university of groningen / faculty of science and engineering / artificial intelligence

Validation of Uncertainty in Classification under Point Cloud Downsampling

Bachelor's Project Thesis

Jay Khalil; s3167240; a.l.khalil@student.rug.nl,
Supervised by: Dr M.A. Valdenegro Toro

Abstract: The rapid evolution of artificial intelligence (AI) highlights the need to understand uncertainties in machine learning (ML) predictions. This research explored uncertainties in point cloud data, a high-dimensional structure pivotal in many fields, utilising Bayesian neural networks. Hence, we integrated Monte Carlo Dropout, Monte Carlo DropConnect, Flipout, and Deep Ensemble within a custom PointNet model. Through iterative downsampling, the models unveiled varying abilities to manage uncertainties. Our findings suggest a prevalent underconfidence due to point cloud data complexity. Importantly, model sensitivities varied across classes, indicating the intricacies of class representation and training complexities. Notably, the extent of downsampling influenced uncertainty in non-linear ways, challenging the assumption that more sampling points always yield better results. These insights underscore the study's contribution to ML's uncertainty management, hinting at avenues for model optimisation. Future research should delve deeper into this promising domain.

1 Introduction

Artificial Intelligence (AI) is the mantra of the current era." This intriguing statement by the author Jordan summed up the advancement of AI perfectly. AI has permeated every aspect of modern life, transforming the landscape of various industries and reshaping our everyday experiences. A testament to this monumental shift is the rise of OpenAI's large language models, such as ChatGPT, that can comprehend and generate human-like text, opening up new frontiers in communication, content creation, and knowledge extraction. Another example can be Tesla's Autopilot, which can observe the dynamic environment and react under extreme time-stress situations, striving for a reaction with high accuracy.

Despite these advancements, the proliferation of AI hinges on the ability to handle and interpret high-dimensional, complex data structures—an area where machine learning (ML), a subset of AI, shines, Krizhevsky et al., 2017. Machine learning algorithms can learn from data, recognise patterns, and make decisions, contributing significantly to the evolution of AI. As such, ML has been at the forefront of AI's growth, facilitating significant improvements in areas such as image and speech recognition, natural language processing, and automation, Rayhan, 2023.

1.1 Point Clouds

One of the most shining achievements of machine learning lies in image classification. This revolutionary technology produced significant results since it has been under development for decades Sultana et al., 2020. However, in terms of real-world accuracy, image recognition can be less trustworthy based on the physical world. This drawback is raised because images consist of 2D data. Hence, point clouds capture the precise 3D geometry and spatial relationships of objects or scenes. This information is crucial for tasks that require understanding objects' shape, structure, and positioning in three-dimensional space. Therefore, scientists have developed advanced technologies to harness the power of such a classification. The crucial form of point cloud data is fundamental to various fields, including computer vision, geospatial analysis, and robotics. Point cloud data is characterised

by its high dimensionality and complexity, presenting unique challenges and opportunities for AI algorithms, such as PointNet, DGCNN, and PointCNN Kim & Kim, 2020.

An instance of these uncertainties in point cloud data handling arises from the inherent noise produced by the data collection technologies, such as Laser or LIDAR systems. These systems capture data by emitting pulsed laser beams to measure distances to the Earth's surface. Inherent to this process is the introduction of noise, primarily due to the less-than-perfect precision of the detection mechanisms, atmospheric interference, and variable surface reflectivity. The noise can result in spurious data points within the point cloud, creating a 'false feature' that could lead the Neural Network to classify incorrectly.

This inherent noise represents a form of data uncertainty, a concept that expands to all fields where data is collected and interpreted, not only point cloud data. Similarly, there is a parallel to be drawn with the concept of model uncertainty in machine learning, where even with perfect data, the model's predictions may still be uncertain due to the inherent limitations and assumptions of the chosen model. Both forms of uncertainty - data and model - pose significant challenges for machine learning applications and require careful consideration when designing and implementing these systems.

1.2 Uncertainty

In machine learning, uncertainties can broadly be categorised into two types: aleatoric uncertainty, which stems from inherent data noise, and epistemic uncertainty, which arises from incomplete knowledge about the model. Identifying, quantifying, and managing these uncertainties is critical to the reliable deployment of every task utilising a neural network Valdenegro-Toro & Mori, 2022.

Nevertheless, the handling and interpretation of point cloud data are rife with uncertainties. An example of such uncertainty can be seen in a classification task when the Neural Network must classify an object by choosing between various objects. Such a task can be straightforward for humans. However, it is not easy for the algorithms, especially if we consider two similar objects with certain features yet differs in categories, or training with class imbalanced datasets Yun & Lee, 2023.

1.3 State of the art

There has been a significant amount of publications enriching our understanding of point cloud classifications since Qi, Su, et al. pioneered the field with the PointNet architecture. Subsequent research expanded upon this foundational work and introduced advanced versions with improved accuracy, such as PointNet++ (Qi, Yi, et al., 2017). However, there remains a noticeable lack of discussion concerning crucial factors like the number of selected point clouds per entity and their impact on uncertainty prediction.

While one might naturally presume that a greater quantity of point clouds per entity would yield enhanced accuracy, the study by Lumban-Gaol et al. suggests otherwise. They investigated the semantic segmentation of an indoor scene of a railway station captured in point cloud using three distinct models, contrasting their outcomes across various factors (Lumban-Gaol et al., 2021).

Experimental results demonstrated a trade-off between computational efficiency and preserving geometric details. Their results indicated a negative correlation between the number of points per class and segmentation task performance. Additionally, a detailed study on the sensitivity of PointNet's hyperparameters provided valuable insights into their relationship with the number of points (Nurunnabi et al., 2021).

Moreover, Valdenegro-Toro & Mori work on understanding uncertainty in neural networks proposed a generalised approach to identify and differentiate between aleatoric and epistemic uncertainties. Their contributions significantly advanced our comprehension of the intricate relationship between different uncertainty types and their application in neural networks. The state of the art work demonstrated an outstanding results using the latest advancement in model architecture to capture uncertainty.

1.4 Current work

In light of previous pioneering work, this work seeks to evaluate the effectiveness of Bayesian neural networks in determining both aleatoric and epistemic uncertainties in point cloud classification tasks with varying numbers of sample points per object. We utilise the PointNet architecture (Qi, Su, et al.,

2017) for 3D object recognition. While more recent algorithms have emerged after PointNet, they often draw upon its foundational concepts. The research will employ the benchmark ModelNet10 dataset (Wu et al., 2015). The primary research question we seek to address is: 'How significant is the influence of downsampling on introducing uncertainty in cloud classification, and what approaches are suitable for its validation?' Addressing this question will shed light on uncertainty quantification in machine learning algorithms tailored for point cloud data, propelling the development of dependable and accurate AI systems.

Our approach employs several techniques to evaluate the performance and robustness of different uncertainty quantification methods across diverse data volume scenarios: Monte Carlo Dropout, Monte Carlo DropConnect, Flipout, and Deep Ensemble. We will examine each method under various downsampling strategies, progressively reducing point cloud numbers from 4096 to a minimum of 16 points per object. This systematic exploration will thoroughly assess these uncertainty techniques in different point cloud data volume contexts.

2 Method

In this coming section we will discuss the theoretic implementation of the approached method.

2.1 Dataset

The ModelNet10 dataset is a subset of the larger ModelNet40 dataset, both introduced by (modelnet10) in their seminal paper on 3D ShapeNets. The dataset comprises ten categories, each containing CAD models of objects typically found in indoor environments. These categories include beds, desks, chairs, bathtubs, tables, sofas, monitors, dressers, nightstands, and toilets. ModelNet10 contains 3,991 3D CAD models for training and 908 for testing. The objects in the dataset are represented as point clouds, and each point cloud has been normalised into a unit sphere, which is beneficial for tasks related to 3D object recognition. However, the dataset comes unfortunately imbalanced, but this can be beneficial as it brings better simulation to the real world, Appendix A Table A.1 & Table A.2. Despite that, the widespread use and acceptance of ModelNet10 in the research community can be attributed to its well-curated nature, which makes it an ideal benchmark dataset for evaluating 3D object classification algorithms. In Figure 2.1, we can see examples of objects from ModelNet10 dataset.

Figure 2.1: ModelNet10 Dataset, An example from all the categories, left to right, up to down: Bathtub, Bed, Chair, Desk, Dresser, TV, Nightstand, Couch, Sofa, Table, and Toilet

However, the data in our model originates from a subset of points within an Euclidean space, with these points forming a point cloud. This point cloud possesses three key attributes. First, it's unordered, differing from pixel or voxel arrays, necessitating a model immune to variations in input point orders. Second, these points interact within their defined metric space, creating meaningful local structures whose interactions the model must discern. Lastly, the model must ensure invariant under transformations, such as rotation or translation, to maintain accurate object representation and classification, Qi, Su, et al., 2017.

2.1.1 Sampling

Point sampling is performed on the point clouds in our dataset, using a random process for each cloud. The goal is to ensure that the distribution of points in each cloud remains representative of the original, while reducing the computational complexity by limiting the total number of points, Qi, Su, et al., 2017.

Given a triangle formed by three points in the cloud, $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$, the area of the triangle is calculated using Heron's formula:

$$A = \sqrt{s(s-a)(s-b)(s-c)} \tag{2.1}$$

where s is the semi-perimeter of the triangle given by $s = \frac{a+b+c}{2}$, and a, b, and c are the lengths of the sides of the triangle.

A point within the triangle is then randomly sampled based on barycentric coordinates (s, t), ensuring that the probability of selection is proportional to the triangle's area. The barycentric coordinates are used to compute the position of the sampled point $\mathbf{p}$ as follows:

$$\mathbf{p} = s\mathbf{p}_1 + (t - s)\mathbf{p}_2 + (1 - t)\mathbf{p}_3 \tag{2.2}$$

This process is repeated until the desired number of points is sampled, Figure 2.2.

In addition, to achieve a representative sample of the mesh, triangles (or faces) are sampled based on their areas. This ensures that larger triangles, which occupy more of the mesh's surface, are more likely to be sampled, thus retaining the overall structure and features of the original mesh in the downsampled point cloud.

2.1.2 Data Normalisation

Normalising the point cloud data is a crucial step in our methodology. This process involves translating and scaling the objects within the point cloud to a common coordinate system and size, respectively. This normalisation aligns with the "translation" process used in the work by Qi, Su, et al. (2017). Thus, the translation is achieved by subtracting the mean value of the point cloud coordinates from each point, effectively placing the objects at the origin of the coordinate system. After that, we scale these translated points to fit within a unit sphere by dividing each point by the maximum norm of the point clouds, as shown in Figure 2.2. The mathematical representation of these operations is as follows:

Given a point cloud $P = p_1, p_2, \ldots, p_n$, where p_i are the coordinates of each point, we compute the normalised point cloud P'' by:

1. Centring the Point Cloud:

$$\begin{aligned} \text{mean}_j &= \frac{1}{n}\sum_{i=1}^{n} \text{pc}_{i,j} \\ \text{npc}_{i,j} &= \text{pc}_{i,j} - \text{mean}_j \\ &\text{for } i = 1, 2, \ldots, n \\ &\text{and } j = 1, 2, \ldots, d \end{aligned} \tag{2.3}$$

2. Normalising the Point Cloud:

$$\begin{aligned} \text{norm}(\text{npc}_i) &= \sqrt{\sum_{j=1}^{d} \text{npc}_{i,j}^2} \\ &\text{for } i = 1, 2, \ldots, n \\ \text{max\_norm} &= \max_i \left(\text{norm}(\text{npc}_i)\right) \\ \text{normalised\_pc}_{i,j} &= \frac{\text{npc}_{i,j}}{\text{max\_norm}} \\ &\text{for } i = 1, 2, \ldots, n \\ &\text{and } j = 1, 2, \ldots, d \end{aligned} \tag{2.4}$$

2.1.3 Data Pre-processing

In our methodology, data preprocessing is a key step to ensure the robustness of our model Qi, Su, et al., 2017. The process increases the diversity of our training dataset, reducing the risk of overfitting while accommodating potential changes in the input data. Our approach primarily relies on two essential techniques:

1. Random Rotation: We randomly rotate each object in the point cloud around the Z-axis during the training process. The rotation angle θ is chosen randomly within the range $[0, 2\pi]$. The rotation operation is conducted with a rotation matrix R, resulting in the rotated point cloud P', as shown in 2.5.

$$\begin{aligned} \theta &= \text{random}(0, 2\pi) \\ R &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ P' &= RP^T \end{aligned} \tag{2.5}$$

2. Gaussian Noise: To better equip the model to cope with potential sensor noise in real-world applications, we add Gaussian noise to the point cloud. The noise N is drawn from a Gaussian distribution with mean 0 and standard deviation 0.02, and is added to each point in the point cloud, resulting in the noisy point cloud P'', as shown in 2.6.

$$\begin{aligned} N &\sim \mathcal{N}(0, 0.02) \\ P'' &= P' + N \end{aligned} \tag{2.6}$$

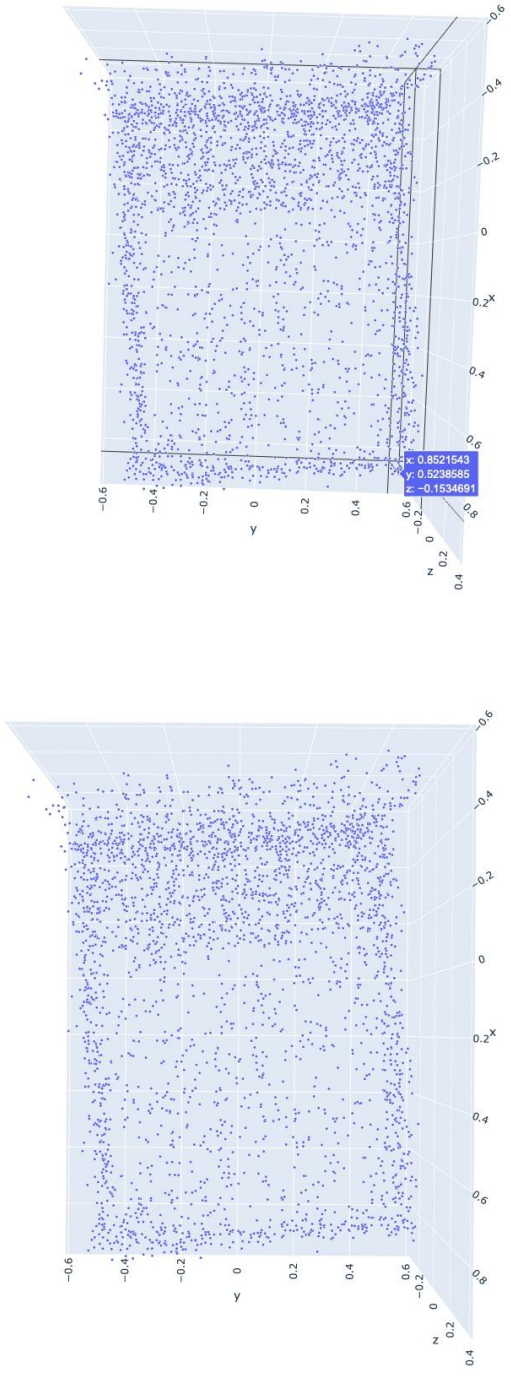

Figure 2.2: Point Clouds Processing; The top sample illustrates a bed object before augmentation, while the bottom sample represents the same bed post-processing, where data has been modified through rotation around the Z-axis and the addition of noise was added.

Through these augmentation techniques, we manage to increase the variability within our training data, resulting in a more generalisable and robust model, Figure 2.2.

2.2 Models

In this study, we utilise the PointNet model, closely aligning with the original version as described by Qi, Su, et al.. Furthermore, as we integrated various uncertainty models into our study, additional modifications were required. Specifically, alterations to the final layers of each PointNet model to become an uncertainty model were made to ensure alignment with their respective theoretical underpinnings. The following subsections will delve further into these models and highlight the specific differences and modifications.

2.2.1 PointNet Architecture

PointNet architecture is a type of neural network that directly consumes point clouds, which are unstructured sets of points in space. This model is beneficial for 3D geometric data processing. The model's architecture is implemented in Python using the Keras and TensorFlow libraries. PointNet's pipeline consists of several key components, including transformation networks (T-Nets), a classification network, and a custom loss computation. Figure 2.3 will show the simplicity of the pipeline.

The particular component of the model is the T-Nets, which include an input and feature transform, which are employed to align input data and learn the features, respectively. These transformation networks capture global spatial information about point clouds. The transformation matrices generated by the T-Nets are also regularised to be close to orthogonal matrices, which helps to ensure the transformation learned is a rotation, Figure 2.7.

$$\|X \cdot X^T - I\| = 0 \tag{2.7}$$

The classification network comprises several layers, including 1D convolutional layers, dense layers, and a final softmax layer. It takes the transformed data from the T-Nets, applies a series of transformations, and finally classifies the point cloud into one of the predefined classes.

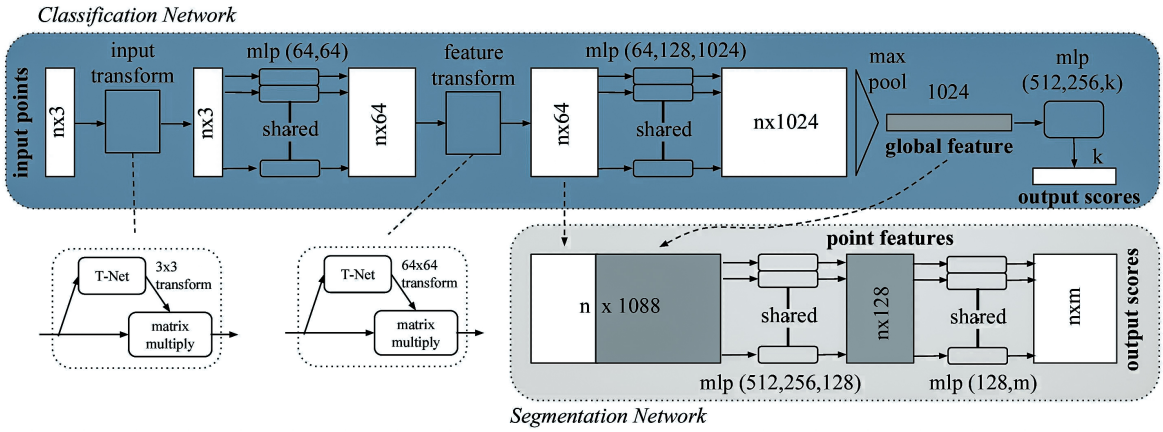

Figure 2.3: PointNet architecture. The model accepts n points, performs input and feature transformations, and aggregates point features via max pooling. This process yields classification scores for k classes. The segmentation network extends this, combining global and local features to output per point scores. 'mlp' refers to multi-layer perceptron, with bracketed numbers indicating layer sizes. All ReLU layers utilise batch normalisation, and dropout layers are applied in the final mlp of the classification network, Qi, Su, et al., 2017.

Loss function

The loss function in this model combines classification loss and regularisation. Classification loss, calculated using Sparse Categorical Crossentropy, measures how accurately the model classifies point clouds. The regularisation term ensures transformation matrices (m3x3 and m64x64) approximate orthogonal matrices, effectively learning rotation transformations.

The loss function takes the following form:

$$L_{total} = L_{classification} + \alpha(\|I - AA^T\|_F^2 + \|I - BB^T\|_F^2) \tag{2.8}$$

In this equation, L_{total} is the total loss, $L_{classification}$ is the classification loss, A and B are the transformation matrices, and I is the identity matrix. AA^T and BB^T are the result of matrix multiplication of each transformation matrix and its transpose. The symbol α represents a hyperparameter, and $|.|_F^2$ is the Frobenius norm. This structure encourages both proper classification and the learning of mostly rotation transformations.

Metrics

The performance of the model is tracked using two metrics: mean loss and accuracy. Mean loss provides an overview of the model's error magnitude, while accuracy measures the percentage of correct predictions, assessing the model's prediction capabilities. Both metrics are monitored during training and validation for comprehensive performance understanding.

2.2.2 Bayesian Neural Networks

A Bayesian Neural Network (BNN) is a variant of traditional neural networks where the weights are assigned a probability distribution rather than a single value. Such a methodology can motivate the model to account for uncertainty in its predictions, improving reliability. Formally, Bayesian inference is used to update our prior beliefs $p(\boldsymbol{\lambda})$ about the weights $\boldsymbol{\lambda} = \{w_1, w_2, ..., w_K\}$ given the observed input/output pairs $(\mathbf{x}, \mathbf{y})$, resulting in a posterior distribution $p(\boldsymbol{\lambda}|\mathbf{x}, \mathbf{y})$, Abdar et al., 2021. This is done using Bayes' theorem:

$$p(\boldsymbol{\lambda}|\mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\lambda})p(\boldsymbol{\lambda})}{p(\mathbf{y}|\mathbf{x})} \tag{2.9}$$

For a new test sample x_*, the model averages its predictions over all possible weights, weighted by the updated belief about those weights, to make a prediction:

$$p(y_*|x_*, \mathbf{x}, \mathbf{y}) = \int p(y_*|x_*, \boldsymbol{\lambda})p(\boldsymbol{\lambda}|\mathbf{x}, \mathbf{y})d\boldsymbol{\lambda} \tag{2.10}$$

However, due to the high-dimensional nature of the weight space in neural networks, computing this integral is computationally expensive, and hence, various techniques such as Monte Carlo Dropout and Monte Carlo DropConnect are employed to approximate it, Gawlikowski et al., 2023.

2.2.3 Monte Carlo Dropout

Monte Carlo Dropout (MC Dropout) serves as a potent technique for approximating integrals that are generally computationally challenging, as detailed by Gal & Ghahramani, 2016, and illustrated in equation 2.11. The technique pivots on the principles of stochastic sampling. In this context, each execution of a forward pass through the neural network yields a unique sample from the underlying posterior distribution. This approach is of particular value as it sidesteps the extensive computational resources usually associated with traditional Monte Carlo simulations.

During the training process, MC Dropout employs random binary markers to denote specific nodes within the network. The interpretation is that if a node's marker value equates to 0, the said node is omitted or "dropped" from the network. This action has the advantage of curtailing overfitting by diminishing the prevalent co-adaptation effect among different layers.

The mathematical representation for estimation using MC Dropout can be described as follows:

$$p(y_*|x,x,y) = \frac{1}{M}\sum_{i=1}^{M} p(y_|x_*,\lambda)p(\lambda|x,y) \quad (2.11)$$

Here, M stands for the number of forward passes through the network. Furthermore, $p(y_*|x_*,\lambda)$ is a representation of the distinct samples generated during each of these forward passes.

2.2.4 Monte Carlo DropConnect

Monte Carlo DropConnect (MC DropConnect) operates as a variant of MC Dropout. However, instead of eliminating nodes from the neural network, it strategically omits weights, a method discussed in Oberdiek et al., 2018. This methodology offers an alternate route for injecting randomness into the network's structure and behavior.

Mirroring the process in MC Dropout, during the training phase, DropConnect assigns random binary indicators to the weights within the network. Weights assigned a value of 0 are subsequently dropped. This mechanism serves as an effective means of mitigating overfitting, a situation particularly prominent in larger and more complex models.

The estimation formula for MC DropConnect aligns closely with that of MC Dropout:

$$p(y_*|x,x,y) = \frac{1}{M}\sum_{i=1}^{M} p(y_|x_*,\lambda)p(\lambda|x,y) \quad (2.12)$$

In this equation, M symbolizes the number of forward passes through the network, while $p(y_*|x_*,\lambda)$ signifies the distinct samples procured during each forward pass. For both MC Dropout and MC DropConnect, we calculate the predictive mean, $\mu(x)$, and standard deviation, $\sigma(x)$, as follows:

$$\mu(x) = \frac{1}{M}\sum_{i=1}^{M} f_i(x) \quad (2.13)$$

$$\sigma(x) = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(f_i(x) - \mu(x))^2} \quad (2.14)$$

Here, $f_i(x)$ corresponds to the output of the network for the i^{th} forward pass. These predictive mean and standard deviation values are then used to construct the model's confidence interval.

2.2.5 Flipout

The Flipout strategy brings a new level of refinement to the Bayes by Backprop (BBB) approach by injecting stochasticity into the weight distributions of a neural network. While BBB treats weight sampling as an additive disturbance on the mean (expressed as $w = \mu + \sigma z$, where $z \sim N(0,1)$ and $w \sim N(\mu,\sigma)$), as documented in Wen et al., 2018.

$$\Delta W_n = \Delta\hat{W} r_n s_n^T \quad (2.15)$$

The equation 2.15, ΔW_n represents the per-sample perturbation. Meanwhile, $\Delta\hat{W} = \sigma z$ stands

as the per-sample perturbation metric, and r_n, s_n are independent samples drawn from a Rademacher distribution.

2.2.6 Deep Ensemble

The Deep Ensemble strategy synergies the predictive capacities of multiple models, also known as ensemble members, to boost overall performance and refine uncertainty quantification, as discussed in Lakshminarayanan et al., 2017. The output of each ensemble member for a specific input x is represented by $f_i(x)$, where i symbolises the index of the ensemble member. The collective output from the DE, symbolised as $f_E(x)$, is the arithmetic mean of all individual member outputs. For an ensemble with M members, this is captured in equation 2.16:

$$f_E(x) = \frac{1}{M} \sum_{i=1}^{M} f_i(x) \tag{2.16}$$

In tasks that involve classification, the ensemble's collective prediction is computed by taking the softmax of the averaged logits from all ensemble members. This is illustrated in equation 2.17:

$$f_E(x) = \frac{1}{M} \sum_{i=1}^{M} \text{softmax}(f_i(x)) \tag{2.17}$$

2.3 Downsampling

Downsampling functions as an effective method for decreasing the quantity of observations in a dataset, with the primary objective of maintaining the fundamental characteristics of the original data Chen et al., 2022. Given a collection of data points $X = x_1, x_2, ..., x_n$, where n represents the number of data points, downsampling facilitates the generation of a smaller set $X' = x'i_1, x'i_2, ..., x'_m$, in which $m < n$.

A downsampling function, designated as $D : X \to X'$, is defined for this purpose, where the set X' contains fewer points than the initial set X. In the particular context of this investigation, we introduce a downsampling factor f, defined such that $f = 2^k$, with k representing a non-negative integer. Consequently, the number of points m in the reduced set X' is given by $m = n/2^k$. This approach results in a sequence of successively down-sampled datasets, wherein the number of points is halved at each progressive stage.

2.4 Experiment Design

The experiment is built around the concept of downsampling the point clouds data points 2.3. By varying the volume of data in this manner, we aim to examine the behaviour of selected uncertainty methods under different data circumstances.

In the following sections, we detail the construction and training of the respective models under each method. The choice of hyperparameters, model architecture, and training regimen are explained, providing a comprehensive view of the experimental setup.

2.4.1 PointNet Baseline Model

Our experimental framework commences with a baseline model built on the architecture of PointNet, a deep learning model specifically tailored for processing point cloud data, as previously outlined in Section 2.2.1. This architecture serves as the foundation for all subsequent model variants in this study. The model was custom designed according to the specifications outlined in Qi, Su, et al., 2017.

In an effort to augment this architecture with a capability for uncertainty quantification, we introduce a Stochastic Classifier to perform multiple stochastic forward passes for each input, with each pass resulting in a different output due to the randomness introduced by the dropout. The model doesn't directly deliver these multiple outputs as the final result; instead, they are processed further. The model calculates the mean over all stochastic forward passes to create a single prediction that encapsulates the predictive uncertainty in the form of a distribution over possible outcomes. The Stochastic Classifier class was imported from keras_uncertainty library Valdenegro-Toro, 2021

However, the Stochastic Classifier will not have that much of an impact regarding measuring the uncertainty on the Baseline model since it is design with an ordinary dropout that is activated only during training. Yet, it creates unification when compared to another models.

2.4.2 Monte Carlo Dropout model

The Monte Carlo (MC) Dropout model is a modified version of the baseline PointNet architecture by incorporating the Stochastic Dropout layer, which applies dropout during both training and inference phases, Section 2.2.3. This MC Dropout layer is introduced after the second dense layer with a dropout rate of 0.3, followed by standard batch normalisation and ReLU activation functions.

The model concludes with a dense output layer, mapping to the classes of the classification problem, with a softmax activation function. The model's distinctive feature is the application of the Stochastic Classifier during the inference phase. With this modification, the model effectively acts as a Monte Carlo simulator by performing multiple forward passes for each input, each pass generating a different prediction due to the MC Dropout layer.

The Stochastic Dropout layer was imported from keras_uncertainty library Valdenegro-Toro, 2021.

2.4.3 Monte Carlo dropConnect model

The Monte Carlo (MC) dropConnect model, as described 2.2.4, was similar to the MC Dropout except it instead introduces the concept of dropConnect into the architecture of the PointNet model. Hence, a DropConnect Dense layer replaces the second dense layer in the Base model, randomly dropping connections instead of neurons with a rate of 0.3.

Similar to the MC Dropout model, the MC dropConnect model uses the Stochastic Classifier to perform multiple forward passes during inference, effectively turning the model into a Monte Carlo simulator. These multiple passes take advantage of the DropConnect Dense layer's behaviour, which randomly drops connections, allowing the model to generate a distribution of predictions for each input and thus offering a probabilistic understanding of the inherent uncertainty in the outcomes.

The Stochastic DropConnect layer was imported from keras_uncertainty library Valdenegro-Toro, 2021

2.4.4 Flipout Model model

The Flipout model adapts the PointNet Base model, integrating a Flipout Dense layer that introduces Bayesian variational inference into the network, as explained in section 2.2.5. This transformation creates a Bayesian version of PointNet, capable of capturing weight uncertainty.

Further, the Flipout Dense layer, which substitutes the second dense layer in the baseline architecture, is characterised by several defining parameters. The KL weight, a critical factor representing the distance between two probability distributions, is set as the inverse of the total number of batches in the dataset. In this case, the KL weight was computed as 1.0/number of batches, where the number of batches is the dataset length divided by the batch size of 32.

For the prior, the Flipout Dense layer assumes a mixture of two Gaussian distributions. The standard deviations were set as 5.0 and 2.0, while the prior weight was set as 0.5, although these values can be adjusted to fit the problem context.

Similar to the MC Dropout and MC dropConnect models, the Flipout model employs the Stochastic Classifier during inference. This approach involves multiple forward passes for each input, generating a distribution of predictions, and offering a probabilistic understanding of the outcomes.

The Flipout Dense layer was incorporated using the keras_uncertainty library Valdenegro-Toro, 2021.

2.4.5 Deep Ensemble model

The Deep Ensemble model is another adaptation of the baseline PointNet model, with the architecture remaining identical but without incorporating a Dropout layer. It employs the concept of ensemble learning, wherein multiple instances of the same model are created and trained independently on the same dataset, section 2.2.6.

The layers in the Deep Ensemble model are identical to the baseline PointNet model, following the same sequence of dense layers, batch normalisation, and ReLU activation functions. However, the differentiating feature is the utilisation of the Deep Ensemble Classifier during the inference phase. This classifier creates multiple instances of the model, each trained independently on the same dataset, leading to an ensemble of predictions. The ensemble's mean prediction is then normalised to adjust the probabilities for each class.

The model generates a distribution of predictions

for each input, allowing for a probabilistic interpretation of the predictions.

The Deep Ensemble Classifier class was imported from keras_uncertainty library Valdenegro-Toro, 2021

2.4.6 Downsampling Method

In our experiment, we introduced a straightforward downsampling technique, described in 2.1.1. The sampling deprivation is performed with a set of downsampling factors, $K = \{0, 1, 2, ..., 9\}$, which results in a number of points $F = \{4096, 2048, 1024, 512, 256, 128, 64, 32, 16\}$. For each $f \in F$, all models (Base, MC Dropout, MC DropConnect, Flipout, and Deep Ensemble) are trained and evaluated to study their performance and robustness under varying data volumes.

2.4.7 Test-time Data Augmentation Method

In this study, we also explore aleatoric uncertainty using a test-time data augmentation method, as proposed by Wang et al., 2019. This method estimates uncertainty by introducing small perturbations to the data during the testing phase, generating various versions of the same sample. In our case, these perturbations include adding random noise and rotating points around the z-axis, as described in section 2.1.3.

However, unlike the original methodology outlined by Wang et al., we did not create multiple versions of each data sample in our implementation. Since our dataset's inherent characteristics, such as fundamental similarities between objects within the same category, made it well-suited with various objects shares similar features. Furthermore, we averaged the entropy of all model predictions for each individual sample to measure uncertainty per sample.

2.4.8 Uniform Training Setup for Standardised Results

We standardised our experiment using a universal random seed of 42, applied across all random generator libraries. Our models were trained using GPU-based methods for optimised performance, with each model completing a full run through the five models in about 8 hours using NVIDIA RTX 3090 GPU.

The models, including various PointNet iterations and their uncertainty modifications, employed identical hyperparameters. They underwent 16 epochs, with batch normalisation at 0.75 momentum. An L2 bias regulariser (0.01 rate) was used on the 1D convolutional layer, and the second-to-last dense layer featured an L2 kernel regulariser (0.01 rate). Uniformity was further maintained with each stochastic classifier model using 50 stochastic samples, a parameter mirrored in the Stochastic model's number of estimators.

2.5 Metrics used for evaluation

2.5.1 Calibration Analysis

Calibration analysis assesses the alignment between a classifier's predicted probabilities and the actual outcomes. As outlined in this paper Silva Filho et al., 2021, it consists of several evaluation methods, including the Classifier Calibration Error, Classifier Calibration Curve, and Classifier Accuracy Confidence Curve, detailed below.

Classifier Calibration Error

The classifier calibration error is used to evaluates the agreement between the predicted probabilities and the observed frequencies. The calibration error classifier is the expected absolute difference between the predicted probabilities and observed probabilities over a set of instances grouped by predicted probability into bins. This can be calculated using equation 2.18:

$$E_{\text{cal}} = \frac{1}{N} \sum_{m=1}^{M} n_m |P_{\text{pred}m} - P\text{obs}_m| \qquad (2.18)$$

In equation 2.18, E_{cal} is the calibration error, N is the total number of instances, n_m is the number of instances in bin m, $P_{\text{pred}m}$ is the average predicted probability in bin m, and $P\text{obs}_m$ is the actual observed probability in bin m.

Classifier Calibration Curve

The classifier calibration curve provides a visual inspection of the model's calibration by plotting the

average predicted confidence against the bin accuracy for each bin. Ideally, for a perfectly calibrated model, this curve should be close to the line of equality (y=x).

Classifier Accuracy Confidence Curve

The classifier accuracy confidence curve offers another way of assessing calibration. It plots the achieved accuracy against the confidence threshold. For each threshold, it considers only the examples with confidence greater than or equal to that threshold and calculates the accuracy of predictions within this subset. The plot provides a perspective on how the model's accuracy varies with its own confidence in its predictions.

The Classifier Calibration Error, Classifier Calibration Curve, and Classifier Accuracy Confidence Curve were imported from keras_uncertainty library Valdenegro-Toro, 2021

2.5.2 Negative Log-Likelihood (NLL)

Negative Log-Likelihood (NLL) quantifies the dissimilarity between the predicted probabilities and the true labels, Quinonero-Candela et al., 2005. It is calculated using equation 2.19:

$$NLL = -\frac{1}{N}\sum_{i=1}^{N}\log(y_{\text{pred}_i}[y_i]) \tag{2.19}$$

In equation 2.19, NLL is the negative log-likelihood, N is the total number of instances, and $y_{\text{pred}_i}[y_i]$ is the predicted probability for the true label y_i of instance i.

The Negative Log-Likelihood (NLL) was method imported from keras_uncertainty library Valdenegro-Toro, 2021

2.5.3 Entropy

Entropy is a measure of the uncertainty or randomness of the predictions, often used in information theory. The standard Shannon entropy, as defined in the paper Valdenegro-Toro & Mori, 2022, is used in this context. For a single instance, the entropy is calculated by summing the product of the predicted probability and the logarithm of the predicted probability for each class. The overall entropy is then computed as the average of the individual entropies across all instances, as described by equations 2.20 and 2.21:

$$H_i = -\sum_{j=1}^{C} P_j \log(P_j) \tag{2.20}$$

$$H = \frac{1}{N}\sum_{i=1}^{N} H_i \tag{2.21}$$

In equations 2.20 and 2.21, H_i is the entropy of instance i, C is the number of possible classes, P_j is the predicted probability for class j, H is the overall entropy, and N is the total number of instances.

The Shannon entropy method was imported from keras_uncertainty library Valdenegro-Toro, 2021

3 Results

This section outlines the findings from our study, highlighting the model performance through Calibration Analysis, Negative Log-Likelihood (NLL), and Entropy metrics.

3.1 General Model Performance

Our models underwent rigorous training and validation procedures, utilising two sets from the dataset Training and Validation, detailed in section 2.1. We prioritised optimising computational efficiency, which included aspects such as model design and algorithmic improvements, to ensure minimal interference with our primary research focus. Similar to the original implementation of Qi, Su, et al., 2017 we considered the use of the base model with 1024 points sample as an example here. The model demonstrated a substantial accuracy of 87%, while a reduction of Loss went down to 0.42, as illustrated in Figure 3.2, Figure 3.1 shows every recorded accuracy for the same model under different point sampling. Moreover, the downsampling had an interesting effect on the variation in the model's accuracy. The points sampling from 4096 down to 1024 was relatively close, around 87%, then it went up to the highest on 512, reaching 88%, then down to stay within the range of 80s% until descending to lowest at 40.31% which is the lowest sampling point of 16, Table 3.1. However, this accuracy can also be observed considering the

struggle of the models to find the correct classification choice when we observe the Loss. The lowest reached Loss was at the sampling point 512 with 0.3671 but reached up to 1.0790 on the 16 sampling points, Table 3.1.

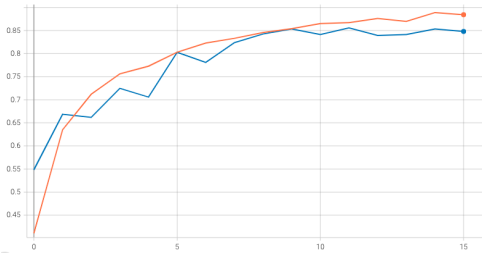

Figure 3.1: Model Accuracy; Accuracy achieved at the 14^{Th} training epochs using 1024 point cloud samples

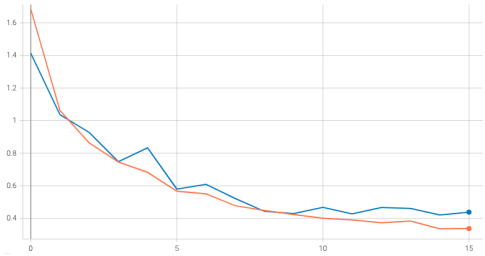

Figure 3.2: Model Loss; Loss minimisation recorded at the 14^{Th} training epochs with 1024 point cloud samples

Additionally, the randomly selected prediction example underscores the model's accuracy, correctly identifying 28 out of 30 objects, as shown in Figure A.1. Despite the high accuracy, the confusion matrix reveals some misclassifications among the categories, highlighting areas for potential improvement. This is further illustrated in Figure A.2.

3.2 Calibration Analysis

Our analysis on calibration provided several key insights. We examined three primary facets of calibration: the Calibration plot "Reliability Diagram", the Calibration Curve, and the Calibration Error. These methods are discussed in depth in section 2.5.1.

The Calibration plot illustrates the relationship between a model's predicted confidence probabilities and its observed accuracy. An ideal Calibration plot would have outcomes closely aligned with the diagonal, signifying that confidence estimates are well-calibrated. However, deviations from this ideal were observed for our models, as seen in Figures A.6, A.7, and A.8. While all models started near the diagonal, many displayed divergences beginning around a confidence of 0.25. An upward divergence indicates regions of overconfidence, which was evident in models like "Base", "MC Dropout", "MC DropConnect", and "Deep Ensemble" at specific sample sizes, such as 4096. A downward divergence, conversely, indicates underconfidence, which was noted in models like "MC Dropout", "MC DropConnect", "Flipout", and "Deep Ensemble". Notably, "Deep Ensemble" at 1024 samples and "MC DropConnect" at 256 samples demonstrated the closest adherence to the ideal calibration. In stark contrast, "MC Dropout" at both 256 and 4096 samples performed poorly.

Table 3.1: Sampling Accuracy

Sample	Accuracy	Loss
4096	87%	0.4058
2048	86%	0.4021
1024	87%	0.3671
512	88%	0.3657
256	85%	0.4306
128	85%	0.4437
64	82%	0.5291
32	76%	0.6903
16	62%	1.0790

The effect of down-sampling also offered insights into model performance. For instance, approximately 80% of models trained with the highest sampling displayed tendencies towards overfitting. Meanwhile, models trained with the lowest sampling showed the least variability.

For a more nuanced understanding, we also assessed the Calibration Error, visualised in Figure A.12. "Deep Ensemble" consistently performed well across multiple sample sizes, particularly 4096, 2048, and 1024. In comparison, "MC DropOut" showcased heightened errors at smaller sample sizes, such as 256, hinting at its sensitivity to variations in data size and quality.

The Calibration Curve revealed that our models typically exhibited a non-linear trajectory. Starting relatively flat, they converged near a quarter of the

confidence axis, subsequently displaying a notable rise in accuracy towards the final point on the plot, as illustrated in Figures A.9, A.10, A.11. This pattern indicates regions where the model's stated confidence did not align with a commensurate increase in accuracy. It suggests that all models tended to be under-confident in their predictions, particularly at lower confidence levels. Notably, a direct correlation was evident between the model's confidence increase and the accuracy of the projections, predominantly around the third section of the confidence axis. It is intriguing to note that models which incorporated some form of uncertainty, such as MC Dropout, Flipout, and Deep Ensemble, generally exhibited higher calibration errors compared to the Base model.

3.3 Quantitative Analysis

3.3.1 Entropy & Negative Log-Likelihood (NLL)

The entropy values for each point cloud sample were determined by averaging the results of each data point five times generated from different models, section 2.4.7. The results revealed varying trends in Figure A.4 through histogram analysis across different point cloud sizes. Plots with larger sampling sizes, 4096 to 512, had entropy data tend to concentrate mainly on the side of the plot, suggesting less certainty. In contrast, smaller sizes range from 256 to 16 sampling points and seem to spread out, indicating potential uncertainty. Specifically, towards lower entropy values, Points 64 and 32 exhibited a notable increase in the middle range. In addition, the 16th sampling point was remarkably distinct, with a significant presence of a bell shape, which is an alarming sign of a substantial uncertainty level.

We have observed various trends and insights from the NLL analysis across all models, classes, and varying points, table B.1. The NLL metric generally tends to range immensely between all the involved factors. Comparing the results of table B.3, revealed that the choice of the sampling number affected the uncertainty and the model evidently since averaging the categories NLL individually and then extracting the mean considering all models for shows the sampling of 4096 got a total average of 0.744, sampling 2048 got a total average of 0.726 while 1024 had an average of 0.722. These values were emphasised further looking at the Entropy values in values in table B.4, where 4096 showed 0.41, while 2048 and 1024 resulted in 0.36.

To measure the performance in uncertainty regarding each model, we have used the average of the Entropies and NLL of each model across all categories in that sample class. Hence, the Deep Ensemble model consistently achieved a lower NLL across all points, with its minimum at 0.63 for 4096 points and, on average, around 0.6 until reaching 0.67 at 256 sampling points. The second best was Flipout, with its average $\tilde{0}.7$ until reaching the extreme downsampling level around 128 sampling points, table B.1. However, that was not the case in terms of Entropy, Flipout was the optimal model considering the average Entropy across all samples scoring 0.32 at each of the largest sampling points. The second was MC DropConnect, table B.1.

When we analyse individual classes and exclude the results after 128 sampling since it is evident that all models were not optimal, a few patterns emerge. Looking at Entropy table B.2, and NLL table table B.1, the values remain consistently low for the monitor and chair across all models, even across different point sampling, e.g. the chair's scoring was as low as 0.09 at 4096, 2048 and reaching 0.2 on 1024. While some other categories were better with some sampling but not the others, such as monitor sofa and bed, as their performance changed across different sampling points. In contrast, the worse categories' performance was desk, nightstand, and dresser, and the last in their list was a bathtub, table B.1.

3.4 Qualitative Analysis

3.4.1 Entropy & Negative Log-Likelihood (NLL)

Distribution Analysis Across Sample Sizes

The entropy scatter plot, like the histogram, derives its values by averaging each data point from five distinct model outcomes. Interpreting the scatter plot hinges on the y-axis distribution: a higher positioning signifies greater entropy and vice versa, Figure A.5. For larger sample sizes, namely 4096 to 512, specific categories, like the monitor and sofa, predominantly cluster near the y-axis' base,

suggesting lower entropy values. Conversely, nightstands and desks display a more dispersed y-axis distribution, starting notably above the base and sometimes reaching beyond the 1.25 mark. In mid-sized samples from 256, a more obvious spread was observed. Then below 64 and 32, the categories allocate more evenly, culminating in a near-uniform distribution by the 16-sample plot. Despite that, The monitor category remains consistently close to the y-axis base, indicating its stability against downsampling. In contrast, the desk, nightstand, and dresser were the least stable under downsampling conditions.

Model-wise Analysis Across Categories and Samples

Analysing the models based on their Negative Log Likelihood (NLL) values across different point cloud samplings in Figure A.3 unveiled clear patterns and tendencies in their performances.

The Base model displayed noticeable fluctuations dependent on the number of points sampled. Particularly, the bathtub category was unpredictable, pointing towards challenges or intricacies in this specific category. Similarly, the MC DropOut model, while relatively consistent, also faced challenges in the bathtub category, suggesting shared complexities across models for this category. The desk and Nightstands within this model further stood out due to their unique performance curves, potentially highlighting category-specific challenges. Further, the MC DropConnect model have presented a consistent trend but was not devoid of occasional spikes, especially in the dresser and nightstand categories. This pattern emphasises the existence of inherent challenges within certain categories that might require further exploration. The Flipout model, on the other hand, performed with a balance of highs and lows.

3.5 Downsampling

To better understand the effects of downsampling, we plotted both the Negative Log-Likelihood (NLL) and Entropy on the same graph. This allows us to visualise the impact of downsampling across different categories.

Upon examining the downsampled results at 4096, 2048, and 1024 as shown in Figure A.13, notable discrepancies emerge between the NLL and Entropy results. A striking observation is that the performance at the 1024 sample size often surpasses that of the larger 4096 and 2048 sample sizes. Detailed analysis indicates that sample sizes above 1024 consistently register elevated average values for both Entropy and NLL. For instance, at the 4096 sample size, many models exhibit an NLL nearing 1.0 and an Entropy around 2.0, making its performance inferior to both the 2048 and 1024 sample sizes. Furthermore, a closer inspection of the 1024 sample size reveals superior performance over the 2048 sample size, particularly within individual categories of models such as Deep Ensemble, Flipout, MC Dropout, and MC DropConnect. In Deep Ensemble, while categories like desks and nightstands display subpar results at 1024, others like Bathtub, Bed, Desk, Dresser, and Sofa exhibit marked improvements. The table category, for example, demonstrated over a 50% enhancement in both NLL and Entropy when compared to the 2048 sample size.

Transitioning to the 512 to 128 downsample range, Figure A.14 illustrates a significant uptick in both NLL and Entropy, implying deteriorated performance. Models at the 128 sample size, specifically Flipout, MC DropOut, and MC DropConnect, display Entropy values often surpassing 2.0. However, the NLL values remained somewhat stable, typically staying slightly below 1.0.

Further downsampling paints a grim picture. As shown in Figure A.15, the plots begin to saturate at the 64 sample mark. Here, we witness instances where the Entropy of MC DropConnect exceeds 2.0, and for MC DropOut, it approaches a staggering 3.0. High NLL values become increasingly commonplace, often reaching 1.0. At the lowest sample sizes of 32 and 16, Entropy and NLL values dominate the plot, with some models displaying Entropy values beyond 3.5.

Throughout this downsampling analysis, the Deep Ensemble model consistently better on average than other models, showcasing consistency in reporting its prediction. Conversely, post the Base model, MC DropConnect emerged as the most adversely affected model.

4 Discussion

This study investigated the capability of Bayesian neural networks to measure the uncertainties of point cloud classification tasks. The inquiry behind this research asked: 'How significant is the influence of downsampling on introducing uncertainty in cloud classification, and what approaches are suitable for its validation?' The findings of this study gave essential knowledge that can help progress in the production of solid and precise AI systems which can skillfully address the intricacies and uncertainties existing in point cloud data.

The application of Monte Carlo Dropout, Monte Carlo DropConnect, Flipout, and Deep Ensemble, each implemented within a specialised version of the PointNet model, served as the foundation of our methodology. These modified models were trained and validated alongside a basic PointNet model for comparison purposes. Additionally, the models were enhanced with techniques that utilise a stochastic classifier to generate predictions that can quantify the level of uncertainty. In order to conduct a comprehensive evaluation, the models underwent a sequential downsampling process, gradually reducing the number of points from 4096 to 16 points. This technique allowed us to assess the models' performance across different volumes of point cloud data using various metrics to measure performance and uncover uncertainty. Through this process, we gained insight into the models' ability to classify under increased stress caused by varying point cloud densities.

This calibration analysis of different models indicated a general tendency for underconfidence, likely due to the complexity of point cloud data. Despite this, they maintained a high accuracy rate. As confidence levels rose, prediction accuracy also improved, providing evidence of their reliability. However, models that incorporated elements of uncertainty (such as MC Dropout, MC DropConnect, Flipout, and deep ensemble) had higher calibration errors and thus were less reliable. This finding suggests that introducing uncertainty may cause model miscalibration.

It is worth noting that the models had different sensitivities to different classes, which suggests that the relationship between class representation, class complexity, diversity within a class, and biases in model training is complex. The results also highlight the importance of selecting the appropriate model for specific tasks. The MC Dropout model had higher uncertainty and lower accuracy for the Desk class compared to the MC DroConnect model, possibly due to class imbalance or functional differences between the models. On the other hand, the deep ensemble model had fairly consistent performance across classes, although there were some variations due to data complexities and uncertainties. Certain classes like Desk, dresser, and Nightstand had higher NLL and entropy, indicating that the model struggled to confidently predict these classes, possibly due to data noise or a lack of informative features. This might be due to inherent data noise or a lack of sufficiently informative features.

The NLL and entropy scores specific to each class indicated that different models handle class-specific uncertainties in various ways. Notably, categories with higher entropy generally have higher NLL scores, suggesting a correlation between model uncertainty and classification performance. Furthermore, the data in tables (Reference tables here) aligned with the results' analysis (reference here the analysis section) of class representation versus NLL, demonstrating that models respond differently to class representation and inherent complexities. Despite having low representation, certain classes did not necessarily exhibit the highest NLL or Entropy, indicating intricate dynamics within model predictions influenced by factors beyond class representation. Many categories displayed confusion, which could be attributed to a class imbalance in the datasetA.1 and A.1. For instance, the models were tested with classes like a bathtub, which accounted for only 5.51% of the entire dataset, while a nightstand comprised 9.47% of the validation dataset. The first class performed relatively well, but the second class was one of the worst in the entire dataset. Analysing the training dataset clarifies this phenomenon, as the distribution in the training sets was the opposite.

Moreover, exploring downsampling's impact on uncertainty in various models revealed intriguing findings. The results unveiled a noteworthy

surge in uncertainty with an increase in the downsampling rate. However, this effect was inconsistent across different classes, displaying various outcomes. Certain classes consistently displayed heightened uncertainty across all models, while others showcased improvements. Meagre downsampling rates resulted in increased uncertainty for most classes. Nonetheless, some classes maintained or enhanced their positions, indicating potential robustness in their corresponding model architectures. Surprisingly, the results indicate that higher sampling points do not necessarily guarantee better outcomes. On the contrary, it might have the opposite effect. Consequently, a prominent sensitivity is observed in the point cloud classification surrounding the PointNet architecture regarding the number of points.

Building upon the foundational work by Lumban-Gaol et al., 2021 and Valdenegro-Toro & Mori, 2022, our study adds to the literature highlighting the significance of uncertainty quantification in point cloud data classification. The authors, Lumban-Gaol et al. and Nurunnabi et al., provided key insights into model accuracy, pointing to a recurring problem of class sensitivity in point clouds. While revealing their findings, they did not probe deeply into uncertainty aspects. Our research not only supports their conclusions but also accentuates the prevalence of the issue based on our data. Their selection of data, which had its own set of limitations, particularly in volume, can be better understood in the context of our study. We have determined that the quality and quantity of data can indeed influence predictive outcomes.

In addition, the groundbreaking research of Valdenegro-Toro & Mori enriched our understanding of the intricate interplay between different types of uncertainties and underscored the need for more robust quantification techniques. Their findings about the unreliability of aleatoric uncertainty in out-of-distribution settings shaped our research and evaluation criteria. This study adds to this discourse by providing empirical evidence of the models' differential performance across different classes and their sensitivity to downsampling. This emphasises the potential of model customisation based on specific tasks, classes, or data complexities.

Our research contributes to the flourishing domain of uncertainty quantification in machine learning, especially in point cloud data classification. While we found encouraging results, it is crucial to recognise the dynamic and swiftly evolving nature of AI and machine learning technologies. Therefore, exploring increasingly sophisticated methods for handling uncertainties remains a continuous journey, requiring relentless research, exploration, and refinement. Future studies must aim to disentangle these uncertainties and dive deep into identifying aleatoric and epistemic uncertainties.
Furthermore, the models' differential performance across different classes and their sensitivity to downsampling also point towards the potential of model customisation based on specific tasks, classes, or data complexities. Additionally, the influence of dataset imbalance on the model's performance and uncertainty estimations underscores the need for extra attention to categorical balancing techniques.

5 Conclusions

This research attempted to examine the capabilities of Bayesian neural networks for quantifying uncertainties in point cloud classification tasks under progressive downsampling, a crucial aspect in the evolution of precise AI systems. The methodologies employed, including variants of the PointNet model, offered valuable insights into the dynamics of point cloud data under varying data volumes and complexities. Our findings resonate with the implications of prior studies, underscoring the challenge of class sensitivity in point clouds. Moreover, they emphasise the nuanced relationship between class representation, complexity, and inherent biases in training. A key revelation from our study is the potential miscalibration introduced by incorporating elements of uncertainty, pointing towards the necessity for more rigorous validation techniques in future works.

The differential model performance across classes and the observed sensitivities to downsampling emphasise the potential for task-specific model customisation. This adaptability will be

paramount in addressing specific challenges posed by unique datasets, classifications, or underlying data complexities. Additionally, our observations concerning dataset imbalance highlight the importance of employing balanced training techniques to enhance model reliability and accuracy.

In summary, while our study contributes valuable insights to the realm of uncertainty quantification in machine learning, it also accentuates the ever-evolving nature of AI. As the domain progresses, so does the need for refined methodologies that address these uncertainties with increasing precision. Our work serves as a stepping stone, encouraging future endeavours to disentangle further and understand the uncertainties inherent in machine learning models, especially concerning point cloud data classification.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... others (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, *76*, 243–297.

Chen, C., Liu, D., & Xu, C. (2022). Point clouds downsampling based on complementary attention and contrastive learning. In *Igarss 2022-2022 ieee international geoscience and remote sensing symposium* (pp. 1872–1875).

Gal, Y., & Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning* (pp. 1050–1059).

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., ... others (2023). A survey of uncertainty in deep neural networks. *Artificial Intelligence Review*, 1–77.

Jordan, M. I. (2019, 7). Artificial intelligence—the revolution hasn't happened yet. *Harvard Data Science Review*, *1*(1). Retrieved from `https://hdsr.mitpress.mit.edu/pub/wot7mkc1`

Kim, H., & Kim, C. (2020). Deep-learning-based classification of point clouds for bridge inspection. *Remote Sensing*, *12*(22), 3757.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, *30*.

Lumban-Gaol, Y., Chen, Z., Smit, M., Li, X., Erbaşu, M., Verbree, E., ... Van Der Vaart, N. (2021). A comparative study of point clouds semantic segmentation using three different neural networks on the railway station dataset. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *43*, 223–228.

Nurunnabi, A., Teferle, F. N., Li, J., Lindenbergh, R., & Parvaz, S. (2021). Investigation of pointnet for semantic segmentation of large-scale outdoor point clouds. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *46*, 397–404.

Oberdiek, P., Rottmann, M., & Gottschalk, H. (2018). Classification uncertainty of deep neural networks based on gradient information. In *Artificial neural networks in pattern recognition: 8th iapr tc3 workshop, annpr 2018, siena, italy, september 19–21, 2018, proceedings 8* (pp. 113–125).

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 652–660).

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, *30*.

Quinonero-Candela, J., Rasmussen, C. E., Sinz, F., Bousquet, O., & Schölkopf, B. (2005). Evaluating predictive uncertainty challenge. In *Machine learning challenges workshop* (pp. 1–27).

Rayhan, A. (2023). *Artificial intelligence in robotics: From automation to autonomous systems.* DOI.

Silva Filho, T., Song, H., Perello-Nieto, M., Santos-Rodriguez, R., Kull, M., & Flach, P. (2021). Classifier calibration: How to assess and improve predicted class probabilities: a survey. *arXiv e-prints*, arXiv–2112.

Sultana, F., Sufian, A., & Dutta, P. (2020). Evolution of image segmentation using deep convolutional neural network: A survey. *Knowledge-Based Systems*, *201*, 106062.

Valdenegro-Toro, M. (2021). *Keras uncertainty.* `https://github.com/mvaldenegro/keras-uncertainty`. (GitHub repository)

Valdenegro-Toro, M., & Mori, D. S. (2022). A deeper look into aleatoric and epistemic uncertainty disentanglement. In *2022 ieee/cvf conference on computer vision and pattern recognition workshops (cvprw)* (pp. 1508–1516).

Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., & Vercauteren, T. (2019). Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, *338*, 34–45.

Wen, Y., Vicol, P., Ba, J., Tran, D., & Grosse, R. (2018). Flipout: Efficient pseudo-independent weight perturbations on mini-batches. *arXiv preprint arXiv:1803.04386*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 1912–1920).

Yun, J., & Lee, J.-S. (2023). Learning from imbalanced data using triplet adversarial samples. *IEEE Access*, *11*, 31467–31478.

A Appendix

Table A.1: Distribution of Classes in the Training Dataset for the ModelNet10

Class	Count	Percentage
Bathtub	106	2.66%
Bed	515	12.90%
Chair	889	22.28%
Desk	200	5.01%
Dresser	200	5.01%
Monitor	465	11.65%
Night Stand	200	5.01%
Sofa	680	17.04%
Table	392	9.82%
Toilet	344	8.62%

Table A.2: Distribution of Classes in the Validation Dataset for the ModelNet10

Class	Count	Percentage
Bathtub	50	5.51%
Bed	100	11.01%
Chair	100	11.01%
Desk	86	9.47%
Dresser	86	9.47%
Monitor	100	11.01%
Night Stand	86	9.47%
Sofa	100	11.01%
Table	100	11.01%
Toilet	100	11.01%

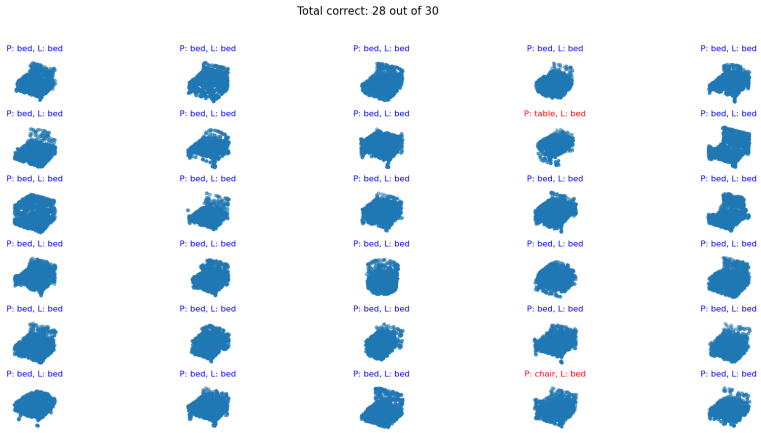

Figure A.1: Predicted examples; Predicted randomly sampled objects.

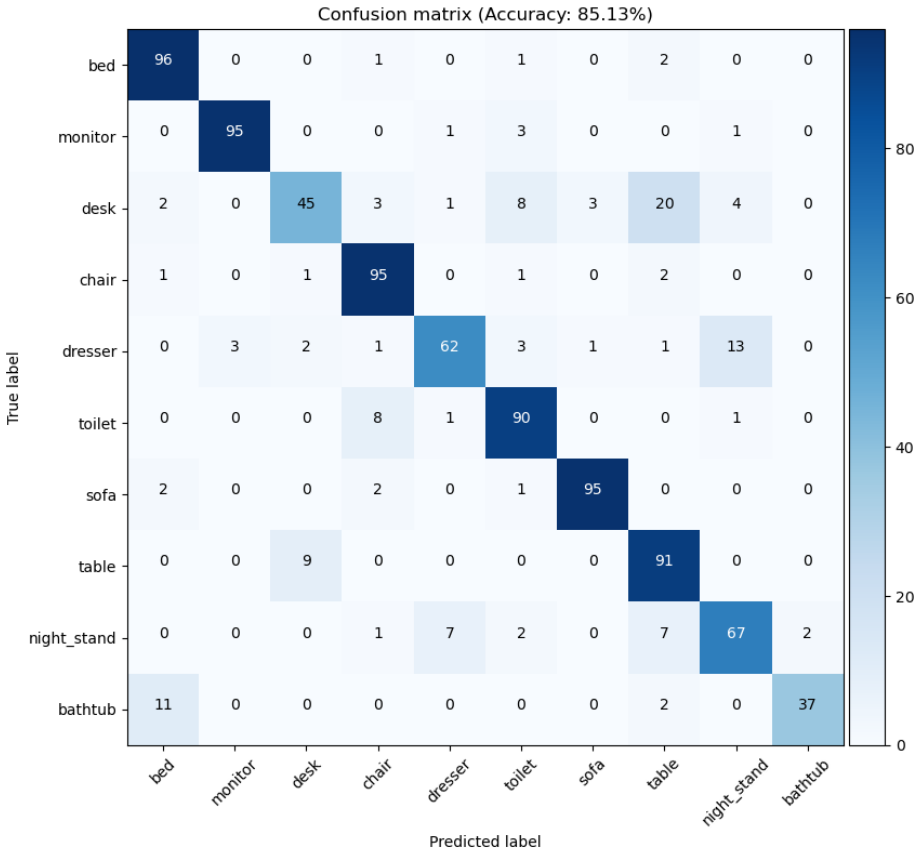

Figure A.2: Confusion Matrix; Illustration of the model's confusion without Normalisation

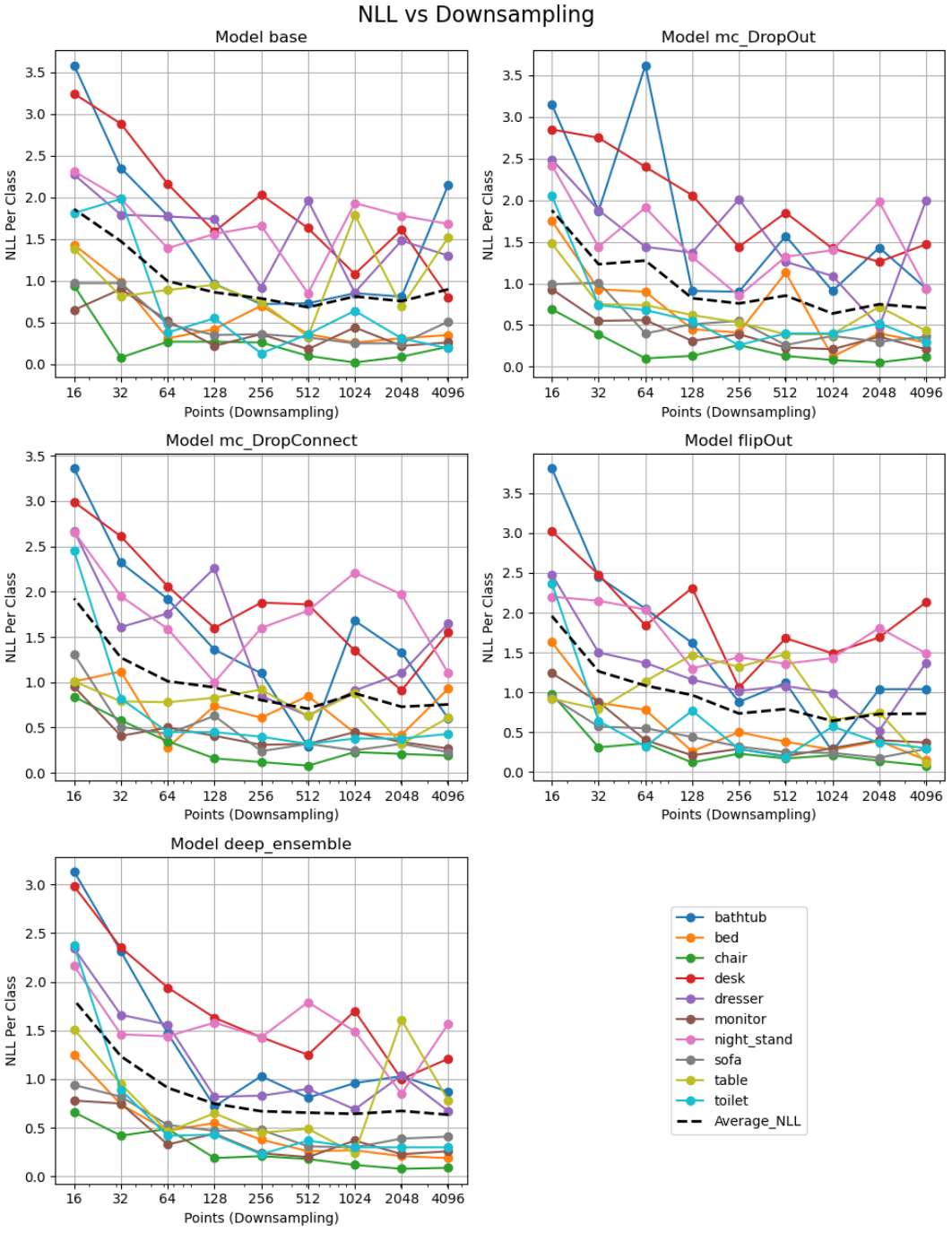

Figure A.3: Comparative analysis of model performances based on NLL values across different point cloud samplings. The figure consists of five plots, each representing one of the models described in Models 2.2. The x-axis indicates the NLL per class, while the y-axis represents the downsampling point clouds. Category labels are provided on the bottom-left, corresponding to each line in the plots.

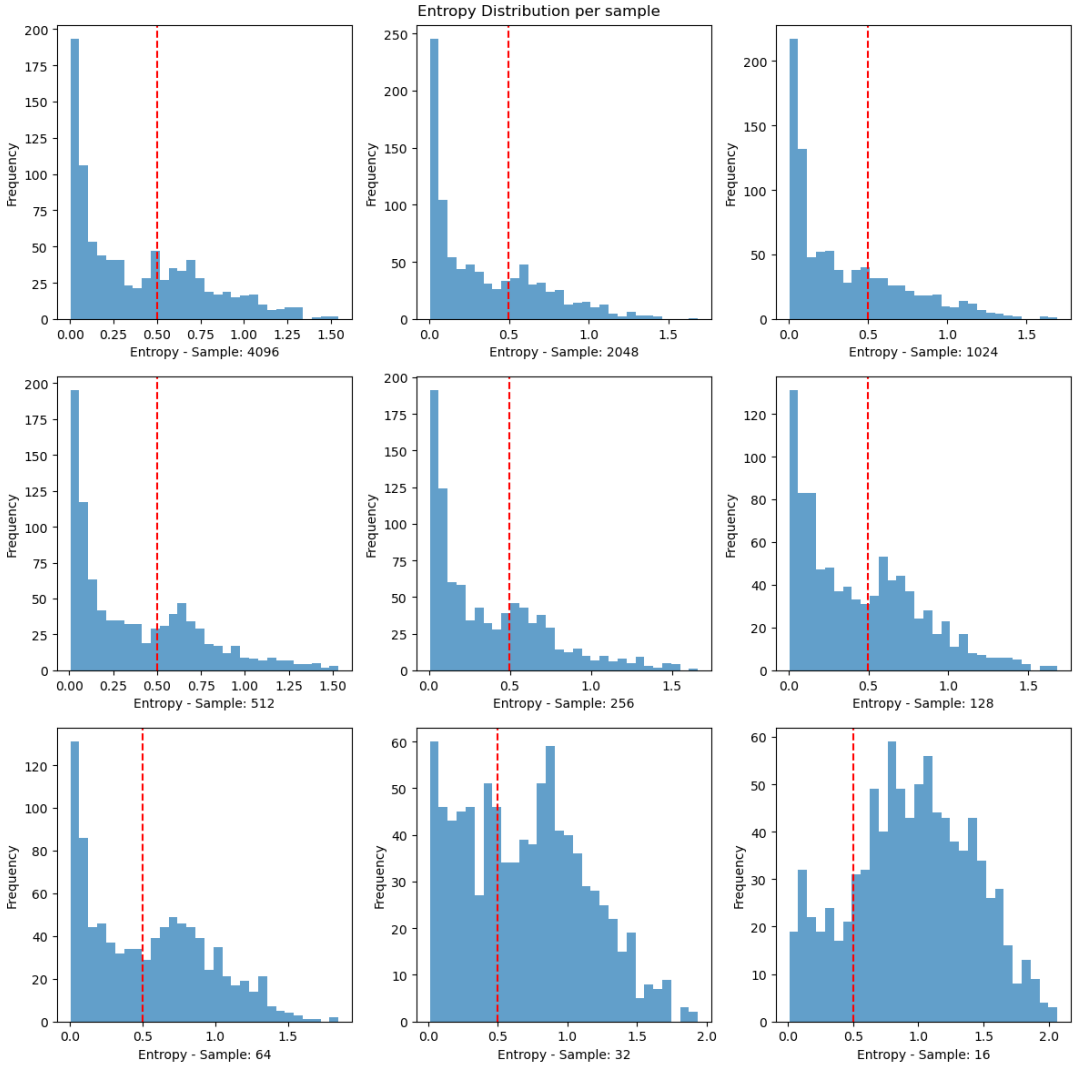

Figure A.4: Average Entropy per Sample; Average Entropy per Sample histogram represents every sampling frame, starting from left to right and top to bottom: 1024, 512, 256, 128, 64, 32, 16, 8, 4

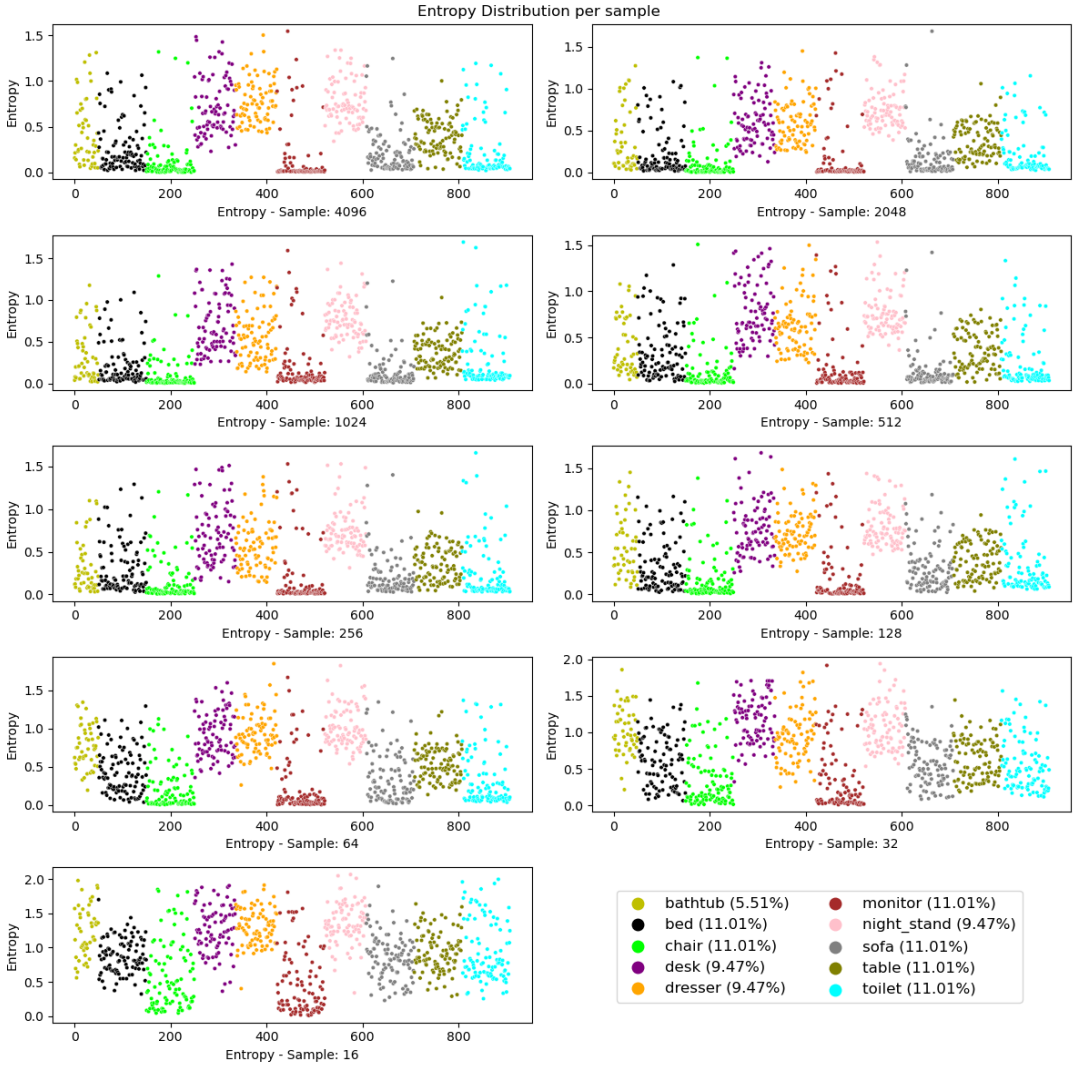

Figure A.5: Average Entropy per Sample; Average Entropy per Sample histogram represents every sampling frame, starting from left to right and top to bottom: 1024, 512, 256, 128, 64, 32, 16, 8, 4

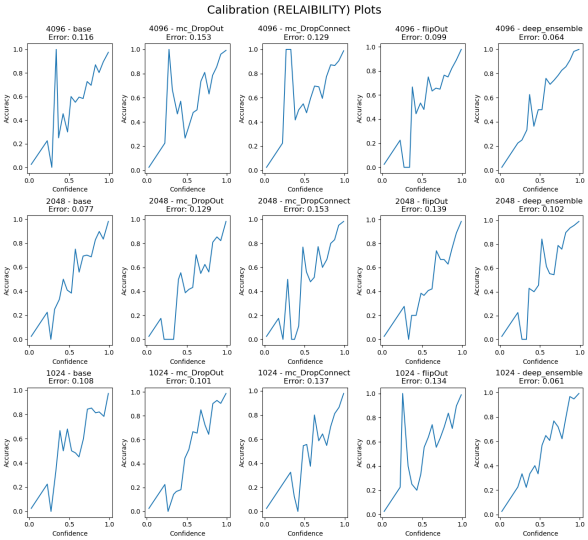

Figure A.6: Calibration Plot "Reliability diagram"

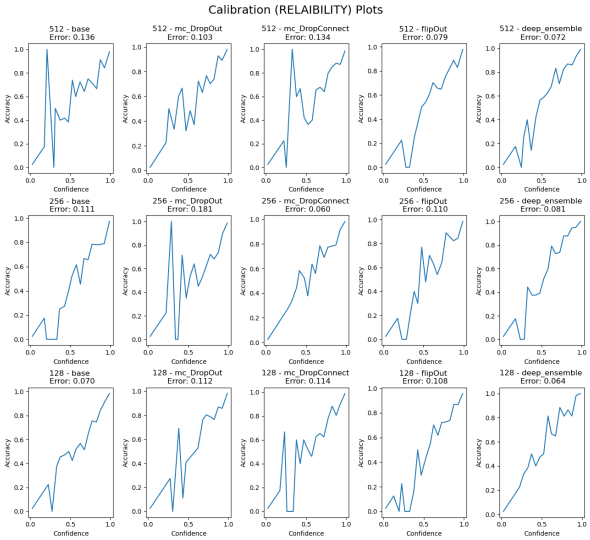

Figure A.7: Calibration Plot "Reliability diagram"

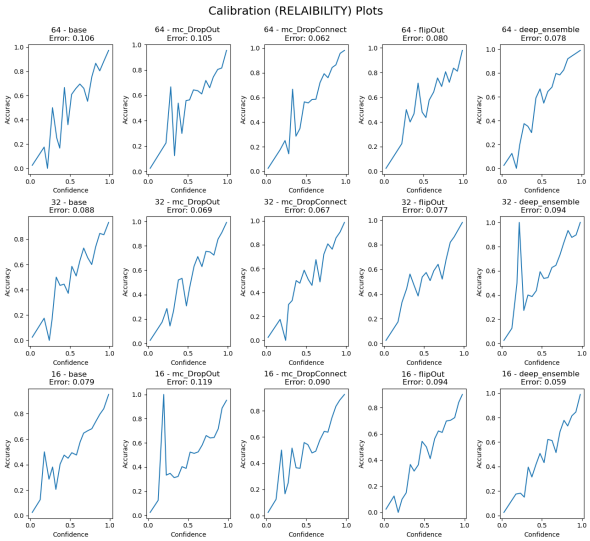

Figure A.8: Calibration Plot "Reliability diagram"

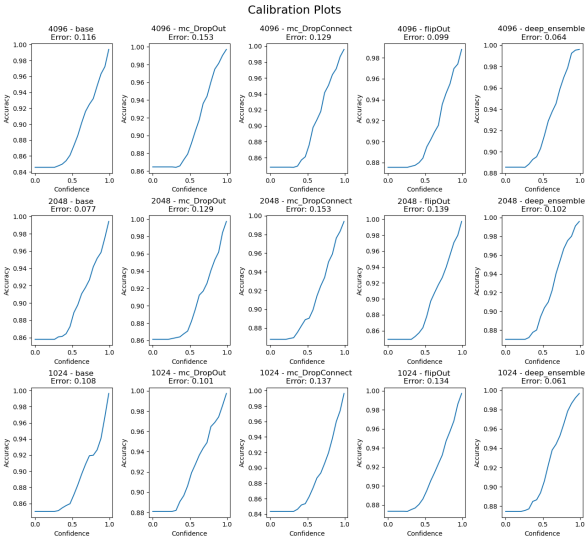

Figure A.9: Calibration Curve

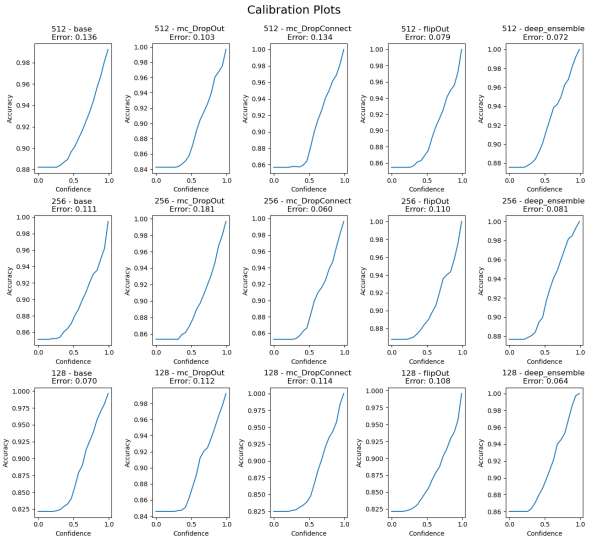

Figure A.10: Calibration Curve

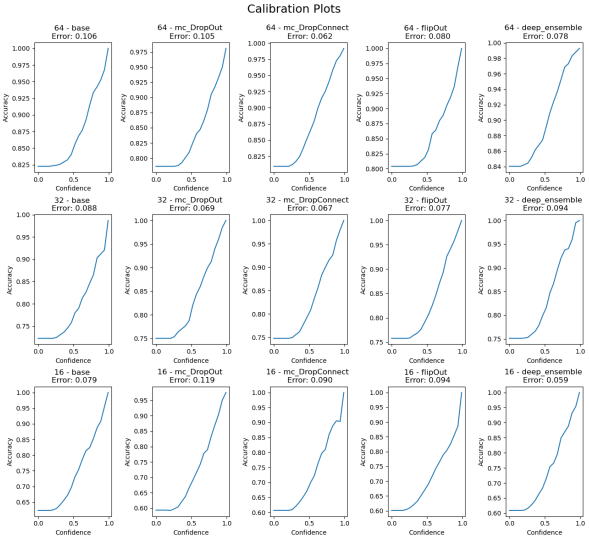

Figure A.11: Calibration Curve

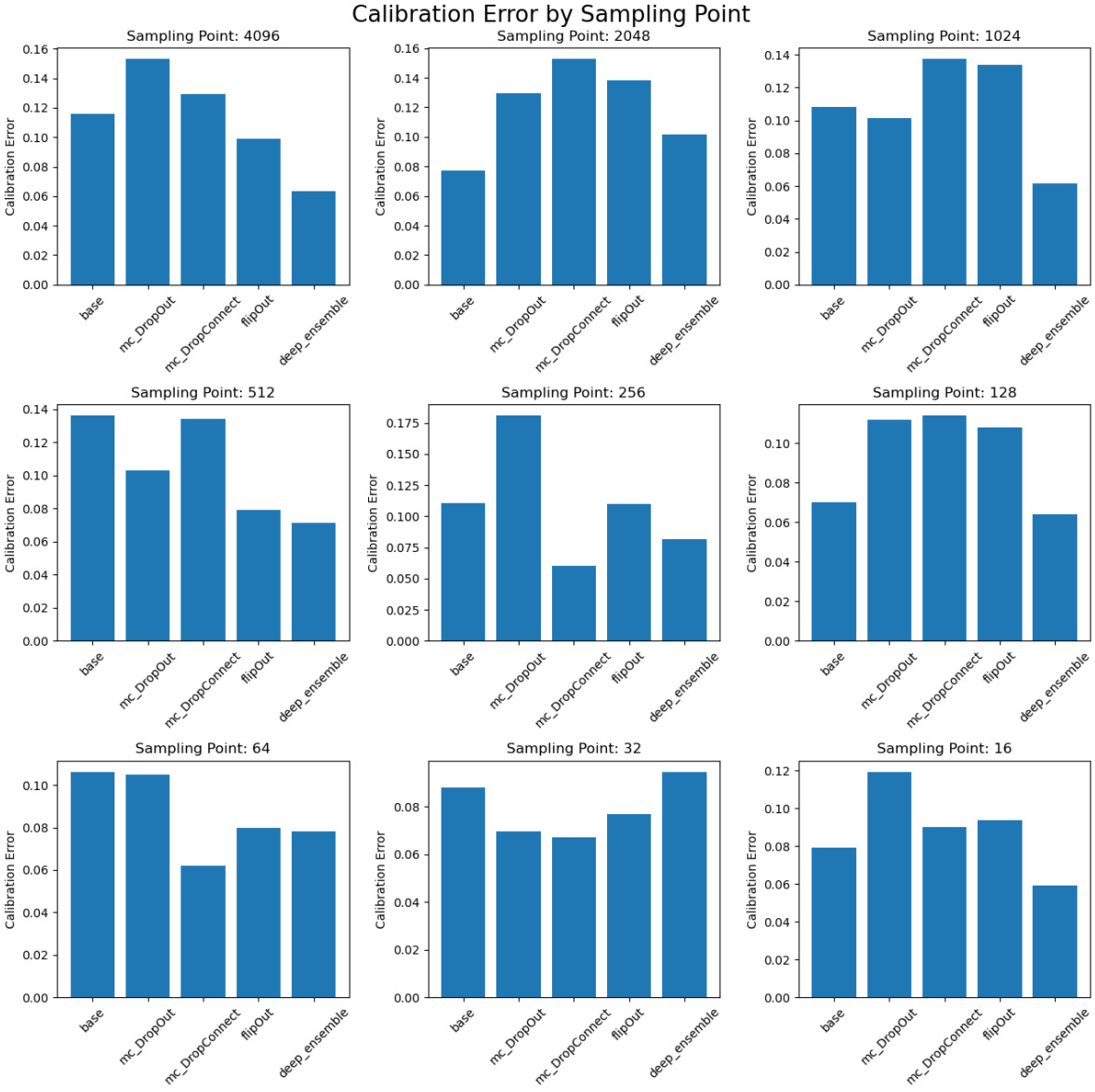

Figure A.12: Calibration Error

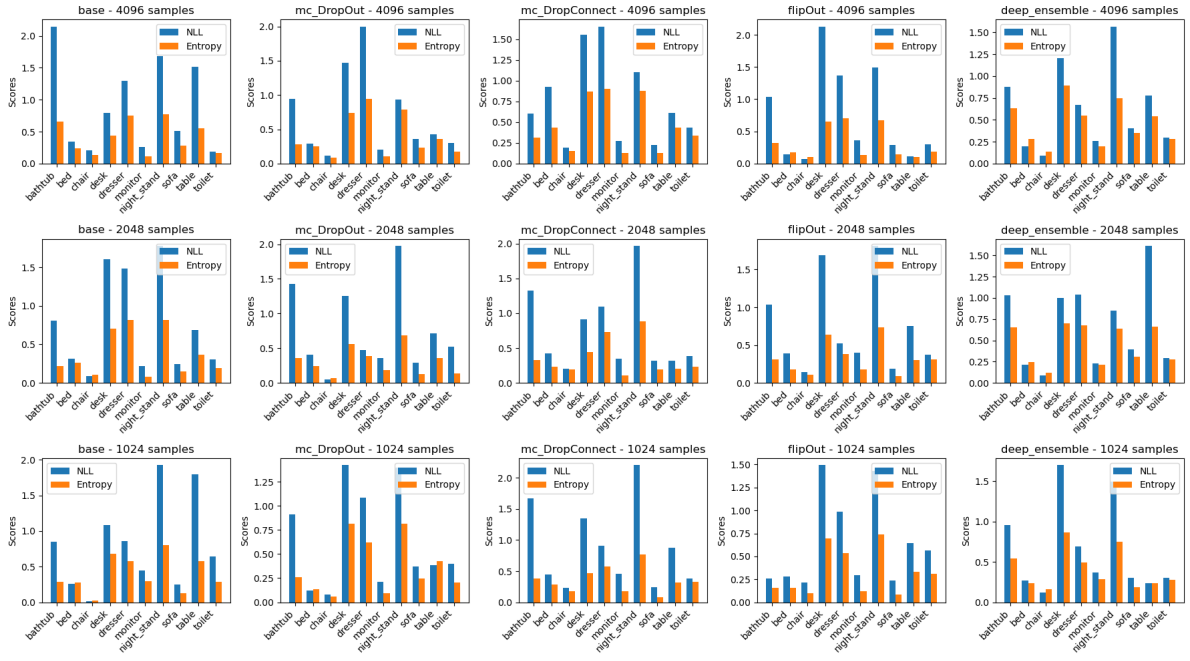

Figure A.13: NLL and Entropy scores of the model for each category across all sample sizes. Models are plotted on the x-axis, and their scores are on the y-axis. Each model's name is displayed at the top, followed by the corresponding sample points.

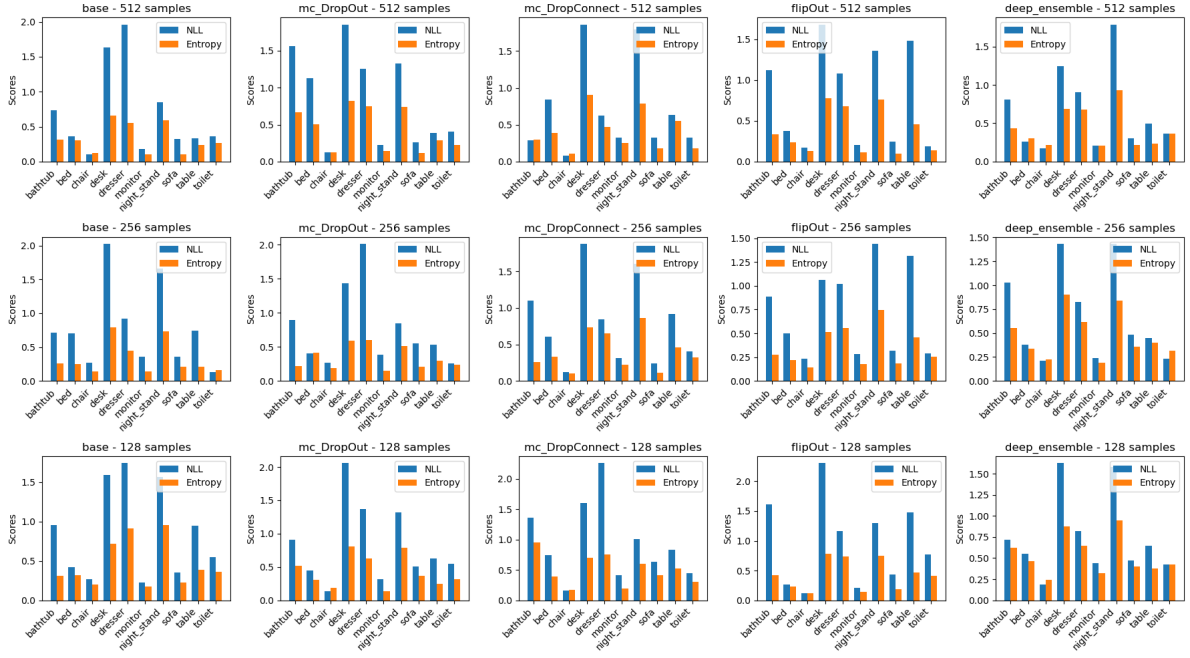

Figure A.14: NLL and Entropy scores of the model for each category across all sample sizes. Models are plotted on the x-axis, and their scores are on the y-axis. Each model's name is displayed at the top, followed by the corresponding sample points.

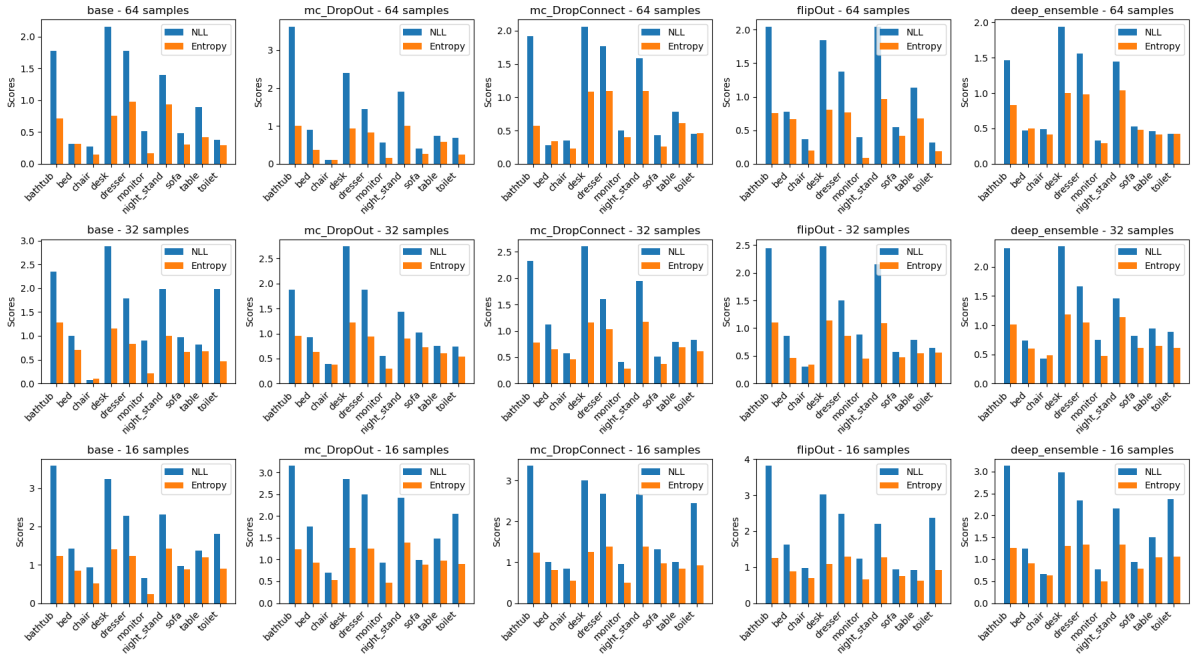

Figure A.15: NLL and Entropy scores of the model for each category across all sample sizes. Models are plotted on the x-axis, and their scores are on the y-axis. Each model's name is displayed at the top, followed by the corresponding sample points.

B Appendix

Models	Sample	NLL.	Bath.	Bed	Chai.	Desk	Dres.	Moni.	NigSt.	Sofa	Tabl.	Toil.
Base	4096	0.90	2.15	0.35	0.21	0.80	1.30	0.26	1.68	0.51	1.52	0.19
MC DO	4096	0.70	0.94	0.29	0.12	1.47	2.00	0.21	0.94	0.36	0.43	0.30
MC DC	4096	0.76	0.60	0.93	0.19	1.55	1.65	0.27	1.10	0.23	0.61	0.43
FlipOt	4096	0.73	1.04	0.15	0.08	2.13	1.37	0.37	1.49	0.29	0.11	0.30
Ensemb	4096	0.63	0.87	0.19	0.09	1.21	0.67	0.26	1.57	0.41	0.78	0.30
Base	2048	0.75	0.81	0.31	0.09	1.61	1.48	0.22	1.78	0.25	0.69	0.31
MC DO	2048	0.75	1.43	0.40	0.05	1.26	0.48	0.36	1.98	0.30	0.72	0.52
MC DC	2048	0.73	1.33	0.42	0.21	0.91	1.10	0.34	1.97	0.32	0.32	0.38
FlipOt	2048	0.73	1.04	0.39	0.14	1.69	0.52	0.40	1.81	0.18	0.75	0.37
Ensemb	2048	0.67	1.03	0.21	0.08	1.00	1.04	0.23	0.85	0.39	1.61	0.30
Base	1024	0.81	0.85	0.26	0.02	1.08	0.86	0.44	1.93	0.25	1.79	0.64
MC DO	1024	0.64	0.91	0.12	0.08	1.42	1.09	0.21	1.40	0.37	0.39	0.40
MC DC	1024	0.88	1.68	0.44	0.23	1.35	0.91	0.45	2.21	0.25	0.88	0.38
FlipOt	1024	0.64	0.26	0.28	0.21	1.49	0.99	0.30	1.43	0.24	0.65	0.57
Ensemb	1024	0.64	0.96	0.27	0.12	1.70	0.69	0.37	1.49	0.30	0.24	0.30
Base	512	0.68	0.73	0.36	0.10	1.63	1.96	0.18	0.85	0.32	0.33	0.36
MC DO	512	0.85	1.57	1.13	0.13	1.85	1.26	0.23	1.32	0.26	0.39	0.40
MC DC	512	0.71	0.29	0.85	0.08	1.86	0.63	0.32	1.79	0.32	0.63	0.32
FlipOt	512	0.79	1.12	0.38	0.17	1.68	1.08	0.20	1.36	0.25	1.48	0.19
Ensemb	512	0.66	0.81	0.26	0.18	1.25	0.90	0.20	1.79	0.31	0.49	0.37
Base	256	0.79	0.72	0.70	0.26	2.03	0.92	0.36	1.66	0.36	0.74	0.13
MC DO	256	0.76	0.90	0.41	0.26	1.44	2.01	0.39	0.85	0.55	0.53	0.26
MC DC	256	0.80	1.10	0.61	0.12	1.88	0.85	0.31	1.60	0.24	0.92	0.40
FlipOt	256	0.74	0.88	0.50	0.23	1.06	1.02	0.29	1.44	0.32	1.32	0.29
Ensemb	256	0.67	1.03	0.38	0.21	1.43	0.83	0.24	1.43	0.48	0.45	0.23
Base	128	0.86	0.96	0.42	0.27	1.59	1.74	0.22	1.56	0.35	0.95	0.55
MC DO	128	0.82	0.91	0.45	0.13	2.06	1.37	0.31	1.32	0.51	0.62	0.55
MC DC	128	0.94	1.36	0.74	0.16	1.60	2.26	0.41	1.00	0.63	0.83	0.45
FlipOt	128	0.97	1.62	0.26	0.12	2.31	1.16	0.21	1.30	0.44	1.47	0.77
Ensemb	128	0.75	0.72	0.55	0.19	1.63	0.82	0.44	1.58	0.47	0.65	0.43
Base	64	1.00	1.78	0.31	0.27	2.16	1.77	0.52	1.39	0.48	0.89	0.38
MC DO	64	1.27	3.62	0.90	0.10	2.40	1.44	0.56	1.91	0.40	0.74	0.68
MC DC	64	1.01	1.92	0.28	0.35	2.06	1.76	0.50	1.59	0.43	0.78	0.45
FlipOt	64	1.08	2.05	0.78	0.36	1.84	1.37	0.40	2.04	0.55	1.14	0.32
Ensemb	64	0.91	1.47	0.47	0.49	1.94	1.56	0.33	1.44	0.53	0.46	0.42
Base	32	1.47	2.35	0.99	0.08	2.88	1.79	0.90	1.98	0.97	0.81	1.98
MC DO	32	1.23	1.87	0.93	0.39	2.75	1.88	0.55	1.44	1.01	0.75	0.74
MC DC	32	1.27	2.32	1.12	0.58	2.61	1.61	0.41	1.95	0.51	0.79	0.82
FlipOt	32	1.26	2.45	0.87	0.31	2.48	1.50	0.88	2.15	0.57	0.79	0.64
Ensemb	32	1.23	2.31	0.74	0.42	2.35	1.66	0.75	1.46	0.82	0.95	0.89
Base	16	1.86	3.58	1.43	0.94	3.24	2.27	0.65	2.31	0.97	1.38	1.81
MC DO	16	1.88	3.15	1.75	0.69	2.85	2.49	0.93	2.42	0.99	1.48	2.05
MC DC	16	1.93	3.36	1.01	0.84	2.99	2.67	0.95	2.66	1.31	1.01	2.45
FlipOt	16	1.96	3.81	1.63	0.98	3.02	2.48	1.24	2.20	0.94	0.92	2.37
Ensemb	16	1.81	3.13	1.25	0.66	2.98	2.34	0.78	2.16	0.94	1.51	2.37

Table B.1: The Negative Log Likelihood results of five different models (Base, MC DropOut, MC DropConnect, FlipOt, and Deep Ensemble) evaluated on the dataset, Section 2.1 across nine downsampling levels. Each column after the "Sample" column represents the entropy for the specified category in the label.

Models	Sample	Entr.	Bath.	Bed	Chai.	Desk	Dres.	Moni.	NigSt.	Sofa	Tabl.	Toil.
Base	4096	0.41	0.66	0.24	0.14	0.44	0.76	0.11	0.77	0.28	0.55	0.17
MC DO	4096	0.40	0.28	0.25	0.08	0.74	0.95	0.10	0.78	0.23	0.35	0.18
MC DC	4096	0.46	0.32	0.44	0.15	0.87	0.90	0.13	0.88	0.12	0.43	0.34
FlipOt	4096	0.32	0.32	0.18	0.10	0.65	0.70	0.13	0.68	0.14	0.10	0.19
Ensemb	4096	0.46	0.63	0.28	0.14	0.89	0.55	0.20	0.75	0.35	0.54	0.28
Base	2048	0.37	0.22	0.26	0.10	0.71	0.81	0.08	0.82	0.15	0.37	0.19
MC DO	2048	0.31	0.35	0.24	0.07	0.56	0.39	0.19	0.69	0.12	0.36	0.14
MC DC	2048	0.35	0.33	0.23	0.19	0.45	0.73	0.11	0.88	0.19	0.20	0.23
FlipOt	2048	0.32	0.31	0.18	0.11	0.64	0.38	0.18	0.74	0.09	0.30	0.31
Ensemb	2048	0.45	0.65	0.25	0.12	0.70	0.68	0.22	0.64	0.31	0.66	0.27
Base	1024	0.39	0.29	0.28	0.03	0.68	0.57	0.29	0.80	0.13	0.58	0.29
MC DO	1024	0.37	0.26	0.13	0.06	0.82	0.62	0.09	0.82	0.25	0.43	0.20
MC DC	1024	0.36	0.38	0.28	0.18	0.47	0.58	0.18	0.77	0.08	0.32	0.33
FlipOt	1024	0.32	0.16	0.16	0.10	0.69	0.54	0.12	0.74	0.08	0.33	0.31
Ensemb	1024	0.40	0.54	0.24	0.16	0.86	0.49	0.28	0.75	0.19	0.23	0.28
Base	512	0.32	0.31	0.30	0.12	0.66	0.55	0.10	0.59	0.11	0.24	0.26
MC DO	512	0.44	0.66	0.50	0.13	0.82	0.75	0.14	0.74	0.12	0.29	0.22
MC DC	512	0.41	0.30	0.39	0.11	0.91	0.47	0.25	0.79	0.18	0.55	0.18
FlipOt	512	0.37	0.33	0.24	0.13	0.78	0.68	0.11	0.76	0.10	0.46	0.14
Ensemb	512	0.43	0.43	0.30	0.22	0.68	0.67	0.20	0.93	0.22	0.24	0.37
Base	256	0.33	0.26	0.25	0.14	0.80	0.44	0.14	0.73	0.21	0.21	0.16
MC DO	256	0.34	0.22	0.41	0.18	0.59	0.60	0.15	0.52	0.20	0.29	0.24
MC DC	256	0.41	0.26	0.33	0.10	0.73	0.65	0.22	0.86	0.11	0.46	0.32
FlipOt	256	0.35	0.27	0.22	0.14	0.52	0.56	0.18	0.74	0.19	0.46	0.25
Ensemb	256	0.47	0.55	0.34	0.22	0.90	0.61	0.19	0.84	0.36	0.40	0.32
Base	128	0.46	0.31	0.32	0.20	0.71	0.92	0.17	0.95	0.23	0.39	0.36
MC DO	128	0.43	0.52	0.31	0.19	0.81	0.63	0.13	0.79	0.37	0.25	0.31
MC DC	128	0.50	0.95	0.39	0.18	0.70	0.76	0.19	0.60	0.42	0.52	0.30
FlipOt	128	0.42	0.43	0.23	0.12	0.78	0.74	0.14	0.75	0.19	0.47	0.41
Ensemb	128	0.53	0.62	0.47	0.25	0.88	0.65	0.32	0.95	0.40	0.38	0.42
Base	64	0.50	0.72	0.32	0.14	0.76	0.98	0.17	0.93	0.30	0.42	0.30
MC DO	64	0.55	1.00	0.36	0.11	0.93	0.83	0.16	1.00	0.26	0.58	0.25
MC DC	64	0.61	0.58	0.34	0.23	1.09	1.09	0.40	1.09	0.26	0.61	0.46
FlipOt	64	0.55	0.76	0.67	0.20	0.81	0.76	0.09	0.97	0.42	0.67	0.19
Ensemb	64	0.64	0.83	0.50	0.41	1.00	0.98	0.29	1.04	0.48	0.41	0.42
Base	32	0.71	1.29	0.71	0.10	1.15	0.83	0.21	1.00	0.66	0.67	0.47
MC DO	32	0.72	0.96	0.64	0.37	1.23	0.95	0.30	0.91	0.72	0.60	0.53
MC DC	32	0.72	0.78	0.65	0.46	1.16	1.03	0.28	1.18	0.38	0.69	0.62
FlipOt	32	0.70	1.10	0.46	0.34	1.13	0.86	0.45	1.09	0.48	0.55	0.56
Ensemb	32	0.78	1.01	0.60	0.49	1.18	1.05	0.48	1.14	0.61	0.64	0.61
Base	16	0.99	1.23	0.85	0.52	1.40	1.23	0.24	1.41	0.89	1.20	0.90
MC DO	16	0.98	1.23	0.92	0.54	1.27	1.26	0.47	1.39	0.88	0.97	0.90
MC DC	16	0.99	1.24	0.81	0.55	1.26	1.38	0.51	1.39	0.97	0.85	0.92
FlipOt	16	0.94	1.25	0.88	0.69	1.09	1.29	0.66	1.28	0.75	0.62	0.92
Ensemb	16	1.02	1.25	0.90	0.63	1.30	1.34	0.50	1.34	0.79	1.04	1.06

Table B.2: The Entropy results of five different models (Base, MC DropOut, MC DropConnect, FlipOt, and Deep Ensemble) evaluated on the dataset, Section 2.1 across nine downsampling levels. Each column after the "Sample" column represents the entropy for the specified category in the label.

Class	4096	2048	1024	512	256	128	64	32	16
bathtub	1.12	1.128	0.932	0.904	0.926	1.114	2.168	2.26	3.406
bed	0.382	0.346	0.274	0.596	0.52	0.484	0.548	0.93	1.414
chair	0.138	0.114	0.132	0.132	0.216	0.174	0.314	0.356	0.822
desk	1.432	1.294	1.408	1.654	1.568	1.838	2.08	2.614	3.016
dresser	1.398	0.924	0.908	1.166	1.126	1.47	1.58	1.688	2.45
monitor	0.274	0.31	0.354	0.226	0.318	0.318	0.462	0.698	0.91
night_stand	1.356	1.678	1.692	1.422	1.396	1.352	1.674	1.796	2.35
sofa	0.36	0.288	0.282	0.292	0.39	0.48	0.478	0.776	1.03
table	0.69	0.818	0.79	0.664	0.792	0.904	0.802	0.818	1.26
toilet	0.304	0.376	0.458	0.328	0.262	0.55	0.45	1.014	2.21
Total Model Avg.	0.744	**0.726**	**0.722**	0.738	0.752	0.868	1.054	1.292	1.888

Table B.3: Performance Averages per Class for all Model Configuration Based on Point Sampling including Avg. model performance: NLL results of different models across downsampling levels. Each column represents the NLL for the specified sample in the label.

Class	4096	2048	1024	512	256	128	64	32	16
bathtub	0.442	0.372	0.326	0.406	0.312	0.566	0.778	1.028	1.24
bed	0.278	0.232	0.218	0.346	0.31	0.344	0.438	0.612	0.872
chair	0.122	0.118	0.106	0.142	0.156	0.188	0.218	0.352	0.586
desk	0.718	0.612	0.704	0.77	0.708	0.776	0.918	1.17	1.264
dresser	0.772	0.598	0.56	0.624	0.572	0.74	0.928	0.944	1.3
monitor	0.134	0.156	0.192	0.16	0.176	0.19	0.222	0.344	0.476
night_stand	0.772	0.754	0.776	0.762	0.738	0.808	1.006	1.064	1.362
sofa	0.224	0.172	0.146	0.146	0.214	0.322	0.344	0.57	0.856
table	0.394	0.378	0.378	0.356	0.364	0.402	0.538	0.63	0.936
toilet	0.232	0.228	0.282	0.234	0.258	0.36	0.324	0.558	0.94
Total Model Avg.	0.41	**0.36**	**0.36**	0.394	0.38	0.468	0.57	0.726	0.984

Table B.4: The average Entropy results of the five models (Base, MC DropOut, MC DropConnect, FlipOt, and Deep Ensemble) evaluated across the downsampling levels. Each column after the "Class" column represents the entropy for the specified sample in the label.