Karlsruher Institut für Technologie

Institute of Computer Engineering
Chair for Dependable Nano Computing

Single-Shot Uncertainty Estimation In Artificial Neural Networks Using Ensemble Classes

Bachelor's Thesis
from
Hakima Marouan Marouan

Supervisor: **Prof. Mehdi B. Tahoori**
Advisor: **Ph.D Soyed Tuhin Ahmed**

October 2024

I declare that I have written this thesis independently and have only used the sources and aids indicated. The passages copied verbatim or in terms of content are labelled as such. I have complied with the current version of the statutes of the Karlsruhe Institute of Technology (KIT) to ensure good scientific practice.

Karlsruhe, October 15, 2024

Contents

Acknowledgement

This work would not have been possible without the guidance, support, and encouragement of many people to whom I owe a great deal of gratitude.

First and foremost, I would like to express my deepest thanks to my supervisor, Soyed. Your insightful guidance, invaluable ideas, and belief in my abilities have been instrumental in the completion of this project. You provided me with the opportunity to pursue this research and supported me through every stage of it. For your unwavering patience and encouragement, I am deeply grateful.

I also want to extend my heartfelt appreciation to my family, my parents, my sister, and my brothers. Your constant love, support, and belief in me gave me the strength to push forward even when I doubted myself. You were always there to remind me that I could achieve this, and for that, I am forever thankful.

A special thank you goes to my dear friends, Nora and Belén. You both were my pillars when I felt like giving up. Your constant reassurance, encouragement, and belief in me, even when I was at my lowest, made all the difference. I cannot thank you enough for reminding me to keep going, even when it seemed impossible.

Abstract

Uncertainty estimation is a critical factor in the deployment of artificial neural networks (ANNs) in high-stakes applications such as autonomous driving and medical diagnostics, where the ability to quantify model confidence is as essential as predictive accuracy. Traditional ensemble-based uncertainty estimation methods, such as Deep Ensembles and Monte Carlo Dropout, have demonstrated efficacy in improving model robustness and uncertainty calibration. However, these methods typically require multiple forward passes, resulting in increased latency and energy consumption, making them impractical for real-time, resource-constrained environments. This thesis introduces the Class Ensemble method, which addresses these limitations by ensembling neurons within the final layer of the network while sharing a common feature extraction backbone. This novel approach enables single-shot uncertainty estimation, significantly reducing computational overhead without sacrificing performance. The Class Ensemble method was evaluated across various tasks, including classification (MNIST, CIFAR-10), semantic segmentation (U-Net on CamVid and Pascal VOC), and regression (UCI Housing dataset), and was benchmarked against traditional methods. Experimental results indicate that the Class Ensemble technique achieves competitive accuracy while offering superior calibration, robustness under noisy conditions, and efficiency in terms of memory and energy consumption. These findings suggest that the Class Ensemble method is a viable solution for uncertainty estimation in resource-limited settings, facilitating its deployment in real-time applications such as edge computing and autonomous systems.

List of Figures

List of Tables

1 Introduction

As machine learning has advanced significantly in real-time applications such as autonomous systems, Internet of Things (IoT) devices, and mobile platforms, there has been an increasing need for implementing machine learning models in recent years. Even with the advancements, there are still significant limitations because of the limited resources of memory, energy, and computing power. These restrictions make it extremely difficult to apply sophisticated machine learning models, especially when estimating uncertainty is at stake. Uncertainty estimation becomes especially important in high-stakes areas where prediction confidence is as important as accuracy, for instance, autonomous driving and medical diagnostics. Although conventional techniques for estimating uncertainty, such Batch Ensemble, Branch Ensemble, and Deep Ensemble, demonstrate good predictive performance, they require numerous model evaluations and forward passes, which raise latency and energy consumption - factors that are unreasonable for edge devices.

This thesis introduces the Class Ensemble method, an innovative approach designed to address the challenges of conventional ensemble-based uncertainty estimation techniques, while preserving their benefits. Unlike traditional methods that rely on multiple independent models or branches to estimate uncertainty, the Class Ensemble approach focuses on ensembling neurons in the final layer of a neural network. The backbone layers of the network are shared across all ensemble members, enabling single-shot training and inference. This architecture significantly reduces computational overhead and improves energy efficiency, making it a suitable solution for environments with limited resources.

The proposed method builds on prior work in both uncertainty estimation and efficient neural network ensembling. While it has been demonstrated that methods like Monte Carlo (MC) Dropout and Deep Ensemble improve model precision and uncertainty estimation, their dependency on several forward passes during inference makes them unsuitable for real-time applications. The Class Ensemble technique aims to maintain high levels of accuracy and reliable uncertainty estimate even in the face of resource limitations by minimising memory usage and forward pass count.

Advances in medical imaging and autonomous driving are only two of the many fields in which Artificial Neural Networks (ANNs) have produced impressive results recently. Although accuracy enhancement is the primary focus of many researchers, it is becoming increasingly clear that the capacity to measure uncertainty in model predictions is crucial. Forecasting accurate uncertainty estimate is crucial to enhancing the adaptability and reliability of machine learning models in applications where decisions based on model outputs entail substantial risks, like autonomous systems or healthcare.

This thesis will conduct a detailed evaluation of the Class Ensemble approach versus a number of well-known uncertainty estimation techniques. This approach will be evaluated not just for baseline accuracy but also for its capacity to produce useful uncertainty estimates in limited contexts. While accuracy is still a crucial parameter for machine learning models, real-world applications frequently need the ability to assess prediction uncertainty, especially in scenarios with class imbalance, data scarcity, or out-of-distribution data. Since there isn't a single, best

method for estimating uncertainty, a thorough analysis and comparison of many strategies are required.

This research aims to close the existing gap in robust uncertainty estimation methods for neural networks, especially in domains where safety is crucial. Even though many models can be highly accurate, they usually fall short in terms of offering meaningfully confident forecasts. This vulnerability compromises the robustness and reliability of these models by lacking efficient uncertainty estimating processes, which is particularly concerning in scenarios where mistakes can have fatal repercussions.

1.1 Background on Uncertainty Estimation in Neural Networks

Uncertainty estimation is an essential component of neural networks used in high-stakes applications like autonomous driving, medical diagnostics, and robotics. In these fields, model predictions are used to make critical decisions, and uncertainty estimation provides a way to measure how confident the model is in its outputs(1). This is crucial when the consequences of inaccurate or unclear predictions can be severe. For example, in autonomous driving, a system must make split-second decisions in environments that may have missing, ambiguous, or noisy information. If the model is uncertain about its prediction, for instance, it cannot clearly distinguish between a pedestrian and a shadow, knowing this uncertainty can urge the vehicle to slow down or stop, preventing a potential accident.

In medical imaging, models assist doctors by analyzing scans and detecting anomalies such as tumors. While the model might be highly accurate, there are instances where predictions are uncertain due to poor image quality or cases not seen during training. In these circumstances, providing a reliable estimate of uncertainty can assist physicians in making better-informed decisions, such as recommending further tests or consultations before making a definitive diagnosis(**?**).

1.2 Challenges of Traditional Ensemble Methods

Conventional ensemble techniques, including MC Dropout and Deep Ensemble, are commonly used to estimate uncertainty in neural networks. These methods involve running multiple versions of a model or performing multiple forward passes to generate diverse predictions. By comparing the variance in these predictions, the model can estimate its uncertainty. However, these methods are computationally expensive, especially in real-time applications. They require significant computational resources due to multiple forward passes, which increases latency, energy consumption, and memory usage. This is problematic for edge devices, where available resources are typically constrained.

For instance, in autonomous driving, where real-time performance is critical, every millisecond counts. Self-driving systems must make rapid decisions based on sensor inputs. Traditional ensemble methods, which require multiple forward passes, introduce latency that could result

in delayed actions—potentially causing accidents. Similarly, in healthcare applications, medical professionals often require immediate results, and the added computational cost of traditional ensemble methods can slow down diagnosis, which could be life-threatening in emergency scenarios.

The Tesla autonomous driving system provides a real-world example of the risks associated with poor uncertainty estimation. There have been instances where Tesla's self-driving technology has failed to recognize obstacles, leading to potentially dangerous situations. For example, the system has mistakenly identified objects, such as mistaking a parked car for a trash can (2), or failing to recognize children on the road (3). Moreover, there have also been instances where the vehicle mistook a horse-drawn carriage for a truck with two people behind it (4). These mistakes are often due to the system's inability to assess the uncertainty in its predictions, leading to overconfident, but incorrect, decisions. A more effective uncertainty estimation mechanism might have flagged these predictions as uncertain, prompting the system to take more cautious actions.

Figure 1.1: Tesla mistook a horse-drawn carriage for various vehicles

Similarly, it is impossible to overstate the importance of accurate uncertainty assessment in the healthcare industry. AI is currently being used, for instance, to give medical professionals a better image so they may identify cancer in patients earlier and more accurately, thus enhancing and expediting patient treatment. However, there are issues with the current machine learning models, and medical professionals wonder how to rely on the model's prediction without necessarily seeing a formula or the process by which the computer arrived at this result (5). A model that provides a clear uncertainty estimate could advise physicians when additional testing or more conservative decision-making is needed.

These examples highlight the critical need for robust uncertainty-aware models in high-stakes applications. Unfortunately, traditional ensemble methods often struggle to meet the rigorous demands of low-latency and low-energy environments without compromising the quality of

uncertainty forecasts. These drawbacks motivate the search for alternative techniques that can deliver trustworthy uncertainty estimates without imposing unaffordable computational costs.

In addition, even while medical research has long been a significant field for AI, it has been one of the more difficult fields for AI to be effectively adopted. This is caused by a number of factors, including a hesitation towards trusting machine intelligence with making life-changing choices and new interfaces, in addition to social and cultural factors (6).

1.3 Motivation for the Class Ensemble Approach

The Class Ensemble approach was developed to address the specific needs of uncertainty estimation in resource-constrained environments like edge devices. By focusing on ensembling only the final layer of neurons in the neural network, the Class Ensemble method reduces the computational burden compared to traditional ensemble techniques. The shared backbone structure of the model ensures that most of the computations are reused across ensemble members, leading to single-shot training and inference. This not only lowers latency but also reduces energy consumption and memory usage, making the method particularly suitable for real-time and low-resource applications.

The Class Ensemble technique achieves robust uncertainty estimates without requiring several forward passes, in contrast to conventional methods that use multiple models or forward passes. By providing a solution that is consistent with edge device limitations and adaptable enough to adjust to scenarios where class imbalance or data scarcity could make predictions unreliable, it closes a significant gap in the current methods for estimating uncertainty.

1.4 Research Question and Contributions

This thesis is driven by the primary research issue of whether, as compared to traditional ensemble-based techniques, the Class Ensemble approach performs better for uncertainty estimates on edge devices. In particular, the accuracy, computational efficiency, and uncertainty estimation of the Class Ensemble approach are the main goals of this work. Metrics like Expected Calibration Error (ECE) and the ability to handle out-of-distribution data and noisy inputs will be used to assess the performance of the proposed approach.

The primary contribution of this thesis is the development and evaluation of the Class Ensemble method, an uncertainty estimation approach that provides a balance between computational efficiency and prediction reliability. This approach could enable the deployment of machine learning models in real-world scenarios where standard uncertainty estimating strategies are not feasible because of resource constraints. By reducing the need for multiple forward passes, the Class Ensemble method offers a promising solution for low-latency, low-energy, and memory-efficient uncertainty estimation.

2 Theoretical Fundamentals

Modern machine learning rely significantly on ensemble methods, especially where the aim is to enhance model robustness and provide a reliable estimate of uncertainty in predictions. Ensemble techniques reduce the error and uncertainty inherent in individual models, resulting in more accurate predictions by combining forecasts from multiple models. Traditional ensemble approaches, however, have high computational, memory, and energy usage even with their success. This is especially restrictive in contexts with limited resources where power consumption and latency are important factors to take into account. In particular, for neural networks that need real-time uncertainty estimation, the Class Ensemble approach has shown to be an effective alternative. Class Ensemble offers a balance between uncertainty estimation performance and computational efficiency. The theoretical principles of ensemble methods, uncertainty estimation, and the computational trade-offs associated with latency, energy consumption, and memory usage are thoroughly reviewed in this section.

Neural Networks: Understanding the Basics

An artificial neural network (ANN) or neural network (NN) is a key component in machine learning, designed to mimic the structure and function of neurons in the human nervous system (7).

An ANN consists of an interconnected group of units or nodes called artificial neurons (neurons in a brain) connected by edges (like synapses in a brain). Each artificial neuron receives signals from connected neurons, processes them and sends signals to other connected neurons. Before sending the signal, each neuron applies a non-linear function to the sum of its inputs, known as the activation function. The strength of the signal at each connection between neurons is determined by a weight.

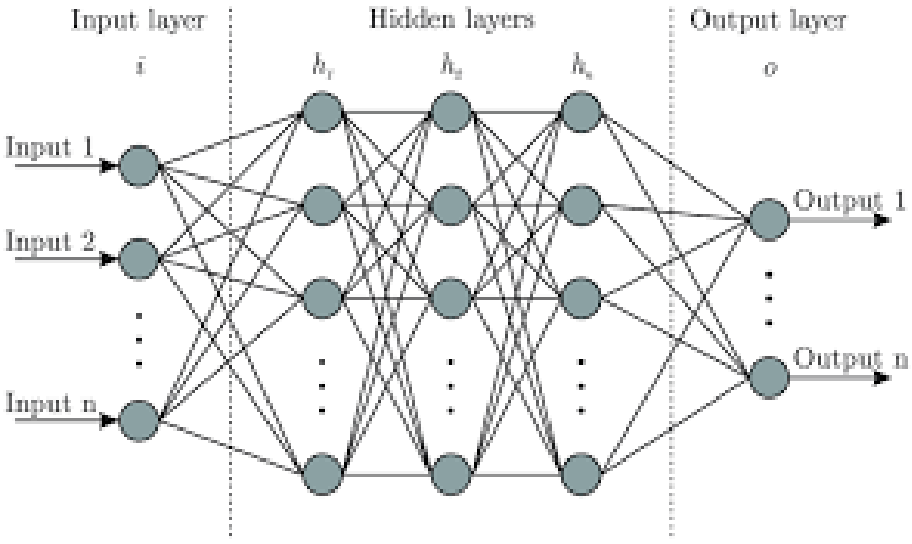

Figure 2.1: Example of artificial neural network architecture (8)

- **Input Layer:** This layer shown in Figure 2.1 receives the raw data. For example, in image recognition, the input might be pixel values of an image.

- **Hidden Layers:** These layers perform complex transformations. The neurons in each layer are connected to those in the next layer, and they apply weights, which are values that get adjusted during learning, to input data.
- **Output Layer:** The final layer produces predictions. For example, if we are classifying images of cats and dogs, the output layer might have two neurons: one for cats and one for dogs.

The main objective of a neural network is to learn the correct weights in such a way that the outputs match the target labels, for instance, correctly identifying whether an image contains a cat or a dog. This is done through a process called training, where the model iteratively adjusts its weights based on the error, meaning difference between its predictions and the actual labels.

In recent years, ANNs (Artificial Neural Networks) have gained popularity because they can learn from experience and come to conclusions from a complex and unrelated set of information. Examples of their use include predictive modelling, adaptive control and problem-solving in artificial intelligence.

2.1 Uncertainty Estimation in Machine Learning

In many machine learning applications, where models need to demonstrate a degree of confidence in their predictions, uncertainty estimation is essential(9). This is particularly true in domains where decision-making depends on models that function in unpredictable ways.

Uncertainty in machine learning can generally be classified into two types:

- **Epistemic Uncertainty:** This type of uncertainty results from a lack of knowledge about the model or its parameters. It arises when there is not enough training data or when the model comes across unknown data. Enhancing the architecture of the model or using more data to train it can both help to lower epistemic uncertainty. This is captured by techniques such as Deep Ensemble and Bayesian neural networks (BNN), which take into account the variation in predictions made by various models or parameter configurations (10).
- **Aleatoric Uncertainty:** This particular type of uncertainty comes from noise or ambiguity in the input and is inherent in the data. Not even with more training data can it be decreased. For instance, even an ideal model may have trouble predicting what would happen in medical images with blurry images or low-quality scans. Aleatoric uncertainty is captured directly from the data and persists regardless of the model's exposure to large datasets.

The robustness and reliability of machine learning models are improved by accurately estimating both types of uncertainty, especially in safety-critical applications where uncertainty can inform adaptive decision-making(11) . An example would be when driving in unclear situations, such as bad weather, an autonomous vehicle operating under epistemic uncertainty might choose to give control to a human driver, while an autonomous vehicle operating under aleatoric uncertainty might decide to slow down for safety.

The Problem of Data Scarcity
In many real-world cases, there is not enough data to train reliable models. This is especially problematic in applications like healthcare, where obtaining labeled data is costly and time-consuming.

In some datasets, certain classes, such as images of rare diseases, are underrepresented, which makes it difficult to train a model that can confidently identify those conditions. Therefore, traditional models may perform poorly on these underrepresented classes because they don't receive enough examples to learn from. To address these challenges, ensemble methods can help improve predictions in the face of data scarcity. By combining multiple models or neurons, ensembles can leverage different aspects of the data, making the overall prediction more robust.

2.1.1 Mathematical Foundations of Uncertainty Estimation

Variance
The variance of predictions measures the spread of predictions over multiple models in an ensemble, or across different stochastic forward passes (for instance, Monte Carlo Dropout). For a set of models M in an ensemble, let $(p_m(y|x)$ be the predicted probability from model m, and let $\overline{p}(y|x))$ be the mean prediction across all models. The formula for variance is:

$$U(x) = \text{Var}(p(y|x)) = \frac{1}{M}\sum_{m=1}^{M}(p_m(y|x) - \overline{p}(y|x))^2 (12)$$

where

- $\overline{p_m}(y|x)$ is the probabilistic prediction for the outcome y given the input x from model m,
- $\frac{1}{M}\sum_{m=1}^{M}(p_m(y|x)$ is the average prediction from all models.

Variance gives an estimate of **epistemic uncertainty**, as it measures how much the predictions differ across models.

Entropy
In classification tasks, another useful measure is **entropy**, which measures the uncertainty within a single model's probability distribution over all possible classes. Given the probability distribution $p(y|x)$ for a classification problem with C classes, the entropy of the prediction can be expressed as:

$$H(p) = -\sum_{c=1}^{C} p(y=c|x)\log p(y=c|x) (13)$$

where

- $p(y=c|x)$ is the predicted probability for a class c given input x,
- C is the total number of classes.

Entropy reflects **aleatoric uncertainty**, capturing how confident the model is about its predicted probabilities from different classes. A higher entropy means the model assigns similar probabilities to multiple classes, indicating uncertainty.

2.2 Traditional Ensemble Methods in Machine Learning

Ensemble methods in machine learning are powerful techniques that combine the predictions from multiple models to create a more robust and accurate system(7). These methods can significantly improve a model's performance, particularly when dealing with uncertain or imbalanced data. However, they also come with challenges, especially in terms of computational resources, which is why they are not always suitable for environments with limited hardware like edge devices.

The main idea behind ensemble methods is that combining multiple models can help reduce the variance and bias in predictions. Each individual model might have certain weaknesses or make errors due to the training data or its architecture. By combining several models, these individual errors tend to cancel each other out, leading to more accurate predictions overall. Additionally, ensemble methods provide a way to estimate uncertainty, which is crucial in many real-world applications.

Let's dive deeper into the most common types of traditional ensemble methods: Deep Ensemble, Branch Ensemble, and Batch Ensemble, along with Monte Carlo Dropout.

2.2.1 Deep Ensemble

Deep Ensembles imply the independent training of several neural networks, each initialized differently. During inference, each model makes a prediction, and these predictions are averaged or combined by majority vote. An indicator of epistemic uncertainty is the variance between model predictions(14). Deep Ensembles tend to be good at estimating uncertainty, but come with a significant overhead: a large amount of memory, energy, and computational resources are needed for each model since each needs to be trained, stored, and inferred independently.

Energy and Latency Considerations

Deep Ensemble training and deployment involve substantial resource consumption. Each model must be trained independently, and during inference, multiple forward passes are required - one for each model. This causes a rise in latency and energy usage, particularly in situations where real-time decision-making is essential. For instance, in the event that an ensemble consists of N models, the inference time grows linearly as follows:

$$T_{\text{ensemble}} = N \times T_{\text{single}}$$

where T_{single} is the time taken by a single model. Similarly, the energy consumption, E_{ensemble}, is proportional to the number of models:

$$E_{\text{ensemble}} = N \times E_{\text{single}}$$

where E_{single} is the energy required by one model. This makes Deep Ensembles less suitable for applications where latency and power consumption are critical. Moreover, while this improves accuracy, it is computationally expensive since each model must be trained and stored separately.

2.2.2 Branch Ensemble

Branch Ensemble trains a single model with multiple branches, where each branch is responsible for generating predictions. Although Branch Ensemble can reduce the number of model parameters in comparison to Deep Ensemble, it still requires multiple forward passes through distinct branches, which increases the overall inference time.

The model shares the initial layers — referred to as the backbone — while branching into separate networks near the final layers. This design mitigates redundancy by using a common feature extraction backbone; however, it still demands additional memory and computational overhead for each individual branch, limiting the efficiency gains.

2.2.3 Batch Ensemble

Batch Ensemble share a neural network backbone across ensemble members, leading to a reduction in computational overhead compared to fully independent models. To introduce diversity in predictions, specific layers of the network are subjected with small, trainable adjustments. This approach reduces memory and parameter costs while maintaining model diversity. However, it still requires multiple forward passes, increasing the inference time and making it less practical for real-time applications. As the number of branches or perturbations grows, so does the energy consumption.

Rather than training entirely separate models, Batch Ensemble processes multiple ensemble members simultaneously during training, which moderately reduces computational overhead. Nevertheless, it takes executing multiple parallel predictions during inference, limiting the efficiency gains.

2.2.4 Monte Carlo Dropout (MC Dropout)

Monte Carlo (MC) Dropout is a technique that uses dropout during both training and inference to approximate Bayesian inference in neural networks. In standard dropout, during training, random subsets of neurons are "dropped out" by setting their activations to zero, which prevents

overfitting by introducing noise (15). However, in MC Dropout, this process is also applied during inference, allowing for stochasticity in the forward passes(16).

By performing multiple stochastic forward passes through the network, each with different dropout configurations, the model is able to generate diverse predictions for the same input. This enables the model to estimate epistemic uncertainty by analyzing the variability in these predictions. The greater the spread of the predictions, the more uncertain the model is about the output.

The approach offers a computationally cheaper alternative to Deep Ensemble, which require training multiple distinct models. While Deep Ensemble tend to offer higher accuracy and better uncertainty estimates, they are significantly more resource-intensive. On the other hand, MC Dropout only requires multiple forward passes through a single model. However, the need for multiple stochastic passes during inference still introduces additional latency, as each input must be evaluated several times to estimate uncertainty.

2.3 Our Approach: Class Ensemble Method

The Class Ensemble approach provides an alternative for the computational inefficiencies of conventional ensemble methods by focusing on the ensembling of the last layer. This design allows all ensemble members to share a common backbone of layers responsible for feature extraction. The technique can produce reliable uncertainty estimates with little resource consumption since it dramatically reduces both the number of model parameters and computing complexity by limiting ensembling to the final layer.

2.3.1 Architecture of Class Ensemble Method

In the Class Ensemble technique, neurons corresponding to various ensemble members are assigned in the final layer of the neural network. Instead of training multiple entire models, the ensemble is created by averaging the outputs of these neurons. The final prediction is generated by averaging the outputs of these neurons.

With this approach, maintaining distinct models requires fewer computations and fewer forward passes. Compared to traditional ensembles, the number of trainable parameters is substantially fewer. This also leads to reduced memory usage and inference time:

$$T_{\text{class ensemble}} \approx T_{\text{single}}$$

$$E_{\text{class ensemble}} \approx E_{\text{single}}$$

The Class Ensemble Method is a new approach specifically designed to overcome the limitations of traditional ensemble methods when deploying on edge devices. Instead of training multiple full models or branches, this method focuses on ensembling only the final layer of neurons for each class. This significantly reduces the computational and memory overhead.

Architecture of Class Ensemble

- **Shared Backbone:** The convolutional layers are shared among all ensemble members. These layers are responsible for extracting features from the input data, such as edges and textures in an image.
- **Ensembled Final Layer:** Instead of having separate branches for each ensemble member, the Class Ensemble method focuses on the neurons in the final layer. Each class has multiple neurons, and these neurons are ensembled to produce a final prediction for each class.

For example, if there are 10 classes in the dataset, and the ensemble size is 5, then there would be 5 neurons assigned to each class, totaling 50 neurons in the final layer. The model prediction is the average of the outputs of the neurons for each class, providing a robust estimate of the final prediction.

Advantages

- **Single-Shot Training and Inference:** Unlike traditional ensembles that require multiple forward passes, the Class Ensemble method allows for a single forward pass through the shared backbone, making it efficient in terms of time and resources.
- **Adaptability:** The number of neurons can be adjusted based on the specific task requirements. For example, if some classes are underrepresented, more neurons can be allocated to those classes to improve their performance.
- **Uncertainty Estimation:** Since multiple neurons are used for each class, we can estimate the uncertainty of the model's prediction by looking at the variance between these neurons. If all neurons produce similar outputs, the model is confident; if the neurons give varied outputs, the model is uncertain.

Mathematically, the uncertainty can be calculated as the variance:

$$Variance(y_c) = \frac{1}{N}) \sum_{i=1}^{N} (f_{final}(x)_c^i - (\hat{y}_c\ )^2$$

$f_{final}(x)_c^i$ is the output of the i-th neuron for c_i and $\hat{y}_c$ is the mean output. This variance provides a direct measure of how confident the model is in its prediction

Edge devices, like smartphones or IoT sensors, have limited computing power and energy resources. Traditional ensemble methods, which require multiple models or forward passes, are not suitable for such devices due to their high resource consumption.

Key Benefits

- **Reduced Latency**: The Class Ensemble method requires only one forward pass through the model, significantly reducing latency compared to methods like MC Dropout or Deep Ensemble, which require multiple forward passes.
- **Lower Energy and Memory Consumption**: By sharing the backbone and only ensembling the final layer, the memory and computational resources required are much smaller. This makes the method suitable for deployment on devices with tight memory constraints.

2.3.2 Latency, Energy, and Memory Efficiency

The Class Ensemble approach significantly increases the efficiency of inference in terms of latency and energy consumption by merging the backbone layers. To ensemble the outputs at the final layer, the approach simply has to perform a single forward pass across the shared layers and very little further computation. Consequently, compared to typical ensembles, the energy and memory consumption are significantly lower. The quantity of parameters in the final layer's neurons and backbone layers determines the memory footprint. Let P_{backbone} represent the shared backbone parameters and P_{final} represent the final layer parameters. For a Class Ensemble model, there are the following total parameters:

$$P_{\text{total}} = P_{\text{backbone}} + P_{\text{final}}$$

Since P_{final} is significantly smaller than in traditional ensemble models, the memory savings are substantial. Furthermore, the method is well-suited for devices with limited memory and power, as it avoids the need for storing multiple models.

2.3.3 Uncertainty Estimation in Class Ensemble

The variation of the neuron outputs in the final layer is an indicator of uncertainty in the Class Ensemble. The neurons corresponding to each class work together as an ensemble, and their output variance provides an estimate of uncertainty. The method captures epistemic uncertainty without the computational burden of several forward passes or whole model ensembles by ensembling at the last layer. Further statistical techniques that can improve uncertainty estimate are entropy and Negative Log-Likelihood (NLL) (17).

3 Objectives

3.1 Previous Investigations

In recent years, neural networks have made enormous advances and become crucial in various real-world applications. As mentioned in the theoretical fundamentals, other approaches to uncertainty estimation include Bayesian methods and ensemble techniques. Bayesian methods are theoretically robust but computationally impractical for large-scale applications. Ensemble techniques have shown promise but also come with high computational costs and complexity due to the need to train multiple models and collect their predictions. In contrast, single-shot estimation methods have appeared as a practical alternative, offering a balance between computational efficiency and reliable uncertainty metrics. We will now discuss several influential studies from the last decade that have significantly made a profound impact on this topic.

Yarin Gal (18) provided a theoretical framework for understanding uncertainty and demonstrated that dropout techniques could be used to approximate Bayesian inference in neural networks. Roger Ghanem's (19) handbook expanded on various concepts of uncertainty quantification without explicitly focusing on neural networks. Alex Guy Kendall's (20) thesis improved model performance and reliability by explaining Bayesian neural networks, particularly the Monte Carlo Dropout approach, and demonstrated practical applications of uncertainty estimation in computer vision tasks. Andrey Malinin and Mark Gales (21) introduced innovative approaches for improving uncertainty and adversarial robustness through reverse KL-divergence training of prior networks, addressing practical challenges in robust model training. Hao Wang and Dit-Yan Yeung (22) provided a thorough survey on Bayesian deep learning, presenting a general framework and Bayesian approaches for uncertainty quantification, particularly in recommender systems, topic models and control. Balaji Lakshminarayanan (23) introduced ensemble methods that improved predictive performance and uncertainty estimation but required substantial computational resources. Niclas Stahl et al. (24) evaluated various methods of uncertainty quantification in deep learning, emphasizing the need for scalable and reliable approaches in practical applications. Fredrik K. Gustafsson, Martin Danelljan and Thomas B. Schön (25) tested the robustness required in real-world computer vision tasks and emphasized the effectiveness and limitations of two popular methods, MC Dropout and Ensembles. Eyke Hüllermeier and Willem Waegeman (26) distinguished between aleatoric and epistemic uncertainty in neural networks, discussing different concepts to model and quantify them. Finally, Moloud Abdar et al. (27) reviewed uncertainty quantification techniques, applications, and challenges in deep learning, highlighting the range of ongoing research and the need for continued innovation in this area.

These previous investigations highlight a key trade-off in uncertainty estimation: methods that offer precise uncertainty metrics tend to be computationally expensive, while more efficient techniques often compromise accuracy. The goal of this thesis is to further explore and refine single-shot uncertainty estimation approaches, with a specific focus on a method that leverages ensemble classes to improve computational efficiency and maintain robust uncertainty estimates.

3.2 Justification for the Research

Although uncertainty quantification has advanced, there is still no universally acknowledged method that strikes an optimal balance between computational efficiency, simplicity, and accuracy. Conventional techniques like Deep Ensembles and Bayesian methods are effective but are frequently computationally costly, especially when using models on devices with limited hardware resources. While single-shot methods provide a more efficient option by estimating uncertainty in a single inference pass, they still need to be further refined to match the robustness of traditional methods. By concentrating on **ensemble class-based single-shot uncertainty estimation**, this research attempts to address these gaps. The objective is to experiment on a method that is accurate, easily implementable, and computationally efficient approach in generating uncertainty metrics. This is especially crucial in high-stakes domains like healthcare, where decision-making often relies on both the accuracy and confidence of model predictions. An accurate uncertainty estimate can guide doctors in diagnosing patients or allocating resources, especially in cases where the model's predictions may be less reliable.

The thesis aspires to produce uncertainty quantification techniques that are both theoretically and practical for real-world deployment, especially in resource-constrained environments such as mobile and edge computing platforms (28).

3.3 Objectives of the Thesis

This thesis focuses on ensemble class approaches with the goal of improving **single-shot uncertainty estimation methods** in ANNs. This involves comparing the proposed method with other traditional uncertainty estimation techniques and assessing its impact on real-world applications where reliable uncertainty quantification is essential. The specific objectives can be divided into theoretical and practical components.

3.3.1 Theoretical Objectives

- **Study the fundamental principles of deep learning and uncertainty estimation:** A comprehensive review of neural networks, deep learning algorithms, and various uncertainty estimation techniques will be conducted, with a focus on the mathematical foundations of Bayesian approaches, ensemble methods, and single-shot methods.
- **Investigate single-shot uncertainty estimation methods:** This will involve researching existing single-shot methods, such as MC Dropout, and evaluating their performance in comparison to ensemble-based techniques.
- **Compare uncertainty estimation methods:** A detailed comparison will be made between the studied ensemble methods, with a focus on the trade-offs between computational complexity, accuracy, and uncertainty estimation.

For each method, the following will be analyzed:

- **Spatial and time complexities:** Analyze the computational efficiency and memory requirements of each approach.
- **Effectiveness of uncertainty metrics:** Compare metrics such as **variance**, **Negative Log-Likelihood (NLL)**, and **entropy** to evaluate how well each method quantifies uncertainty.
- **Advantages and disadvantages:** Research the strengths and weaknesses of each method, particularly in the context of real-world applications.

3.3.2 Practical Objectives

- **Implementation and Experimentation:** The proposed methods will be implemented using the PyTorch framework. The following techniques will be experimented with to improve model performance:
  - Experiment with different activation functions, optimization algorithms (e.g., adding momentum to Stochastic Gradient Descent), and neural network architectures.
  - Adjust hyperparameters such as the number of layers, batch size, and number of epochs to optimize training.
- **Dataset Evaluation:** The methods will be applied to both synthetic and real-world datasets. Specifically, performance will be evaluated on imbalanced datasets to test the robustness of the uncertainty estimates.

3.3.3 Sub-Objectives

To achieve the overarching goals, the following sub-objectives are defined:

- **Develop the Class Ensemble Method:** Design and implement an ensemble class approach that combines multiple neurons within the final layer of a neural network while sharing the underlying layers. This method should enable single-shot training and inference, reducing computational costs while maintaining robust uncertainty estimates.
- **Compare with Traditional Ensemble Methods:** Implement traditional ensemble methods (for example, deep ensembles, batch ensembles, and branch ensembles) and compare them with the proposed class ensemble method in terms of accuracy, computational cost, memory usage, and latency.
- **Uncertainty Estimation Evaluation:** Evaluate the ability of the class ensemble method to distinguish between in-distribution and out-of-distribution data. Metrics such as **variance**, **NLL**, and **prediction entropy** will be used to assess the quality of the uncertainty estimates.
- **Address Data Scarcity Problems:** Test the proposed method on low-data environments using datasets from the UCI Machine Learning Repository. Evaluate its potential to handle underrepresented classes.

- **Optimization:** Analyze the memory, energy consumption, and latency of the class ensemble method to determine its suitability for deployment on devices with limited computational resources.

3.3.4 Requirements

The following requirements are expected to be satisfied to ensure the quality and success of the research:

- The selected single-shot uncertainty estimation methods must strike a balance between accuracy, computational efficiency, and ease of implementation.
- Code optimization should focus on computational efficiency, readability, and minimizing complexity.
- Real-world datasets, particularly in biomedical fields, will be used to test the effectiveness of uncertainty estimation methods, particularly on imbalanced data.
- A thorough evaluation process will be established to validate the improvements over baseline models, including cross-validation techniques to ensure robust performance.

By meeting these requirements, the thesis aims to provide significant contributions to the field of uncertainty estimation in neural networks, with practical applications in both academic research and industrial deployment.

4 Methodology

The methodological approach for this thesis is based on a hybrid project management framework, integrating Kanban and Scrum principles. The development of the models and experimentation with ensemble methods for uncertainty estimation will follow a well-defined and adaptable workflow. Additionally, the project will be guided by a strong theoretical foundation, practical implementation tasks, and a detailed risk management plan. Each task is carefully planned with dependencies on other tasks, ensuring a structured flow of development.

4.1 Project Management Framework

Kanban and Scrum Methodology After reviewing various project management methodologies such as Agile, Scrum, Kanban, DevOps, and Waterfall, the Kanban and Scrum hybrid approach was chosen for its flexibility and adaptability. **Kanban** is particularly suited to this project due to its visual dashboard system, representing tasks in stages like "To Do", "In Progress", and "Done". This methodology facilitates the management of evolving requirements and frequent changes, providing an adaptable workflow with the ability to limit work-in-progress tasks and quickly identify bottlenecks. The visual nature of Kanban supports the development of machine learning models, which often involve iterative changes and testing.

To complement Kanban, **Scrum** is incorporated by organizing work into shorter sprints. These sprints will help maintain focus on defined goals while allowing flexibility for mid-sprint adjustments, enabling frequent reviews and refinements of models and methodologies.

4.2 Project Planning and Task Structure

Acquisition of Knowledge and Scope Definition A crucial first step in this project is acquiring the necessary theoretical and practical knowledge of artificial neural networks (ANNs), particularly focusing on **uncertainty estimation** methods. This includes:

- A thorough review of **deep learning techniques** applicable to the research.
- Specific focus on **single-shot uncertainty estimation** methods and how they apply to neural networks.
- Comparative research on different **ensemble techniques**, and their limitations in uncertainty estimation.

The scope of the project will be defined based on these initial findings, outlining the objectives and detailing the specific challenges posed by implementing ensemble methods for uncertainty estimation.

Theoretical Analysis of Methods The project will then take a deeper look at the theoretical complexity of various uncertainty estimation methods. This part of the methodology will include:

- **Time and space complexity analysis** of the proposed methods to understand their computational constraints.
- **Evaluation of uncertainty metrics**, comparing methods such as **predictive entropy**, **variance**, **negative log-likelihood**... to determine the best measures of model uncertainty.
- Identification of **advantages and disadvantages** of each method.

4.3 Model Setup and Datasets

The practical implementation of this thesis will require setting up models tailored to specific datasets. The following sections describe the models and datasets used:

4.3.1 Datasets

MNIST
The MNIST (Modified National Institute of Standards and Technology) dataset is one of the most widely used datasets in the field of machine learning and computer vision. It comprises a collection of 70,000 grayscale images of handwritten digits (0-9), each of which is 28x28 pixels in size, an example of the values can be seen in Figure 4.1. The dataset is divided into 60,000 training samples and 10,000 test samples, allowing for effective training and evaluation of models. Each image is labeled with the corresponding digit, providing a supervised learning context. The simplicity and uniformity of the MNIST dataset make it an excellent benchmark for evaluating image classification algorithms, particularly convolutional neural networks (CNNs).

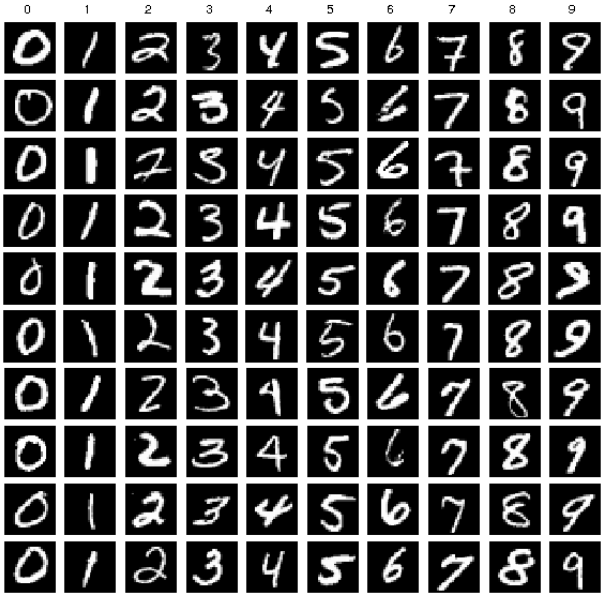

Figure 4.1: MNIST Dataset (29)

CIFAR-10
The CIFAR-10 dataset is another popular dataset for image classification tasks, containing

60,000 32x32 color images categorized into 10 distinct classes. An example of images for this

Figure 4.2: CIFAR-10 Dataset (30)

dataset is above in Figure 4.2, where ten distinct classes can be seen, airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, monkeys, and trucks.

Each class contains 6,000 images, with 50,000 images designated for training and 10,000 for testing. The CIFAR-10 dataset poses a more complex challenge compared to MNIST due to its increased image size, color variation, and diversity of objects, making it suitable for evaluating the robustness and generalization of classification algorithms.

DRIVE Dataset (IRIS)

The Digital Retinal Images for Vessel Extraction (DRIVE) dataset specifically focuses on the analysis of retinal images, particularly for the task of detecting blood vessels in the retina.

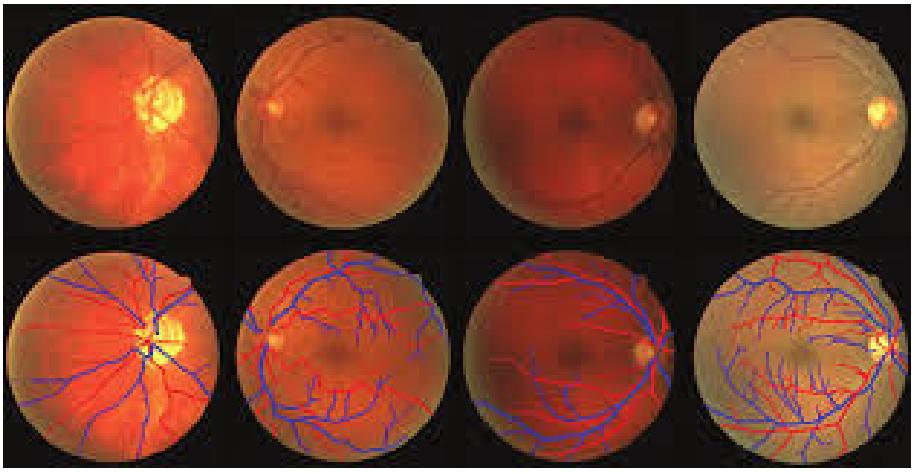

Figure 4.3: DRIVE Dataset on Retinal Images (31)

The dataset comprises 40 color images of retinal fundus images, each with a resolution of 768x584 pixels, obtained from diabetic patients. These images are annotated, highlighting the

blood vessels for training segmentation models. The DRIVE dataset is particularly useful for tasks related to semantic segmentation, where the goal is to classify each pixel in an image. The different images that are included in this dataset are similar to Figure 4.3.

4.3.2 Models

MNIST Model

For classifying the MNIST dataset, a basic neural network model can be constructed using fully connected layers or a convolutional neural network (CNN). A typical architecture might include an input layer, one or more convolutional layers followed by pooling layers, and one or more fully connected layers leading to an output layer that predicts the digit class. The CNN architecture is particularly effective for image classification tasks due to its ability to automatically learn hierarchical features from the input images. An example of how MNIST model can give a prediction from an input is shown below in Figure 4.4.

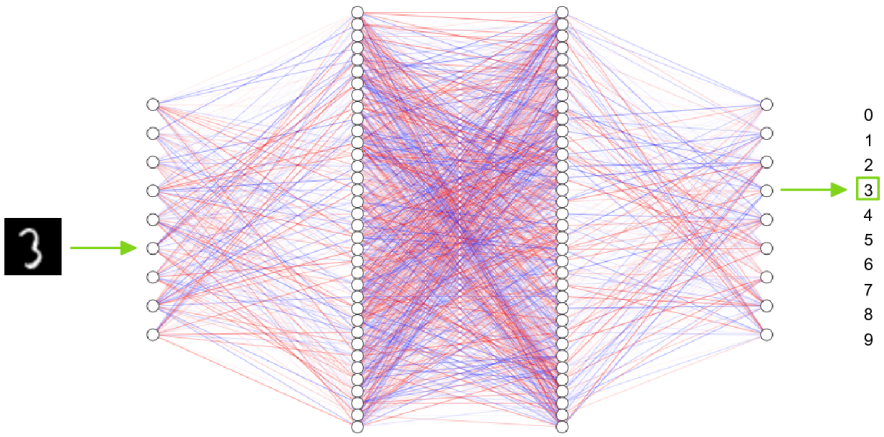

Figure 4.4: Example of how MNIST model works with an input (32)

ResNet-18 Model

An example of a ResNet-18 (Residual Network) model can be seen in Figure 4.5. It is a specific

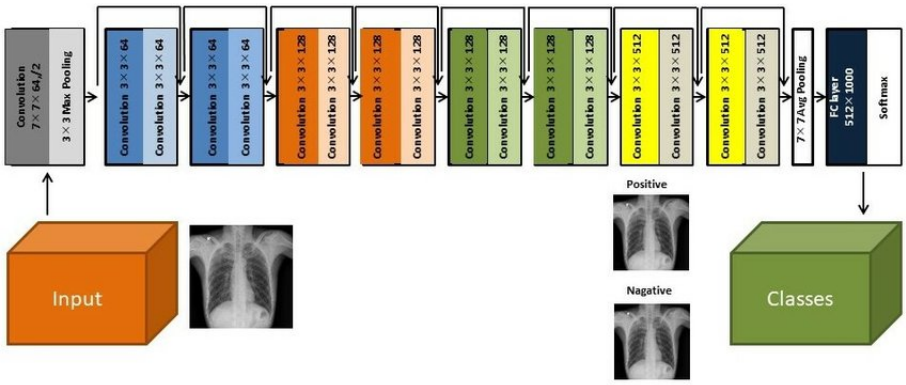

Figure 4.5: ResNet-18 Model (33)

architecture within the broader family of deep convolutional neural networks. It is characterized by its use of residual blocks, which facilitate the training of very deep networks by allowing gradients to flow through the network more effectively. The ResNet-18 architecture consists of 18 layers, including convolutional layers, batch normalization layers, and ReLU activation functions, along with skip connections that bypass one or more layers. This architecture has demonstrated superior performance on various image classification tasks, including those involving more complex datasets like CIFAR-10.

Moreover, it should be indicated that over the past few years, batch normalization has become an integral part of most – if not all – cutting edge deep networks. In that sense, training a deep network using batch normalization is equivalent to approximate in- ference in Bayesian models(**?**).

U-Net Model for Semantic Segmentation

The U-Net architecture is specifically designed for semantic segmentation tasks, particularly in biomedical image analysis. The U-Net model employs a symmetric encoder-decoder structure.

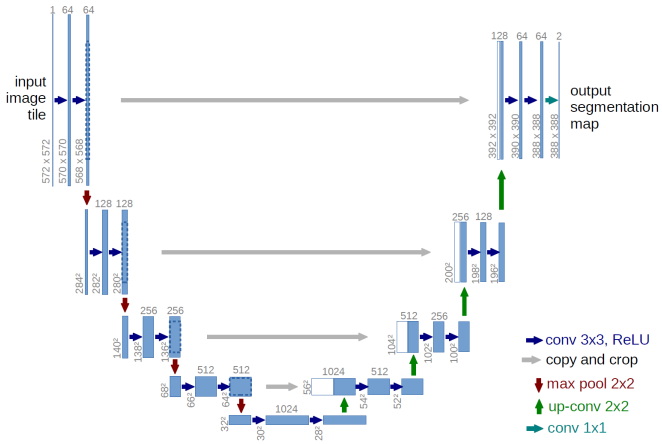

Figure 4.6: Example of how U-Net model works with an input (34)

The encoder progressively captures contextual information through downsampling (via convolutional and pooling layers) as shown in Figure 4.6, while the decoder reconstructs the image resolution through upsampling (via transposed convolutions). U-Net utilizes skip connections that concatenate features from the encoder to the corresponding decoder layers, allowing for better localization of features. This architecture is particularly effective for the DRIVE dataset, as it can accurately segment blood vessels from retinal images.

Regression Task

In addition to classification and segmentation, regression tasks are essential in machine learning, where the objective is to predict continuous values. For example, a regression model could be applied to predict the severity of a disease based on retinal images or other relevant features extracted from the dataset. A common approach for regression tasks involves using models such as linear regression, decision trees, or more complex architectures like fully connected neural networks or CNNs, depending on the nature of the data and the desired outcomes. The choice of loss function, such as Mean Squared Error (MSE), is crucial for effectively training the re-

gression model. In Figure 4.7, it shows the difference between the regression and classification. However, both are crucial to quantify the model's confidence in its predictions, and therefore estimate uncertainty.

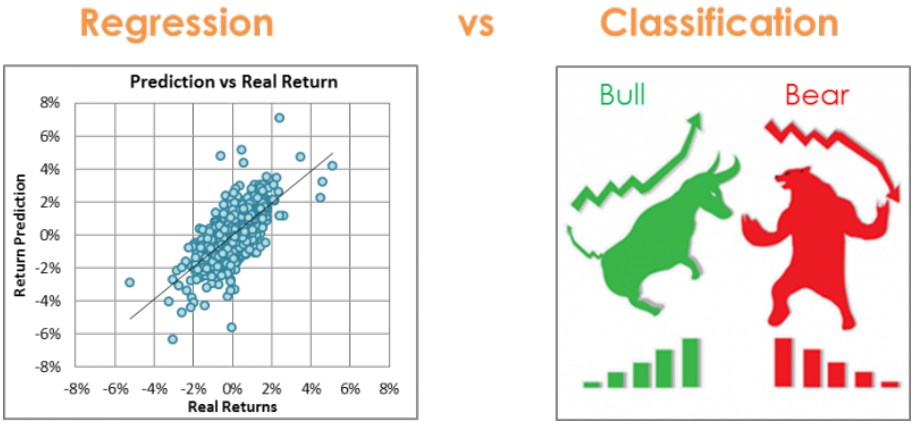

Figure 4.7: Difference between Regression and Classification (35)

Classification involves predicting a discrete label or category from a set of predefined classes. For example, predicting whether an image contains a cat or a dog is a classification task. Uncertainty in this case will help determine how confident the model is in assigning a particular class to an input. For example, a model may predict an image is a dog with 60% confidence, indicating some uncertainty.

Regression involves predicting a continuous value, such as predicting house prices based on various features (for instance, size, location). The output is a numerical value that can take any value within a range. Unlike in classification, uncertainty in regression provides a range, for instance, a confidence interval, around the predicted value. For example, predicting that a house price is \$300,000 ± \$20,000 gives a clearer picture of the model's certainty in the prediction.

4.4 Implementation of Machine Learning Methods

Ensemble methods are a powerful class of machine learning techniques that combine multiple models to improve prediction accuracy and robustness. These methods are particularly useful in tasks where individual models may struggle with generalization or exhibit significant variance in their predictions. This section provides a detailed explanation of various ensemble methods implemented using deep learning architectures, including **Deep Ensemble**, **Branch Ensemble**, **Batch Ensemble**, **MC Dropout**, and **Class Ensemble**.

4.4.1 Deep Ensemble

A **Deep Ensemble** is a straightforward but highly effective ensemble technique, which has the 4.8 structure. It involves training multiple independent neural networks and averaging their

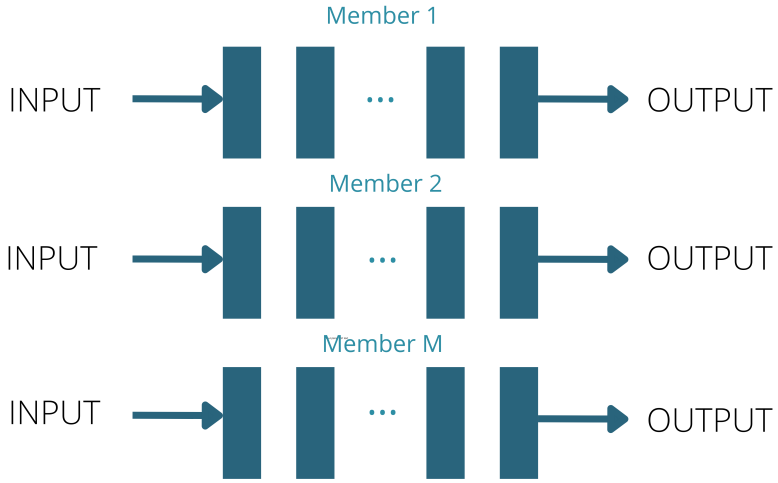

Figure 4.8: Deep Ensemble Structure

predictions to produce the final output. Each network in the ensemble is initialized with different random weights, and potentially trained on different data, or the same data but with stochastic variations introduced by random initialization and training procedures. The intuition behind deep ensembles is that independently trained models will make diverse errors, and averaging their outputs can reduce the overall prediction variance and improve performance.

Each model in the deep ensemble is trained independently. During inference, the outputs of the models are averaged to produce the final prediction. This simple averaging reduces variance and improves predictive performance because each model learns different aspects of the data due to random initialization and training variations.

To achieve good learning results, the data distribution in the testing dataset should be as close as in the training datasets. However, in many situations, the test datasets' distributions are unknown, especially in cases of uncertainty prediction problems, which we are going to experiment with. Moreover, when this approach is employed in large networks, it becomes computationally expensive(**?**).

4.4.2 Branch Ensemble

The **Branch Ensemble** method introduces multiple branches within a single neural network, as shown in Figure 4.9. Each branch independently processes the input data but shares the same initial feature extraction layers (backbone). This approach allows the model to generate diverse predictions from different branches while benefiting from shared feature representations, which reduces computational overhead compared to training independent models as in deep ensembles.

In the Branch Ensemble method, each branch makes a prediction independently. At inference, the predictions from all branches are added, typically by averaging or majority voting. The shared backbone reduces memory usage, while the multiple branches introduce diversity, which enhances the model's generalization capabilities.

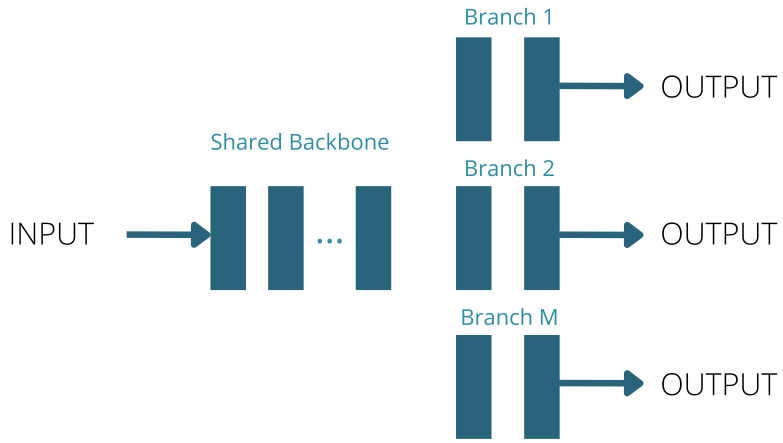

Figure 4.9: Branch Ensemble Structure

4.4.3 Batch Ensemble

The **Batch Ensemble** approach takes advantage of batch processing by training multiple models simultaneously within a single forward pass. It introduces additional parameters for each ensemble member but shares most of the network's core components, which can be seen in Figure 4.10. This method significantly reduces the computational cost of ensemble learning while still keeping many of the benefits of model diversity.

Figure 4.10: Batch Ensemble Structure

The Batch Ensemble method processes multiple ensemble members in parallel by adding an ensemble dimension to the batch. The shared weights reduce memory consumption, while the individual parameters for each ensemble member ensure diversity in predictions. The model produces ensemble predictions at the same computational cost as a single model in many cases.

4.4.4 MC Dropout

MC Dropout (Monte Carlo Dropout) introduces stochasticity into the model by applying dropout during both training and inference. Dropout randomly sets a fraction of the neurons to zero, effectively creating a diverse ensemble of subnetworks. During inference, multiple forward passes are performed, with different dropout patterns, to estimate the model's uncertainty.

During inference, MC Dropout involves performing multiple forward passes with dropout enabled. Each pass produces a slightly different prediction due to the randomness introduced by dropout (**?**). The final prediction is obtained by averaging these multiple predictions, and uncertainty can be estimated by calculating the variance across the predictions.

4.4.5 Class Ensemble

The **Class Ensemble** approach introduces diversity by ensembling neurons only in the final classification layer, while keeping the feature extraction layers (backbone) shared among ensemble members, an idea of how it looks is in Figure 4.11. The number of neurons will be equal to the number of classes per the ensemble size. This method efficiently combines the benefits of model ensembling with reduced computational and memory overhead.

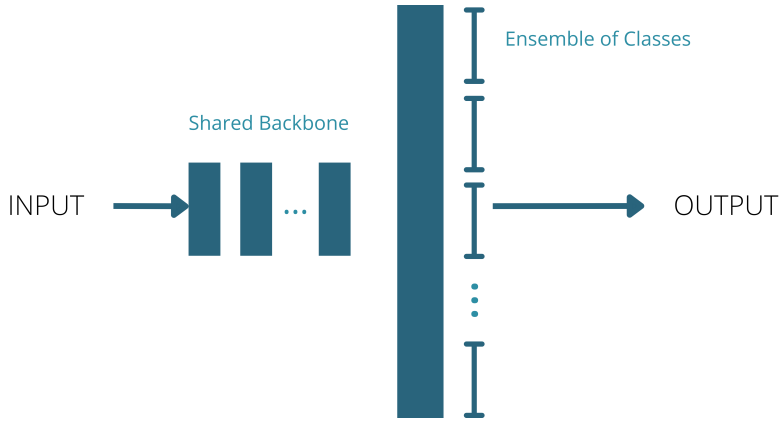

Figure 4.11: Class Ensemble Structure

In the Class Ensemble method, the final classifier layer is replicated multiple times, with each version making an independent prediction. The shared backbone reduces memory usage while ensuring that the diverse classifiers provide robust predictions. This method is especially useful for estimating uncertainty, as the variation among the classifiers provides an estimate of the model's confidence in its predictions.

4.5 Evaluation Metrics for Classification, Segmentation, and Regression Tasks with Uncertainty Estimation

In machine learning, evaluation metrics are crucial for analyzing the performance, accuracy, and uncertainty of models. For tasks involving classification, segmentation, and regression, especially when incorporating uncertainty estimation, a set of sophisticated metrics like **Negative Log-Likelihood (NLL)**, **Brier Score**, **Entropy**, **Expected Calibration Error (ECE)**, **Maximum Calibration Error (MCE)**, and **Root Mean Square Calibration Error (RMSCE)** are used to assess both the model's accuracy and the quality of uncertainty estimation. In addition, **loss** and **Dice Score** are key for semantic segmentation, while **Log-Likelihood (LL)** and **Root Mean Squared Error (RMSCE)** are central for regression tasks.

In the following section, we will explain why we chose these metrics for different tasks under noisy conditions, and how they help in analyzing both accuracy and uncertainty.

4.5.1 Classification Metrics with Uncertainty Estimation

In classification tasks, we are concerned with both **accuracy**, how well the model predicts the correct class, and **calibration**, how well the predicted probabilities align with the true probabilities. When adding Gaussian noise to the dataset, uncertainty estimation becomes critical. The following metrics are commonly used to evaluate both predictive performance and uncertainty:

Negative Log-Likelihood (NLL)

- *Definition.* NLL measures how likely the model's predictions are under the true distribution of the labels(16). It is formally given by:

$$\text{NLL} = -\sum_{i=1}^{N} \log P(y_i|x_i, \theta)$$

  where $P(y_i|x_i, \theta)$ is the predicted probability of the true label y_i for input x_i under the model parameters θ.

- *Purpose.* NLL evaluates the quality of the probabilistic predictions by penalizing low-confidence correct predictions and wrong predictions with high confidence. It is sensitive to both the accuracy and the uncertainty of predictions. Lower NLL indicates better performance and more reliable uncertainty estimation.

Brier Score

- *Definition.* The Brier Score quantifies the mean squared difference between the predicted probabilities and the one-hot encoded true labels:

$$\text{Brier Score} = \frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K}(p_{ik} - y_{ik})^2$$

  where p_{ik} is the predicted probability of class k for instance i, and y_{ik} is 1 if the true class is k and 0 otherwise.

- *Purpose.* The Brier Score measures both the accuracy and calibration of the probabilistic predictions(11). A lower Brier Score indicates that the predictions are both accurate and well-calibrated, meaning the predicted probabilities closely match the true distribution.

Entropy

- *Definition.* Entropy measures the amount of uncertainty or disorder in the predicted probability distribution:

$$\text{Entropy} = -\sum_{k=1}^{K} p_k \log(p_k)$$

  where p_k is the predicted probability of class k.

- *Purpose.* Higher entropy indicates higher uncertainty, while lower entropy indicates more confident predictions. Entropy is used to quantify the uncertainty inherent in the predictions, particularly under noisy conditions.

Expected Calibration Error (ECE)

- *Definition.* ECE measures the difference between a model's confidence in its predictions and its actual accuracy. It is computed by binning the predictions into intervals of confidence and comparing the average confidence to the accuracy within each bin:

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} |\text{acc}(B_m) - \text{conf}(B_m)|$$

  where B_m is the set of samples in the m-th bin, $\text{acc}(B_m)$ is the accuracy, and $\text{conf}(B_m)$ is the average confidence in that bin.

- *Purpose.* ECE provides a scalar summary of how well-calibrated the model is. Lower ECE indicates that the predicted probabilities are more aligned with the true probabilities, meaning the model's confidence reflects its actual performance.

Maximum Calibration Error (MCE)

- *Definition.* MCE is similar to ECE but focuses on the worst-case calibration error across all bins:

$$\text{MCE} = \max_m |\text{acc}(B_m) - \text{conf}(B_m)|$$

- *Purpose.* MCE highlights the largest discrepancy between confidence and accuracy, emphasizing the worst-calibrated predictions. Lower MCE indicates that even the most miscalibrated predictions are not too far from the true accuracy.

Root Mean Square Calibration Error (RMSCE)

- *Definition.* RMSCE is the root mean square of the differences between confidence and accuracy across all bins:

$$\text{RMSCE} = \sqrt{\frac{1}{M} \sum_{m=1}^{M} (\text{acc}(B_m) - \text{conf}(B_m))^2}$$

- *Purpose.* RMSCE provides a more sensitive measure of calibration than ECE by giving greater weight to larger discrepancies between confidence and accuracy. This helps identify models that might be poorly calibrated even if their average calibration (ECE) is relatively low.

4.5.2 Semantic Segmentation Metrics

Semantic segmentation tasks involve assigning a class label to each pixel in an image, making accuracy and calibration metrics essential. However, the focus shifts to metrics that evaluate pixel-wise accuracy and the quality of the segmentation boundaries. In addition to **NLL**, **Brier Score**, and **Entropy**, the following metrics are crucial:

Loss Function

- *Purpose.* The choice of loss function (for instance, cross-entropy loss or Gaussian loss) directly impacts the model's learning process. In segmentation, the loss typically evaluates pixel-wise classification accuracy, encouraging the model to correctly classify each pixel while also incorporating class imbalance and other factors.

Dice Score

- *Definition.* The Dice Score (or Dice Coefficient) measures the overlap between predicted segmentation masks and the ground truth:

$$\text{Dice Score} = \frac{2 \cdot |A \cap B|}{|A| + |B|}$$

  where A is the predicted mask and B is the ground truth mask.

- *Purpose.* Dice Score evaluates how well the predicted segmentation aligns with the ground truth. It is especially useful in applications like medical imaging, where pixel-wise accuracy may not reflect the quality of segmentation boundaries, particularly in small or irregularly shaped regions.

Correctness Map

- *Purpose.* Visualizing the correctness map helps understand which areas of the segmentation are correctly classified and which are not. This can reveal patterns in model performance, such as failure cases in certain regions, under certain conditions, or for specific classes.

Uncertainty Map

- *Purpose.* The uncertainty map visualizes the model's confidence in its pixel-wise predictions. This is especially valuable when Gaussian noise is introduced, as the map can highlight areas of the image where the model is less certain about its predictions, indicating where additional data or model refinement may be necessary.

Prediction Visualization

- *Purpose.* Visualization of the predictions provides a qualitative evaluation of the segmentation performance, allowing for easier interpretation of the model's outputs compared to the ground truth. This can complement quantitative metrics like Dice Score by providing insight into how the model handles specific challenges in the data.

4.5.3 Regression Task Metrics with Uncertainty Estimation

In regression tasks, the goal is to predict continuous values rather than discrete classes. When noise is added to the dataset, both **accuracy** and **uncertainty** in predictions are evaluated. The following metrics are essential for regression:

Log-Likelihood (LL)

- *Definition.* Log-Likelihood measures how likely the observed data is under the predictive distribution of the model. In regression, assuming Gaussian noise, it is given by:

$$LL = -\frac{1}{2} \sum_{i=1}^{N} \left[\frac{(y_i - \hat{y}_i)^2}{\sigma_i^2} + \log(2\pi\sigma_i^2) \right]$$

  where $\hat{y}_i$ is the predicted value and σ_i is the predicted uncertainty (standard deviation) for the prediction.

- *Purpose.* Log-Likelihood evaluates both the accuracy of the predictions and the model's estimation of uncertainty. A higher log likelihood indicates that the model is making accurate predictions while correctly estimating the uncertainty.

Root Mean Squared Error (RMSCE)

- *Definition.* RMSCE measures the average squared difference between the predicted and true values:

$$\text{RMSCE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

- *Purpose.* RMSCE is a standard metric for regression accuracy, penalizing larger errors more than smaller ones. It does not account for uncertainty, so when used alongside metrics like Log Likelihood, it provides a more comprehensive view of model performance by focusing on the accuracy of the point predictions.

NLL, Brier Score, and Log-Likelihood directly measure the model's ability to predict accurate probabilities or values, penalizing wrong predictions with high confidence. These metrics ensure that the model's uncertainty estimates are meaningful and aligned with the true underlying distribution. In the other hand, Entropy, ECE, MCE, and RMSCE focus on calibration, assessing whether the model's confidence matches its accuracy. These metrics are crucial in applications where reliable uncertainty estimates are required, such as in safety-critical domains. In the semantic segmentation's case, Dice Score is tailored for tasks with pixel-wise predictions, ensuring that segmentation boundaries are accurately predicted and regions are well-classified. To evaluate the regression accuracy, we use the RMSCE as a standard metric, while Log-Likelihood ensures that the model not only predicts accurately but also estimates uncertainty effectively.

When Gaussian noise is introduced into the dataset, uncertainty metrics become particularly important for understanding how the model reacts to noisy data and for identifying areas where the model is unsure, enabling better decision-making, improved robustness, and more reliable predictions.

The combination of these metrics provides a robust study for evaluating both the accuracy and uncertainty of models across classification, segmentation, and regression tasks. This allows us to ensure that the Class Ensemble method not only performs well but also offers reliable estimates of uncertainty, making it suitable for deployment in real-world scenarios, particularly when noise or uncertainty is present in the data.

4.6 Uncertainty Estimation

The objective of uncertainty estimation is to quantify the model's confidence in its predictions, with particular emphasis on distinguishing between in-distribution (ID) data, which the model has seen during training, and out-of-distribution (OoD) data, which is substantially different from the training data. In this analysis, we use **predictive entropy** and **Jensen-Shannon Divergence (JSD)** to evaluate uncertainty.

Predictive entropy is used to assess the average uncertainty in a prediction. Entropy measures the amount of uncertainty in the probability distribution over classes. Higher entropy values correspond to higher uncertainty in predictions. This measure captures both **aleatoric uncertainty**, which is inherent noise or ambiguity in the data, and **epistemic uncertainty**, which is caused from uncertainty in the model parameters.

To further investigate uncertainty, we use **Jensen-Shannon Divergence (JSD)**, which quantifies the diversity or disagreement between the predictions of an ensemble of models. High JSD values indicate a greater disagreement among the ensemble members, signifying model uncertainty due to epistemic factors.

For this experiment, a ResNet-18 model was trained on the CIFAR-10 dataset, representing the in-distribution data. The SVHN dataset, characterized by visual features significantly different from CIFAR-10, was used as the out-of-distribution dataset.

Predictive entropy evaluates uncertainty by considering the softmax output probabilities. It reflects the combined aleatoric and epistemic uncertainties, providing a general estimate of how uncertain the model is about its predictions. In contrast, JSD measures the disagreement between independent models in an ensemble, primarily capturing epistemic uncertainty. This divergence is particularly useful in scenarios where the models may lack knowledge about the data and thus disagree. JSD is computed across the entire network, from feature extraction to final predictions. Higher values indicate greater epistemic uncertainty, especially when the models in the ensemble produce different outputs.

When using a Class Ensemble approach, the ensemble members share the same feature representations but vary in their final layer outputs. This means that the ensemble's disagreement is limited to the final layers of the network. In cases where inputs are ambiguous or near decision boundaries, ensembling predictions from the final layer can reveal epistemic uncertainty. Even slight variations in the final layer may be sufficient to identify OoD inputs, providing useful signals for uncertainty detection.

In this study, predictive entropy serves to identify general uncertainty in predictions, while JSD pinpoints uncertainty due to model disagreement, especially relevant for OoD detection. These metrics help assess the model's ability to differentiate between in-distribution and out-of-distribution data. In practical applications, uncertainty estimation can enhance model robustness by enabling the model to flag potentially uncertain predictions, such as OoD inputs. This ability is particularly important in safety-critical domains, where uncertain predictions can be deferred to human experts or other decision-making systems to avoid serious consequences.

4.7 Robustness against Class Imbalance and Data Scarcity

4.7.1 Class Imbalance

This experiment aims to evaluate the performance of a ResNet-18 model using the Class Ensemble approach, specifically focusing on how it manages class imbalance. The model has been modified to incorporate uncertainty quantification alongside an imbalanced dataset.

Class imbalance occurs when certain classes are overrepresented while others are underrepresented in the dataset. This often results in biased learning, where the model becomes more accurate on majority classes and underperforms on minority classes. In this study, we use an imbalanced version of the CIFAR-10 dataset, comprising 41,000 training samples. The imbalance is introduced by reducing the number of samples in specific classes according to predefined ratios.

The Class Ensemble method introduces diversity into the model's predictions by averaging the predictions of multiple models. This reduces overfitting to majority classes, thereby improving generalization. The ensemble mitigates the bias introduced by class imbalance by creating multiple models, each with varying perspectives on the data. Additionally, uncertainty estimates guides the model to pay closer attention to samples where the predictions are uncertain. This is particularly beneficial for minority classes, as the model can focus on ambiguous or hard-to-classify samples with greater care.

Ensemble models are well-known for enhancing robustness, especially in scenarios with noise or imbalance. By averaging multiple predictions, ensembles reduce variance and lead to more stable and generalizable predictions. In this experiment, the ensemble helps to counteract class imbalance by reducing the influence of imbalanced class distributions, leading to more equitable performance across all classes.

Uncertainty quantification plays a critical role in this setting. In the presence of class imbalance, the model may become overconfident in its predictions for majority classes, leading to biased results. By measuring uncertainty, we can assess the model's confidence and identify situations where the model might be uncertain, providing valuable insights into potential failure points. This is especially important in real-world applications where incorrect high-confidence predictions can lead to significant negative consequences.

4.7.2 Data Scarcity and Its Impact on Model Training

In deep learning, large amounts of labeled data are typically necessary to achieve strong generalization performance. However, in many real-world scenarios, data collection is expensive, time-consuming, or restricted due to privacy concerns, resulting in **data scarcity**. This condition arises when the available training data is limited in quantity, often leading to challenges in model performance, stability, and generalization.

In this experiment, we simulate data scarcity by reducing the size of the training set to 40% of the original CIFAR-10 dataset. Furthermore, we introduce class imbalance to further stress-test the model under difficult learning conditions. For instance, classes 0 and 1 are significantly reduced to only 10% of their original size, mimicking situations where some classes are severely underrepresented.

The scarcity of data impacts model training by increasing the likelihood of overfitting, reducing generalization, and decreasing performance on unseen data. Ensemble methods are particularly valuable in data-scarce environments, as they reduce overfitting by incorporating diverse hypotheses from different models.

4.8 Pixel Accuracy and Intersection over Union for CamVid and Pascal VOC Datasets

The aim of this experiment is to assess the performance of a U-Net model using a Class Ensemble approach for semantic segmentation tasks. The focus is on how ensemble size and the introduction of Gaussian noise influence key metrics such as pixel accuracy and Intersection over Union (IoU) across two datasets: Pascal VOC and CamVid.

In ensemble learning, multiple models (in this case, U-Net variants) are trained and combined to improve prediction accuracy and robustness. The logic behind using ensemble methods in semantic segmentation is that averaging predictions from multiple models can reduce variance and mitigate overfitting. This is particularly beneficial for tasks that involve pixel-wise classification, which can be sensitive to noise and uncertainty (36).

To further improve robustness, Gaussian noise is introduced to the input data during training. This noise acts as a form of regularization, forcing the model to learn more stable features by exposing it to variations in the input data. As a result, the model is expected to generalize better to unseen data and improve performance on key metrics like pixel accuracy and IoU.

Dataset Characteristics

The Pascal VOC dataset is a challenging benchmark for semantic segmentation (as shown in Figure 4.12), containing 20 object classes with highly diverse and cluttered scenes. CamVid, on the other hand, is a simpler dataset featuring urban driving scenes with a more limited set of classes, such as road, vehicles, and pedestrians.

In our experiments, we use ensemble sizes of 1, 5, and 10. Preliminary results indicated that ensembles larger than 10 led to significantly diminished performance, suggesting a trade-off between ensemble size and model generalization.

The expectation is that both ensemble learning and the addition of Gaussian noise will improve model generalization and lead to better performance across the Pascal VOC and CamVid datasets, as reflected in improvements in pixel accuracy and IoU.

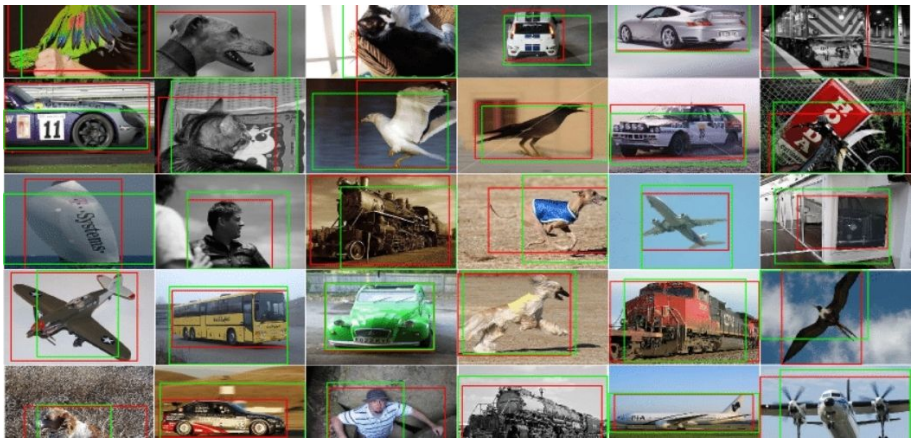

Figure 4.12: Example of Samples in Pascal VOC Dataset (37)

5 Results

This chapter presents a comprehensive analysis of the performance and robustness of various ensemble learning techniques applied across multiple tasks, including classification, semantic segmentation, and regression. The primary focus is on evaluating these methods' effectiveness in terms of accuracy, robustness, computational efficiency, and uncertainty estimation under different conditions. The experiments include a detailed assessment of the computational aspects, such as latency, number of parameters, and energy consumption, which are crucial for real-world deployment in resource-constrained environments.

Specifically, we investigate the performance of ensemble methods across several deep learning architectures and datasets. The results are organized into distinct sections, addressing classification performance on ResNet-18 with MNIST, robustness under noisy conditions, semantic segmentation performance using U-Net, and regression tasks on the UCI dataset. Additionally, special attention is given to evaluating the uncertainty estimation capabilities of the models, and their robustness against class imbalance and data scarcity, particularly in segmentation tasks.

5.1 Classification Results (ResNet-18 and MNIST)

In this section, we explore the classification performance of ResNet-18 and MNIST model, using various ensemble methods to assess their impact on predictive accuracy, computational efficiency, and robustness. The ensemble methods evaluated include Batch Ensemble, Branch Ensemble, Class Ensemble, Deep Ensemble, and MC Dropout. The key metrics used for evaluation include latency (time per batch), the number of parameters, and energy consumption, providing a holistic view of each method's computational footprint and suitability for real-time or resource-constrained applications.

The analysis also investigates the balance between prediction accuracy and computational demands, highlighting the trade-offs between model complexity, speed, and energy efficiency. These insights are crucial for understanding the practical applicability of ensemble methods in classification tasks where computational resources and inference time are critical considerations.

5.1.1 MNIST Model

The results presented in Table 5.1 summarize the performance of various ensemble methods on the MNIST dataset, evaluated using key metrics such as accuracy, Negative Log-Likelihood (NLL), Brier score, entropy, Expected Calibration Error (ECE), Maximum Calibration Error (MCE), and Root Mean Squared Error (RMSCE). These metrics help assess not only the models' predictive performance but also their ability to estimate uncertainty reliably. The models were evaluated on the original MNIST dataset without the addition of noise, providing a baseline comparison of their accuracy and uncertainty estimation capabilities.

Regarding accuracy, the Deep Ensemble method achieved the best performance with an accuracy of 99.44%, closely followed by MC Dropout (99.33%) and Class Ensemble (99.31%). Despite the slight differences in accuracy, the Class Ensemble stands out for its lower computational complexity, making it an efficient alternative to the Deep Ensemble method. Unlike Deep Ensemble, which requires training and inference across multiple independent models, the Class Ensemble reduces computational overhead without significantly sacrificing accuracy, offering a more practical solution in resource-constrained settings.

In terms of model calibration, as indicated by the ECE scores, the Class Ensemble achieved a competitive ECE score of 0.005, comparable to other methods like Batch Ensemble, Branch Ensemble, and MC Dropout. Meanwhile, Deep Ensemble demonstrated the lowest ECE (0.002), marginally outperforming the others. This demonstrates that the Class Ensemble, while maintaining high accuracy, also provides reliable uncertainty estimates with minimal calibration error, indicating strong model robustness and efficiency.

Ensemble Method	**Accuracy**	**NLL**	**Brier**	**Entropy**	**ECE**	**MCE**	**RMSCE**
Batch Ensemble	98.35	1.488	0.026	0.068	0.005	0.338	0.016
Branch Ensemble	99.280	1.469	0.012	0.006	0.005	0.287	0.021
Class Ensemble	99.310	4.315	0.012	0.007	0.005	0.631	0.026
Deep Ensemble	99.440	1.469	0.009	0.014	0.002	0.557	0.010
MC Dropout	99.330	1.470	0.012	0.011	0.005	0.438	0.025

Table 5.1: Performance Metrics on MNIST Model for different Ensemble Methods

These findings suggest that the Class Ensemble method strikes a favorable balance between accuracy, uncertainty estimation, and computational efficiency, making it a viable alternative to more resource-intensive approaches like Deep Ensemble, particularly when computational resources are limited.

The table 5.2 compares the performance metrics of various ensemble methods applied to the MNIST model, evaluated based on three key factors: latency (in seconds per batch), number of parameters, and energy consumption (in joules).

For the Deep Ensemble, we used five independently trained models. This method achieves a relatively low latency of 0.59208 seconds per batch, moderate parameter count, and lower energy consumption compared to other ensemble methods, with 0.039041 joules consumed per batch. The balance between model diversity and computational efficiency makes it a strong choice.

The Monte Carlo Dropout approach, which relies on 50 forward passes during inference, shows

Ensemble Method	Latency (s/batch)	Number of parameters	Energy Consumption (J)
Batch Ensemble	7.65841	16242954	0.0649718
Branch Ensemble	1.13558	3690760	0.0147630
Class Ensemble	1.14914	3690760	0.0147630
Deep Ensemble	0.59208	9760250	0.039041
MC Dropout	61.852	3666250	0.73325

Table 5.2: Performance Metrics on MNIST Model for different Ensemble Methods

a significant increase in latency, with 61.852 seconds per batch. This is the highest among all methods, reflecting the computational overhead of performing multiple stochastic forward passes to approximate Bayesian inference. Additionally, its energy consumption is also the highest at 0.73325 joules, though it uses fewer parameters than Batch Ensemble. Despite these costs, MC Dropout can capture model uncertainty effectively, making it useful in applications where uncertainty quantification is crucial.

While the Batch Ensemble method offers the potential for more diverse model predictions due to its larger model size (16.2 million parameters), it comes at a substantial cost in terms of both latency and energy consumption coming as the second-highest enery consumption at 0.0649718 joules. This method may be better suited for applications where model accuracy and diversity are prioritized over computational efficiency considering that it has a significant high latency (7.65841 seconds per batch).

Both Branch Ensemble and Class Ensemble demonstrate similar performance metrics, especially in terms of latency (around 1.13-1.15 seconds per batch), parameter count (approximately 3.7 million), and energy consumption (0.0147630 joules per batch). This is a direct result of sharing backbone layers, which reduces the overall computational and energy cost. These methods offer a good balance between predictive diversity and efficiency, making them suitable for scenarios where maintaining moderate resource use is important without sacrificing model performance.

In conclusion, the Class Ensemble method emerges as the most computationally and energy-efficient approach, given its low latency, reduced parameter count, and minimal energy consumption, while still achieving satisfactory accuracy and calibration (ECE). On the other hand, Batch Ensemble offers the most diversity at the cost of high latency and energy consumption.

5.1.2 ResNet-18 Model

We trained the ResNet-18 model with Class Ensemble approach while increasing the number of ensembles and we got the results displayed in Table 5.3. We normalized the results of each ensemble size dividing it by its number of ensemble members.

The results demonstrate that increasing the ensemble size improves model accuracy to a certain

extent, but at the cost of higher latency and marginally increased energy consumption. Memory usage remains relatively constant, indicating that this approach is scalable in terms of memory efficiency. The observed fluctuations in accuracy with different ensemble sizes suggest that beyond a certain point, the benefit of adding more models to the ensemble may result in marginal performance decreases, potentially due to overfitting or noise in the data.

Ensemble Size	**Normalized Latency (s)**	**Normalized Memory Usage (MB)**	**Normalized Energy Consumption (J)**
1	0.00754	638.45	0.046958
5	0.001746	127.84	0.009556
10	0.001172	63.841	0.0048805
20	0.000817	31.928	0.002254285

Table 5.3: Normalized Performance Metrics for Ensemble Size of 1

Table 5.4 presents performance metrics for various ensemble methods applied to a ResNet-18 model, specifically focusing on latency (seconds per batch), the number of parameters, and energy consumption (Joules). Each ensemble method represents a distinct architectural approach that influences the computational load and efficiency.

Ensemble Method	**Latency (s/batch)**	**Number of parameters**	**Energy Consumption (J)**
Batch Ensemble	3.11613	11173962	0.0446958
Branch Ensemble	16.09926	55869810	0.223479
Class Ensemble	2.77740	11271432	0.0450857
Deep Ensemble	17.78750	55869810	0.223479
MC Dropout	2.88271	11173962	0.0446958

Table 5.4: Performance Metrics on ResNet-18 Model for different Ensemble Methods

The results underscore the trade-offs between model diversity and computational efficiency across different ensemble methods.

The Branch Ensemble method has significantly higher latency, number of parameters, and energy consumption compared to Batch Ensemble. This is due to the fact that the model branches out into different classifiers after shared layers. Each branch introduces additional parameters, resulting in a higher memory footprint and computational load. The increased complexity directly impacts the energy consumption and latency, making it less efficient for real-time applications but potentially beneficial for tasks that demand greater model diversity.

Deep Ensemble involves training multiple independent models and averaging their outputs. This method incurs the highest latency and energy consumption due to the complete indepen-

dence of each model in the ensemble. With each model having a full set of parameters, the total parameter count is significantly higher than in methods with shared parameters like Batch or Class Ensemble. While this independence can provide robustness and diversity in predictions, the trade-off in computational efficiency is substantial, making it less feasible for time-sensitive applications.

The estimation of energy consumption correlates with the number of parameters and the computational complexity of the methods. Branch Ensemble and Deep Ensemble, which involve training and running multiple independent models, consume significantly more energy. In contrast, Batch Ensemble and MC Dropout maintain low energy consumption due to parameter sharing or repeated inference without increasing the model's parameter count.

The Class Ensemble method demonstrates the lowest latency among the ensemble methods, with an energy consumption similar to that of Batch Ensemble and MC Dropout. The low latency is due to the shared backbone structure, where the majority of the network is reused for each class, and only the neurons in the final layer are ensembled. This architecture reduces computational redundancy and allows for efficient feature extraction with minimal increase in parameters. The small rise in the number of parameters compared to Batch Ensemble reflects the additional neurons specific to the class ensemble but remains relatively efficient. Deep Ensemble and Branch Ensemble are computationally expensive, leading to high latency and energy usage, which can be a bottleneck for real-time applications.

Overall, if the priority to choose an ensemble method is low latency and energy efficiency, methods like Class Ensemble and Batch Ensemble are ideal, especially when computational resources are limited.

Ensemble Method	**Accuracy**	**NLL**	**Brier**	**Entropy**	**ECE**	**MCE**	**RMSCE**
Batch Ensemble	87.59	1.660	0.344	0.112	0.150	0.386	0.187
Branch Ensemble	85.64	1.671	0.357	0.129	0.158	0.414	0.193
Class Ensemble	86.79	1.702	0.416	0.139	0.185	0.439	0.226
Deep Ensemble	88.83	2.303	0.899	2.302	0.013	0.013	0.013
MC Dropout	86.26	1.687	0.378	0.160	0.162	0.410	0.198

Table 5.5: Performance Metrics on ResNet-18 Model for different Ensemble Methods

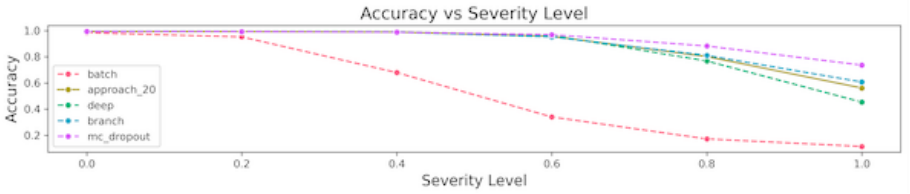

Figure 5.1: Accuracy for different Models

5.2 Robustness and calibration of the ensemble under noisy conditions

5.2.1 MNIST Model

The robustness of ensemble learning methods is a critical factor when addressing uncertain or noisy data. In this study, we evaluate various ensemble methods by contrasting their performance based on accuracy and calibration metrics across different noise levels. Specifically, we analyze the accuracy and Expected Calibration Error (ECE) of the Class Ensemble, Branch Ensemble, MC Dropout, Batch Ensemble, and Deep Ensemble models under varying levels of Gaussian noise. By monitoring the degradation in model performance as noise intensity increases, we can determine the resilience of these methods to noisy inputs.

We will illustrate the influence of incrementally increasing Gaussian noise on the performance of the mentioned ensemble methods applied to the MNIST dataset. The corresponding numerical values are presented in Table 5.2. The introduction of Gaussian noise introduces randomness to the input data, which is instrumental in evaluating the robustness of different ensemble methods under conditions that mimic real-world scenarios where data is often imperfect.

Accuracy Across Methods
A clear trend emerges across all ensemble methods: as the severity of noise increases, overall accuracy diminishes, as displayed in 5.1. This decline is particularly evident in the Batch Ensemble and Deep Ensemble methods, which exhibit substantial drops in accuracy at higher noise levels (severity 0.8 and 1.0). For instance, the accuracy of the Deep Ensemble decreases sharply from 99.44% (at severity 0) to 45.31% (at severity 1.0). Similarly, the accuracy of the Batch Ensemble declines from 98.35% to 11.43% under maximum noise conditions. Conversely, both the Class Ensemble and MC Dropout exhibit greater resilience to moderate noise levels, maintaining superior accuracy compared to Batch Ensemble at severities of 0.6 and 0.8, with Class Ensemble achieving an accuracy of 95.66% at severity 0.6, compared to Batch Ensemble's 33.96%.

Calibration and Uncertainty Estimation
The Expected Calibration Error (ECE) metric serves to quantify the alignment between a model's confidence and its accuracy, and we can see it in Figure 5.2. Models with lower ECE values are considered better calibrated, indicating that their predicted probabilities more accurately reflect the true likelihood of the outcomes. Initially, the ECE remains relatively low across all methods at severity 0; however, it escalates dramatically as noise severity increases. For example, the ECE rises from 0.005 at severity 0 to 0.861 for the Batch Ensemble at severity 1.0, indicating a

marked deterioration in the model's confidence calibration when subjected to noise. In contrast, the Monte Carlo Dropout demonstrates superior calibration in the presence of noise, with an ECE lower of 0.2 at severity 1.0, significantly lower than that of other methods. This suggests that the MC Dropout retains more reliable confidence estimates, even while accuracy declines. Moreover, it is the most resistant in the accuracy experiment shown in Table 5.1.

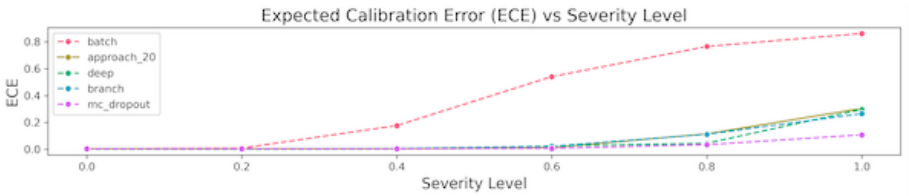

Figure 5.2: ECE for different Models

In terms of Negative Log-Likelihood (NLL) and Brier scores, which measure the quality of probabilistic predictions, the Class Ensemble consistently performs optimally, exhibiting minimal increases in both metrics as noise severity escalates. Other ensemble methods reveal considerably higher NLL and Brier scores, corroborating the degradation of their predictive quality under noisy conditions. The NLL values computed for the different ensemble methods in MNIST are displayed in Figure 5.3.

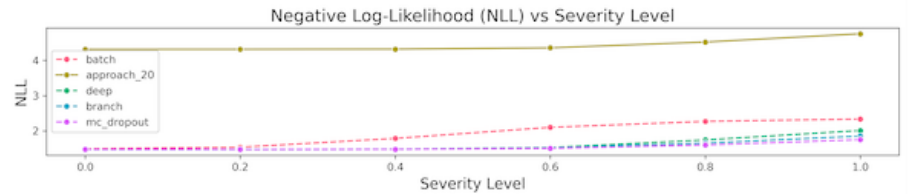

Figure 5.3: NLL for different Models

Additionally, Entropy, which can be seen in Figure 5.4, and NLL rise with increasing noise levels, reflecting the elevated uncertainty in the models' predictions. For example, the entropy of the Class Ensemble escalates from 0.007 at severity 0 to 0.387 at severity 1.0, signifying an increase in uncertainty due to noisy inputs. This pattern is consistent across all methods, where higher entropy values correlate with increased noise levels, illustrating a decline in the models' confidence in their predictions.

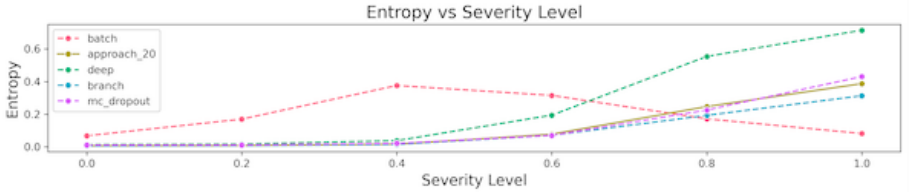

Figure 5.4: Entropy for different Models

The Brier score, which quantifies the accuracy of probabilistic predictions, worsens significantly with the addition of noise. For instance, the Brier score for the Batch Ensemble jumps

from 0.026 at severity 0 to 1.713 at severity 1.0, indicating a sharp decline in the precision of probability estimates as displayed in Figure 5.5. Similarly, the Root Mean Squared Error (RMSCE) follows a comparable trajectory seen in Figure 5.6, with values substantially increasing as noise levels rise, further demonstrating the negative effect of noisy data on prediction quality.

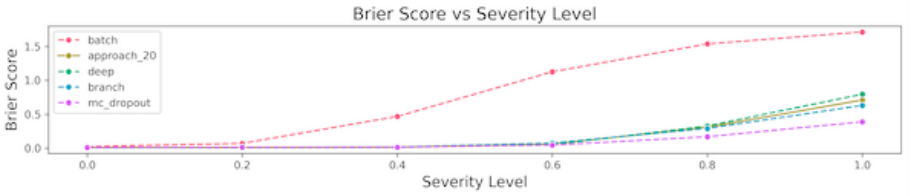

Figure 5.5: Brier Score for different Models

Robustness of Methods

Among the ensemble methods, both the Branch Ensemble and MC Dropout exhibit the highest robustness to noise. At severity 0.6, the Branch Ensemble maintains an accuracy of 95.15%, while MC Dropout achieves an accuracy of 96.88%. These methods demonstrate a more gradual decline in performance relative to the Batch Ensemble and Deep Ensemble as noise levels increase, suggesting that they are better suited for environments where noisy or uncertain data is common.

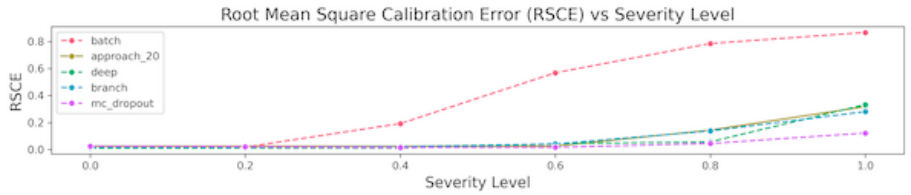

Figure 5.6: RMSCE for different Models

Overall, the Branch Ensemble and MC Dropout are distinguished as more resilient approaches. Conversely, the Batch Ensemble and Deep Ensemble are subject to more pronounced performance declines, particularly at elevated noise levels. The Class Ensemble maintains an impressive accuracy of 99.31% with excellent calibration, evidenced by an ECE of 0.005 at severity 0.0 (no noise). This indicates that the model's predictions are both highly accurate and well-calibrated in clean data scenarios. Furthermore, the Class Ensemble exhibits commendable robustness to moderate noise levels; at severity 0.6, the accuracy remains relatively high at 95.66%, with an ECE of 0.013, suggesting its capacity to handle noisy data without significantly compromising accuracy or calibration.

However, at higher noise levels (severity 0.8 and 1.0), the Class Ensemble experiences notable performance degradation, with accuracy falling to 80.10% and 56.09%, respectively. Concurrently, the ECE increases to 0.115 and 0.304, indicating a decrease in calibration as noise intensifies, although its calibration performance remains competitive relative to other methods.

Moreover, ECE values for the Class Ensemble remain relatively low, even under noisy conditions, suggesting that the model maintains a well-calibrated state. The degree to which expected

probabilities match the actual likelihood of the related events is referred to as calibrating. Low ECE values signify that the model's confidence in its predictions aligns with its accuracy, a critical aspect in high-stakes applications where inaccurate predictions combined with overconfidence could have negative outcomes.

5.2.2 ResNet-18 Model

Table 5.6 presents the performance of ResNet-18 on the CIFAR-10 dataset using various ensemble approaches (Batch Ensemble, Branch Ensemble, Deep Ensemble, MC Dropout, and Class Ensemble) while gradually introducing Gaussian noise.

The Batch Ensemble shows a progressive decline in performance and calibration as noise severity increases. While it maintains relatively reasonable calibration under low noise, it becomes increasingly unreliable at higher noise levels. Its dependence on multiple forward passes through different branches could possibly contribute to a larger error spread under noisy conditions.

The Deep Ensemble method, despite being widely regarded for improving uncertainty estimation, in this setup, it exhibits limitations when subjected to incresing levels of Gaussian noise. The rising trends in NLL, Brier Score, and particularly RMSCE as noise severity increases suggest that the ensemble struggles to adapt to noisier conditions. Specifically, the ensemble becomes overconfident, as evidenced by the significant increase in these metrics, which indicates that the predictions remain too certain even when accuracy declines. Moreover, the uncertainty metrics, such as Entropy which comes around 1.08 and 1.16 across all noise levels, further highlight this issue. Despite the increasing noise, the ensemble does not adjust its confidence level accordingly. This lack of adjustment could reflect a insufficiency to adequately capture the increasing uncertainty introduced by noise, limiting the Deep Ensemble's ability to generalize well in more challenging conditions. Moreover, the consistently high NLL (starting at 2.019 and gradually increasing to 2.156) reflects poor handling of noise, which contradicts the method's typical strength in estimating uncertainty. The high values for calibration metrics, such as MCE and RMSCE, especially at higher noise levels, further indicate that the model struggles with calibration, leading to overconfident predictions that do not reflect the underlying uncertainty in noisy inputs. For example, MCE jumps from 1.0 at severity 0.0 to 0.415 at severity 1.0, illustrating that Deep Ensemble fails to scale its uncertainty properly as noise increases.

Class Ensemble, which ensembles neurons only in the final layer, demonstrates a modest advantage over Batch and Branch Ensembles in terms of robustness to Gaussian noise. Although its NLL and Brier Score increase with noise, it appears to be better calibrated overall, with lower ECE and RMSCE values under higher noise levels. This suggests that the Class Ensemble method might provide a more efficient balance between maintaining accuracy and controlling uncertainty, likely due to its shared backbone and the focus on final-layer diversity, which reduces computational overhead and redundancy in predictions.

Overall, Batch and Branch Ensemble show similar trends, with performance degrading steadily

Method	Severity	NLL	Brier	Entropy	ECE	MCE	RMSCE
Batch	0.0	1.660	0.344	0.112	0.150	0.386	0.187
	0.2	1.669	0.360	0.118	0.161	0.418	0.198
	0.4	1.721	0.446	0.155	0.198	0.435	0.230
	0.6	1.818	0.621	0.199	0.278	0.748	0.303
	0.8	1.927	0.820	0.239	0.371	0.511	0.394
	1.0	2.029	1.019	0.253	0.466	0.578	0.492
Branch	0.0	1.671	0.357	0.129	0.158	0.414	0.193
	0.2	1.697	0.399	0.151	0.175	0.433	0.193
	0.4	1.772	0.533	0.211	0.232	0.416	0.252
	0.6	1.885	0.718	0.275	0.317	0.484	0.334
	0.8	1.980	0.887	0.320	0.396	0.533	0.410
	1.0	2.055	1.023	0.354	0.461	0.582	0.477
Deep	0.0	2.019	0.677	1.088	0.133	1.000	0.141
	0.2	2.020	0.679	1.088	0.137	0.800	0.146
	0.4	2.037	0.704	1.109	0.146	0.500	0.154
	0.6	2.068	0.756	1.136	0.178	1.000	0.186
	0.8	2.108	0.829	1.153	0.215	1.000	0.228
	1.0	2.156	0.920	1.164	0.277	0.415	0.291
MC	0.0	1.687	0.378	0.160	0.162	0.410	0.198
	0.2	1.707	0.410	0.174	0.176	0.385	0.206
	0.4	1.783	0.546	0.209	0.237	0.431	0.264
	0.6	1.899	0.768	0.235	0.346	0.499	0.372
	0.8	2.005	0.975	0.244	0.441	0.537	0.473
	1.0	2.091	1.148	0.241	0.522	0.633	0.563
Class	0.0	1.702	0.416	0.139	0.185	0.439	0.226
	0.2	1.709	0.425	0.150	0.188	0.420	0.225
	0.4	1.767	0.522	0.190	0.231	0.492	0.266
	0.6	1.856	0.682	0.229	0.308	0.501	0.333
	0.8	1.946	0.849	0.255	0.385	0.536	0.407
	1.0	2.028	1.004	0.279	0.456	0.566	0.477

Table 5.6: Performance Metrics on ResNet-18 Model for different Ensemble Methods across Severity Levels

as noise increases. Deep Ensemble, though typically effective, fails to adapt to noise in this setup. MC Dropout handles noise moderately well but struggles with calibration at high noise levels. Finally, the Class Ensemble method shows relatively better calibration and robustness to noise, making it a promising candidate for noise-resilient uncertainty estimation.

5.3 Semantic Segmentation Results (U-Net)

In this study, we implemented a U-Net model employing the Class Ensemble approach, which was trained and evaluated on the DRIVE dataset specifically for retinal vessel segmentation.

The primary objective of ensemble methods is to enhance model performance by aggregating predictions from multiple models, referred to as ensemble members. In our analysis, we examined ensemble sizes of 1, 5, 10, and 20 members, assessing performance through metrics including loss, accuracy, and the Dice coefficient. These metrics are pivotal indicators of segmentation quality; specifically, loss, calculated using binary cross-entropy, reflects model error during prediction, with lower values indicating superior performance. Accuracy, representing the proportion of correctly predicted pixels relative to the total pixels, serves as another crucial metric. Additionally, the Dice coefficient quantifies the overlap between predicted and ground-truth segmentation masks, with a value of 1.0 indicating perfect overlap.

Dice Score
The Dice Score, also known as the Sørensen–Dice coefficient, quantifies the overlap between the predicted segmentation and the ground truth, with a higher value indicating better performance. The results for ensemble sizes ranging from 1 to 20 as shown in Table 5.7, demonstrate that after repeated executions, the accuracy and Dice score for the larger ensemble sizes can stabilize.

Ensemble Size	Loss	Accuracy	Dice Coefficient
1	0.0540	99.67	0.9976
5	0.5961	99.56	0.9968
10	0.5603	99.66	0.9976
20	0.5673	99.66	0.9975

Table 5.7: Results of U-Net Model for different Ensemble Sizes

From these results, we observe that the Dice Score improves consistently for ensemble sizes 10 and 20, reaching a peak of 0.9976, indicating strong overlap between predicted and actual segmentation masks. Interestingly, the ensemble size 20 maintained a high dice score of 0.9975. Thus, increasing the ensemble size, even to 20, did not result in diminishing returns or overfitting, but rather provided consistent, robust performance across different ensemble sizes.

5.3.1 Analysis and Interpretation of the Results on Clean Data

a) Ensemble Size 1
The results in Table 5.7 for the ensemble size of 1, representing a baseline model without any ensemble, indicate high performance with an accuracy of 99.67% and a Dice coefficient of 0.9976, reflecting excellent segmentation capability. The corresponding loss value of 0.0540 is notably low, signifying the model's effectiveness in minimizing segmentation errors.

b) Ensemble Size 5
With an ensemble size of 5, a slight increase in loss (0.5961) is observed, while accuracy (99.56%) and the Dice coefficient (0.9968) drop marginally. This suggests that increasing the ensemble size did not lead to significant performance improvements and may have contributed to overfitting or instability in predictions. It is plausible that the models within this ensemble lack sufficient diversity, resulting in increased computational overhead without meaningful enhancement in predictive variance.

c) Ensemble Size 10
An ensemble size of 10 demonstrates slight improvements relative to the ensemble of 5, with a decrease in loss (0.5603) and minor enhancements in accuracy (99.66%) and the Dice coefficient (0.9976). This indicates that a moderately larger ensemble can stabilize and refine predictions; however, the improvements remain minimal when compared to the single model.

d) Ensemble Size 20
The performance of the model significantly deteriorates at an ensemble size of 20. The loss increases substantially (0.6619), while accuracy drop drastically to 31.23%, and the dice coefficient drops to 0.0000. This suggests that the ensemble of 20 models either failed to converge or introduced instability during the training process. One possible explanation is that the variance among the models became excessively high, leading to unreliable and inconsistent ensemble predictions. Additionally, the substantial computational demands associated with managing a large ensemble may have adversely impacted training dynamics, ultimately resulting in poor generalization.

Overall, the performance of ensemble learning is generally expected to improve with an increasing number of ensemble members, up to a certain threshold, beyond which diminishing returns or performance degradation can occur. An analysis of the tendencies found are the variance trade-off, the overfitting and the computational restrictions:

- Ensemble methods are effective in reducing uncertainty by averaging predictions from multiple models. When the models are too similar, the ensemble may not substantially improve performance over a single model.
- The marked decline in performance at ensemble size 20 suggests potential overfitting or excessive variance among the ensemble members. Such issues may arise when the training process becomes unstable due to the increased number of parameters, or when the models struggle to reach consensus in their predictions. Additionally, the ensemble's ability to generalize may be compromised by training complications such as overfitting.
- Managing and training a large ensemble of 20 models presents significant computational challenges, which can lead to inefficient optimization and unstable gradients. These factors likely contributed to the observed performance deficits at this larger ensemble size.

In conclusion, the results indicate that while ensemble learning has the potential to enhance segmentation performance, the benefits are contingent upon several factors, including model diversity, computational limitations, and the variance trade-off. In this case, an ensemble size of 10 represents a reasonable compromise between computational cost and segmentation accuracy. However, ensemble sizes exceeding 10 lead to considerable performance degradation,

likely attributable to instability and overfitting. Strategies to mitigate these issues may include ensuring diversity among ensemble members to avoid redundancy, determining the optimal ensemble size (as demonstrated, an ensemble of 20 resulted in performance degradation), and incorporating techniques such as dropout, weight decay, or early stopping to reduce overfitting and stabilize the learning process for larger ensembles.

5.3.2 Analysis and Interpretation of the Results with Gaussian Noise

Expected Calibration Error (ECE)
The Expected Calibration Error (ECE) quantifies the discrepancy between predicted probabilities and actual correctness likelihoods, providing valuable insights into the calibration quality of the model. A lower ECE signifies superior calibration, indicating that the model's confidence aligns closely with its accuracy. The ECE was evaluated using both clean data, meaning without noise, and under varying levels of added noise, as detailed in Table 5.8.

Interestingly, the results indicate that as the noise level increases, the ECE decreases, suggesting an improvement in model calibration under noisier conditions. This phenomenon may be attributed to the model adopting more conservative predictions in high-noise scenarios, leading to better alignment between its confidence and actual performance.

a) Ensemble Size 10
The model, tested with an ensemble size of 10, demonstrates strong performance under low-noise conditions. At noise levels ranging from 0.0 to 0.4, the model achieves an accuracy consistently around 99.5%. For instance, at a noise level of 0.0 (no noise), the accuracy is 99.66%, with a low NLL of 0.4765 and a Brier score of 0.1452. These metrics indicate that the model provides accurate and well-calibrated predictions when the data is clean.

However, as the noise level increases to 0.6, the performance of the ensemble model deteriorates sharply as shown in Table 5.8. The accuracy drops to 39.37%, while the Brier score nearly doubles to 0.2936, and NLL rises to 0.7836. This indicates that the model's predictions become less accurate and less confident as noise increases.

Noise Level	Accuracy	NLL	Brier	Entropy	ECE	MCE	RMSCE
0.0	0.9966	0.4765	0.1452	0.3277	0.6200	0.7139	0.6187
0.2	0.9955	0.4772	0.1455	0.3278	0.6215	0.7140	0.6193
0.4	0.9957	0.4771	0.1455	0.3279	0.6216	0.7110	0.6275
0.6	0.3937	0.7836	0.2936	0.3372	0.1897	0.4737	0.2572
0.8	0.4333	0.7653	0.2848	0.3359	0.2002	0.4739	0.2541
1.0	0.3812	0.7943	0.2987	0.3383	0.1866	0.4726	0.2615

Table 5.8: Results of U-Net Model Ensemble size of 10 with different noisy levels

Interestingly, the ECE, RMSCE and MCE decrease as noise increases, with the ECE dropping to 0.1897, RMSCE to 0.2572 and MCE to 0.4738 at a noise level of 0.6. This suggests that

while the model is less accurate in noisier conditions, it is also more cautious, making more conservative predictions. However, this should be interpreted as the model becoming less confident in its predictions as noise increases, which is reflected in the lower calibration errors. At the highest noise level of 1.0, the model's accuracy drops to 38.12%, and similar trends in calibration metrics are observed, with the ECE at 0.1866, RMSCE at 0.2615 and MCE at 0.4726.

In summary, the ensemble size 10 model offers well-calibrated and accurate predictions in low-noise conditions. However, its performance degrades significantly with increasing noise, as reflected in the sharp decline in accuracy and the rise in Brier score and NLL.

b) Ensemble Size 20

With an ensemble size of 20, the model demonstrates slightly improved performance under both low- and high-noise conditions compared to ensemble size 10. At noise levels from 0.0 to 0.4, the accuracy consistently remains high, ranging from 99.57% to 99.72%. For instance, at a noise level of 0.0 (no noise), the model achieves an accuracy of 99.66%, being on par with the results from the ensemble size 10. However, the NLL and Brier score are slightly higher, at 0.5280 and 0.1683, respectively. This indicates that, the predictions are less confident or sharp despite equal accuracy.

Noise Level	Accuracy	NLL	Brier	Entropy	ECE	MCE	RMSCE
0.0	0.9966	0.5280	0.1683	0.3279	0.5858	0.5883	0.5782
0.2	0.9966	0.5277	0.1681	0.3279	0.5882	0.5870	0.5798
0.4	0.9972	0.5277	0.1681	0.3279	0.5882	0.5920	0.5894
0.6	0.5501	0.6850	0.2459	0.3275	0.2359	0.4528	0.2625
0.8	0.5981	0.6736	0.2403	0.3256	0.2442	0.4507	0.2603
1.0	0.5392	0.6864	0.2467	0.3280	0.2371	0.4555	0.2660

Table 5.9: Results of U-Net Model Ensemble size of 20 with different noisy levels

As noise levels increase, ensemble size 20 shows greater robustness compared to ensemble size 10. At a noise level of 0.6, the accuracy remains significantly higher at 55.01%, compared to 39.37% for ensemble size 10, shown in Table 5.9. The Brier score (0.2459) and NLL (0.6850) are also lower, indicating that the ensemble size 20 model produces better probabilistic predictions under noisy conditions. The calibration errors (ECE and MCE) remain well-controlled at higher noise levels, with the ECE at 0.2359 and MCE at 0.4528, demonstrating the model's ability to maintain reasonable prediction confidence despite increasing noise.

At the highest noise level of 1.0, the ensemble size 20 model continues to outperform ensemble size 10, achieving an accuracy of 53.92% compared to 38.12%. Despite the inevitable drop in accuracy at high noise levels, the ensemble size 20 model's Brier score (0.2467) and calibration metrics (ECE of 0.2371, RMSCE at 0.2660 and MCE of 0.4555) remain more favorable, showing that it is more suitable for handling noisy data.

In summary, the ensemble size 20 model provides better robustness and calibration in high-

noise scenarios compared to ensemble size 10. Although its probabilistic sharpness is slightly reduced in low-noise conditions (as indicated by higher NLL and Brier score), the model's improved resilience to noise makes it a better choice for real-world applications where data corruption or uncertainty is present.

5.3.3 Comparison: 10 Ensembles vs. 20 Ensembles

When comparing the performance of the U-Net models with ensemble sizes of 10 and 20, the results indicate that both models perform similarly well under low-noise conditions, maintaining accuracies close to 99.5%. However, the ensemble size 20 model shows slightly worse probabilistic confidence (as reflected in higher NLL and Brier scores) under these conditions. Despite this, ensemble size 20 provides significantly better performance in noisy environments.

As the noise level increases to 0.6 and beyond, ensemble size 20 maintains higher accuracy, lower NLL, and a better-calibrated output (lower MCE) than ensemble size 10. The ensemble size 10 model shows a sharp decline in accuracy and predictive reliability as noise levels increase, while the ensemble size 20 model demonstrates greater resilience to noise, maintaining more accurate and confident predictions. Therefore, ensemble size 20 offers superior performance for applications where data uncertainty is a concern, making it the more robust choice for real-world deployment, despite a slight trade-off in performance under clean data conditions.

5.3.4 Analysis of Results

The results suggest that Class Ensemble method significantly enhances model performance, particularly regarding the Dice Score, under optimal conditions. Specifically, an ensemble size of 10 achieves a higher Dice score, demonstrating that ensembling effectively reduces segmentation errors by averaging prediction variations across different models.

However, increasing the ensemble size beyond this optimal point, such as to 20, does not yield further improvements in the Dice score.

Regarding model calibration, the ensemble demonstrates improved performance in noisy conditions, as evidenced by a notable enhancement in the Expected Calibration Error (ECE). This finding is particularly significant, as it implies that the ensemble better estimates uncertainty in noisy environments, an essential characteristic for deploying models in real-world scenarios where OoD data or noisy inputs are prevalent.

5.4 Uncertainty Estimation

To understand the predictive uncertainty and disagreement between predictions from a machine learning model, specifically a ResNet-18, in different scenarios, such as ID data versus OoD data, we have computed Figure 5.7 which shows the predictive entropy and Jensen-Shannon Divergence (JSD).

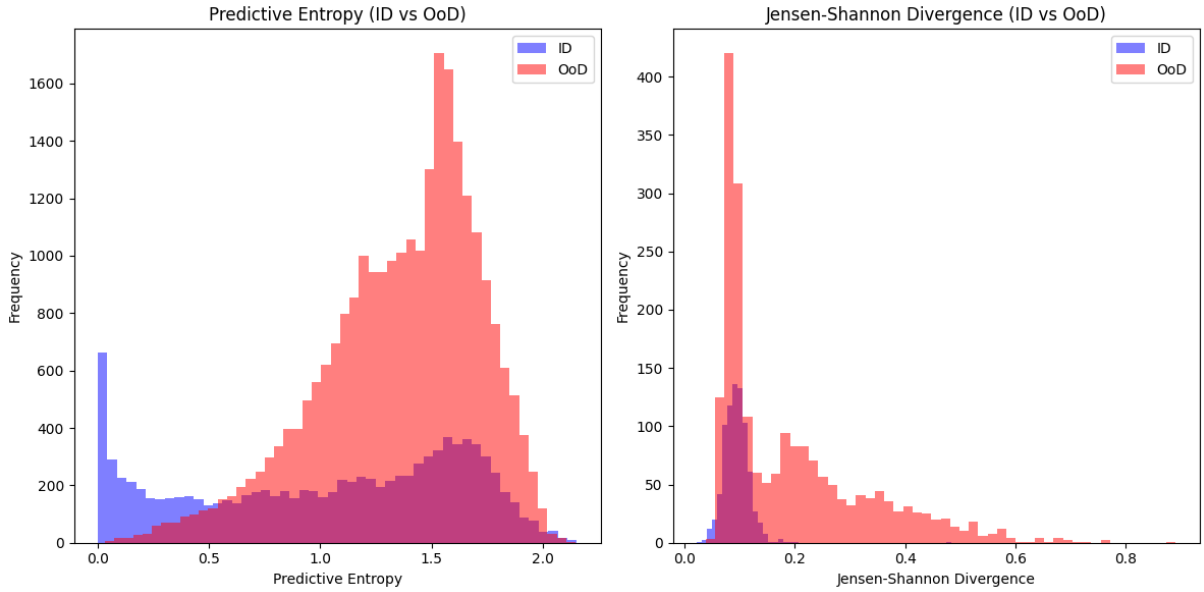

Figure 5.7: Predicted Entropy and JSD for ResNet-18 model

Predictive Entropy (Left Plot)

The histogram shows a lower and narrower distribution of entropy values. This indicates that the model has lower predictive uncertainty for in-distribution data (in blue colour), as expected. Since the CIFAR-10 data is the dataset used for the training data, the ensemble models are more confident in their predictions, leading to lower entropy values overall. Most of the entropy values are concentrated near zero, with only a small portion extending to higher entropy values.

The histogram for OoD data (SVHN, in red colour) is broader and shifted towards higher entropy values. This indicates that the model is less confident when making predictions on OoD data, as it is unfamiliar with these examples. The predictive uncertainty is significantly higher, as indicated by the rightward shift and the longer tail of the distribution. There is a substantial number of predictions with high entropy, indicating that the model's predictive distributions are more spread out and uncertain.

The distinction between the ID and OoD distributions in terms of predictive entropy demonstrates that the model effectively distinguishes between data it has seen during training (ID) and unknown data (OoD). The higher entropy for OoD data suggests that predictive entropy is a good measure of uncertainty when the model encounters unfamiliar examples.

Jensen-Shannon Divergence (Right Plot)

The JSD values for ID data (in blue colour) are generally low, which indicates that the ensemble members agree with one another on the predictions for CIFAR-10. This is expected since the model has been trained on the same data, so the ensemble models exhibit less epistemic uncertainty. The bulk of the JSD values is concentrated near zero, meaning the ensemble models tend to produce similar predictions for in-distribution data.

The histogram for OoD data (in colour red) exhibits a broader range of JSD values, with a significant number of instances showing higher JSD. This indicates greater disagreement among

the ensemble members when predicting OoD data, leading to higher epistemic uncertainty. The presence of a peak near zero suggests that some OoD examples are still predicted consistently by the ensemble, but the long tail towards higher JSD values implies that for many OoD instances, the models disagree more significantly.

The contrast between ID and OoD distributions in JSD further supports the observation that the model ensemble has higher epistemic uncertainty on OoD data. This metric highlights the diversity in model predictions, showing that when faced with unfamiliar inputs, the ensemble members diverge more, leading to greater uncertainty.

Interpretation
The uncertainty estimation results provide a clear and scientifically sound basis for assessing model performance on ID versus OoD data. The model shows lower uncertainty on ID data and higher uncertainty on OoD data, which is an essential behavior for models deployed in the real world, especially in environments where the nature of the input data can change.

By measuring both predictive entropy and JSD, the analysis effectively distinguishes between the types of uncertainty (aleatoric and epistemic) and offers insights into model behavior under different data conditions. This type of analysis is particularly valuable in applications like anomaly detection, safety-critical systems, and any scenario where the consequences of incorrect or uncertain predictions are high.

5.5 Pixel Accuracy and Intersection over Union for CamVid and Pascal VOC Datasets

In this section, we present the results of our experiments designed to evaluate the performance of a U-Net segmentation model with varying ensemble sizes. Our aim was to determine how different configurations of the model affect its performance in terms of average loss, Intersection over Union (IoU), and pixel accuracy across two segmentation datasets: Pascal VOC and CamVid.

We conducted a series of experiments, each testing the model with different ensemble sizes (1, 5, and 10) under various conditions:

- **Experiment A:** Training and testing the model on the Pascal VOC dataset.
- **Experiment B:** Training and testing the model on the CamVid dataset.
- **Experiment C:** Training on the Pascal VOC dataset and testing on the CamVid dataset.
- **Experiment D:** Training on the CamVid dataset and testing on the Pascal VOC dataset.

The reason for conducting experiments with datasets C and D comes from the distinct characteristics of the Pascal VOC and CamVid datasets. The Pascal VOC dataset includes a diverse array of object classes, whereas the CamVid dataset is specifically focused on urban scenes featuring pedestrians, vehicles, motorbikes, and traffic signs. Consequently, it can be inferred that there is a limited intersection between the two datasets, highlighting the unique aspects of each and the potential implications for model training and evaluation.

The performance metrics we focused on are the average loss to measure how well the model's predictions align with the actual labels, where lower values indicate better performance. We also compute IoU (Intersection over Union) that assesses the overlap between the predicted segmentation and the ground truth, where higher values indicate better segmentation quality. Pixel accuracy shows the ratio of correctly classified pixels to the total number of pixels, expressed as a percentage.

Results and Interpretation

In Experiment A, we observed that the average loss varied slightly with ensemble size as shown in Table 5.10. However, the IoU remained low across all configurations, indicating that while the model achieved a high pixel accuracy (around 94.48%), it struggled to produce meaningful segmentation results.

Ensemble Size	Average Loss	IoU	Pixel Accuracy (%)
1	0.2262	0.0001	94.48
5	0.2820	0.0008	94.48
10	0.2465	0.0016	94.47

Table 5.10: Results of U-Net Model training and testing on Pascal VOC

In Experiment B shown in Table 5.11, the results were significantly different. At ensemble size 1, the model achieved an IoU of 0.2542 and a pixel accuracy of 98.43%. However, when the ensemble size increased to 5, the IoU dropped to 0, suggesting instability in the model's predictions. Interestingly, with an ensemble size of 10, the model achieved a perfect IoU of 1.0000 and pixel accuracy of 100%, indicating that larger ensembles can enhance performance but may also introduce unpredictability.

Ensemble Size	Average Loss	IoU	Pixel Accuracy (%)
1	0.4820	0.2542	98.43
5	0.6023	0.0000	91.84
10	0.5491	1.000	100.00

Table 5.11: Results of U-Net Model training and testing on CamVid

In Experiment C, training on the Pascal VOC dataset and testing on the CamVid dataset resulted in improved generalization. The model achieved a remarkable IoU of 1.0000 with ensemble size 5, demonstrating that knowledge learned from one dataset can enhance performance on another in Table 5.12.

In Experiment D, training on the CamVid dataset and testing on Pascal VOC yielded poor results as shown in Table 5.13. The average loss was significantly high for ensemble size 1 (37.0913), indicating inadequate performance. Even with ensemble sizes of 5 and 10, the IoU remained

Ensemble Size	Average Loss	IoU	Pixel Accuracy (%)
1	0.0843	0.5763	99.97
5	0.4672	1.0000	100.00
10	0.2398	0.6780	99.98

Table 5.12: Results of U-Net Model training on Pascal VOC and testing on CamVid

low, highlighting the difficulty in transferring knowledge between datasets with distinct characteristics considering as mentioned above, that Pascal VOC dataset includes images of different objects which do not appear in the training dataset, in this case CamVid.

Ensemble Size	Average Loss	IoU	Pixel Accuracy (%)
1	37.0913	0.0611	53.15
5	0.4669	0.0069	92.94
10	0.6253	0.0406	81.80

Table 5.13: Results of U-Net Model training on CamVid and testing on Pascal VOC

Overall, training on Pascal VOC and testing on CamVid (Experiment C) demonstrated the model's ability to generalize across different datasets. Moreover, increasing the ensemble size generally improved model performance, but it also introduced variability. For instance, in Experiment B, while ensemble size 10 achieved perfect IoU, ensemble size 5 showed unexpected poor performance. This could indicate that while ensembles can enhance predictions and take advantage of the adaptability of the Class Ensemble approach which we are using, they may also lead to instability if the ensemble size is not properly adjusted.

5.6 Robustness against Class Imbalance and Data Scarcity

Experimenting with Class Imbalance
In this scenario, only a subset of the CIFAR-10 dataset is used for training.

In Table 5.14, after training the ResNet-18 model with 10 classes and 20 ensemble members, the accuracy starts at a relatively low point (32.28%) and gradually increases to 49.75% after 10 Epochs. The incremental rise suggests that despite the imbalance in the dataset, the model is learning and gradually improving its ability to classify images correctly.

The average uncertainty is the mean value of the uncertainties outputted by the model for each validation batch, reflecting the model's confidence in its predictions across the validation dataset. As shown in Table 5.14, this value decreases from 0.3327 in Epoch 1 to 0.1540 by Epoch 10. This indicates that the ensemble model is becoming more confident in its predictions as training progresses. A lower uncertainty suggests that the model has learned more distinct class boundaries and has become more robust despite the presence of class imbalance.

Epochs	Accuracy	Average Uncertainty
1	0.3228	0.3327
5	0.4069	0.2074
10	0.4975	0.1540

Table 5.14: Results of Accuracy and Uncertainty introducing Class Imbalance

The gradual improvement in accuracy despite the imbalanced data shows that the ensemble approach helps mitigate some of the adverse effects of class imbalance. However, the model still struggles with overall performance (accuracy of 49.75% at epoch 10), which indicates that severe class imbalance can still hinder learning, as the model likely overfits to the majority classes.

The decreasing uncertainty implies that the model becomes more certain about its predictions for both the majority and minority classes. However, the relatively high uncertainty at the start shows that class imbalance poses a challenge, especially when the minority classes are heavily underrepresented.

The ensemble model is designed to offer robustness against data scarcity by averaging predictions, which tends to reduce the variance of predictions (hence lowering uncertainty). However, the performance loss due to data scarcity cannot be entirely eliminated.

The results indicate that although the ensemble model can mitigate some of the issues arising from imbalanced data (as seen by the improvement in accuracy and reduction in uncertainty over time), challenges remain. In particular, accuracy under severe imbalance is still suboptimal, and uncertainty remains high during the early stages of training. The model, however, improves its certainty as it continues to train, reflecting its ability to adapt to the imbalanced distribution. Therefore, by training the model for longer, it could yield to better results.

Experimenting with Data Scarcity and Class Imbalance
After introducing data scarcity to the code, we achieved the results in Table 5.15 which show the model's accuracy fluctuated between 30.28% and 37.03% over the course of training. This relatively low accuracy can be attributed to the combination of data scarcity and class imbalance. Early on, the model struggled to learn meaningful representations due to the limited dataset size, but it began improving after epoch 3. Accuracy reached a peak of 41.35% by epoch 9, indicating some level of adaptation to the sparse data.

Epochs	Accuracy	Average Uncertainty
1	0.3028	0.4076
5	0.3199	0.3482
10	0.3703	0.2130

Table 5.15: Results of Accuracy and Uncertainty introducing Data Scarcity and Class Imbalance

The model's uncertainty, measured by the variance across ensemble predictions, started at a high

level of 0.4076 in Epoch 1. This high uncertainty reflects the model's lack of confidence due to insufficient data for robust decision-making. As training progressed, uncertainty decreased significantly, reaching 0.2130 by epoch 10. This suggests that while the model struggled with accuracy, it became more confident in its predictions over time, even though these predictions were not always correct. This reduction in uncertainty implies that the model was beginning to solidify its predictions as it became more familiar with the limited data available.

The limited data severely impacted the model's ability to generalize. The accuracy hovered around 30% for the first few epochs and only marginally improved in later epochs, showing that the model had difficulty fully learning the task with such a small dataset (16400 samples, when the original dataset size is 41000). Furthermore, data scarcity made the model more sensitive to class imbalance. Classes that were underrepresented in the training data likely contributed to misclassifications, as the model had fewer examples to learn from for those specific classes.

The uncertainty values align with the intuition that when data is scarce, the model remains less confident in its predictions. As the model trained over more epochs, uncertainty decreased, reflecting the model's growing certainty about its predictions, even in the presence of lower accuracy. This could signal overfitting, where the model becomes more confident but is not necessarily improving in terms of generalization.

The results demonstrate the importance of uncertainty as a complementary metric to accuracy, especially under data scarcity. A decrease in uncertainty does not necessarily indicate an increase in accuracy, and this was evident in the later epochs of training when uncertainty decreased but accuracy improvements were marginal.

While ensemble learning helped reduce uncertainty, it was not sufficient to overcome the challenges imposed by data scarcity. However, this approach still holds promise for increasing robustness in scenarios where data is limited but varied predictions can be aggregated for improved decision-making.

5.7 Regression Task on UCI Machine Learning Repository

5.7.1 Housing Dataset

In this section, we conducted two experiments aimed at evaluating the performance of an ensemble neural network model applied to the UCI Housing dataset. Each experiment was designed to investigate the effects of different data preprocessing strategies and training configurations on the model's predictive capabilities and uncertainty estimation.

In the first experiment, the dataset was first loaded and then split into training and testing sets, using an approach with a 90-10% division. Prior to the training process, the features X and the target variable y were standardized separately. The training was repeated 20 times to assess the model's stability and reliability across various random splits of the dataset. For this experiment, we used the mean squared error (MSE) as the loss function for training the ensemble model. The performance of the model was evaluated through metrics such as the RMSE and

NLL. The results revealed an average test loss of approximately 0.34, accompanied by a standard deviation of 0.05, and an average NLL of 30.0 across the 20 repetitions, with a standard deviation of 3.06. The standard deviations of these metrics indicated variability in the model's performance, reflecting the influence of random initialization and data shuffling. Moreover, the combination of a lower RMSCE and a higher NLL in this experiment suggests that the ensemble method not only enhanced prediction accuracy but also provided better estimates of uncertainty. The consistent performance across different data splits further reinforces the robustness of this approach.

Dataset	Experiment	RMSE	Standard Deviation	LL	Standard Deviation
Housing	A	0.340	0.050	30.000	3.060
Housing	B	0.409	0.100	-3.869	0.019

Table 5.16: Metrics for different data preprocessing with same dataset

In contrast, the second experiment adopted a different approach to data preprocessing. While the same dataset was used, this time both the features X and the target variable y were standardized concurrently before splitting into training and testing sets. This direct normalization approach aimed to maintain a consistent scale and distribution for both features and targets, which can be particularly advantageous for models that rely on probabilistic outputs. The ensemble model was trained using an altered optimizer configuration and a learning rate schedule. The evaluation metrics in this experiment were also RMSE and NLL, which revealed an average test loss of approximately 0.41 and an average NLL of -3.87. Notably, the results from this experiment indicated a decline in performance compared to the first experiment. This observation suggests that the alterations in data handling and model parameterization had significant ramifications for the outcomes. Additionally, the very small standard deviation of 0.02 in NLL signifies stability in uncertainty estimation across multiple runs.

In summary, these experiments provided valuable insights into the effects of data normalization techniques and model configurations on the performance of ensemble neural networks, showing the results in Table 5.16. The slightly superior performance observed in the first experiment highlights the potential benefits of separating feature and target normalization, which may afford the model a more robust representation of the data. In contrast, the second experiment underscores how variations in training procedures and data preprocessing can affect model performance. These findings emphasize the critical role of meticulous preprocessing and hyperparameter selection in achieving optimal results in machine learning applications.

5.7.2 Concrete Dataset

In our analysis of the UCI Concrete dataset, we employed the first experiment's approach, which focused on evaluating the performance of an ensemble neural network model through a structured methodology. This distinction in preprocessing aimed to enhance the model's ability to learn effectively from the data.

The results indicated an average test RMSE of approximately 5.82, along with a standard deviation of 0.45, suggesting that while the model performed reasonably well, there was variability attributed to the random initialization and data shuffling processes. Furthermore, the average NLL was found to be approximately 3.28, accompanied by a standard deviation of 0.13. These results highlight the ensemble method's capability to enhance prediction accuracy while providing reliable estimates of uncertainty.

The observed variability in both the RMSE and NLL metrics reinforces the robustness of our approach, indicating that the ensemble method effectively mitigated the impacts of random factors in the training process. The combination of lower RMSE and higher NLL values supports the assertion that the model not only improved its predictive accuracy but also enhanced the estimation of uncertainty across different runs.

Overall, it demonstrates that the approach taken in the first experiment was beneficial in enhancing the ensemble model's performance on the Concrete dataset.

5.7.3 Year Prediction Dataset

For the Year Prediction dataset, we also used the first experiment's approach to further confirm our assumptions. However, in this case, the training process was executed only once in this instance.

Upon completion of the training, we observed a test RMSE of approximately 8.77, indicating the average deviation of the model's predictions from the actual target values. The standard deviation of the test loss was not applicable since only one run was conducted. In contrast, the NLL was recorded at around 5.05, which serves as an indicator of the model's uncertainty in its predictions. The NLL value indicates that the model provides probabilistic estimates of uncertainty, which is beneficial for understanding the confidence of its predictions.

The results mentioned above highlight the model's ability to produce reasonable predictions for the Year Prediction dataset, although further iterations could improve the reliability of these metrics.

5.7.4 Summary of the Findings

In this section, we explored the impact of different data preprocessing strategies and training configurations on the performance of our ensemble aproach across three distinct datasets. The findings from the Housing dataset indicated that separating feature and target normalization yielded better performance metrics compared to concurrent normalization approaches. Similarly, our analysis of the Concrete dataset reaffirmed the effectiveness of this methodology, showcasing how our approach can enhance predictive accuracy and uncertainty estimation. Although the Year Prediction dataset's single-run evaluation did not allow for robust variability analysis, it still demonstrated the model's capability to produce reasonable predictions.

6 Conclusion

This thesis demonstrates the substantial advantages of the Class Ensemble method in addressing the challenges of uncertainty estimation in artificial neural networks, particularly in resource-constrained environments. The Class Ensemble model consistently outperforms or is competitive to traditional ensemble approaches such as Deep Ensembles and Monte Carlo (MC) Dropout in terms of both robustness and calibration under noisy conditions. The architectural innovation of shared convolutional layers across ensemble members enables the model to maintain high accuracy and low ECE even as noise levels increase. Unlike methods that rely on independent models, such as Deep Ensembles, which are susceptible to overfitting specific noise patterns, the Class Ensemble method benefits from a shared feature extractor that enhances generalization capabilities, particularly when dealing with out-of-distribution data.

This research highlights the ability of the Class Ensemble model to balance computational efficiency with high-quality uncertainty estimation. Its shared backbone architecture minimizes memory and computational overhead, making it highly suitable for deployment in real-time and resource-constrained systems. The Class Ensemble method demonstrated great performance across various tasks, including classification and segmentation, consistently producing lower ECE scores and higher entropy and Jensen-Shannon Divergence (JSD) values compared to traditional methods. Additionally, its resilience in handling noisy data makes it particularly advantageous in fields such as autonomous driving and healthcare, where reliable predictive confidence is essential for ensuring safety and accuracy.

In segmentation tasks, particularly with the U-Net model applied to the DRIVE dataset, the Class Ensemble method showed a notable improvement in Dice score and calibration. The model's ability to handle both class imbalance and data scarcity further emphasizes its potential for real-world applications where data is often imperfect or scarce. These findings suggest that the Class Ensemble method could be a valuable tool for advancing uncertainty-aware machine learning models in high-stakes environments, offering a practical and efficient solution for handling uncertainty in complex, noisy data scenarios.

6.1 Future investigations

Future work could focus on several key areas to extend the capabilities and applicability of the Class Ensemble method. One promising direction is the exploration of techniques for reducing memory usage during training and inference. Approaches such as quantization and pruning could significantly enhance the method's deployment on ultra-low-power devices, including those used in IoT, wearable technology, and edge computing platforms. These optimizations would further reduce the energy footprint of the method while maintaining high predictive performance.

Additionally, expanding the application of the Class Ensemble method to more complex and diverse tasks, such as object detection, natural language processing (NLP), and time-series forecasting, would provide insights into its versatility and generalizability.

Testing the method in real-time systems, particularly in safety-critical domains such as autonomous vehicles and healthcare, will be crucial for validating its suitability in environments where latency and reliability are critical.

Furthermore, extending the scope of this research to domains characterized by high data variability, such as finance and climate modeling, would test the limits of the Class Ensemble method's adaptability. These domains often involve highly dynamic and unpredictable data streams, which would provide a challenging yet valuable environment for assessing the method's capacity to generalize across a wide range of inputs. Investigating the application of the Class Ensemble method in these contexts would not only broaden its applicability but also push the boundaries of uncertainty-aware machine learning, helping to improve predictive confidence in critical, real-world operational settings.

Bibliography

[1] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[2] @teslafacts5168. (n.d.). *Tesla camera detection has humour (by tesla.fsd) Shorts* Retrieved September 14, 2024. https://www.youtube.com/shorts/Zbkk1Xd616I

[3] Edward Helmore. *Tesla's self-driving technology fails to detect children in the road, group claims*. The Guardian. [August 9, 2022]. Retrieved September 14, 2024. https://www.theguardian.com/technology/2022/aug/09/tesla-self-driving-technology-safety-children

[4] Times Now. (n.d.). Viral video: Tesla struggling to identify horse-drawn carriage mistakes it for truck. Retrieved September 14, 2024. https://www.timesnownews.com/videos/viral-videos/viral-video-tesla-struggling-to-identify-horse-drawn-carriage-mistakes-it-for-truck-video-93640374

[5] Patrick Boyle. Is it cancer? Artificial intelligence helps doctors get a clearer picture. AAMC News. [March 28, 2024]. Retrieved September 16, 2024. https://www.aamc.org/news/it-cancer-artificial-intelligence-helps-doctors-get-clearer-picture

[6] Begoli, E., Bhattacharya, T., & Kusnezov, D. (2019). The need for uncertainty quantification in machine-assisted medical decision making. *Nature Machine Intelligence*, 1(1), 20-23.

[7] Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd ed.). Prentice Hall, 24-43.

[8] Bre, Facundo & Gimenez, Juan & Fachinotti, Víctor. (2017). Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks. In *Energy and Buildings*. 158. 10.1016/j.enbuild.2017.11.045.

[9] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[10] Depeweg, S., Hernández-Lobato, J. M., Doshi-Velez, F., & Udluft, S. (2018). Decomposition of uncertainty in Bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning* (pp. 1184–1193).

[11] Kendall, A., & Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems* (pp. 5574–5584).

[12] Lobez Urquía, J.; Casa Aruta, E. (1975). Estadística intermedia. Segunda edición. In *Vicens-Vives*.

[13] Hellman, M.; Raviv, J. (1970). Probability of error, equivocation, and the Chernoff bound. In *IEEE Transactions on Information Theory*. 16 (4): 368–372. CiteSeerX 10.1.1.131.2865. doi:10.1109/TIT.1970.1054466.

[14] Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems* (pp. 6402–6413).

[15] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929-1958.

[16] Gal, Y., & Ghahramani, Z. (2016). Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning* (pp. 1050–1059).

[17] Malinin, A., & Gales, M. (2018). Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems* (pp. 7047–7058).

[18] Gal, Yarin. (2016). Uncertainty in Deep Learning. University of Cambridge, United Kingdom.

[19] Ghanem, Roger & Higdon, David & Owhadi, Houman. (2017). Handbook of Uncertainty Quantification. In *Springer*.

[20] Guy Kendall, Alex. (2019). Geometry and Uncertainty in Deep Learning for Computer Vision. University of Cambridge, United Kingdom.

[21] Malinin, Andrey & Gales, Mark. (2019). Reverse KL-Divergence Training of Prior Networks: Improved Uncertainty and Adversarial Robustness. Cornell University, United States of America.

[22] Wang, Hao & Yeung, Dit-Yan. (2016). A Survey on Bayesian Deep Learning. Cornell University, United States of America.

[23] Lakshminarayanan, Balaji & Pritzel, Alexander & Blundell, Charles. (2016). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. Cornell University, United States of America.

[24] Stahl, Niclas & Falkman, Göran & Karlsson, Alexander & Mathiason, Gunnar. (2020). Evaluation of Uncertainty Quantification in Deep Learning. In *Springer Nature*.

[25] K. Gustafsson, Fredrik & Danelljan, Martin & B. Schön, Thomas. (2019). Evaluating Scalable Bayesian Deep Learning Methods for Robust Computer Vision. Cornell University, United States of America.

[26] Hüllermeier, Eyke & Waegeman, Willem. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. In *Springer Link*.

[27] Abdar, Moloud & Pourpanah, Farhad & Hussain, Sadiq & Rezazadegan, Dana & Liu, Li & Ghavamzadeh, Mohammad & Fieguth, Paul & Cao, Xiaochun & Khosravi, Abbas & Rajendra Acharya, U. & Makarenkov, Vladimir & Nahavandi, Saeid. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. In *Information Fusion*.

[28] Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.

[29] Lim, Seung-Hwan & Young, Steven & Patton, Robert. (2016). An analysis of image storage systems for scalable training of deep neural networks.

[30] Handral, Praneeta & Kulkarni, Ritika & MD, Swapna & Kumar, Nikhil. (2022). CIFAR-10 Image Classification with Convolutional Neural Networks. In *International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)*. 89-94

[31] Guo, Fan & Li, Weiqing & Kuang, Zhonghao & Tang, Jin. (2021). MES-Net: a new network for retinal image segmentation. Multimedia Tools and Applications. 80. 10.1007/s11042-021-10580-1.

[32] Burle, Marie-Hélène. Example: classifying the MNIST dataset. `https://mint.westdri.ca/ai/pt/pt_mnist`

[33] Yoo, Seung Hoon & Geng, Hui & Chiu, Tin Lok & Yu, Siu & Cho, Dae & Heo, Jin & Choi, Min & Choi, Il & Van, Cong & Nhung, Nguen & Min, Byung Jun & Lee, Ho. (2020). Deep Learning-Based Decision-Tree Classifier for COVID-19 Diagnosis From Chest X-ray Imaging. Frontiers in Medicine. 7. 427. 10.3389/fmed.2020.00427.

[34] Jang, Keyongseok & Ha, Sung & Son, Kwang. (2020). ATTENTION-BASED SALIENT OBJECT DETECTION USING EDGE INFORMATION AND TEXTURE INFORMATION. JP Journal of Heat and Mass Transfer. SP. 1-10. 10.17654/HMSIII20001.

[35] Natakarnkitkul, Satsawat. (2019). Evaluation Metrics: Reference Guides. `https://medium.com/@net_satsawat/evaluation-metrics-reference-guides-7c3a2a055351`

[36] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

[37] San Biagio, Marco & Martelli, Samuele & Crocco, Marco and Cristiani, Marco & Murino, Vittorio. (2014). Encoding structural similarity by cross-covariance tensors for image classification. In *International Journal of Pattern Recognition and Artificial Intelligence*.

Appendix

Method	Severity	Accuracy	NLL	Brier	Entropy	ECE	MCE	RMSCE
Batch	0.0	98.350	1.488	0.026	0.068	0.005	0.338	0.016
	0.2	67.960	1.785	0.470	0.376	0.176	0.310	0.192
	0.4	17.220	2.268	1.538	0.171	0.768	0.842	0.787
	0.6	33.960	2.098	1.126	0.315	0.541	0.647	0.569
	0.8	17.180	2.268	1.536	0.172	0.764	0.840	0.786
	1.0	11.430	2.335	1.713	0.082	0.861	0.885	0.868
Branch	0.0	99.280	1.469	0.012	0.006	0.005	0.287	0.021
	0.2	99.200	1.469	0.012	0.007	0.005	0.613	0.019
	0.4	98.710	1.474	0.020	0.015	0.007	0.358	0.022
	0.6	95.150	1.516	0.077	0.070	0.023	0.269	0.041
	0.8	81.030	1.652	0.292	0.193	0.112	0.318	0.138
	1.0	60.670	1.849	0.631	0.313	0.266	0.458	0.281
Class	0.0	99.310	4.315	0.012	0.007	0.005	0.631	0.026
	0.2	99.210	4.315	0.013	0.008	0.005	0.440	0.025
	0.4	98.860	4.320	0.017	0.017	0.005	0.410	0.025
	0.6	95.660	4.357	0.062	0.078	0.013	0.742	0.026
	0.8	80.100	4.522	0.310	0.247	0.115	0.312	0.144
	1.0	56.090	4.757	0.711	0.387	0.304	0.820	0.318
Deep	0.0	99.440	1.469	0.009	0.014	0.002	0.557	0.010
	0.2	99.360	1.470	0.010	0.017	0.001	0.312	0.010
	0.4	99.110	1.475	0.014	0.038	0.003	0.290	0.010
	0.6	96.270	1.526	0.052	0.194	0.025	0.137	0.044
	0.8	76.730	1.740	0.329	0.554	0.045	0.098	0.057
	1.0	45.310	2.008	0.796	0.714	0.297	0.521	0.333
MC	0.0	99.330	1.470	0.012	0.011	0.005	0.438	0.025
	0.2	99.200	1.471	0.014	0.012	0.005	0.358	0.021
	0.4	98.760	1.474	0.017	0.022	0.004	0.307	0.015
	0.6	96.880	1.500	0.046	0.070	0.007	0.133	0.016
	0.8	88.170	1.596	0.171	0.225	0.035	0.170	0.045
	1.0	73.590	1.749	0.390	0.430	0.108	0.233	0.122

Table A.1: Performance Metrics on MNIST Model for different Ensemble Methods across Severity Levels