data, there are several different predictions as well, and in the end the network's final prediction can be given as an average of all the predictions, as

$$y' = \frac{1}{N}\sum_{i=1}^{N} f_i(x,\theta_i) \tag{2.9}$$

where N is the number of forward passes. Having several predictions, it is now possible to measure the

prediction's uncertainty by calculating the variance as

$$var = \frac{\sum_{i=1}^{N}(y' - f_i(x, \theta_i))^2}{N} \tag{2.10}$$

So, [25] states that dropout layers can emulate this behavior. In a network with dropout layers each neuron has some p probability (dropout rate) of being switched off. In each training step a different neuron is turned off, according to the probability p. It is as if we get a slightly different network each time, which means we can get several different predictions as well. Most commonly dropout layers are only used during training since they are being used to reduce overfitting [26]. However, in the method proposed by the authors, dropout layers also have to be used during inference. This way we get several predictions to average and are able to calculate the variance. This method is referred to as the Monte Carlo Dropout Method (MCDM).

Many articles have already successfully used the MCDM to modify already existing networks to make them account for uncertainty, achieving positive results as they were able to improve model performance. [28] uses dropout to modify SegNet - a deep convolutional encoder-decoder semantic segmentation network [29]. They add dropout layers to the network and use it during test time to obtain the posterior distribution of class probabilities, taking the mean of the samples as the prediction and the variance to output the model uncertainty for each class. The authors state that the altered network [28] produces a reliable measure of model uncertainty that is also very effective when modeling smaller datasets. Further, the authors tested how the accuracy of the model improved with the number of Monte Carlo (MC) samples, the results showed that the accuracy converged after 40 MC samples. This article also studies the optimal position of the dropout layers in the network, stating that in practice applying dropout after each convolutional layer is not optimal. For them the optimal result was obtained when the central encoder and decoder were dropped out. The dropout probabilities were set at 0.5. In the end the authors stated that modeling uncertainty improved segmentation performance by 2-3%.

Likewise, [30] modifies the PoseNet network [31] - a regression deep learning network that regresses a 6-DOF camera position from an image. In this work the authors fit a unimodal Gaussian to the MC samples, treating the mean of the samples as the point estimate for the pose, and the trace of the unimodal Gaussian's covariance matrix as the uncertainty measurement. This paper also stated that the MC sampling converged after 40 samples, and that using dropout after each convolutional layer in practice diminished the networks performance. Concluding that applying dropout layers only before any layer where the weights were randomly initialized was the optimal architecture design to improve the performance of the model. In this paper the authors also set the dropout probabilities at 0.5. In the end, the results showed that the prediction given by the mean of the MC samples originated a more accurate solution. In this way it became possible to improve the localization accuracy.

3

Methodology

Contents

Numerous deep learning models yield good results for the NBV prediction, as was demonstrated in Chapter 2. These models, however, do not take into consideration the uncertainty of their predictions, which has been shown to have the ability to enhance a model's performance. In this thesis particular case a prediction with high uncertainty would mean that the NBV was probably not being chosen, but instead just some other viewpoint possibly not as relevant. This affects the efficiency of the network, since more scans could be needed to achieve the same coverage scores.

To address this limitation and improve the performance of NBV prediction models, this dissertation proposes modifying a NBV deep learning model as presented in [25] to make it account for uncertainty. The modification effects on the prediction will be examined, and two uncertainty-based metrics will be studied to improve view selection. The uncertainty measurements will be analyzed by evaluating their relevance as decision-support measures, and analyzing their potential ability to improve the model's final coverage score prediction.

In brief, this thesis aims to enable the model to predict an uncertainty measurement, along with its regular prediction, that reflects how sure the model is of each prediction. This uncertainty measurement could then be used to assist the NBV choice process, and could ideally be used to improve the model's outcome, making it closer to reality.

The chosen NBV model to modify was PC-NBV [1], as it was the reviewed model that showed better performance results. As a matter of fact, as seen before, DB-NBV [5] showed slightly better results compared to PC-NBV. However,at the time this work was produced, the authors had not made the code publicly available, making it harder to use in this work.

This chapter provides an overview of the work developed in this dissertation. Describes the used dataset and how it was created, as well as the model training process. Moreover, it explains how the PC-NBV model was modified and the followed steps to obtain and evaluate the uncertainty metrics.

3.1 Dataset Description

3.1.1 Generation

The PC-NBV's authors did not provide the partitions of the training/testing set they used to train the model on, but made available the process they followed to generate it. To be able to compare the results achieved in this dissertation to the ones achieved by the original model it is important to train the model with the same data, to reduce the number of differences between the two works. Therefore the first critical task was to follow the same process to create the training and testing datasets.

The adopted approach to generate the dataset was by making a simulated 3D reconstruction of objects whose full 3D models are previously known. These models were obtained from ShapeNet [21] - a dataset with hundreds of 3D models organized in several categories (called classes) of objects. The

original paper only refers which classes of objects were used, not referring which models inside each class. Therefore, the dataset build for this work will not have the exact same models.

For the training set 4000 models from 8 categories (airplane, cabinet, car, chair, lamp, sofa, table and vassel) were randomly chosen (500 models from each category), and another different 400 models for the validation set (50 models from each category). Regarding inference two sets were created - one with 400 models selected from these 8 categories known known in the training/validation sets, and another set - the novel set - with 400 models from other 8 categories not learned by the model. This last testing set was created to evaluate the performance of the model on unknown objects.

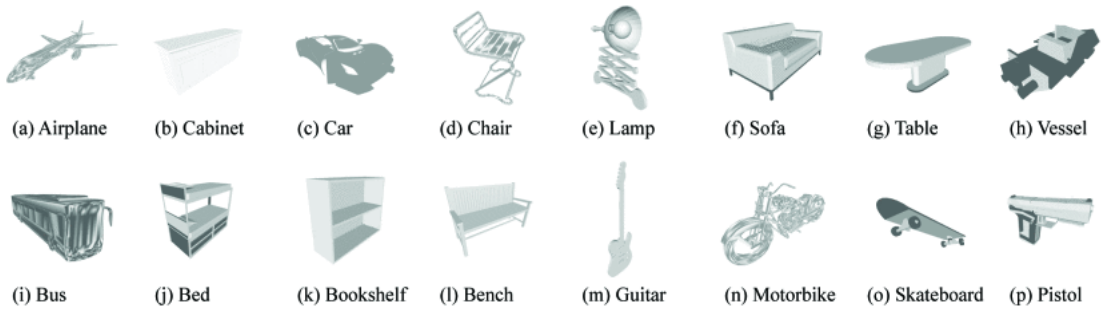


Figure 3.1: First row: Classes of objects used for training, validation and testing. Second row: Classes of objects used for the novel dataset. - Source: [1]

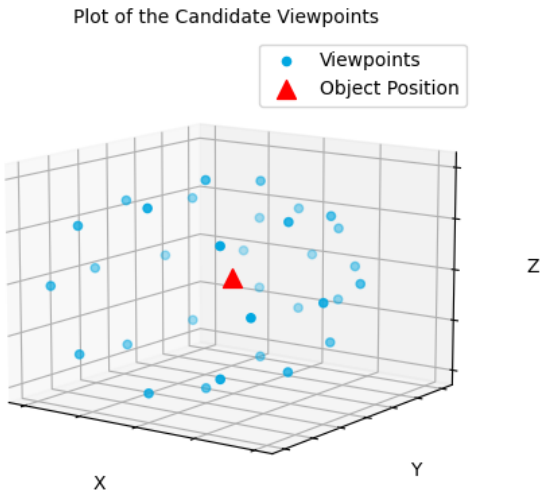


Figure 3.2: Candidate View Space - $n_v = 33$ viewpoints with direction pointing towards the origin, from each point in the plot.

The selected ShapeNet models were then processed to create the NBV data. ShapeNet models are represented in triangular meshes, so firstly every object mesh is uniformly sampled to create the complete point cloud of the object, P_O. Then the view space where the NBV is going to be searched in, needs to be defined. The authors provided a file with the coordinates of each viewpoint, so that was the used view space, to decrease the differences between works. This view space is represented in Figure 3.2, it is composed of 33 uniformly sampled viewpoints (n_v) on a sphere around the object.

Then, as briefly explained in Chapter 2, an artificial reconstruction of the 3D object is made for each model. At each viewpoint from the candidate view space it is simulated the point cloud acquisition with a virtual camera in the 3D environment. Then a first viewpoint is randomly selected and the search for the NBV begins. All the viewpoints are evaluated calculating the surface coverage score each view would achieve if selected (2.2). The one with a higher score is selected as the NBV. This scanning process occurs for 10 scans. The training data then consists of 3 types of information that is stored in each scan, during the reconstruction process:

- The selection state of each viewpoint, V_{state} - when a viewpoint is selected as the NBV its selection states changes from 0 to 1. This information is stored in a vector with the length equal to the number of viewpoints ($n_v = 33$).

- The coverage score of each viewpoint, $C(P)$ - whose value belongs to the interval $[0, 1]$. If a viewpoint was already chosen as the NBV in an eralier scan it's coverage score is zero.

- The accumulated point cloud, P_{part} - the point cloud of the reconstructed object until that point of the scanning process. This point cloud starts empty in the beginning of the process, and in each scan the point cloud from the selected NBV is appended to it. In the last scan P_{part} will be the complete point cloud of the reconstructed object.

The data acquisition process follows the Algorithm 3.1. In this algorithm P_{new} the point cloud of points that are capturable by a viewpoint and are not yet in the P_{part} point cloud.

Algorithm 3.1: NBV Training Data Preparation. Source: [1]

Input: O, P_o, V_c, $max_{iter} = 10$
$V^k_{state} \leftarrow 0$ $(k = 1, 2, 3, \ldots, m)$
$iter \leftarrow 0$
$P_{part} \leftarrow \emptyset$
$i \leftarrow Random(1 : m)$
while $iter < max_{iter}$ **do**
  $V^i_{state} \leftarrow 1$
  $P_{part} += P(v_i, O)$
  $max \leftarrow 0$
  for $j \leftarrow 1 : m$ **do**
    if $C(P^j_{new}) > max$ **then**
      $max \leftarrow C(P^j_{new})$
      $i \leftarrow j$
    end
  end
  $Save(P_{part}, V_{state}, C(P_{new}))$
  $iter++$
end

Each scan of this artificial reconstruction is fed to the model as an independent sample, i.e the model is not informed of the coverage scores of the previous scans, it only is informed of the views that have already been used (View State), and of the point cloud of the reconstructed object up to that phase (P_{part}). The coverage scores of each viewpoint is what the model aims to predict.

3.1.2 Data Characterization

It is natural that the number of new points is higher in the first scans and gets increasingly lower throughout the following scans. Therefore, in each set of 10 scans it is expected that a large portion of the final scans have low coverage scores, while the first's scans achieve higher coverage scores. This behavior is represented in Figure 3.3 that shows the $C(P)$ and P_{part} contents of the training data for a single object. In this sample it is visible that the first two scans are the ones with a higher coverage score, while in the proceeding ones the coverage scores take very small values - here more than half of the scans have small coverage scores. Accordingly, it is expected the distribution of the NBV coverage score per sample not to be uniform, having a higher number of samples with small coverage scores. The distribution of the NBV coverage scores of the test set is presented in Figure 3.4, where this behavior can be observed. Table 3.1 shows the View State representation of the sample presented in Figure 3.3.

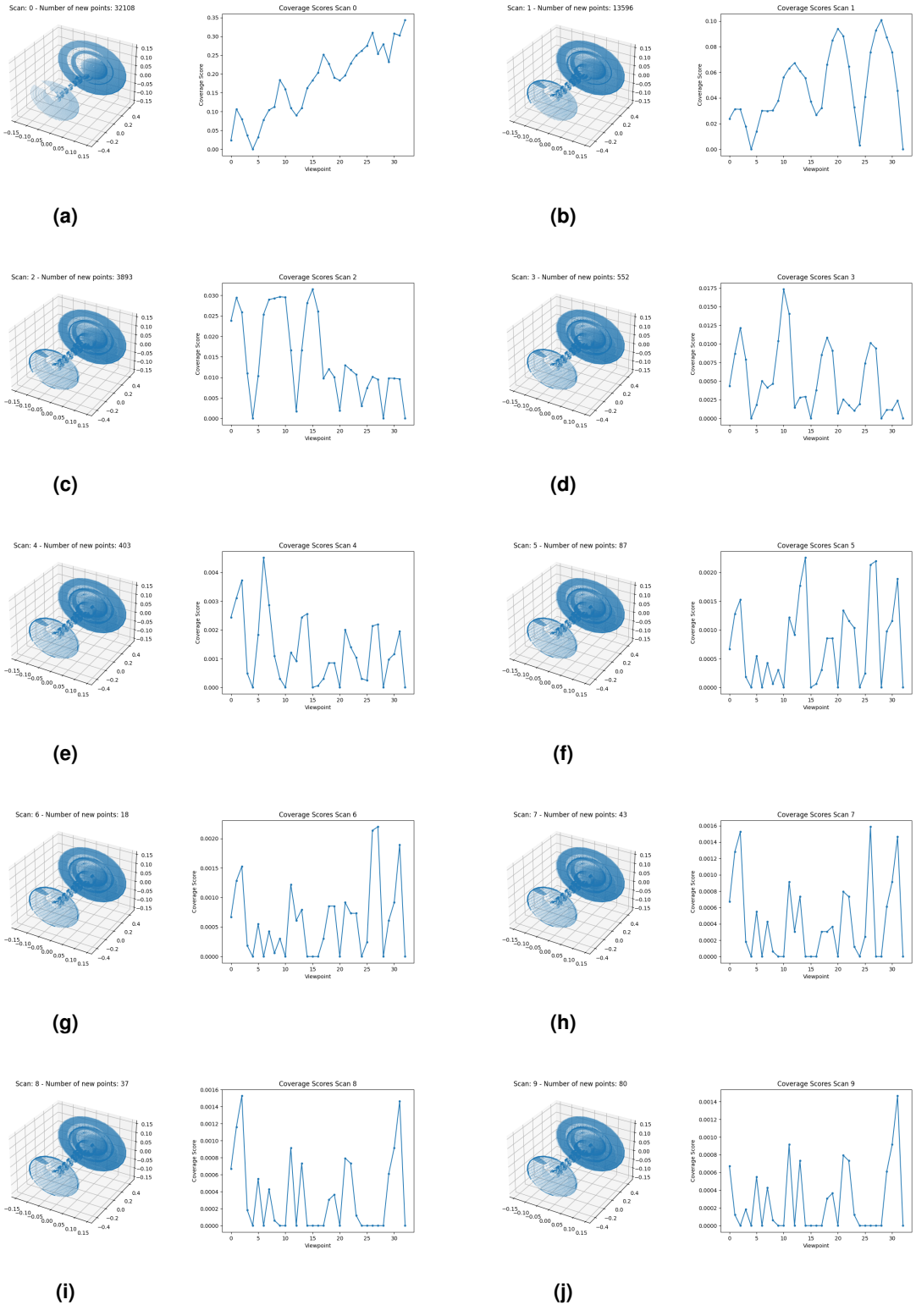


Figure 3.3: Training Sample. $C(P)$ and P_{part} for each scan of a training object. (Represents 10 test samples.)

Table 3.1: View State of an object throughout the 10 scans.

Scan 0	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Scan 1	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
Scan 2	0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
Scan 3	0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
Scan 4	0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
Scan 5	0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
Scan 6	0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1
Scan 7	0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 1
Scan 8	0 0 0 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1
Scan 9	0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1

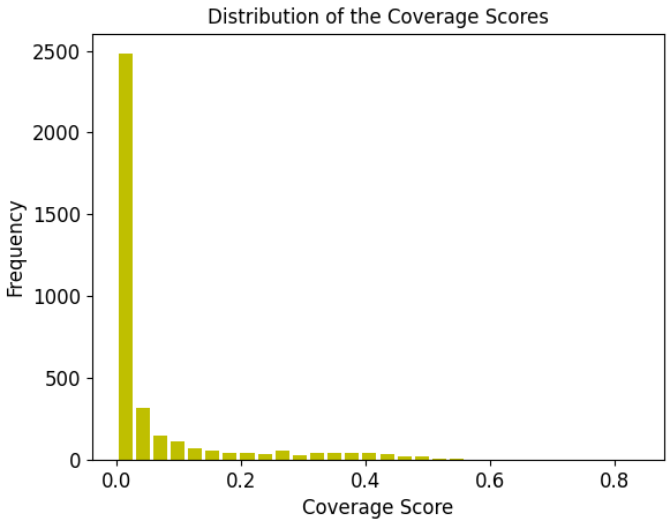


Figure 3.4: Distribution of the coverage scores along the test set.

3.2 Training and Model Architecture

3.2.1 Baseline training

To be able to evaluate the results of the network that is going to be created by modification of the PC-NBV model it is first important to get a baseline to compare the results to. This baseline is given by the results of the original model. So it is important to train and test this model on the created dataset.

The model code provided by the authors is implemented in a *Tensorflow* environment [1]. However, when testing the code it presented some incompatibilities with some functions used to serialize the data. So an equivalent implementation in *Pytorch* [2] that is publicly available and reportedly achieves the same performance levels was used.

[1] https://github.com/Smile2020/PCNBV/tree/master
[2] https://github.com/xdHKPC/PCNBV_pytorch/tree/main

The PC-NBV model was trained in the training set for 300 epochs, with a batch size of 32, a 0.01 learning rate and a 10^{-6} weight decay. The trained model of the epoch that achieves the lowest training loss was then used to evaluate the model performance on the two test sets.

3.2.2 Architectural Changes

To enable the model to account for uncertainty by implementing the MCDM dropout layers need to be applied after each convolutional layer. The new model architecture is presented in Figure 3.5, where the added dropout layers are signalized in dark green. Since adding dropout layers to the model is the same as approximating it to a Bayesian model, from now on this new architecture will be referred to as Bayesian PC-NBV.

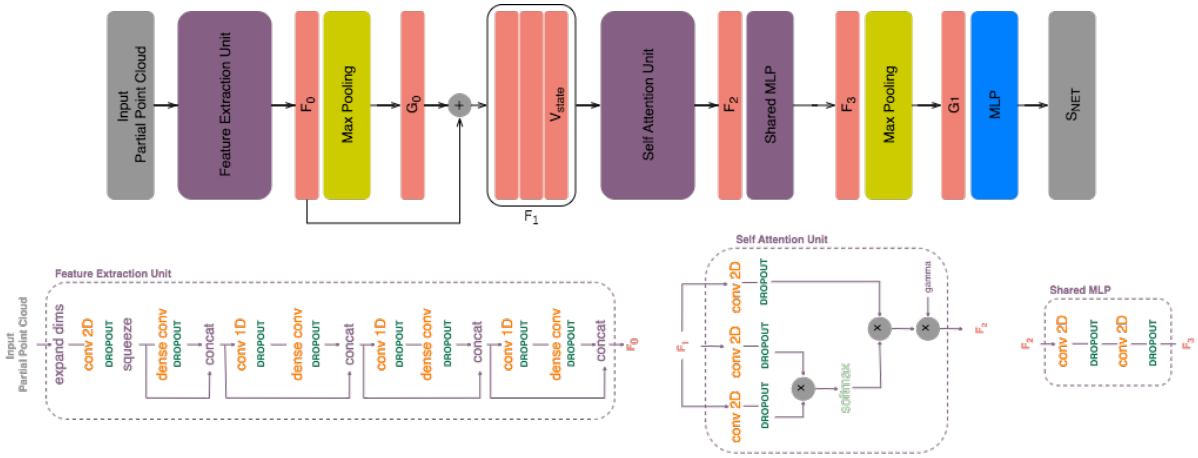


Figure 3.5: Bayesian PC-NBV architecture. Input and Output: Gray; Larger Units: Purple; Pooling Layers: Yellow; Fully Connected Layers: Blue; Features: Pink; Convolutional Layers: Orange; Activation Layers: Light Green; Dropout Layers: Dark Green.

When setting the dropout layers, a dropout probability needs to be defined. Usually, the dropout probability falls between 0.2 and 0.5. This is an hyperparameter that could be optimized, however, due to time constraints, it was decided to set it to 0.5, as done in [28, 30], and leave that optimization to future work.

The model was then trained with the same parameters used in the PC-NBV training.

3.2.3 Evaluation metrics

In this dissertation the performance of the models is evaluated with two metrics besides the loss. These two metrics are the error and accuracy of the predictions. The error measures the difference between the model prediction and the ground truth, it differs from the loss (2.3) since it does not account for the regularization loss. The accuracy measures if the model's predicted NBV matches the groundtruth's

NBV.

As the model predicts a coverage score for each viewpoint we are measuring the error between two 2D variables. Various error metrics were evaluated, in order to pick the one that would be the most relevant, these results will be shown in the next chapter.

$$Euclidean\ Distance = |gt - fp| = \sqrt{\sum_{i=1}^{n_v} (gt_i - fp_i)^2} \tag{3.1}$$

measures the error by calculating the Euclidean Distance between the groundtruth (gt) and the model prediction (fp). As

$$Squared\ Error = \sum_{i=1}^{n_v} (gt_i - fp_i)^2 = Euclidean\ Distance^2 \tag{3.2}$$

measures the sample error by calculating the sum of each view Squared Difference between gt and fp, which is the square of the Euclidean Distance. The model error is then given by the mean of the sample errors.

The accuracy is another metric analyzed in this work. Despite the fact that PC-NBV is a regression model, the scores it regresses are used to choose a NBV, i.e - in a real world problem the closeness of the coverage score predictions to the groundtruth is not 100% important, the most important aspect is that the model predicts the highest coverage score on the right viewpoint. And for this is important to evaluate if the model has the ability to predict the right NBV viewpoint or not.

In most of the cases, in the groundtruth there is only one NBV - the viewpoint with the highest coverage score. However, up on further examination, when visualizing the data, it is possible to understand that in some cases more than one view achieve coverage scores close to the NBV coverage score value. In practice, if we assume independence between a sequence of selections, it wouldn't make a lot of difference to choose one of these views over another, therefore in some cases there is more than one possible NBV. To calculate the accuracy of each sample this circumstance is taken into consideration. We analyze if the predicted NBV belongs to a set of views whose coverage scores lay in the interval given by Equation (3.4), i.e. a set of top views with close coverage scores between each other. If the model chooses a viewpoint that is inside this set we assume the prediction is right, if not the prediction is set as wrong - Equation (3.3). In the end the model accuracy is given by the ratio between the number of right predictions and the total number of samples - Equation (3.5). The graphs in Figure 3.6 allow for a better comprehension of what is considered a right prediction - the viewpoints that lay on the gray area are the ones considered as NBVs.

$$accuracy(fp) = \begin{cases} 1, \text{if } fp_{NBV} \in \{\text{NBVS}\} \\ 0, \text{otherwise} \end{cases} \tag{3.3}$$

$$\{NBVS\} = \{v_{i^*}\}, \{i^*\} = \{i^* \in arg(C),$$

$$C \in [max(fp) - 0.15\alpha, max(fp)]\} \tag{3.4}$$

$$\alpha = max(C) - min(C)$$

$$model\ accuracy = \frac{\sum_{i=1}^{n_s} accuracy(fp_i)}{n_s} \tag{3.5}$$

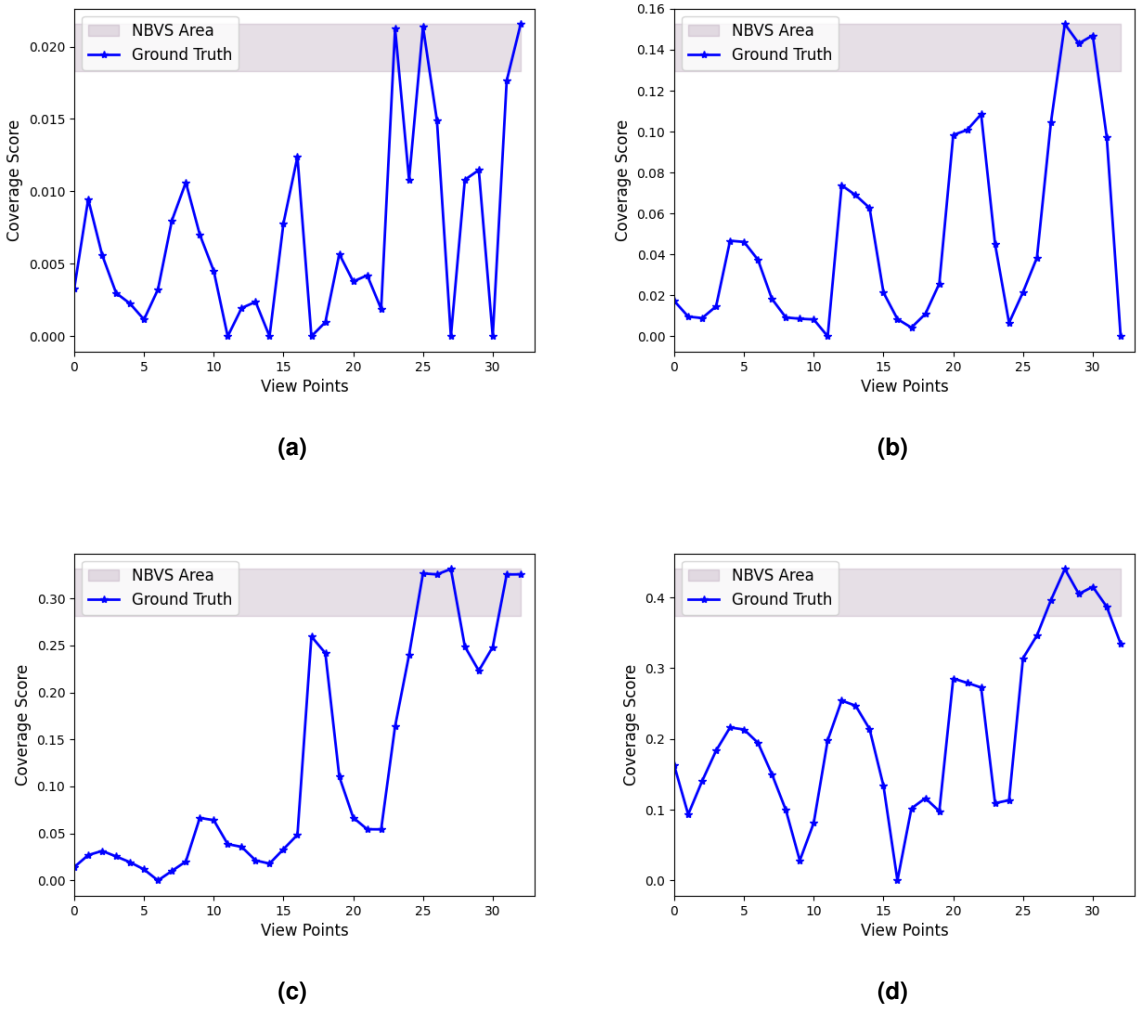


Figure 3.6: Coverage Score groundtruth plot with signalized NBVS interval.

3.3 Bayesian PC-NBV Inference and Uncertainty Measurements

3.3.1 Bayesian PC-NBV Inference

The usual behavior of dropout layers is to be on during testing and off during inference. However, in this case, to be able to get several different predictions for the same test sample, it is desired that the dropout layers are on during inference as well. For this a small routine that specifies that during inference dropout layers should keep their training behavior was implemented.

The MCDM requires a somewhat large number of inferences of the same test sample. Based on the conclusions of the works [28, 30] presented on Chapter 2 this number (n_{mc}) was set to 40. The 40 MC samples (mc) are then averaged to obtain the model's final prediction:

$$fp = \widehat{C(P)} = \frac{\sum_{i=1}^{n_{mc}} mc_i(x)}{n_{mc}} \tag{3.6}$$

Just like before, also this model is going to be analyzed in terms of sample error and accuracy.

3.3.2 Uncertainty Measurements

As seen in Chapter 2, the uncertainty measurements are given by the differences along the MC samples. In theory, samples that show larger differences between themselves reflect a higher model uncertainty. Hence, the uncertainty measurements are obtained by calculating standard deviations between the different samples, with reference to the mean of the samples - the final prediction. The standard deviation was chosen over the variance since the latter presented very small values.

The model prediction is composed of a coverage score for each of the 33 viewpoints. So it is possible to calculate the standard deviation for each viewpoint, having an uncertainty metric for each view. However, it would be more interesting if the model could return just a single value that reflects how certain the model is of its prediction. So, to address this question several standard deviation metrics were studied:

- Standard Deviation by View, σ_v:

$$\sigma_{v_j} = \sqrt{\frac{1}{n_{mc}} \sum_{i=1}^{n_{mc}} (fp(v_j) - mc_i(v_j))^2}, \quad j = \{1, ..., n_v\} \tag{3.7}$$

  Measures the standard deviation between the model's final prediction and the MC samples for each viewpoint.

- Mean Standard Deviation, $\overline{\sigma}$:

$$\overline{\sigma_v} = \frac{1}{n_v} \sum_{j=1}^{n_v} \sigma_{v_j} \tag{3.8}$$

Is given by the mean of a sample's standard deviation by view. Returns only one value that quantifies the standard deviation of all the views for that sample.

- NBV Standard Deviation, σ_{NBV}:

$$\sigma_{NBV} = \sigma_{v_{i*}} \tag{3.9}$$

$$v_{i*} = argmax(fp)$$

It is given by the standard deviation of the view the model selected as the NBV, returning only one value.

- Standard Deviation as a Whole, σ_{whole}:

$$\begin{aligned} \sigma_{whole} &= \sqrt{\frac{1}{n_{mc}} \sum_{i=1}^{n_{mc}} |fp - mc_i|^2} = \\ &= \sqrt{\frac{1}{n_{mc}} \sum_{i=1}^{n_{mc}} \left[\sum_{j=1}^{n_v} \left(fp_j - mc_{i_j} \right)^2 \right]} \end{aligned} \tag{3.10}$$

It measures the standard deviation for the prediction as a whole, by calculating the difference between the final prediction and the MC samples as an Euclidean Distance. Returns only one value that quantifies the uncertainty of the whole prediction.

All of these metrics were evaluated to understand which one could better translate the uncertainty and better adapted to this problem.

In the next chapter all of the results of these experiments are shown and interpreted, and a study of the usage of the uncertainty metrics to improve the model's accuracy and performance is made.

4

Results Analysis

Contents

In this chapter the findings obtained from the experiments and analysis conducted during this dissertation are presented. Following the methodological framework outlined in Chapter 3, this section analyzes the results derived from the modified PC-NBV model - Bayesian PC-NBV - proposing and evaluating uncertainty metrics that aim to improve the model's performance in terms of error reduction and accuracy improvement. The results offer relevant insights into the effectiveness of this type of model modifications in quantifying uncertainties and improving the model's predictive capability.

The chapter begins by providing an overview of the results of both the baseline and the modified model training and testing processes, using the performance metrics described in the previous chapter - loss, error and accuracy. Later it evaluates the uncertainty measurements generated by the Bayesian PC-NBV model, evaluating their quality and relevance.

Through comparative analysis and visualization techniques, this chapter studies the relation between prediction accuracy and uncertainty estimation, enlightening the potential efficacy of uncertainty-aware NBV prediction models.

4.1 Training and Testing Results

4.1.1 PC-NBV

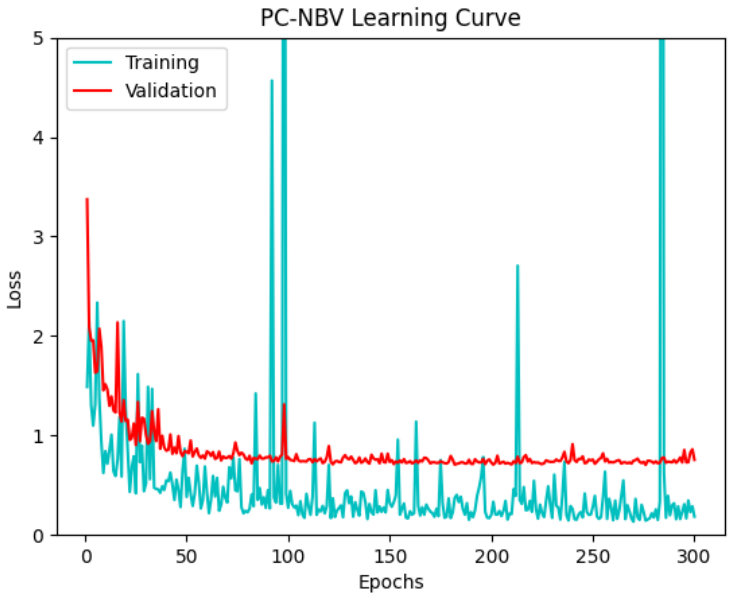


Figure 4.1: PC-NBV Learning Curve. Loss given by Equation (2.3)

The PC-NBV model was trained in the training set with the parameters described in the previous chapter - it was trained for 300 epochs, with a batch size of 32, a 0.01 learning rate and a 10^{-6} weight decay. Its

learning curve is shown in Figure 4.1. By looking at it we can see that the model was able to learn the data without overfitting it. Table 4.1 presents the training, validation and testing loss results (2.3), as well as the loss values the PC-NBV authors presented on their article.

Table 4.1: PC-NBV Loss (2.3) - Comparison between the results achieved by this paper and the results achieved in the original paper.

PC-NBV	Loss	Loss Original Paper
Train	0.144	-
Valid	0.691	0.189
Test	0.662	0.176
Test Novel	1.160	0.316

There is a noticeable difference between this paper values and values presented in the original work. The loss values achieved in this work are around 3 times higher. The PC-NBV model trained in this work did not achieve the same performance level as the the one shown in the original paper. Upon evaluation of the provided loss values, it becomes evident that the performance documented in the original paper could not be replicated in this work, having been established, in this work, a baseline model with a lower performance.

While the same code was used, differences in hardware and software can explain the difference of performance between both works. The computational resources in this work were different from the ones used in the original paper, this can affect the speed and efficiency of the training process, leading to differences in the convergence of the model and in its loss values. Software differences are also likely to have occurred, namely during the processing of the Shapenet models to generate the training data. The model processing required the installation of Blender and libraries such as Open3D and OpenEXR. The installation of the Open3D library posed some problems due to version incompatibilities, being only possible to use it with python 2.7, when the authors mentioned the use of python 3.7. So when this library had to be used the python version used was different, which may have lead to errors in the processing of the point clouds due to behavioral differences between versions. All of these inevitable differences in the training environment can have lead to the difference in model performance.

Since it was not possible to achieve the results presented in the original article, and to guarantee the comparability of the results, the model baseline used in this work will be the one obtained. So, in this work and from now on, the performance of the network that will be modified is going to be compared to this baseline performance.

Table 4.2: PC-NBV Model Error - Euclidean Distance (3.1) and Squared Differences (3.2)), and Model Accuracy (3.5)

PC-NBV	Model Error (Euclidean Distance)	Model Error (Squared Differences)	Model Accuracy
Valid	0.070	0.018	29.76%
Test	0.069	0.017	29.95%
Test Novel	0.093	0.033	26.59%

The error and accuracy results are presented in Table 4.2, these are the metrics that are going to be used to compare model performances.

The model error as well as the model accuracy in the valid and test sets are quite similar. This was expected, since both of these sets have similar sizes and are composed by models in which the network was not trained on but of the object classes the network knows. In the test novel set, as expected as well, these values worsen: the model error increases 35% for the Euclidean Distance and 94% for the Squared Differences; as for the model accuracy it decreases 3.36 percentage points.

Figure 4.2 shows the different error metric distribution in the test set, it is visible that the vast majority of samples show errors with smaller value.

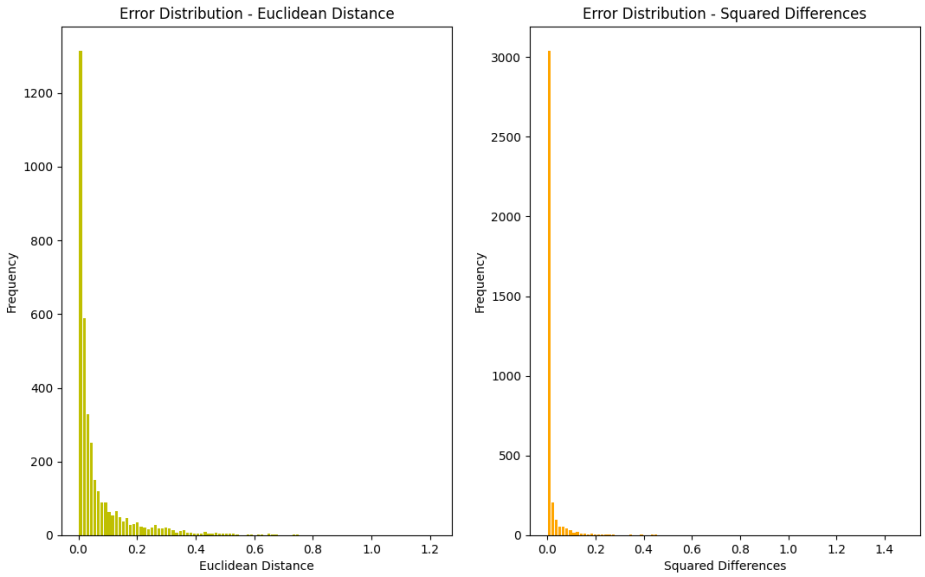


Figure 4.2: Error distribution on the test set - PC-NBV model.

4.1.2 Bayesian PC-NBV

After the modifications described in the previous chapter the Bayesian PC-NBV model was trained with the same parameters used for the PC-NBV model. The model's learning curve is presented in Figure 4.3. It is visible that also here the model did not overfitt the data. However, compared to the previous

training process, the loss seems now slightly more unstable. This is an expected behavior caused by the randomness dropout introduces during the training process, causing the loss to fluctuate more during training compared to when dropout is not used. Table 4.3 shows the network's loss, error and accuracy values.

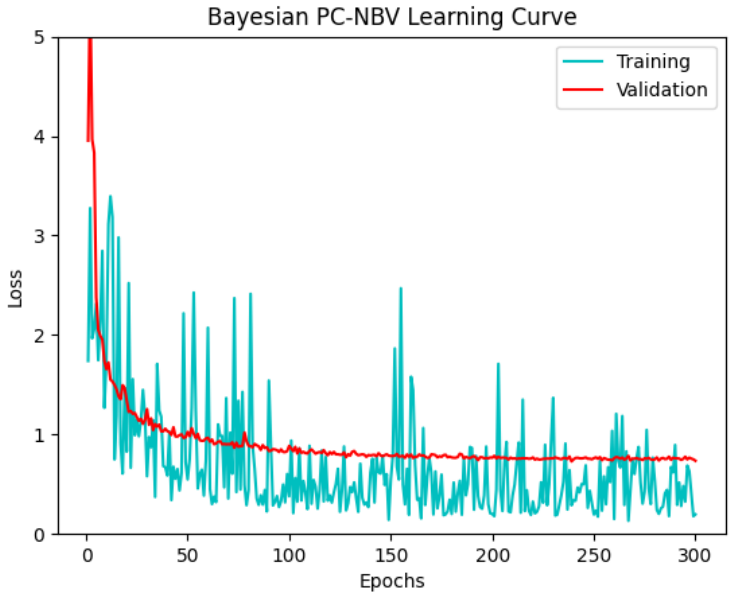


Figure 4.3: Bayesian PC-NBV Learning Curve. Loss given by Equation (2.3).

Table 4.3: Bayesian PC-NBV Model Loss, Model Error - Euclidean Distance (3.1) and Squared Differences (3.2), and Model Accuracy (3.5)

Bayesian PC-NBV	Loss	Error - Euclidean Distance	Error - Squared Differences	Accuracy
Train	0.287	-	-	-
Valid	0.704	0.072	0.018	28.66%
Test	0.662	0.071	0.017	29.02%
Test Novel	1.217	0.097	0.035	25.66%

Comparing the loss, error and accuracy values of the Bayesian PC-NBV model with the PC-NBV model there are very little performance shifts. The loss throughout the sets keeps quite similar values, with a small increase of about 0.1 in some sets. As for the model error values they virtually do not change between both models. Figure 4.4 shows the error distribution along the test set, and just like observed with the PC-NBV model the vast majority of samples show errors with smaller values. When it comes to the model accuracy there is a slight decrease in the performance of the model, it was verified a decrease of about 1 percentage point across all the sets.

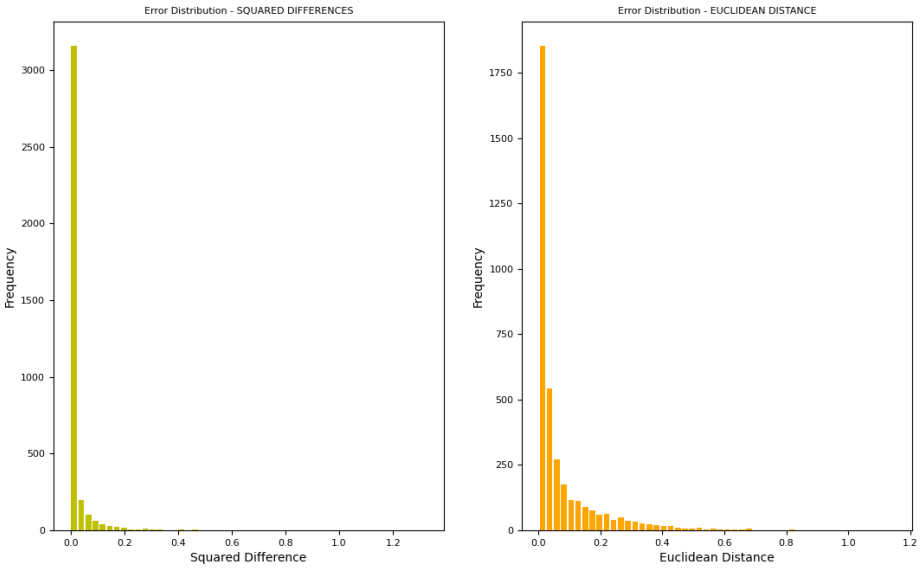


Figure 4.4: Error distribution on the test set - Bayesian PC-NBV model.

Figure 4.5 shows in colors the plots of the 40 MC samples for 4 test samples, in black their mean - the denominated final prediction (3.6) - and in blue the sample's ground truth. While observing these plots it is possible to conclude that the dropout was in fact active during inference - which explains the difference in model predictions for the same input data. It is also interesting to notice that the shape of the predictions is not always the same throughout the 40 inferences, meaning the model does not always predicts the same NBV, which will be relevant in the quantification of the prediction's uncertainty. Also, it is visibly possible to state that the mean of the predictions gives a good approximate of the ground truth.

All in all, it can be said that despite the introduction of dropout layers and the inherent randomness they entail during training, the Bayesian PC-NBV model keeps presenting a good performance when implementing the Monte Carlo sampling method during inference. Although some marginal shifts in loss, error, and accuracy metrics were verified, when comparing to the PC-NBV model, these were quite small and are not considered as a big performance drop. These small changes are the price to pay to enable the model to predict an uncertainty measurement, which will be addressed in the next section.

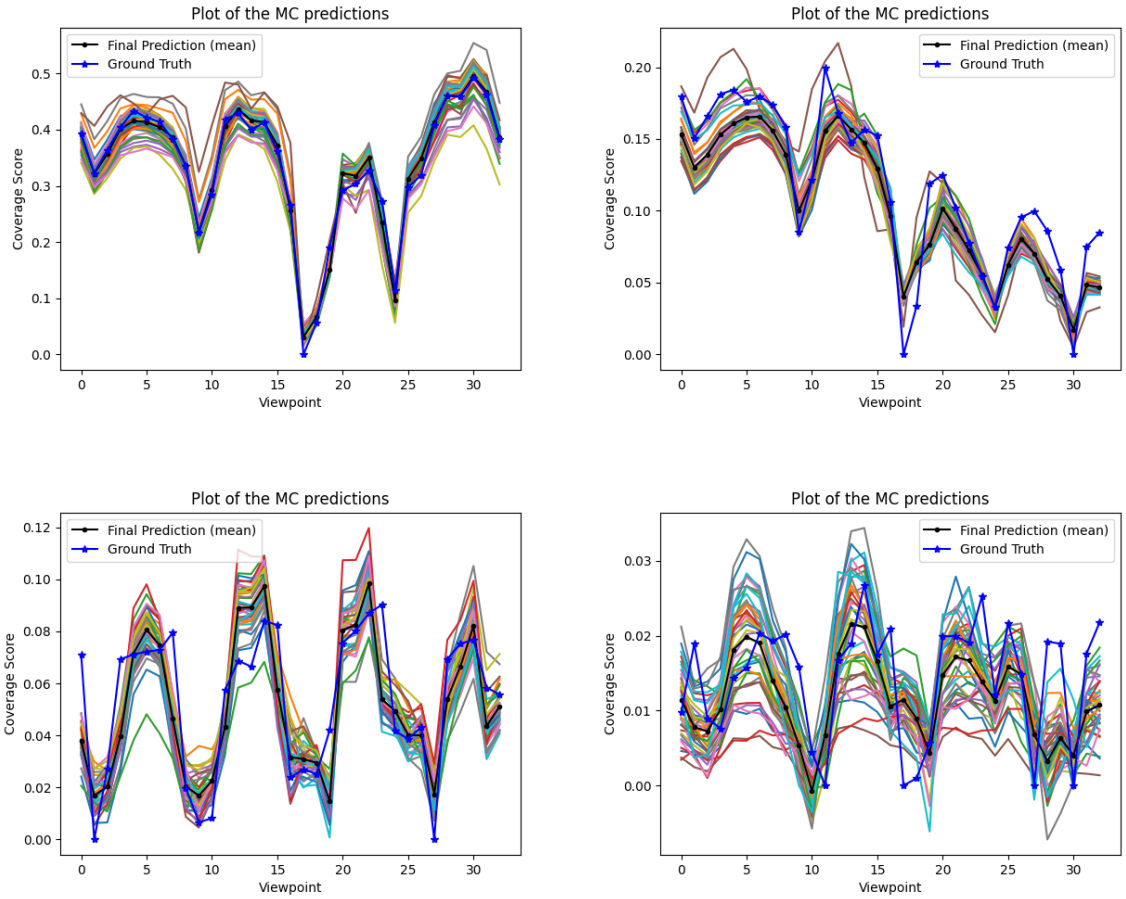


Figure 4.5: Plot of the Monte Carlo samples (in colors), their mean (in black) and the groundtruth (in blue).

4.2 Uncertainty Quantification

The biggest challenge of uncertainty quantification is to find a measurement that correctly reflects the metric we want to predict. In this case it would be interesting to predict a measurement that could model the sample's error and could also model the samples accuracy, i.e - a large uncertainty value would mean that the prediction would have a large error and a higher probability of being wrong (of not having predicted the right NBV). In the previous tests two error metrics were analyzed. Also, in Chapter 3, several uncertainty measurements were presented. Now an analysis of which uncertainty measurement best models which error metric and the accuracy is made.

First the relation between the uncertainty measurement and the error was studied. The desired relation between both metrics is a proportional one - when the uncertainty value is high the error value should be high as well. A purely horizontal or vertical pattern would not be good to differentiate between

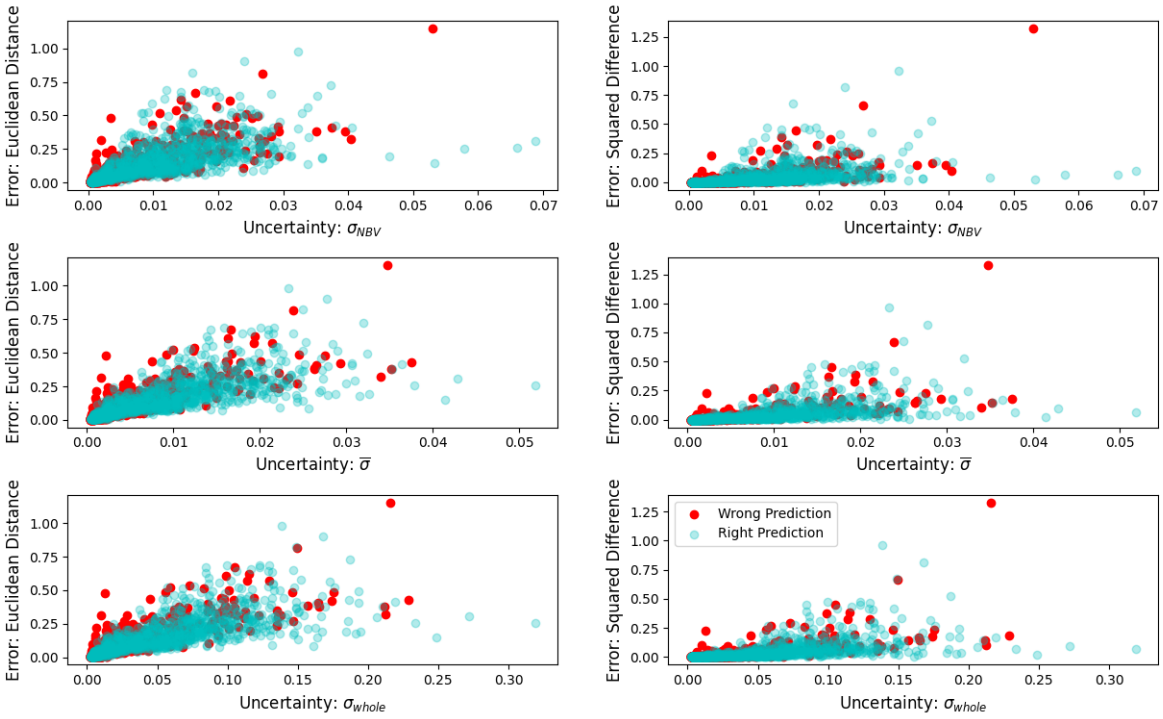


Figure 4.6: Distribution of the sample's error in function of uncertainty - different combinations. From top to bottom - uncertainty: σ_{NBV}, (3.9); $\overline{\sigma}$, (3.8); σ_{whole}, (3.10). From left to right - error: Euclidean Distance, (3.1); Squared Differences, (3.2).

right and wrong predictions. So to study this and understand which uncertainty measurement - $\overline{\sigma}$ (3.8), σ_{NBV} (3.9), and σ_{whole} (3.10) - best models which error metric - Euclidean Distance (3.1) and Squared Differences (3.2) - a plot of the error by the uncertainty of each sample for the different combinations was made, and is displayed in Figure 4.6. In addition, we want to study if the accuracy of the selected view can be modeled as a function of the uncertainty (3.8) - (3.10). To do that, we plot each sample with one of two colors: blue if presents a right view selection and red a wrong view selection.

Looking at the different plots in Figure 4.6 we only see an approximation of the desired proportional relation when the error metric is given by Euclidean Distances. When the metric is given by the Squared Differences the distribution is quite horizontal, with no clear distinction in error as uncertainty increases. Hence, the chosen error metric to model is going to be the Euclidean Distance (3.1).

Looking at the left side of the Figure 4.6, where the used error metric is the Euclidean Distance (3.1), it is possible to verify that the uncertainty metrics that better allow the desired distribution are $\overline{\sigma}$ (3.8) and σ_{whole} (3.10). Both of these metrics allow quite similar distributions. However, σ_{whole} has a larger range of uncertainty values. While the values of $\overline{\sigma}$ only vary in the interval $[0, 0.06]$ the σ_{whole} values vary in the interval $[0, 0.4]$ making it a more suitable metric.

Figure 4.7 shows a closer look at this distribution of the Euclidean Distance error (3.1) in function of the the σ_{whole} uncertainty (3.10), as well as the line that best fits the data. This fit is evidently not perfect,

having a $R^2 = 0.695$. However, it serves to highlight the proportional tendency of the data distribution.

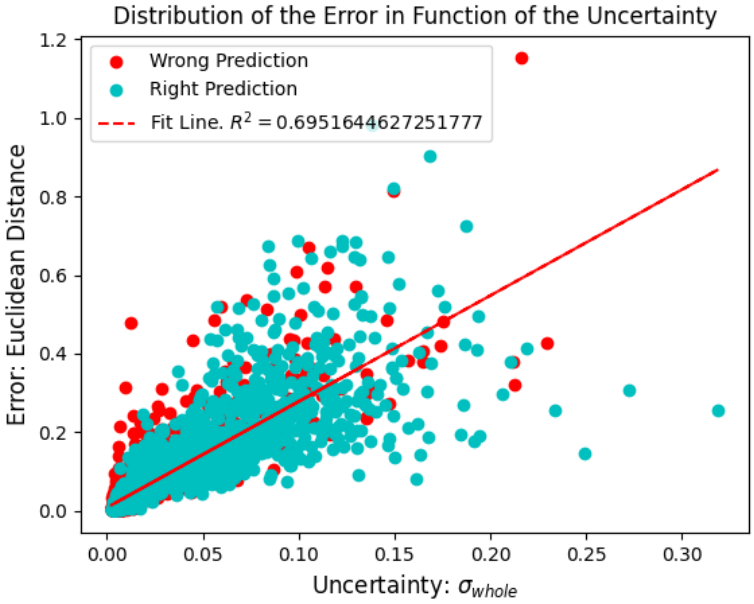


Figure 4.7: Distribution of the sample's Euclidean Distance error in function of the σ_{whole} uncertainty and its approximated fit line (in red) with $R^2 = 0.695$.

Despite this uncertainty metric being able to successfully model the prediction error given by the Euclidean Distances, it is visible that it is not capable of modeling the accuracy of the sample (3.3). The desired behavior would be for the blue dots (the right predictions) to be distributed in the lower range of the uncertainty values and the red dots (the wrong predictions) in the higher range. However, this is not visible - the σ_{whole} uncertainty metric does not model the accuracy of the samples.

As a matter of fact none of the tested uncertainty metrics plotted in Figure 4.6 are able to model the accuracy of the samples, as the above described desired distribution behavior is not visible in any of the plots. This is, however, an expected behavior because the accuracy of a sample (3.3) is not correlated to its error (3.1). The accuracy measures if the highest coverage score was attributed to the groundtruth's NBV while the error measures how close the coverage scores are from the groundtruth's coverage scores. In turn, the σ_{whole} uncertainty (3.10), that is able to model the sample's Euclidean Distance error (3.1), is calculated by computing the standard deviation of the Euclidean Distance between the MC samples and the final prediction (3.10), which is quite similar to the Euclidean Distance error calculation. Therefore, since the σ_{whole} uncertainty calculation is based on the MC samples Euclidean Distance error, it is natural that it can correctly offer an insight of the error, not being able to do so for the sample's accuracy.

Inspired by this train of thought, an uncertainty metric capable of modeling the sample's accuracy (3.3) can be obtained if it is calculated based on the the MC samples accuracy. Thus, we computed the

accuracy of every MC sample for uncertainty estimation as

$$accuracy(mc_i) = \begin{cases} 1, NBV_{mc_i} = NBV_{fp} \\ 0, otherwise \end{cases} \tag{4.1}$$

$$i \in \{1, ...40\}$$

This accuracy checks if the sample's NBV is the same NBV of the final prediction (3.6). If the NBV is the same the MC sample's accuracy is set to 1, otherwise is set to 0. Then the uncertainty is calculated as the Standard Deviation of the MC sample's accuracy, $\sigma_{accuracy}$:

$$\sigma_{accuracy} = \sqrt{\frac{1}{n_{mc}} \sum_{i=1}^{n_{mc}} (1 - accuracy(mc_i))^2} \tag{4.2}$$

It is now important to check if this $\sigma_{accuracy}$ uncertainty is able to model the sample's accuracy. Again, the desired behavior to be observed is that samples that have lower uncertainty values (4.2) are samples that get the NBV prediction right, and the ones with higher uncertainty values get the prediction wrong. Figure 4.8 shows an histogram that quantifies how many samples in each uncertainty bin get the NBV prediction right and how many get it wrong. It is visible that most of the samples in the low range of the $\sigma_{accuracy}$ uncertainty values are samples that got the prediction right, while most of the samples with higher uncertainty values are samples that weren't able to do so. This was indeed the desired behavior for accuracy modeling, and validates the proposition that in order for uncertainty to reflect accuracy it needed to be computed from values that measured this quantity.

So by now we have two uncertainty metrics capable of estimating uncertainty measurements that give insights into the prediction's error - σ_{whole}, (3.10) - and into the prediction's accuracy - $\sigma_{accuracy}$, (4.2). However, an important question arises: How can these metrics help improve the model's performance?

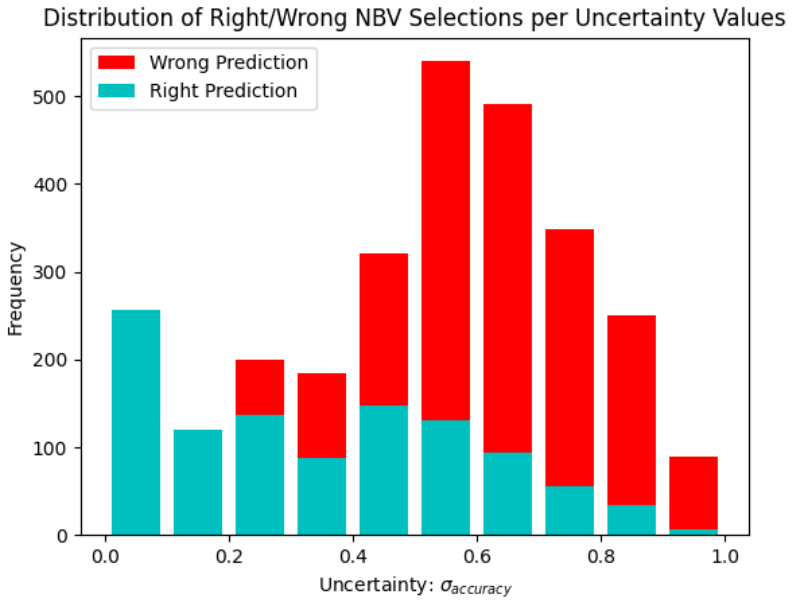


Figure 4.8: Distribution of right/wrong NBV predictions per uncertainty $\sigma_{accuracy}$ values (4.2).

4.3 Leveraging Uncertainty for Model Performance Improvement

The above studied uncertainty metrics have proven to be able to serve as a measure of how certain the model is of its predictions in terms of error and accuracy. In the context of this problem this achievement can be quite relevant. As the major goal is to achieve an efficient 3D model acquisition process, the simple rejection of a prediction with an high uncertainty can help improving the process efficiency.

4.3.1 Coverage Score Improvement

The first analyzed approach for performance improvement was by incorporating the uncertainty predictions with the coverage scores predictions, in an attempt of generating a new improved coverage score. This coverage score would ideally be more similar to the groundtruth, reducing the Euclidean Distance error (3.1). It could also improve the coverage score of the real NBV, increasing that view's coverage score, therefore improving the sample's accuracy (3.3). To do this several different combinations incorporating the uncertainty measurements with the coverage prediction were evaluated. The results are shown on Tables 4.4 to 4.8.

Table 4.4: New Coverage Score formula results with σ_{view} uncertainty. Error: (3.1), Accuracy: (3.5).

		σ_{view}			
		Error	Error Change	Accuracy (%)	Accuracy Change
$fp + \sigma$	Valid	0.078	0.006	28.61	-0.05
	Test	0.077	0.005	28.51	-0.51
	Test Novel	0.101	0.004	25.66	0.00
$fp - \sigma$	Valid	0.074	0.002	28.18	-0.48
	Test	0.074	0.002	29.46	+0.44
	Test Novel	0.098	0.001	25.37	-0.29
$fp \times \sigma$	Valid	0.203	0.131	24.60	-4.06
	Test	0.201	0.129	23.45	-5.57
	Test Novel	0.2145	0.1175	22.19	-3.47
fp/σ	Valid	25.688	25.616	16.88	-11.78
	Test	25.933	25.861	17.17	-11.85
	Test Novel	25.082	24.985	15.92	-9.74

Table 4.5: New Coverage Score formula results with $\overline{\sigma}$ uncertainty (3.8). Error: (3.1), Accuracy: (3.5).

		$\overline{\sigma}$			
		Error	Error Change	Accuracy (%)	Accuracy Change
$fp + \sigma$	Valid	0.077	0.005	28.66	0.00
	Test	0.076	0.004	29.02	0.00
	Test Novel	0.101	0.004	25.66	0.00
$fp - \sigma$	Valid	0.073	0.001	28.66	0.00
	Test	0.074	0.002	29.02	0.00
	Test Novel	0.098	0.001	25.66	0.00
$fp \times \sigma$	Valid	0.203	0.131	28.66	0.00
	Test	0.201	0.129	29.02	0.00
	Test Novel	0.215	0.118	25.66	0.00
fp/σ	Valid	25.739	25.667	28.66	0.00
	Test	25.8	25.728	29.02	0.00
	Test Novel	24.763	24.666	25.66	0.00

Table 4.6: New Coverage Score formula results with σ_{whole} uncertainty (3.10). Error: (3.1), Accuracy: (3.5).

		σ_{whole}			
		Error	Error Change	Accuracy (%)	Accuracy Change
$fp + \sigma$	Valid	0.153	0.081	28.66	0.00
	Test	0.152	0.08	29.02	0.00
	Test Novel	0.176	0.079	25.66	0.00
$fp - \sigma$	Valid	0.148	0.076	28.66	0.00
	Test	0.149	0.077	29.02	0.00
	Test Novel	0.178	0.081	25.66	0.00
$fp \times \sigma$	Valid	0.189	0.117	28.66	0.00
	Test	0.188	0.116	29.02	0.00
	Test Novel	0.2	0.103	25.66	0.00
fp/σ	Valid	4.125	4.053	28.66	0.00
	Test	4.139	4.067	29.02	0.00
	Test Novel	3.953	3.856	25.66	0.00

Table 4.7: New Coverage Score formula results with σ_{NBV} uncertainty (3.9). Error: (3.1), Accuracy: (3.5).

		σ_{NBV}			
		Error	Error Change	Accuracy (%)	Accuracy Change
$fp + \sigma$	Valid	0.08	0.008	28.66	0.00
	Test	0.078	0.006	29.02	0.00
	Test Novel	0.103	0.006	25.66	0.00
$fp - \sigma$	Valid	0.075	0.003	28.66	0.00
	Test	0.075	0.003	29.02	0.00
	Test Novel	0.099	0.002	25.66	0.00
$fp \times \sigma$	Valid	0.203	0.131	28.66	0.00
	Test	0.201	0.129	29.02	0.00
	Test Novel	0.2145	0.1175	25.66	0.00
fp/σ	Valid	22.01	21.938	28.66	0.00
	Test	22.356	22.284	29.02	0.00
	Test Novel	21.387	21.29	25.66	0.00

Table 4.8: New Coverage Score formula results with σ_{whole} uncertainty (3.10) and $\sigma_{accuracy}$ uncertainty (4.2). Error: (3.1), Accuracy: (3.5).

		Error	Error Change	Accuracy (%)	Accuracy Change
$fp^2 + (\sigma_{view} \times \sigma_{normalized})$	Valid	0.16	0.088	27.24	-1.42
	Test	0.161	0.089	27.04	-1.98
	Test Novel	0.175	0.078	25.69	+0.03
$fp^2 - (\sigma_{view} \times \sigma_{normalized})$	Valid	0.172	0.1	25.91	-2.75
	Test	0.17	0.098	26.09	-2.93
	Test Novel	0.186	0.089	22.54	-3.12
$fp + (\sigma_{view} \times (\sigma_{normalized} \times 0.8))$	Valid	0.073	0.001	28.45	-0.22
	Test	0.072	0	29.16	+0.14
	Test Novel	0.097	0	25.80	+0.14
$fp - (\sigma_{view} \times (\sigma_{normalized} \times 0.8))$	Valid	0.072	0	28.71	+0.05
	Test	0.072	0	29.21	+0.19
	Test Novel	0.097	0	25.64	-0.02
$fp + (\sigma_{view} \times \sigma_{normalized} \times \sigma_{accuracy}^{0.4})$	Valid	0.073	0.001	28.45	-0.21
	Test	0.072	0	29.21	+0.19
	Test Novel	0.096	-0.001	25.77	+0.11
$fp - (\sigma_{view} \times \sigma_{normalized} \times \sigma_{accuracy}^{0.4})$	Valid	0.072	0	28.74	+0.08
	Test	0.072	0	29.18	+0.16
	Test Novel	0.097	0	25.66	0.00

Throughout Tables 4.4 to 4.7 it is visible that simply using the error related uncertainties by addition or multiplication with the final prediction does not improve the model error nor the model accuracy. Therefore, it is safe to say that the formulas presented in these tables do not show to be of use for the improvement of the model predicted coverage score.

In Table 4.8, however, more interesting and complex formulas were analyzed. These showed to be more useful to achieve the goal of the prediction improvement, as they were able to achieve slight increments in the model's accuracy. The formulas that multiply the uncertainty of the view's coverage scores by a factor are the ones that achieve these positive results. Here we were trying to find a factor

that could maximize the coverage scores while minimizing the uncertainty of the predictions. The two best factors obtained in this study were: $\sigma_{normalized} \times 0.8$ and $\sigma_{normalized} \times \sigma_{accuracy}^{0.4}$, where $\sigma_{normalized}$ is given by $\sigma_{view}/max(\sigma_{view})$.

$max(\sigma_{view})$ is the maximum measured uncertainty σ_{view} throughout all the test samples, for each view. Therefore, to use this factor we need to have prior information of all the view's uncertainty. To achieve this a calibration set can be used to obtain an estimate of the $max(\sigma_{view})$ factor. Using a set like the validation set, we can compute all the prediction's σ_{view} uncertainties and then find the highest one for each view, obtaining a vector that can be used as a regularization factor on the other sets.

4.3.2 Decision Support

Another possible way of using the uncertainty measurements would be as decision support metrics. By now the model returns the coverage score predictions, predicting along two uncertainty values, one that gives insights on the predictions error (3.10) and another that reflects how certain the model is that it did a right prediction (4.2). By rejecting predictions with an uncertainty value higher than a set limit we would be discarding low quality predictions ensuring a better model performance.

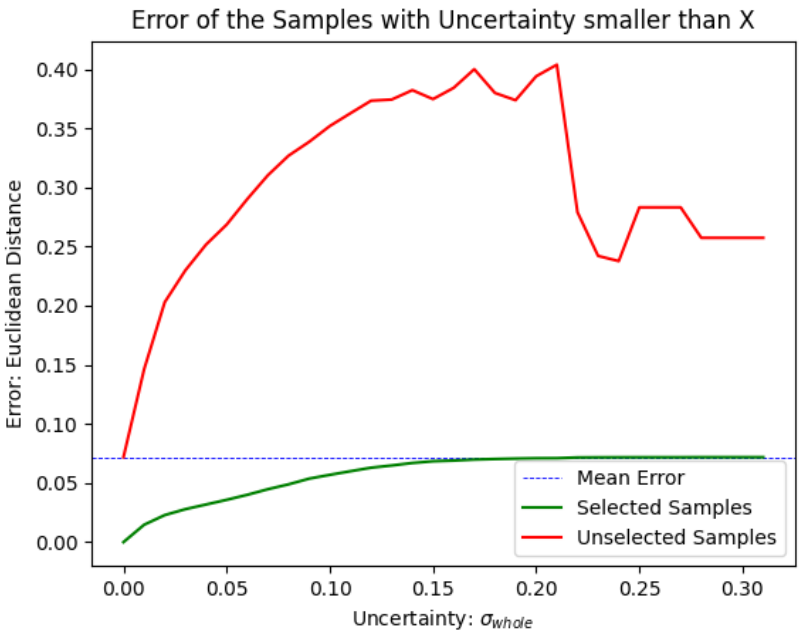


Figure 4.9: Model Error (Euclidean Distance, (3.1) in function of uncertainty σ_{whole} (3.10).

The Figure 4.9 shows in green the plot of the model Euclidean Distance error in the test set when considering only samples with σ_{whole} uncertainty smaller or equal than the value in the x axis. In red it plots the model error considering all the other samples that were not used, i.e. that had a higher

uncertainty value. The dotted blue line marks the model's error when considering all the samples.

This graph shows that in fact when we only account the samples with low uncertainty the model achieves lower error values. In this case if we consider only samples that have an uncertainty lower than 0.15 it is possible to reduce the model's error.

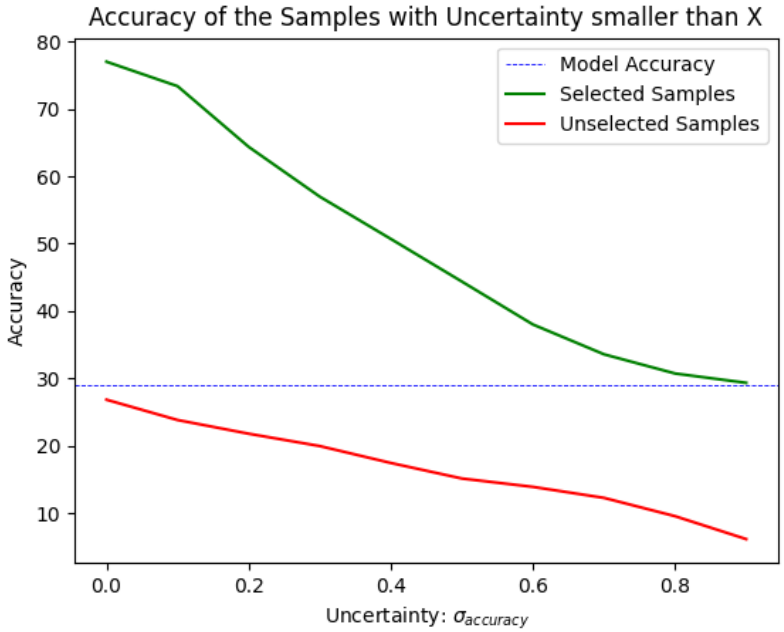


Figure 4.10: Model accuracy (3.5) in function of uncertainty $\sigma_{accuracy}$ (4.2).

Figure 4.10 shows the same behavior but for the model's accuracy. The model achieves high accuracy levels when samples with high uncertainty are discarded, and is visible with the red line plot that the samples that are discarded always achieve an accuracy lower than the model's accuracy when accounting all the samples. It becomes possible to achieve extremely good accuracy percentages comparing to the original model accuracy, detailed in Table 4.2, simply by discarding predictions with high uncertainty. In this case if we want to achieve a model accuracy of 60% we need to discard the samples with an $\sigma_{accuracy}$ uncertainty higher than 0.2.

To use this approach there is a need to set an uncertainty limit that will classify a prediction as acceptable or not. This limit can vary depending on the desired model performance and on the used data. This limit should then be calibrated on a calibration set, where one would examine which uncertainty values allowed the desired model performance.

For instance, assuming that the set used to generate the Figures 4.9 and 4.10 is our calibration set. If we would like to achieve a model accuracy not lower than 60% we would have to discard samples that had an $\sigma_{accuracy}$ uncertainty value higher than 0.2. The same can be calibrated for the error - taking the example of Figure 4.9 if we want to achieve a better model error we need to discard predictions with an

σ_{whole} uncertainty lower than 0.15.

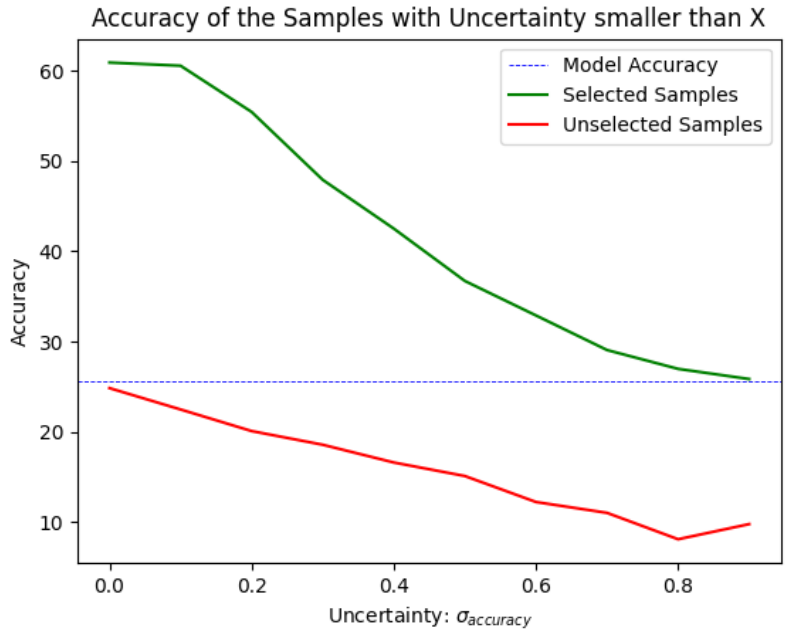


Figure 4.11: Model accuracy (3.5) in function of uncertainty ($\sigma_{accuracy}$, (4.2) on the test novel set.

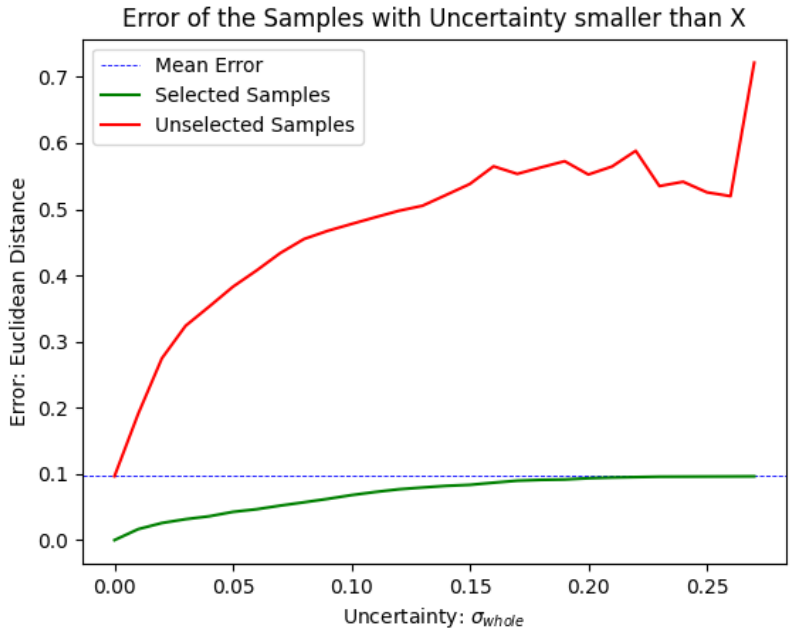


Figure 4.12: Model error (Euclidean Distances, (3.1) in function of uncertainty (σ_{whole}, (3.10) on the test novel set.

By testing these limits on the test novel set, that is composed of objects not known by the model, we

can see on Figures 4.11 and 4.12 that the error related uncertainty measurement with values under 0.15 truly achieved a model error smaller than the error when all the samples were accounted for. As for the accuracy related uncertainty measurement the achieved model accuracy with samples with uncertainty lower than 0.2 actually achieved values lower than 60%. But having into consideration that the possible maximum coverage for this set was 60%, the coverage achieved by the samples with uncertainty values up to 0.2 did not worsened the accuracy that much, having these samples achieved an accuracy of around 55%. In this way it is possible to calibrate the uncertainty measurements on a appropriated set to achieve a desired model performance.

To use these metrics as decision support measurements it would be, however, necessary to implement what to do when a prediction is discarded during the 3D acquisition process, since the model would not provide any information of what the NBV would be. Then tests to show the efficiency of the implemented method in comparison to the efficiency of simply choosing a wrong predicted NBV needed to be run. This is an essential analysis to evaluate the usage relevance of these as measurements decision support metrics.

An example of what could be done to avoid not having a NBV to go to when a prediction is discarded is to choose the viewpoint that least interferes with the prediction's uncertainty. To find it would be necessary to calculate the prediction uncertainty (either as σ_{whole} or as $\sigma_{accuracy}$) accounting for all the views excluding one. The excluded view of the uncertainty that achieves a value closer to the uncertainty of the full prediction could then be chosen as the NBV.

5

Conclusion and Future Work

Deep learning models are always prone to some level of uncertainty. Their ability to make predictions is as strong as the quality of the data they are trained on. Despite being desired for the models to be able to generalize, making them robust to input differences, their predictions always come with some degree of uncertainty. A model is more certain of its predictions on inputs it was trained on and less certain in inputs that it has never seen. NBV deep learning networks are no exception to this. However, most of the NBV models that use deep learning do not account the predictions uncertainty. Since the motivation goal of the dissertation was to make the object's 3D reconstruction process more efficient the picked strategy was to get a measurement of the uncertainty of the model prediction. If the uncertainty showed high levels that prediction could be discarded or manipulated, in order to guarantee that during the reconstruction process no wrong NBV are picked.

To do this the Monte Carlo Dropout method was successfully implemented in an already existing model that showed good performance - PC-NBV. By applying dropout layers after all the convolutional layers of the model and sampling several model predictions for the same input during inference, a functional framework to obtain the prediction's uncertainty was implemented. Several uncertainty metrics were studied with the goal of finding the ones that better reflected the prediction's error and accuracy. It was not possible to find a metric that combined the two, but two separate metrics that successfully reflected these quantities were found. Additionally it was noted that to obtain an uncertainty metric that measures a certain quantity it is necessary to measure that quantity in the MC samples and compute that metric variation between the MC samples and the mean of the samples. In that way, to obtain an uncertainty measurement that reflected the sample's error the error between the MC samples and the mean is calculated. To get an uncertainty measurement that reflected the sample's accuracy the accuracy of the MC samples in relation to the mean was compared.

Moreover, the usage potential of the uncertainty metrics to leverage the model's performance was studied. The first suggested approach was to use them to improve the model's final coverage score prediction, by implementing a new coverage score formula that incorporates the uncertainty measurements with the final prediction. We were able to obtain small model error and accuracy improvements with this approach, by multiplying the uncertainty by a factor and adding it to the final prediction. Another proposed method was to discard any sample that presented an high uncertainty value. This approach seems to be able to improve the model's error, but most important, the model's accuracy. By discarding samples with high uncertainties it was possible to improve the model's accuracy from 30% to 60%-80%. However, this method faces some obstacles that need further studies, as it would need another NBV decision approach to use when the model discarded a prediction. We propose a decision approach to use in this circumstance: choosing as the NBV, when a prediction is discarded , the view that least interferes with the prediction uncertainty. This strategy would, however, require testing to study it's affects on the efficiency of the 3D reconstruction. This was not possible to study during this dissertation, so it is

an important limitation that can be studied in a later work.

Besides these future studies that explore the limitations of this dissertation, future works could also analyze if different dropout layer placements could achieve better results. The works [30] and [28] reviewed in the Chapter 2 showed that in fact applying dropout after each convolutional layer could work as a too strong of a regularizer, and stated that only applying dropout layers in certain parts of their networks showed better results. That analysis was not done in this work, but could also be an interesting study. As well as the optimization of the dropout probability, that in this work was left at 0.5. However, it is possible that another probability could achieve better results. Also, another study not covered in this dissertation was the study of the ideal number of Monte Carlo dropout samples, which is a parameter that could also be optimized. Lastly, another important study to run in the future is the use of this model on a real life scenario.

Bibliography

[1] R. Zeng, W. Zhao, and Y.-J. Liu, "PC-NBV: A point cloud based deep network for efficient next best view planning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 7050–7057.

[2] "RePAIR project," https://www.repairproject.eu/.

[3] M. Mendoza, J. I. Vasquez-Gomez, H. Taud, L. E. Sucar, and C. Reta, "Supervised learning of the next-best-view for 3D object reconstruction," *El Sevier*, vol. Patter Recognition Letters, no. 133, pp. 224–231, 2020.

[4] J. I. Vasquez-Gomez, D. Troncoso, I. Becerra, E. Sucar, and R. Murrieta-Cid, "Next-best-view regression using a 3D convolutional neural network," *Machine Vision and Applications*, vol. 32, no. 42, 2021.

[5] Y. Han, I. H. Zhan, W. Zhao, and Y.-J. Liu, "A double branch next-best-view network and novel robot system for active object reconstruction," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 7306–7312.

[6] S. Kriegel, T. Bodenmüller, M. Suppa, and G. Hirzinger, "A surface-based next-best-view approach for automated 3D model completion of unknown objects," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 4869–4874.

[7] S. Chen and Y. Li, "Vision sensor planning for 3D model acquisition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 35, no. 5, pp. 894–904, 2005.

[8] C. Connolly, "The determination of next best views," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 432–435.

[9] N. A. Massios and R. B. Fisher, "A best next view selection algorithm incorporating a quality criterion," in *Proceedings of the British Machine Vision Conference*. BMVA Press, 1998, pp. 78.1–78.10, doi:10.5244/C.12.78.

[10] C. Potthast and G. S. Sukhatme, “A best next view selection algorithm incorporating a quality criterion.” in *Journal of Visual Communication and Image Representation*, vol. 25, no. 1, 2014, pp. 148–164.

[11] J. I. Vasquez-Gomez, L. E. Sucar, R. Murrieta-Cid, and E. Lopez-Damian, “Next-best-view planning for 3D object reconstruction with positioning error,” in *International Journal of Advanced Robotic Systems*, vol. 11, 2014.

[12] J. I. Vasquez-Gomez, L. E. Sucar, and R. Murrieta-Cid, “View/state planning for three-dimensional object reconstruction under uncertainty.” in *Auton Robot*, vol. 41, 2017, pp. 89–109.

[13] S. Kriegel, C. Rink, T. Bodenmüller, and M. Suppa, “Efficient next-best-scan planning for autonomous 3D surface reconstruction of unknown objects,” in *Journal of Real-Time Image Processing*, vol. 10, 2015, p. 611–631.

[14] F. Bissmarck, M. Svensson, and G. Tolt, “Efficient algorithms for next best view evaluation,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5876–5883.

[15] H. Fan, H. Su, and L. Guibas, “A point set generation network for 3D object reconstruction from a single image,” 2016.

[16] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, “PCN: Point completion network,” 2018.

[17] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, “PU-GAN: a point cloud upsampling adversarial network,” 2019.

[18] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, “Self-attention generative adversarial networks,” 2018.

[19] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” 2016.

[20] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” 2017.

[21] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Z. L. Q. Huang, S. Savarese, M. Savva, S. Song, and H. S. et al., “Shapenet: An information-rich 3D model repository”,” 2015.

[22] W. Yifan, S. Wu, H. Huang, D. Cohen-Or, and O. Sorkine-Hornung, “Patch-based progressive 3D point set upsampling,” in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5951–5960.

[23] E. Hüllermeier and W. Waegeman, "Aleatoric and epistemic uncertainty in machine learning: A tutorial introduction," *CoRR*, vol. abs/1910.09457, 2019. [Online]. Available: http://arxiv.org/abs/1910.09457

[24] R. M. Neal, "Bayesian learning for neural networks," 1996.

[25] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[27] A. Graves, "Practical variational inference for neural networks," in *Neural Information Processing Systems*, 2011. [Online]. Available: https://api.semanticscholar.org/CorpusID:14885866

[28] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian SegNet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *CoRR*, vol. abs/1511.02680, 2015. [Online]. Available: http://arxiv.org/abs/1511.02680

[29] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *CoRR*, vol. abs/1511.00561, 2015. [Online]. Available: http://arxiv.org/abs/1511.00561

[30] A. Kendall and R. Cipolla, "Modelling uncertainty in deep learning for camera relocalization," *CoRR*, vol. abs/1509.05909, 2015. [Online]. Available: http://arxiv.org/abs/1509.05909

[31] A. Kendall, M. Grimes, and R. Cipolla, "Convolutional networks for real-time 6-DOF camera relocalization," *CoRR*, vol. abs/1505.07427, 2015. [Online]. Available: http://arxiv.org/abs/1505.07427