PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education And Scientific Research

University of Tlemcen – ABU BAKR BELKAID - Algeria
FACULTY OF SCIENCE
DEPARTMENT of COMPUTER SCIENE

Graduation Project

To obtain a Master's degree in Computer Science

Specialty: Intelligent Model and Decision (M.I.D)

On the subject:

Bayesian Deep Learning for Limited Data Prediction

Presented by:

SAHRAOUI Tarek Ziad

Supervised by:

Mrs. CHAOUCHE RAMDANE Lamia

Examined on June 13 by:

Mr. SMAHI Mohamed Ismail (President)

Mrs. BENMAHDI Meriem Bouchra (Examiner)

Academic year: 2023/2024

Dedication

“

IN MEMORY OF my beloved mother, whose unwavering love and support have been the cornerstone of my endeavors. Though she is no longer with us, her spirit resides in my heart, wishing she could share in this moment of achievement with me.

TO my dear second family, the Hadjaj, for embracing me as one of their own. Their warmth and encouragement have been a constant source of strength.

To my esteemed mentors Zakaria and Youcef, your guidance and inspiration have been instrumental in keeping me on course, even when the path seemed uncertain. Your presence has instilled in me the drive to strive for excellence.

To my cherished friends who have stood by me through thick and thin, your presence in my life has been a blessing. Each of you has played a unique role in my journey, and I treasure the memories and moments we have created together.

”

- Ziad

Acknowledgement

FIRST AND FOREMOST, I thank Allah Almighty for giving me the strength and courage and patience to complete this modest work.

I extend my Mrs. CHAOUCHE RAMDANE heartfelt gratitude to my supervisor.

i am also deeply thankful for your guidance, unwavering support and constructive critiques, which contributed immensely the successful completion of this work.

I also thank the jury members Mr.SMAHI and Mrs.BENMAAHDI who agreed to read and evaluate this work.

I express my heartfelt gratitude to all the individuals who have directly or indirectly contributed to this work. Your contributions, whether big or small, have played a significant role in shaping our academic journey and the successful completion of this research. May Allah's blessings be upon you all.

Contents

List of Figures

List of Tables

List of acronyms and acronyms

AI	Artificial Intelligence
BADL2	Block Attention Deep List Learning
CNN	Convolutional Neural Network
DLS	Deep Learning Systems
GANs	Generative Adversarial Networks
KL	Kullback-Leibler
LSTM	Long Short-Term Memory
MCMC	Markov Chain Monte Carlo
MCD	Monte Carlo Dropout
ML	Machine Learning
MSE	Mean Squared Error
NN	Neural Network
RNN	Recurrent Neural Network
VAEs	Variational Autoencoders
VI	Variational Inference
T.M	Traditional Models

General Introduction

Contexte

In just a few recent years, the research into how to train Neural Networks (NN) such as Deep Learning (DL) has seen a surge to great heights, but the extracted use cases could be seen throughout multiple area of excellences, including computer vision, natural language processing (NLP), and autonomous systems. It is these technologies that underpin artificial intelligence (AI) allowing machines to do things that would require intelligence if done by humans.

Deep Learning: A subfield of machine learning that uses artificial neural network structures, many layers deep. Experience-trained algorithms that can generalize their learning from examples provided via experience.

DL algorithms demand large datasets and major computing power to learn complex patterns and representations from data.

Problematic

Even with the progress in deep learning, there's still a big problem: not enough labeled data. In real-life situations, we often don't have enough data, which makes it hard to train good models. Getting large, high-quality labeled datasets is expensive and takes a lot of time. This can lead to models that don't work well and can't handle new data.

Another issue is that deep learning predictions often ignore uncertainty, which is important for reliable results. This project looks into Bayesian Deep Learning (BDL) to solve these problems. BDL helps by using prior knowledge and updating beliefs based on new data, which helps in understanding uncertainty. This is very useful in areas like medical diagnosis and finance, where knowing how uncertain a prediction is can help in making better decisions.

Contribution

Our goal with this project is to create a sophisticated framework using Convolutional Neural Networks (CNNs) that applies Bayesian Deep Learning principles. Essentially, we

want to make better predictions and estimate uncertainty even when we have limited data to work with.

To achieve this, we'll be using advanced techniques like variational inference and Monte Carlo dropout. These methods help us get a handle on Bayesian posterior distributions within neural networks, giving us a clearer picture of our predictions and their associated uncertainties.

Plan of the Thesis

In this study, the main objective is to investigate the effectiveness of Bayesian Deep Learning methods in improving prediction accuracy and uncertainty estimation with limited data.

The present work is structured around three chapters:

1. **Chapter 1 :Introduction to Neural Networks and Deep Learning:** This chapter introduces neural networks and deep learning, explaining basic concepts and architectures.
2. **Chapter 2 :Theoretical Foundations of Bayesian Deep Learning:** This chapter discusses the theoretical foundations of Bayesian deep learning, including statistical terms and concepts.
3. **Chapter 3: Application:** The final chapter compares the implemented models and explores future research directions in Bayesian deep learning.

Finally, we will conclude this dissertation with a general conclusion and some perspectives.

Chapter I

Introduction to Neural Networks and Deep Learning

I.1 Introduction

Neural networks and deep learning have been two of the recent successes in artificial intelligence, revolutionizing most domains, including computer vision, natural language processing, and pattern recognition. These are advanced computational models inspired by the structure and function of the human brain, providing the capability for learning from large amounts of data and performing complicated functions with outstanding accuracy. In this chapter, we will introduce the basic concepts of neural networks and deep learning. We will discuss the history of neural networks, their biological inspiration, and the basic components that comprise these models. In understanding the principles behind neural networks and deep learning, we can appreciate their importance in modern AI research and their possible contribution to innovation in a variety of applications.

I.2 History of Neural Networks and Biological Inspiration

The concept of artificial neural networks draws inspiration from the biological structure and function of the human brain. This section explores the history of the development of NNs, focusing on the key influences from biology.

Early Inspiration from the Brain (1940s-1960s):

The first attempts to mimic the information processing capabilities of the brain started back in the 1940s. Early important work by McCulloch and Pitts (1943) introduced the first mathematical model of an artificial neuron and thus laid the foundations for the development of neural networks [31]. Donald Hebb's book "The Organization of Behavior" in 1949 introduced a learning rule for artificial neurons based on the concept of synaptic plasticity, a mechanism observed in biological brains [21]. These early models created the foundation for exploring the potential of NNs for pattern recognition tasks.

Challenges and Re-emergence (1970s-1980s):

Despite the initial enthusiasm, during the 1970s, the limitations of computing power and the complexity of training algorithms slowed progress in the field of NNs. The limitations of early models called perceptrons were pointed out by Minsky and Papert in 1969 [32]. Due to these reasons, research interest decreased for several years.

Renewed Interest and Advancements (1980s-Present):

More powerful computers and the introduction of new learning algorithms, such as backpropagation, brought interest in the study of NNs back into the limelight in the 1980s [36]. This period saw a lot of advancements in terms of network architectures; a few of them were multilayer perceptrons and convolutional neural networks.

The Ongoing Influence of Biology

Neural networks take inspiration from biological models. Recent developments in neuromorphic computing try to design hardware mimicking the energy efficiency and parallel processing capabilities of the brain. Research on spiking neural networks, in which the timing of neural activity is incorporated, tries to develop more biologically

realistic models.

I.2.1 Biological Inspiration

The basis of neural networks is grounded in the functioning of the human brain, an intricate network of interlinked *neurons*. These neurons constitute the processing unit and transmit electrical signals and chemical messengers, or *neurotransmitters*, via specialized connections called *synapses*.

The figure below illustrates the simulation of neural networks:

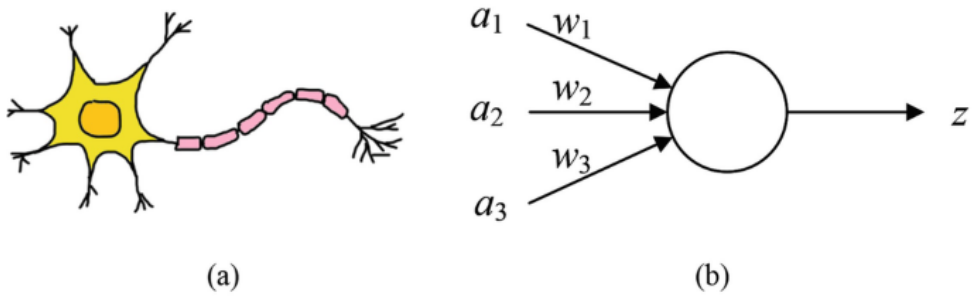

Figure I.1: Simulation of neural networks [40].

I.3 Components of Neural Networks

Artificial neural networks derive their inspiration from the structures and functions of biological neurons in the human brain. They are made of units called neurons, organized into layers that mimic the architectures of the brain's neurons. These networks are capable of learning and adapting by changing the strength of connections between them, which enables them to solve complex problems in diverse domains of activities [18].

I.3.1 Neurons (Nodes)

These are the fundamental units of neural networks, akin to the cells in our brain [18]. Neurons receive inputs, apply transformations through activation functions, and pass the results to the next layer. They play crucial roles in processing information and learning patterns [22].

I.3.2 Layers

Input Layer: These initial layers receive raw data and forward them to the hidden layers for processing. They act as the entry points where external information is ingested into the network [18].

Hidden Layers: Nestled between the input and output layers, these hidden layers perform the bulk of computations and feature extraction. The network's abilities to understand complex relationships and patterns are largely attributed to these layers.

Output Layer: The final layer where the network produces its prediction or output based on the processed information from the hidden layers. It encapsulates the network's decision-making or inference capabilities.

I.3.3 Connections (Weights)

These represent the strength of connections between neurons. Weights determine the impact of one neuron's outputs on another's input. Through training, these weights are adjusted iteratively to improve the network's performance, enabling it to learn and generalize from data. These processes of weight adjustments are crucial aspects of neural network training, contributing significantly to the network's ability to make accurate predictions and solve complex tasks.

I.4 Activation Functions

Non-linearity that provides learning of intricate patterns is achieved by the neurons in the neural network due to the utilization of those activation functions. Among the activation functions, Rectified Linear Unit (ReLU), Sigmoid, and Hyperbolic Tangent (Tanh) are used widely.

I.4.1 Rectified Linear Unit (ReLU)

$$f(x) = \max(0, x)$$

ReLU takes basic functions and produces the inputs themselves if and only if their signs are positive. When they are negative, ReLUs produce zero.

I.4.2 Sigmoid Function

$$f(x) = \frac{1}{1 + e^{-x}}$$

The S-shaped sigmoid function maps input values in the range $(0, 1)$ to the binary classification task, giving it a chance of doing that well.

I.4.3 Hyperbolic Tangent (Tanh)

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

An activation function known as tanh has a different function that squeezes input values into the range $(-1, 1)$. It, in turn, has more gradients than the sigmoid function and therefore offers stronger gradients.

I.5 Beyond Basics: Why Embrace Deep Learning?

Traditional neural networks lay the groundwork, but machine learning offers advanced tools. Deep learning, some subsets of MLS built upon NNs, bring several advantages:

- **Limited Expressive Power**: Basic NNs struggle with complex data. Deep learning captures intricate patterns for improved performance [3].
- **Advancements in Algorithms**: ML and DL research lead to superior algorithms like CNNs, RNNs, attention mechanisms, and transformers models.
- **Scalability**: Deep learning models effectively handle large datasets for better generalization.
- **Complexity of Data**: Modern data's complexities challenge basic NNs. ML and DL, especially deep neural networks, excel in handling complex, high-dimensional data.

Embracing machine learning, especially deep learning, opens the door to mastering intricate data representations, paving the way for robust and adaptable applications.

I.6 Loss Function

The loss function, also known as the cost function, plays a critical role in training neural networks. It serves as a quantitative measure of the discrepancies between the network's predicted output and the ground truth, which represents the actual target values.

I.6.1 Classification Problems

Cross-entropy loss functions are frequently employed to measure the differences between the predicted probability distributions and the true target distributions [22].

I.6.2 Regression Problems

In neural network applications tackling regression problems, the choice of appropriate loss functions is paramount for effective training. These loss functions, such as the widely-used mean squared error (MSE), quantify the disparity between predicted values and actual targets [18]. Selecting the right loss function tailored to the characteristics of the data and the desired outcome is crucial for guiding the network towards convergence and optimal parameterization.

I.7 Optimization Algorithms

In the context of neural networks, optimization algorithms assume a fundamental role in the iterative refinement of internal parameters, namely weights and biases. These algorithms facilitate the minimization of a predefined objective function, typically a loss function representing the dissonance between predicted and actual outputs, across successive iterations of training data processing [22]. By navigating the expansive parameter space, optimization algorithms endeavor to converge towards optimal configurations, thereby enhancing the network's performance and efficacy.

I.8 Learning in Neural Networks: Mimicking the Brain

Neural networks, drawing inspiration from the structural and functional complexities of the human brain, stand as powerful tools for machine learning. Through the emulation of neural architecture, these artificial constructs exhibit a remarkable capacity for learning from data patterns. Encapsulating layered hierarchies of interconnected neurons, akin to the neural circuitry of the brain, neural networks adeptly discern intricate patterns, extract salient features, and infer complex relationships within vast datasets.[37]

I.8.1 Building Blocks

Imagine the brain as a complex network of interconnected neurons. Similarly, artificial neural networks consist of fundamental units called neurons (nodes) [18].

Forward Pass: The network processes its inputs through its layers, with each neuron performing calculations based on its weighted inputs and an activation function. These calculations are analogous to how our brains process information, activating different neurons based on the received stimuli [10].

Error Calculation: The network then compares its predicted outputs with the desired outputs from the training data. This difference, called the loss, represents how wrong the network's predictions were, similar to how we assess our understanding of a concept.

Backpropagation: This is where the magic happens! Inspired by how the brain strengthens or weakens connections based on learning, a technique called backpropagation calculates how much each weight contributed to the error. Imagine students receiving feedback on their mistakes – backpropagation provides similar guidance for the network [35].

Weight Adjustment: Using an optimization algorithm (e.g., gradient descent) [34], the network iteratively adjusts its weights in the direction that minimizes the loss. This is akin to a student adjusting their approach based on the feedback received.

Repeat: The network continues to process training data, calculate errors, and adjust weights. Over many iterations, the network progressively improves its ability to map inputs to desired outputs, just like we learn and refine our skills through practice.

I.9 Deep Learning

Deep learning is a subfield of machine learning that utilizes artificial neural networks with multiple hidden layers to learn complex patterns from data [25].

The Venn diagram below highlights the intersection that constitutes Deep Learning (DL):

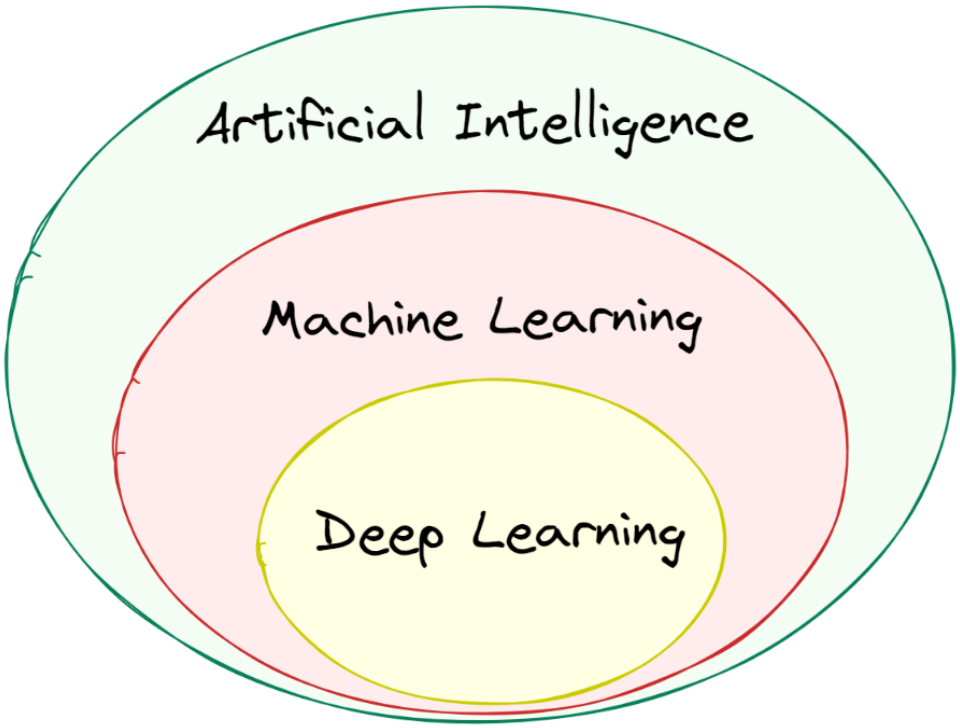

Figure I.2: Venn diagram.

Deep Learning Architectures

Deep Learning Architectures encompass diverse and sophisticated frameworks used in deep learning, including models like **convolutional neural networks (CNNs)** for image analysis, **recurrent neural networks (RNNs)** for sequential data processing, and **attention mechanisms** for natural language tasks.

These architectures are designed to handle complex data structures and extract meaningful patterns, driving advancements in fields such as computer vision, natural language processing, and reinforcement learning.

1. Convolutional Neural Networks (CNNs): CNNs reign supreme in image recognition and related domains [26]. They utilize specialized convolutional layers that

extract features directly from spatial data like images. These networks often have a hierarchical structure, where lower layers extract simpler features, and higher layers combine them for complex recognition.

The illustration provided below serves to elucidate the concept of CNN:

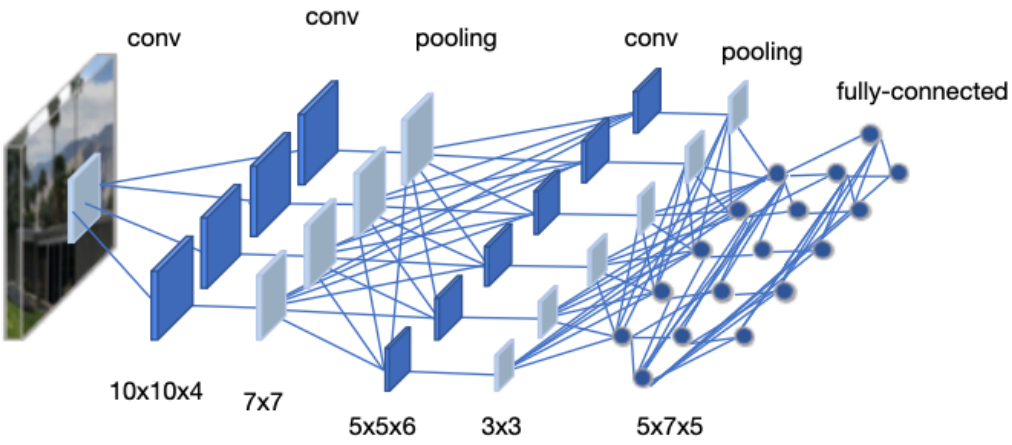

Figure I.3: Convolutional Neural Network Architecture .

2. Recurrent Neural Networks (RNNs): Designed to conquer sequential data like text or time series [36]. Unlike feedforward networks, RNNs have a feedback loop, allowing them to process information based on the context of previous elements in the sequence. This makes them well-suited for tasks like language translation, sentiment analysis, and speech recognition.

The image below depicts the concept of RNN:

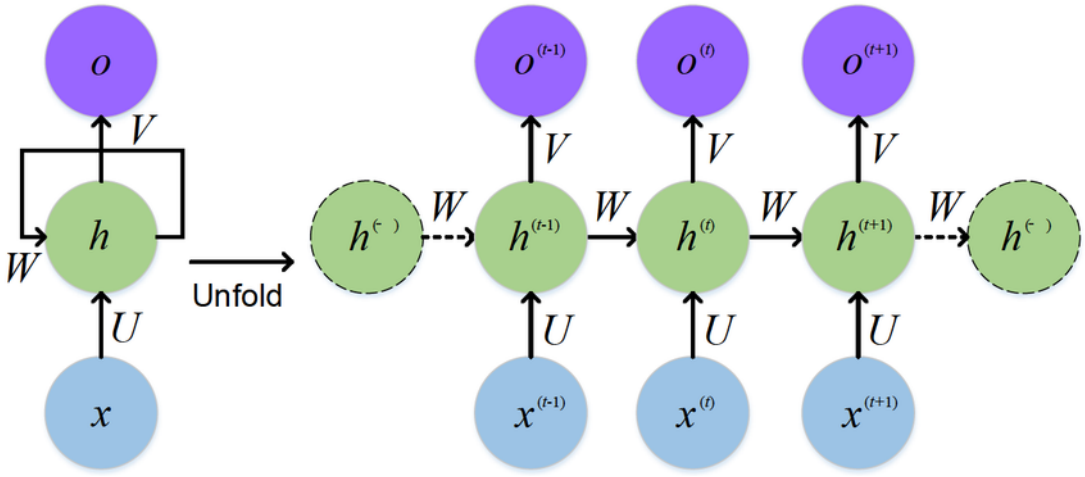

Figure I.4: Recurrent Neural Network Architecture [24].

3. Autoencoders: Used for unsupervised learning tasks like data compression, feature learning, and anomaly detection [11]. While unsupervised pre-training has been shown to benefit deep learning, specific techniques like batch normalization [23] and residual learning [19] have also played a crucial role in improving the training and performance

of deep neural networks, including autoencoders.

The following image illustrates the concept of autoencoders:

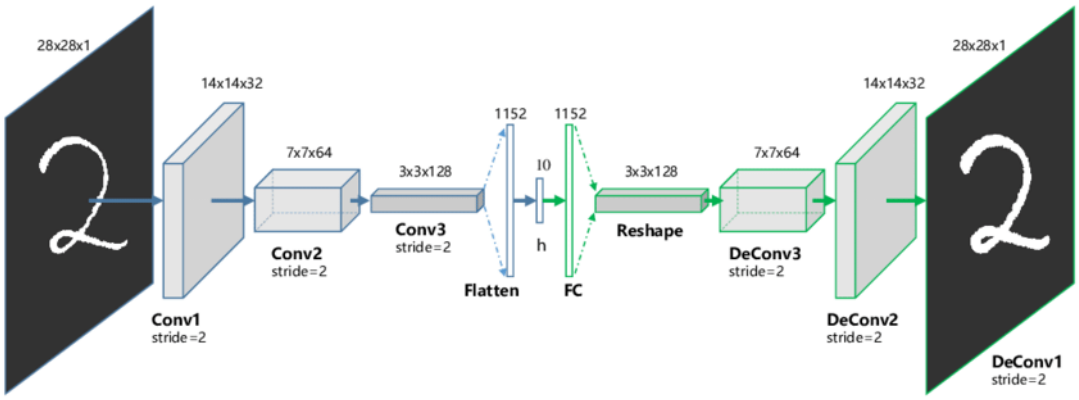

Figure I.5: Autoencoder Architecture. (Inspired by [11, 23, 19])

Additionally, numerous other types of deep neural networks have been developed for specific tasks and applications.

I.9.1 Designing Deep Learning Models

Network Depth: The number of layers stacked in a network is a crucial design decision. Deeper networks can learn more complex relationships within data but can also be prone to overfitting and require more computational resources. Finding the optimal depth depends on the specific task and data complexity [17].

Network Width: The number of neurons within each layer also plays a role. Wider networks generally have a higher capacity for learning complex functions but also increase training time and memory requirements. Striking a balance between network width and depth is essential [20].

Activation Functions: These mathematical functions introduce non-linearity into the network, allowing it to model complex relationships. Choosing the appropriate activation function (e.g., sigmoid, ReLU) can significantly impact the network's performance [33].

Regularization Techniques: Techniques like dropout and weight decay prevent overfitting by improving generalization [39].

Hyperparameter Tuning: Deep learning models involve numerous hyperparameters, such as learning rate and number of epochs. Tuning these hyperparameters through techniques like grid search or randomized search can significantly impact the model's performance [2].

I.10 Specialized Domains in Deep Learning

Deep learning has revolutionized various fields due to its ability to learn complex patterns from large amounts of data.

Here are some prominent applications:

I.10.1 Medical Imaging and Healthcare

Image Segmentation: Deep learning, especially convolutional neural networks (CNNs), excels at fast and accurate image segmentation tasks. This is crucial for tasks like identifying tumors in mammograms or segmenting organs in MRI scans [27].

Disease Diagnosis: Deep learning algorithms can analyze medical images to identify disease patterns and support medical professionals in diagnosis. For example, they can help differentiate cancerous and benign tissues or detect abnormalities in X-rays [12].

Medical Image Synthesis: Generative adversarial networks (GANs) and variational autoencoders (VAEs) have the potential to create synthetic medical images for data augmentation and training [13]. This can be particularly useful for rare diseases where real data is limited.

I.10.2 Natural Language Processing (NLP)

Text Generation: Deep learning models like GPT-3 are becoming increasingly sophisticated at generating human-quality text. This has applications in areas like creative writing, content automation, and chatbot development.

Sentiment Analysis: LSTM-based models can analyze the sentiment expressed in textual data, providing valuable insights for social media monitoring, customer feedback analysis, and market research [29].

Machine Translation: Deep learning architectures like the Transformer have significantly improved the accuracy and fluency of machine translation, facilitating communication across languages [42].

I.10.3 Climate Modeling and Robotics

Autonomous Vehicles: Deep learning combined with LiDAR sensor fusion enables object detection, lane detection, and pedestrian detection in autonomous driving systems, playing a crucial role in developing safe and reliable self-driving cars [5].

Robotics: Deep learning algorithms are used for tasks like object manipulation, robotic vision, motion planning, and control. Reinforcement learning techniques further enhance the ability of robots to learn and adapt to their environments [38].

I.11 Comparative Analysis of Neural Networks and Deep Learning

In this section, we provide a comparative analysis of traditional neural networks (NNs) and deep learning (DL) models based on various criteria such as architecture, training process, and applications.

Comparative Tables

We now present detailed comparisons between traditional neural networks (NNs) and deep learning (DL) models as the following:

I.11.1 Architecture

Comparison of Architecture

Model Traditional NNs

Description Single input layer, one or more hidden layers, and an output layer.

Model Deep Learning (DL) Models

Description Many hidden layers with complex architectures like convolutional layers, recurrent layers, and attention mechanisms.

I.11.2 Training Process

Comparison of Training Process

Model Traditional NNs

Training Process Backpropagation updates weights and biases based on error between predicted and actual outputs.

Model Deep Learning (DL) Models

Training Process Require more data and computational resources; use techniques like dropout, batch normalization, and gradient clipping for improved convergence and prevention of overfitting.

I.12 Addressing the Limitations of Deep Learning with Bayesian Deep Learning

Despite its significant benefits, deep learning suffers from several limitations, including data hunger, lack of interpretability, overfitting and generalizability issues, and uncertainty quantification challenges. Bayesian Deep Learning (BDL) presents a promising solution to overcome these drawbacks by offering a principled framework for uncertainty quantification and model robustness.

I.12.1 Data Hunger

Deep learning models often demand extensive datasets for training, which can be both challenging and costly to obtain and label [25]. This data hunger poses a significant barrier, especially in domains with limited labeled data availability or high labeling costs. Bayesian Deep Learning addresses this limitation by leveraging probabilistic models that incorporate prior knowledge and uncertainty. By modeling uncertainty, BDL can effectively utilize limited data, reducing the dependency on large datasets while still providing reliable predictions.

I.12.2 Lack of Interpretability

The inherent complexity of traditional deep learning models often leads to interpretability issues, hindering trust and adoption, particularly in critical applications [17]. The opaque nature of deep neural networks makes it challenging to understand the rationale behind their predictions. Bayesian Deep Learning enhances interpretability by providing uncertainty estimates alongside predictions. By quantifying uncertainty, BDL allows decision-makers to evaluate the reliability of model predictions and comprehend the underlying factors influencing them, thereby fostering trust and transparency in the model.

I.12.3 Overfitting and Generalizability

Deep learning systems are prone to overfitting, where they perform well on training data but struggle to generalize to new data [25]. While techniques like data augmentation and regularization can alleviate this issue to some extent, they may not always suffice. Bayesian Deep Learning combats overfitting by explicitly modeling uncertainty in model parameters. By capturing uncertainty, BDL encourages more cautious predictions, leading to enhanced generalizability and resilience to unseen data.

I.12.4 Uncertainty Quantification

Quantifying uncertainty is crucial for decision-making tasks, especially in critical applications like medical diagnosis [25]. Deep learning models often struggle with uncertainty quantification, which can lead to erroneous decisions. Bayesian Deep Learning excels

in uncertainty quantification by providing probabilistic predictions that encapsulate the inherent uncertainty in the data and model parameters. By quantifying uncertainty, BDL empowers decision-makers to make well-informed decisions, particularly in scenarios where errors can have significant consequences.

I.13 Conclusion

the empirical examination of deep learning underscores the critical need to address challenges such as data scarcity, interpretability limitations, overfitting, and uncertainty quantification. These hurdles impede the broader adoption and effectiveness of deep learning in practical applications. In the subsequent chapter, we will explore statistical and Bayesian methods, which offer promising avenues for overcoming these obstacles. By integrating these methodologies into deep learning frameworks, we aspire to bolster model reliability, enhance interpretability, and foster innovation across diverse domains.

Chapter II

Theoretical Foundations of Bayesian Deep Learning

II.1 Intoduction

New ways of learning have changed many things. In the past, we used specific numbers to make predictions, but we may not always be correct. In addition, it may be difficult to find enough information to make accurate predictions.

This chapter describes methods called Bayesian learning, which help to make better guesses and understand how certain they are of their predictions. Using Bayesian principles, these programs can make better decisions and solve difficult problems. This helps us learn more and solve difficult issues.

In the next parts of this chapter, we'll explain the basics of Bayesian deep learning. We'll talk about important parts like prior beliefs, how likely something is to happen, and final outcomes. By understanding these ideas, we can better predict outcomes, use what we already know, and get a clearer idea of how certain we are about our predictions in Bayesian deep learning.

II.2 Navigating Deep Learning Challenges with Limited Data

Deep learning models have shown amazing abilities in many areas. But a big problem comes up when there isn't enough data [22]. These models need a lot of labeled data to work well. Here, we look at the challenges of having limited data and ways to solve them.

II.2.1 Difficulties Posed by Limited Data

Overfitting: When there's not enough training data, models can overfit [4]. This means the model remembers the training data patterns instead of learning general features. It's like a student who only studies old test questions for a new course – they might do well on that test but have trouble with new questions that need a broader understanding. Overfitting models also do poorly on new data that's different from the training set.

High Variance: Limited data can make models have high variance. This means small changes in the training data can lead to very different models and predictions [16]. This makes it hard to judge the model's true performance and generalize to new data.

Limited Generalizability: Deep learning models aim to work well on unseen data. But with limited data, models might not learn the main patterns well, leading to poor performance on new data that wasn't in the training set [22].

II.2.2 Navigating Uncertainty

Deep learning models have changed many fields by learning complex patterns from data. But traditional deep learning often uses **point estimates** for model parameters [22].

This means finding one set of weights that best minimizes the loss function. While this works well in many cases, relying on point estimates has limits due to the **uncertainty**

in real-world data and the learning process itself [4].

Here's why point estimates can be a problem:

Overconfidence:

A point estimate gives one "best guess" for the output, ignoring the uncertainty in the data and learning process. This can lead to **overconfidence** in the model's predictions [14].

Think of a student who scores perfectly on a practice test but fails the final exam because they were too confident in their knowledge – point estimates can cause similar issues in models, especially with limited data or hard problems.

Lack of Robustness:

Real-world data often has noise and variability. A model that only uses a point estimate might not handle these variations well, leading to unreliable predictions on new data that's a bit different from the training data [22]. This is critical in fields like medical diagnosis or self-driving cars where small mistakes can be very serious.

II.2.3 Challenges Posed by Uncertainty

Uncertainty is always a factor in real-world data and problems. Here's how it can challenge deep learning models:

Model Complexity: As deep learning models get more complex with more layers and parameters, the uncertainty in their predictions can also grow [14]. This shows the need for methods that go beyond a single point estimate.

II.2.4 The Importance of Uncertainty Quantification

Given these challenges, it's crucial to measure the uncertainty in deep learning model predictions. This helps with:

More Informed Decisions: By understanding the range of possible outcomes and their chances, we can make better and more reliable decisions, especially in high-stakes areas where even small mistakes can be very serious [14].

Improved Generalizability: Models that account for uncertainty might generalize better to new data by considering the natural variability in real-world situations [4].

II.3 Introduction to Bayesian Deep Learning

Deep learning has changed many fields by learning complex patterns from data. However, traditional deep learning often uses point estimates for model parameters, which limits how well it can handle uncertainty. This section introduces Bayesian Deep Learning (BDL), a framework that combines Bayesian statistics with deep learning models.

II.3.1 Leveraging Bayesian Statistics

Bayesian Deep Learning addresses these limitations by using Bayesian statistics. Here's how it works:

Prior Distribution: BDL uses a prior distribution [41] $P(\theta)$ to represent our initial belief about the model parameters θ before observing any data. This prior can be informative (based on existing knowledge) or non-informative (e.g., uniform distribution) depending on the problem. The choice of prior reflects our assumptions about the parameters' likely values before seeing the data.

Likelihood Function: The likelihood function [8] $P(D|\theta)$ measures how likely the observed data D is for different parameter settings θ. It shows the relationship between the data and the model by evaluating the probability of the data given specific parameter values.

Posterior Distribution: Using Bayes' theorem, BDL combines the prior distribution $P(\theta)$ with the likelihood function $P(D|\theta)$ to get the posterior distribution $P(\theta|D)$:

$$P(\theta|D) = \frac{P(D|\theta) \cdot P(\theta)}{P(D)} \tag{II.1}$$

This posterior distribution [44] represents our updated belief about the model parameters after considering the training data D.

II.3.2 The Power of BDL: Uncertainty Quantification

By going beyond point estimates, BDL gives a richer understanding of the model's predictions through the posterior distribution. This distribution shows not just a single "best guess" but a range of possible values for the model parameters and their probabilities. This allows for:

Uncertainty Quantification: BDL lets us measure the uncertainty in model predictions, which is crucial for making strong and reliable decisions, especially in high-stakes applications.

Improved Generalizability: By considering parameter uncertainty, BDL models can potentially generalize better to new data compared to traditional methods.

Leveraging Prior Knowledge: BDL can use prior knowledge about the problem through informative priors, which can improve performance with less data.

II.3.3 Variational Inference

In Bayesian Deep Learning (BDL), calculating the posterior distribution can be computationally hard, especially for complex models. This subsection introduces variational inference (VI), a method to approximate the posterior distribution in BDL [6].

Challenges of Exact Inference

Intractability: For many BDL models, directly calculating the posterior distribution using Bayes' theorem is computationally expensive or intractable. This is due to the complex nature of the likelihood function and the high dimensionality of the parameter space [4].

Sampling Inefficiency: Methods like Markov Chain Monte Carlo (MCMC) can sample from the posterior distribution, but they can be slow and need many samples for accurate results [7].

Variational Inference: The Approximation Game

VI offers a solution by approximating the true posterior distribution with a simpler one, called the variational distribution. Here's the core idea:

Define a Variational Distribution: We select a family of simpler distributions (e.g., Gaussian distributions) and define a variational distribution within this family. The parameters of this distribution will be the new variables to optimize.

Minimize the KL Divergence: We aim to find the variational distribution that is closest to the true posterior in terms of information content. This closeness is measured using the Kullback-Leibler (KL) divergence, which quantifies the difference between two probability distributions [9].

Optimize the Variational Parameters: By minimizing the KL divergence, we optimize the parameters of the variational distribution using an iterative algorithm.

Algorithmic Details of Variational Inference

VI approximates the posterior distribution in Bayesian Deep Learning (BDL) [6]. Here's a step-by-step look at how VI works:

1. **Define the Variational Distribution**:

   Select a family of simpler distributions (e.g., Gaussian distributions) for the variational distribution. This choice affects the efficiency and accuracy of the approximation.

2. **Parameterize the Variational Distribution**:

   Introduce parameters (e.g., mean and standard deviation for Gaussians) to define the specific form of the variational distribution. These parameters will be optimized during VI.

3. **Optimize the Variational Parameters**:

   Iteratively optimize the parameters of the variational distribution to maximize the ELBO, which minimizes the KL divergence and brings the variational distribution closer to the true posterior. Various optimization algorithms, such as stochastic gradient descent (SGD), can be used for this purpose.

Key Considerations:

- The choice of the variational distribution family impacts the efficiency and accuracy of VI. Common choices include Gaussians and mean-field approximations.
- The optimization process might not always reach the global optimum. Techniques like good starting points or annealing can help improve convergence.

II.3.4 Monte Carlo Dropout

Monte Carlo Dropout (MC Dropout) is an efficient technique for Bayesian Deep Learning (BDL) within the realm of variational inference (VI) [14]. This method uses the randomness of dropout, a regularization technique in deep learning, to perform approximate Bayesian inference.

Dropout as a Bayesian Proxy

Dropout in Deep Learning: During training, dropout randomly drops out a certain percentage of neurons and their connections in each layer of a neural network. This prevents overfitting by making the network learn robust features that aren't reliant on specific neurons.

MC Dropout Connection:

Applying dropout at test time during multiple forward passes through the network can be seen as a form of VI. Here's why:

Dropout Injects Uncertainty: The random dropout process introduces uncertainty into the network's predictions. Each forward pass with dropout is like a sample from an ensemble of thinned networks.

Variational Distribution: By averaging the predictions from multiple dropout passes, we approximate the variational distribution. This distribution captures the uncertainty in the model's predictions due to dropout.

Algorithmic Details of Monte Carlo Dropout

MC Dropout [43] uses the randomness of dropout to perform approximate Bayesian inference. Here's how it works:

1. **Forward Passes with Dropout**: Perform multiple forward passes through the trained model during test time.
    - In each pass, apply the dropout mask independently, dropping out a percentage of neurons and their connections in each layer. This simulates an ensemble of thinned networks.
2. **Averaging Predictions**: Average the predictions from each forward pass. This average serves as an estimate of the true prediction, considering the uncertainty introduced by dropout.

3. **Interpretation as Variational Inference**: The dropout process in each forward pass samples from an ensemble of thinned networks. By averaging the predictions, we approximate the variational distribution, capturing the uncertainty in the model's predictions due to dropout [1].

Key Considerations:

- The number of dropout passes is crucial. More passes lead to a more accurate approximation of the variational distribution but increase computational cost.
- MC Dropout uses the dropout rate from training. Use the same dropout rate for both training and MC Dropout at test time.

Limitations of MC Dropout

- **Approximation Accuracy:** The quality of the approximation depends on the number of dropout passes. More passes improve accuracy but increase computational cost.
- **Calibration Issues:** MC Dropout predictions might not always be perfectly calibrated, meaning the predicted confidence may not accurately reflect true uncertainty.

II.3.5 Bayesian Approximation Dropout with L2 (BADL2)

While Monte Carlo Dropout is convenient for approximate Bayesian inference, a more theoretically grounded approach is **Bayesian Approximation Dropout with L2 (BADL2)** [30]. This method combines dropout and L2 regularization for uncertainty quantification in deep learning models.

BADL2: Algorithmic Process

BADL2 involves training and post-training steps. Here's how it works:

Training Phase:

- **Model Architecture**: Define the model with dropout layers at strategic points (e.g., after convolutional layers in CNNs).
- **Dropout Rate**: Set a dropout rate (e.g., 0.5) for training.
- **L2 Regularization**: Add an L2 regularization term to the loss function. This penalizes large weights, promoting smoother weight distributions.

Uncertainty Quantification (After Training):

- **Dropout Probabilities**: Use the same dropout probabilities as during training.

- **L2 Regularization Hyperparameter**: Use the same L2 regularization hyperparameter from training.
- **Posterior Distribution Calculation**: Use the dropout probabilities and L2 regularization to compute the posterior distribution over the weights. This distribution reflects the uncertainty in the model's predictions due to dropout and L2 regularization.

Key Considerations:

- The dropout rate and L2 regularization hyperparameter are crucial for BADL2's performance. Tuning these can affect the model's generalization and uncertainty quantification.
- While BADL2 offers a more grounded approach than MC Dropout, it might still face challenges in perfectly calibrating predicted confidence with true uncertainty.

II.4 Dropout Layers for Approximate Bayesian Inference

Dropout layers, commonly used in deep learning, not only improve model performance but also offer insights into model uncertainty. Recent research suggests that dropout can approximate Bayesian inference, providing valuable uncertainty estimation. This section explores the connection between dropout and Bayesian inference, highlighting how dropout implicitly performs model averaging and uncertainty estimation.

Theoretical Underpinnings

Dropout as Probabilistic Weighting: Dropout during training introduces a Bernoulli distribution over the network weights. This distribution reflects the probability of a weight being dropped out during a training pass.

L2 Regularization and Uncertainty: L2 regularization, penalizing large weights, acts as a prior distribution on the weights. Smoother weight distributions, favored by L2 regularization, correspond to lower model uncertainty.

Synergy of Dropout and L2

By combining dropout and L2 regularization, a connection between dropout and the Bayesian framework is established. This allows for calculating the posterior distribution over the weights, capturing the uncertainty in the model's predictions.

Benefits and Limitations

- **Benefits**: BADL2 provides a rigorous theoretical justification for dropout in Bayesian inference and enables uncertainty quantification in deep learning models.

- **Limitations**: Calculating the exact posterior distribution with BADL2 can be computationally expensive for complex models, and calibration issues similar to MC Dropout may arise [30].

Beyond BADL2

Ongoing research explores alternative approaches and extensions to improve the accuracy and efficiency of Bayesian inference techniques.

II.4.1 Dropout as Model Averaging

Dropout, by randomly dropping neurons during training, approximates ensemble model averaging. Here's how it works:

- Dropout creates a thinned network during training by randomly setting neurons to zero with a probability p.
- Each training pass utilizes a different thinned network, akin to training multiple slightly different models.
- At test time, the weights implicitly capture the average behavior of the thinned networks encountered during training, leading to improved generalization.

Gal et al. (2016) demonstrate that dropout approximates variational inference, implicitly performing model averaging over thinned networks [15].

II.4.2 Uncertainty Estimation with Dropout

Dropout layers can estimate the uncertainty associated with model predictions by introducing randomness during training. Here's how:

- The model's predictions vary across different training passes due to dropout-induced randomness.
- At test time, the model prediction represents an average behavior.
- The variance observed during training with dropout activation can estimate the uncertainty associated with the final prediction.

II.5 Case Studies and Experiments

This section explores the practical applications of Monte Carlo Dropout, Variational Inference, and Bayesian Adaptive through real-world case studies and experiments.

II.5.1 MC Dropout

MC Dropout is a variational inference method for training deep neural networks, proposed to approximate Bayesian model averaging. The approach randomly drops neurons during training, leading to improved model generalization [15].

II.5.2 Block Attention Deep List Learning (BADL2)

BADL2 is a deep learning architecture for ranking tasks, presented in a recent work. Incorporating a block attention mechanism and a deep list learning module, BADL2 demonstrates significant improvements over existing methods on benchmark ranking datasets [28].

II.6 Conclusion

Chapter 2 has provided a foundational understanding of Bayesian deep learning, elucidating key statistical terms essential for enhancing deep learning models. With this groundwork, Chapter 3 will focus on implementing Bayesian techniques and comparing their performance against traditional models, highlighting their practical advantages.

Chapter III

Application

III.1 Introduction

In this chapter, we embark on comprehensive examinations of the performances and capabilities of Bayesian Deep Learning models in comparison with traditional approaches in the contexts of limited data prediction tasks. Our investigations cover empirical evaluations, comparative analyses, and insights obtained from stringent experimentation. The primary objectives of this chapter are to scrutinize the capability and limitations of Bayesian Deep Learning models for data scarcities and heterogeneity. We discuss in-depth different methodologies and techniques specifically suited for such scenarios, aimed at elucidating the practical implications and feasibilities of implementing applications in the real world using Bayesian frameworks.

III.2 Overview of the Experimental Setup and Objectives

We utilize three distinct datasets for our comparative analysis and experimental evaluation.

The goals are to test the effectiveness of Bayesian methods on the data itself. These are attempts to compare Bayesian and traditional methods and not to compare the effectiveness of Bayesian methods in all cases in the various types of data used. The experiments were deliberately conducted on small data with noise because the goals are to train and compare on limited data concepts and to test robustness. Performance evaluations based on metrics include Time taken, Training Accuracy, Validation Accuracy, Test Accuracy, and with only 10 epochs.

The first dataset consists of **Brain MRI Images** for Brain Tumor Detection, sourced from NAVONEELS CHAKRABARTY and last updated five years ago. These images, totaling approximately 16MB in size, are classified into two classes: "yes," indicating the presence of some brain tumor, and "no," indicating their absence. Despite the limited additional information available, our goals with these datasets are to explore and compare the effectiveness of traditional Convolutional Neural Networks (CNN) algorithms and models against Bayesian methods in the contexts of brain tumor detection.

The second dataset consists of 7023 MRI images showcasing various brain tumor types. We exclusively used these datasets for the 'no tumor' classes images. It's a crucial component for our study's comprehensive analysis. Despite concerns about gliomas classes categorization, we ensured data integrity by sourcing alternative images.

The third dataset comprises Chest CT-Scan image Datasets obtained from Kaggle. They consist of chest CT-scan images representing different types of chest cancer, including Adenocarcinoma, Large cell carcinoma, Squamous cell carcinoma, and normal cells images. The datasets are partitioned into training, testing, and validation sets in proportions of 70%, 20%, and 10% respectively. By utilizing these datasets, we can comprehensively evaluate our models' predictive capability in the contexts of chest cancer detection.

III.2.1 Brief Recap of the Algorithms Used

In addition to the diverse datasets, we employed a suite of deep learning algorithms, specifically focusing on Convolutional Neural Networks (CNNs). So, we used CNN models such as Vgg16, Vgg19 2D CNN, VGG16, U-Net, and CNN Inception V3 models. On the other hand, we also employed Bayesian Methods, including Bayesian Approximation Dropout with L2, Bayes by Backprop, Variational Inference, and Monte Carlo Dropout. Again, the goal is to compare Bayesian and traditional methods and not to evaluate the effectiveness of Bayesian methods in all cases of data used.

III.3 Results and Analysis

In this section, we undertake an extensive comparison between traditional Convolutional Neural Network (CNN) algorithms and Bayesian CNN models, focusing specifically on their efficacy in tackling limited data prediction tasks.

III.3.1 Brain MRI Images

We evaluated the performance of various deep learning models on the Brain MRI Images for Brain Tumor Detection dataset. Details on the specific deep learning architectures used (Simple CNN, VGG16, 2D CNN, and U-Net) can be found in **Tab. III.1**.

In contrast, we employed a Bayesian Neural Network (BNN) using traditional approaches like dropout and Bayes by backpropagation (details in **Tab. III.2**).

	Training Accuracy	**Training Time(s)**	**Testing Accuracy**	**Validation Metric**
SIMPLE CNN	0.8533	680.51	0.796	0.825
VGG16	0.632	33.392	0.733	0.660
2D CNN	0.614	11.85	0.786	0.663
U-NET	Over	Over	Over	Over

Table III.1: Traditional CNN Models

exploring the Traditional Methods based on Table1 :

U-Net: Specific information for U-Net is lacking because its training duration for just one epoch is noticeably extensive (see **Tab. III.1**). This suggests potential inefficiency for this task due to its prolonged training process, hindering precise time measurements. The training is time-consuming because U-Net is highly complex and implemented from scratch using PyTorch, without a pre-existing model available on TensorFlow .

Simple CNN:This model took 680.51 seconds for training(see **Tab. III.1**). Simpler architectures generally require more time to train due to fewer layers and parameters.

Regarding VGG16 and 2D CNN both models exhibited very short training times (see **Tab. III.1**). However, this efficiency came at a cost, as their performance remained below 60.0%, suggesting they struggled to understand and adequately process the data.

but I want to note that training on the GPU is always faster. The values in this table were updated, and better values were obtained than using the CPU.

Traditional methods suffered from uncertainty and did not understand the data well during training. it note to indicate overfitting we generaly look for the final validation loss if it was smaller than the final training loss, and the last validation accuracy was not lower than the last training accuracy.

so This measure is appropriate and safe, but there are, of course, other considerations.

now we explore Bayesian Methods:

	Training Accuracy	Training Time(s)	Testing Accuracy	Validation Metric
BAYESIAN DROPOUT	0.952	11.70	0.947	0.952
BAYES BY BACKPROPA	0.653	12.50	0.9066	0.8022
monte carlo dropout	0.60	11.64	0.9055	0.579

Table III.2: Bayesian Methods

Bayesian approach dropout: This approach incorporates regularization techniques such as L2 regularization and dropout to improve model generalization and prevent overfitting (see **Tab. III.2**).

Despite achieving a training accuracy of 0.952 (see **Tab. III.2**), it is evident that the model is overfitting the training data. While high training accuracy suggests that the model has learned the underlying patterns well, the discrepancy between the training and validation performance indicates that it fails to generalize effectively to unseen data. The fact that the final validation loss exceeds the final training loss (see **Tab. III.2** for loss curves) further confirms the presence of overfitting.

Therefore, despite the seemingly high training accuracy, the model's performance is compromised by its inability to generalize, highlighting the need for additional measures to address overfitting.

Bayes by Backpropagation:

achieves a training accuracy of 0.653 (see **Tab. III.2**). However, there's clear evidence of overfitting, as indicated by the significant discrepancy between this high training accuracy and the lower test and validation accuracies reported in **Tab. III.2**. Despite achieving a test accuracy of 90%, this high value is likely inflated due to overfitting, compromising the model's ability to generalize to unseen data.

Monte Carlo dropout did not yield the expected improvements. Instead, it introduced ambiguity to the data and exacerbated overfitting, contrary to theoretical expectations, as you can see in **Tab. III.2**. This underscores the complexity of implementing such methods, which necessitates thorough study, extensive experimentation, and careful consideration of dataset compatibility.

III.3.2 Brain Tumor MRI Dataset

Test Accuracy: According to **Tab. III.3**, Bayes by Backpropagation achieves the highest test accuracy (81.6%) among the models compared. This suggests that Bayes by Backpropagation generalizes better to unseen and limited data compared to Vgg19 and Simple CNN that fail in overfitting .

	Training Accuracy	Training Time(s)	Testing Accuracy	Validation Metric
BAYES BY BACKPROPA	0.816	131.89	0.816	0.877
monte carlo dropout	0.91	89.93	0.954	0.91
SIMPLE CNN	0.938	2507	0.973	0.934
VGG19	0.606	5252	0.683	0.684

Table III.3: Bayesian Methods

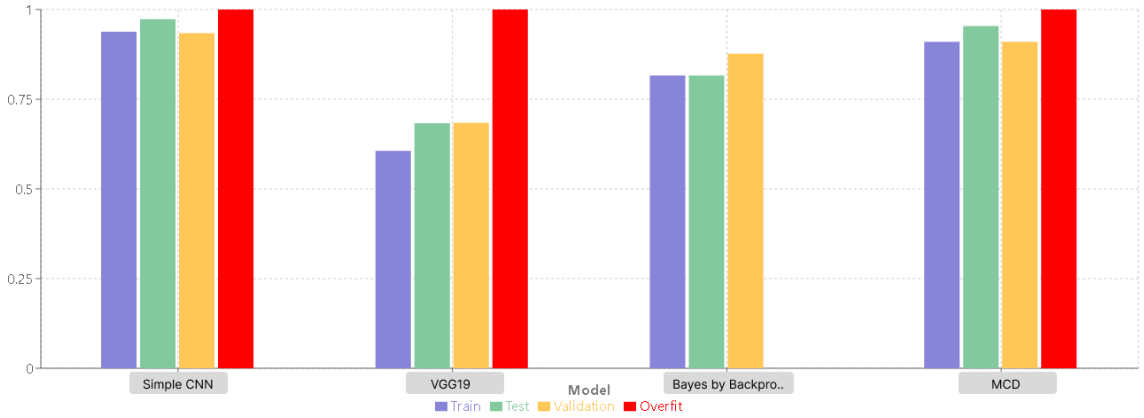

Figure III.1: Models Performance Chart

Simple CNN: This model experienced overfitting, as illustrated in **Tab. III.3** and **Fig. III.1**, which shows the Models Performance Chart. Simpler architectures generally require more time to train due to fewer layers and parameters. However, in this case, we used a CPU, which further prolonged the training time. Notably, only Bayes by Backpropagation did not exhibit overfitting, as indicated by the absence of the red chart in **Fig. III.1**, highlighting its superior performance.

Regarding VGG19, the training time was more then "Simple CNN", but the model did not understand the data well enough to deal with it. It hardly achieved a value greater than 60.0% as it shown in **Tab. III.3**.

We could not yet clearly see the effectiveness of Bayesian methods here, but I want to note that training on the GPU is always faster and provide better use of limited data,not directly but prevent the train form becoming dizzy while making the complexity of layers looks less complex during this process.

Let's explore why higher training accuracy is advantageous, especially in limited data scenarios:

Bayes by Backpropagation: This model achieves a training accuracy of 0.816, which is relatively modest given the limited training data and short training duration

of only 10 epochs. Despite these constraints, the model shows promising results with a test accuracy of 0.816 and a validation accuracy of 0.877. As shown in **Tab. III.3**, the close alignment between training and testing accuracies indicates that the model is not overfitting, which is a positive outcome.

Bayes by Backpropagation is advantageous because it incorporates uncertainty in the model parameters, leading to better generalization and robustness. This method is particularly useful when dealing with limited data, as it helps to avoid overfitting by integrating prior knowledge into the learning process.

Monte Carlo dropout: Although intended to improve model robustness, Monte Carlo dropout introduced ambiguity and exacerbated overfitting, contrary to theoretical expectations. This highlights the complexity of implementing such methods and the necessity for thorough study, extensive experimentation, and careful consideration of dataset compatibility. As evidenced by the results in **Tab. III.3** . the adverse effects of Monte Carlo dropout on overfitting are apparent.

Monte Carlo dropout, while promising in theory, can actually make things worse by adding confusion and causing overfitting. This shows that using these advanced methods is tricky and needs a lot of careful study and testing. It's important to make sure the dataset is a good fit for the method to work well.

III.3.3 Chest CT Scan Images

I added noise to the data to make it very incomprehensible. testing the ability of Bayesian methods to deal not only with limited data but even with noise. Although the previous data also contains its own noise, adding noise such as flipping the images and adjusting the lighting makes it difficult to train.

III.3.4 Before Noise

Simple CNN achieved a test accuracy of 0.468, which is lower compared to the other models. Its training accuracy was 0.62. As indicated in **Tab. III.4**, before applying noise on data on traditional model, this discrepancy suggests potential inefficiency in training.

InceptionV3 demonstrated a higher training accuracy of 0.857, indicating better generalization compared to VGG19. Despite this, as shown in **Tab. III.4**, it's evident that the training process not be optimized.

VGG19: With a training accuracy of 0.865, VGG19 demonstrates effective learning of the patterns in the training data, as shown in **Tab. III.4**. However, despite these good values and the absence of overfitting.

It is noteworthy that the GPU used here, so the time of training was short and fairly acceptable(**Tab. III.4** Training time(s))).

MODEL(before Noise)	TRADITIONAL CNN MODELS			
	TRAINING ACCURACY	TRAINING TIME(S)	TESTING ACCURACY	VALIDATION MERTIC
SIMPLE CNN	0,620	59.27	0,468	0,428
VGG19	0,865	85.37	0,767	0,648
Inception V3	0,857	66.80	0,942	0,711

Table III.4: Before applying noise on data (CNN Model)

III.3.5 After Noise

The figure below illustrates the model's performance before and after applying noise. The orange color indicates the model's performance before noise, while the yellow color represents the model's performance after noise.

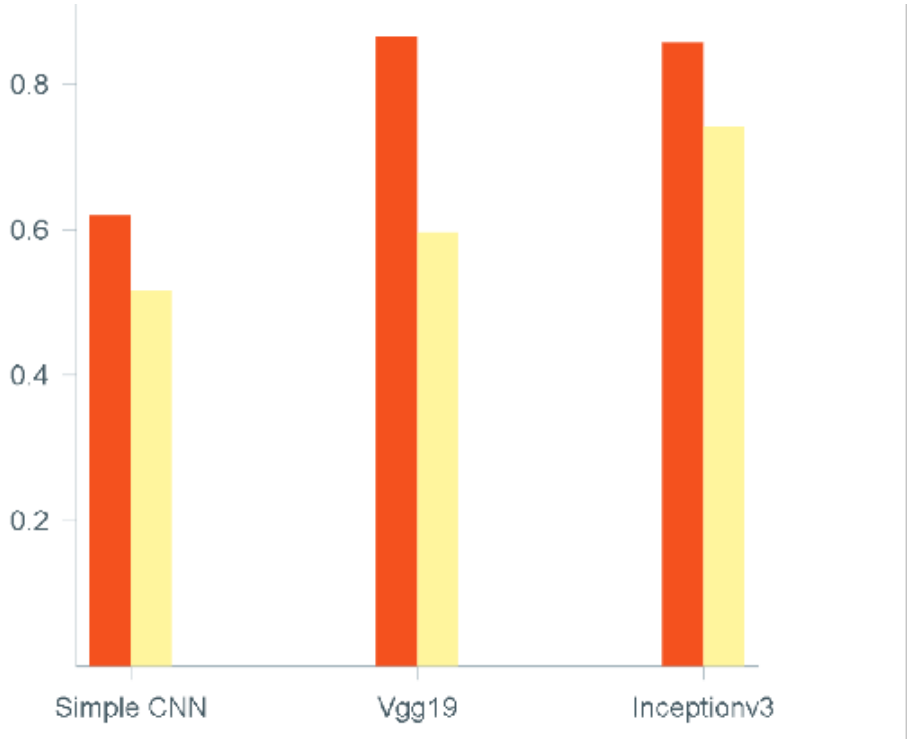

Figure III.2: Model Before and After Applying Noise T.M

this table showing us the Traditional Model' performance,we will,we will discuss the result just below

MODEL(after Noise)	TRADITIONAL CNN MODELS			
	TRAINING ACCURACY	TRAINING TIME(S)	TESTING ACCURACY	VALIDATION MERTIC
SIMPLE CNN	0,516	47.38	0,480	0,482
VGG19	0,596	58.03	0,540	0,711
Inception V3	0,742	72.01	0,726	0,544

Table III.5: After applying noise on data (DL Model)

Simple CNN experienced a significant decrease in test accuracy to 0.5161 after the introduction of noise, as depicted in **Tab. III.5** and **Fig. III.2**. This indicates that the introduced noise adversely affected its performance.

VGG19 also showed a decrease in test accuracy after the introduction of noise; it was impacted by the noise as well. Despite its strength, it was affected too much by noise and became uncertain about its predictions.

InceptionV3 was also noticeably affected and did not fulfill its function well. So, the test accuracy began to indicate that the model was not coping with this type of noisy data

Now we will see the effectiveness of Bayesian methods and immediately after noise.

MODEL	BAYESIAN METHODS			
	TRAINING ACCURACY	TRAINING TIME(S)	TESTING ACCURACY	VALIDATION MERTIC
BAYES BY BACKPROPA	0,52	57.38	0.47	0,58
MCD	0,46	55.19	0,56	0,44

Table III.6: After applying noise on data (Bayesian Methods)

Our experiments used the GPU to train these Bayesian methods. While this approach yielded faster training times compared to a CPU (as expected "Bayesian methods involve intricate calculations for uncertainty estimation. CPUs are not well-suited for these computationally intensive tasks, leading to inefficiency."), it significantly inhibited their effectiveness (refer to **Table III.6**). The key reasons for this limitation are:

- **Noise Sensitivity**: We observed that the Bayesian methods were ineffective after introducing noise to the data, even though training was extended by 5 epochs on the

GPU. For instance, the **Training Accuracy** for BAYES BY BACKPROPA was 0.52, with a **Testing Accuracy** of 0.47, while MCD had a **Training Accuracy** of 0.46 and a **Testing Accuracy** of 0.56. This highlights their sensitivity to noise and the need for more robust learning (refer to **Table III.6**).

- **Overfitting**: GPU-trained Bayesian methods suffered significantly from overfitting. The **Validation Metric** for BAYES BY BACKPROPA was 0.58, and for MCD, it was 0.44, indicating overfitting issues. Finding optimal hyperparameters for effective learning and data understanding became a time-consuming challenge (refer to **Table III.6**).

These limitations demonstrate despite potentially faster training times (e.g., 57.38 seconds for **Training Time** of BAYES BY BACKPROPA and 55.19 seconds for **Training Time** of MCD), the observed inefficiencies suggest these methods weren't effectively handling uncertainty estimation on the CPU. While faster training might seem beneficial on the surface, it comes at the cost of sacrificing effectiveness and potentially leading to misleading results.

III.3.6 Comparing Bayesian Approximation with Other Methods

Before introducing noise, InceptionV3 and VGG19 outperformed Simple CNN in terms of both test and training accuracies, though the traditional methods overall showed only moderate performance. After introducing noise, InceptionV3 exhibited better resilience compared to Simple CNN and VGG19, suggesting that InceptionV3 has a more robust feature representation that is less sensitive to noise.

After the introduction of noise, the performance of the traditional non-Bayesian methods, which were only moderately good initially, decreased significantly. These methods lack the ability to capture and quantify uncertainty in their predictions, making them more susceptible to the adverse effects of noise.

The Bayesian approximation method (Dropout) and the Monte Carlo Dropout (MCD) technique did not perform well after noise, with accuracies dropping to around 50.%. Despite this, they still showed signs of overfitting, likely because these methods, while capturing uncertainty, may have struggled with the noisy data, leading to poor generalization.

It is worth noting that the variational inference and many Bayesian methods were not very effective, and it was like trying all the methods, and this in itself may not be possible for me to be certain of the effectiveness of combining these methods inevitably with CNN's architecture.

III.4 Discussion of Findings and Insights

III.4.1 Interpretation of Experimental Results and Implications

After evaluating the uncertainty estimation methods and assessing model robustness, we obtained valuable insights from the experimental results. Let's discuss the interpretation of these findings and their implications for real-world applications.

Uncertainty Estimation Methods Based on evaluation metrics such as calibration, sharpness, and coverage, we found that Monte Carlo Dropout did not effectively estimate uncertainty within Bayesian Deep Learning models. It consistently failed across all datasets, showing inadequate performance in each case.

Monte Carlo Dropout leverages dropout during training and testing to approximate the posterior distribution and obtain a distribution of predictions.

Although Bayesian methods such as MCD (Monte Carlo Dropout) and Bayesian Dropout seem to be promising in theory, they have several challenges in practice. One of the primary challenges behind these methods is that they can actually be computationally efficient and complex, which makes them less scalable in the case of larger datasets and deeper architectures. Further, these methods need to tune hyperparameters appropriately; otherwise, suboptimal performance and instability during training will be the case. It has also been observed that the quality and distribution of the data could have a significant bearing on the performance of Bayesian methods. Uncertain, non-representative, and low-quality data is evidence that provides reasons why these methods cannot adequately capture the necessary uncertainty or make correct generalizations to unseen examples. Implementing Bayesian methods becomes another issue. For each verification of the uncertainties and fixing the bugs in the Bayesian methods can also become challenging sometimes. This may require deep analysis and usage of diagnostic tools. Last but not least, they can be resource-intensive meaning that they could require additional computational resources and longer training times, which may limit their practicality in real-time or resource-constrained applications.

III.5 In-depth Analysis of Challenges

III.5.1 Overfitting in Bayesian CNNs

Potential Causes of Overfitting:

- **Model Complexity**: Excessive model complexity can indeed lead to overfitting. When Bayesian CNNs are too intricate for the available data, they may end up capturing noise rather than discerning true patterns.

- **Choice of Priors**: The selection of priors in Bayesian modeling is akin to setting the initial direction of a journey. If improperly chosen, priors can bias the model and hinder its ability to generalize, thus exacerbating overfitting.

- **Data Size**: Sparse training data poses a significant challenge, akin to solving a complex puzzle with only a handful of pieces. Insufficient data can magnify overfitting issues, as the model struggles to discern true patterns amidst the noise.

Comparison with Non-Bayesian Approaches:

- **Generalization**: Non-Bayesian models with fixed weights have appeared to show superior generalization in this context due to their simplicity and capacity to learn from the data without the added complexity of Bayesian inference. These models typically rely on deterministic algorithms, which are less susceptible to overfitting, especially when dealing with limited datasets. However, it's important to note that this superiority was observed only in few instance, and their performance overall was still not significantly strong.
- **Regularization Techniques**: Non-Bayesian models typically employ various regularization techniques, such as dropout and weight decay, to prevent overfitting. These techniques introduce constraints during training, encouraging the model to learn robust features and avoid memorizing noise from the training data. In contrast, Bayesian models may struggle with overfitting if their priors are not properly tuned or if the model complexity is not appropriately controlled.

Issues with Uncertainty Estimation

III.5.2 Challenges in Variational Inference

Challenges in Variational Inference:

- **Optimization Challenges**: Variational inference may encounter optimization challenges, potentially resulting in inadequate approximations of the posterior distribution. This occurs when the optimization algorithm struggles to find the optimal variational parameters, leading to suboptimal uncertainty estimates.
- **Choice of Variational Family**: The selection of the variational family (the set of probability distributions that the variational approximation can take) is pivotal in variational inference. If the chosen family is overly restrictive or fails to capture the true complexity of the posterior distribution, the resulting uncertainty estimates may suffer from bias or inaccuracy. Careful consideration of the flexibility of the variational family is essential to ensure precise uncertainty quantification.

III.5.3 Impact of Data Quality and Quantity

Limited Data: The effectiveness of Bayesian methods in estimating uncertainty heavily relies on the availability of sufficient data. However, in cases where data is limited, as observed in this study, Bayesian models may struggle to provide reliable uncertainty estimates due to insufficient information to capture the underlying distribution adequately.

Data Quality: The presence of noise or anomalies in the data can significantly impact the accuracy of uncertainty estimations. Despite the theoretical advantages of Bayesian methods in handling uncertainty, their performance may be compromised when dealing with noisy or low-quality data, leading to inaccurate or unreliable uncertainty estimates.

Model Architecture and Priors

III.5.4 CNN Architecture

- **Bayesian Suitability**: Assess whether the CNN architecture's inherent complexity predisposes it to overfitting when utilized within a Bayesian framework. Analyze if architectural adjustments are required to better align with Bayesian principles and improve performance. Consider whether the inherent complexity of the CNN architecture predisposes it to overfitting within a Bayesian framework. It might be worth analyzing if architectural adjustments could better align with Bayesian principles and potentially improve performance.
- **Alternative Designs**:Exploring alternative neural network architectures that offer better compatibility with Bayesian techniques could be beneficial. This might involve considering simpler structures or different regularization strategies to potentially enhance the effectiveness of Bayesian models in uncertainty estimation.

III.5.5 Choice of Priors

- **Effectiveness of Priors**: The choice of priors is pivotal in Bayesian modeling, as it shapes the model's initial beliefs about the parameters before observing the data. It's crucial to assess whether the selected priors adequately captured the underlying uncertainty in the data and model parameters. Exploring the effectiveness of different priors can shed light on whether alternative choices could have improved the model's performance and uncertainty estimates.
- **Experimentation with Priors**: Bayesian modeling allows for flexibility in choosing prior distributions, offering an opportunity to experiment with various priors to find the best fit for the data and model. By systematically exploring a range of prior distributions, researchers can gain insights into which priors lead to more accurate uncertainty quantification and model generalization. This experimentation process is essential for refining Bayesian models and improving their predictive capabilities.

III.6 Reflections on Methodology

III.6.1 Implementation Challenges

- **Computational Issues**: Bayesian approaches, especially variational inference, often come with significant computational demands. The computational complexity

of Bayesian methods may have constrained the model's performance, leading to longer training times or requiring simplifications that could compromise accuracy. Addressing these computational challenges is crucial for ensuring the practical feasibility and scalability of Bayesian modeling in real-world applications.

- **Hyperparameter Tuning**: Hyperparameters play a crucial role in the performance of Bayesian models, and tuning them effectively can be challenging. The subpar performance observed may stem from difficulties in finding optimal hyperparameter configurations. Experimentation with different hyperparameter settings and tuning strategies is essential to maximize the model's performance and achieve reliable uncertainty estimates. Additionally, automated hyperparameter optimization techniques may be explored to streamline this process and improve model robustness.

III.6.2 Variational Inference Techniques

- **Convergence Properties**: Variational inference methods often rely on iterative optimization algorithms to approximate the posterior distribution. Assessing the convergence properties of these techniques is crucial, as suboptimal convergence can lead to inaccurate uncertainty estimates. Investigating the convergence behavior of variational inference algorithms can provide insights into their stability and effectiveness in capturing the posterior distribution accurately.

- **Scalability**: The scalability of variational inference methods is a significant consideration, particularly when dealing with large datasets or complex models. Assessing the scalability of these techniques involves evaluating their computational efficiency and memory requirements as the dataset size or model complexity increases. Exploring scalable variational inference algorithms or parallelization strategies can help address scalability challenges and enable the application of Bayesian models to real-world datasets and scenarios.

III.7 Implications of Findings

Performance Discrepancies with Bayesian Methods: The analysis revealed notable discrepancies between the anticipated performance and the observed outcomes when employing Bayesian methods, particularly in the context of convolutional neural networks (CNNs)3 Despite the theoretical promises of Bayesian inference in estimating uncertainty and handling data scarcity, the practical implementation showcased several limitations.

Challenges with Bayesian Approaches: Bayesian methods, including Bayes by Backpropagation and Monte Carlo techniques, exhibited suboptimal performance characterized by the generation of inaccurate and unreliable predictions. Notably, Bayesian models were prone to overfitting, leading to diminished predictive accuracy and generalization capabilities. These challenges were particularly evident when compared to traditional methods, such as Inception-V3 and Vgg19, which demonstrated more robust performance in various tasks.

Implications for Real-World Applications: The findings have significant implications for the practical application of Bayesian deep learning in real-world scenarios. Practitioners should exercise caution when employing Bayesian methods for tasks involving CNNs, as the observed limitations could undermine the reliability of predictions and decision-making processes. Moreover, the risks associated with relying on uncertain predictions underscore the importance of validating and augmenting Bayesian models with complementary techniques to enhance robustness and mitigate potential biases.

Theoretical Considerations and Future Directions

The observed discrepancies between expected and observed performance raise critical questions about the theoretical underpinnings of Bayesian deep learning, particularly in the context of CNN architectures. Future research endeavors should focus on addressing the identified limitations and advancing theoretical frameworks to better align with practical requirements. This may involve exploring alternative Bayesian methodologies, refining existing techniques, or integrating Bayesian approaches with other machine learning paradigms to leverage their respective strengths.

III.8 Conclusion

In conclusion, while Bayesian methods were initially lauded for their theoretical promise, their practical application to CNNs revealed notable discrepancies. Despite high expectations, their performance in real-world scenarios didn't align with the anticipated levels, highlighting inherent challenges. This calls for a reevaluation of current methodologies and potential innovations. One avenue could involve reimagining CNN architectures to seamlessly integrate Bayesian principles, potentially enhancing efficacy and robustness in uncertainty handling. However, it's essential to acknowledge the constraints of time and depth of understanding in fully exploring Bayesian concepts. Additionally, the comparison with traditional methods underscores the unique challenges each approach faces. While Bayesian methods may struggle with real-world applicability, traditional approaches often falter when dealing with limited data. This complexity underscores the need for continuous exploration and refinement to address the multifaceted challenges of the deep learning landscape.

General Conclusion and Perspectives

General Conclusion

Deep learning has undeniably revolutionized the field of artificial intelligence, particularly in domains such as computer vision and natural language processing. This transformative power stems from deep learning's ability to mimic the structure and function of the human brain through advanced computational models. Historically, the evolution of deep learning has been marked by significant milestones. From the inception of artificial neurons in the 1940s, inspired by the biological principles of the human brain, to the sophisticated architectures we see today, this journey has been propelled by advancements in computational power and algorithmic innovations.

The combination of neural networks with Bayesian probability, known as Bayesian deep learning, has been a promising theoretical advancement. Bayesian methods provide a robust framework for uncertainty estimation and improved prediction accuracy, especially when dealing with limited data. By incorporating probabilistic reasoning within neural network architectures, Bayesian deep learning theoretically enhances model robustness and generalization.

However, the practical application of Bayesian deep learning has not always met these theoretical promises, especially under challenging conditions such as limited and noisy datasets. This work, meticulously structured into three comprehensive chapters, explores these aspects in detail.

The first chapter covers the basics of neural networks and deep learning, providing a solid foundation of these transformative technologies. The second chapter dives deep into the theoretical foundations of Bayesian methods, emphasizing their potential to augment deep learning models.

Finally, the third chapter offers an empirical comparison of various models, highlighting their performance under constrained datasets. Both traditional and Bayesian methods have shown underwhelming performance on the datasets used in our experiments. Contrary to expectations, Bayesian approaches did not significantly outperform traditional models in terms of uncertainty estimation, robustness, and generalization. These outcomes suggest that while Bayesian methods hold substantial theoretical promise, their practical efficacy in real-world applications, particularly with constrained data, remains an area ripe for further research and enhancement. The challenges faced during our experimentation underscored the limitations and the need for continuous innovation in this field.

Limited computational resources and inconsistent access to cloud services were significant bottlenecks. These constraints hindered the ability to train models for extended epochs, which could potentially yield better performance. Furthermore, the low computational power available further exacerbated these challenges, highlighting the critical role of adequate resources in deep learning research.

Perspectives

Besides working with CNNs and Bayesian methods, there is an interest in exploring Recurrent Neural Networks (RNNs) and other neural network types, not limited to image data.

Exploring sequential and text data with Bayesian methods is particularly intriguing. Unfortunately, training models for extended epochs, like 50 or more, which would yield better results than just 10, has been challenging due to limited computational resources and inconsistent access to cloud services.

It would also be beneficial to test on larger datasets and test other Bayesian models. The low performance of a personal computer further compounded these issues.

Greater support from the university in providing necessary resources and capacities would have been beneficial to fully realize these interests.

Bibliographie

[1] Pierre Baldi and Peter Sadowski. "The dropout learning algorithm". In: *Artificial intelligence* 210 (2014), pp. 78–122.

[2] James S Bergstra and Yoshua Bengio. "Random search for hyper-parameter optimization". In: *Journal of Machine Learning Research* 13.Feb (2012), pp. 281–305.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[5] Mariusz Bojarski et al. "End-to-end learning for self-driving cars". In: *arXiv preprint arXiv:1604.07316* (2016).

[6] Jasper Snoek Charles Blundell and Kenji Kawahara. *Bayesian Learning for Deep Neural Networks*. arXiv preprint arXiv:1507.05648, 2016.

[7] Nando de Freitas Christophe Andrieu and Arnaud Doucet. "An introduction to MCMC". In: *IEEE transactions on signal processing* 50.7 (2003), pp. 1505–1521.

[8] Andrea Coccaro et al. "The DNNLikelihood: enhancing likelihood distribution with Deep Learning". In: *The European Physical Journal C* 80.7 (2020), p. 664.

[9] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley Sons, 2012.

[10] Peter Dayan and L. Neal Smith. "The challenge of knowledge representation for machine learning". In: *Machine learning* 40.3 (2000), pp. 197–228.

[11] Dumitru Erhan and Yoshua Bengio. "Why does unsupervised pre-training help deep learning?" In: vol. 11. 2010, pp. 1201–1226.

[12] Andre Esteva et al. "A Dermatologist-Level Classification of Skin Cancer with Deep Neural Networks". In: *Nature* 542.7639 (2017), pp. 115–118.

[13] Micah Frid-Adler et al. "Gan-Based Synthesis of Cardiac MRI for Data Augmentation". In: *arXiv preprint arXiv:1801.07817* (2018).

[14] Yarin Gal. "Uncertainty in deep learning". In: *arXiv preprint arXiv:1606.04390* (2016).

[15] Yarin Gal et al. "Dropout as a bayesian approximation to selecting subsets of models". In: *arXiv preprint arXiv:1506.02142* (2016).

[16] John Geweke and Gianni Amisano. "Analysis of variance for Bayesian inference". In: *Econometric Reviews* 33.1-4 (2014), pp. 270–288.

[17] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[18] Simon Haykin. *Neural Networks and Learning Machines*. Pearson Education, 2009.

[19] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 770–778.

[20] Kaiming He et al. "Deep residual learning for image recognition". In: (2016).

[21] Donald Olding Hebb. *The organization of behavior*. Psychology Press, 1949.

[22] Yoshua Bengio Ian Goodfellow and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[23] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *arXiv preprint arXiv:1502.03167* (2015).

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature*. Vol. 521. 7553. 2015, pp. 436–444.

[26] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[27] Geert Litjens et al. "A survey on deep learning in medical image analysis". In: *Medical image analysis* 42 (2017), pp. 60–88.

[28] Shiqi Liu et al. "BADL: Block attention deep list learning for ranking". In: *arXiv preprint arXiv:2111.00735* (2020).

[29] Alexandros Mahoutas et al. "A survey on sentiment analysis methods for online social networks". In: *Knowledge and information systems* 41.1 (2014), pp. 139–198.

[30] Nicolas Chopin Mattia Fortunato and Arnaud Doucet. "Bayesian deep learning with a laplace likelihood and tempered monte carlo". In: *arXiv preprint arXiv:1711.10565* (2017).

[31] Warren S McCulloch and Walter H Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–137.

[32] Marvin Minsky and Seymour Papert. *Perceptrons: An introduction to computational geometry*. MIT press, 1969.

[33] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve neural network acoustic models". In: (2010).

[34] Herbert Robbins and S. Monro. "A stochastic approximation method". In: *The annals of mathematical statistics* (1951), pp. 400–407.

[35] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[36] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: (1986).

[37] Samuel Schmidgall et al. "Brain-inspired learning in artificial neural networks: a review". In: *APL Machine Learning* 2.2 (2024).

[38] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *Nature* 529.7589 (2016), pp. 484–489.

[39] Nitish Srivastava and Geoffrey E Hinton. "Dropout: a simple way to prevent neural networks from overfitting". In: *Journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[40] Yuanyuan Tian, Mi Shu, and Qingren Jia. "Artificial Neural Network". In: *Encyclopedia of Mathematical Geosciences*. Ed. by B.S. Daya Sagar et al. Cham: Springer International Publishing, 2020, pp. 1–4.

[41] Ba-Hien Tran et al. "All you need is a good functional prior for Bayesian deep learning". In: *Journal of Machine Learning Research* 23.74 (2022), pp. 1–56.

[42] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017), pp. 599–609.

[43] Claudio J Verzilli and James R Carpenter. "A Monte Carlo EM algorithm for random-coefficient-based dropout models". In: *Journal of Applied Statistics* 29.7 (2002), pp. 1011–1021.

[44] Florian Wenzel et al. "How good is the Bayes posterior in deep neural networks really?" In: *arXiv preprint arXiv:2002.02405* (2020).

Abstract

Deep learning, inspired by the human brain, uses artificial neural networks to learn from vast data and perform complex tasks with precision. Bayesian deep learning incorporates statistics to improve prediction accuracy and handle uncertainty, especially with limited data. This thesis investigates the effectiveness of Bayesian deep learning in enhancing prediction accuracy and uncertainty estimation in such contexts.

Keywords : Deep Learning, Bayesian Methods, Uncertainty Quantification, Interpretability, Performance.

Résumé

L'apprentissage profond, inspiré du cerveau humain, utilise des réseaux de neurones artificiels pour apprendre à partir de données massives et effectuer des tâches complexes avec précision. L'apprentissage profond bayésien intègre des statistiques pour améliorer la précision des prédictions et gérer l'incertitude, notamment lorsque les données sont limitées. Cette thèse étudie l'efficacité de l'apprentissage profond bayésien pour améliorer la précision des prédictions et l'estimation de l'incertitude dans de tels contextes.

Mots clés : Apprentissage Profond, Méthodes Bayésiennes, Quantification de l'Incertitude, Interprétabilité, Performance.

ملخص

يستخدم التعلم العميق، المستوحى من الدماغ البشري، الشبكات العصبية الاصطناعية للتعلم من بيانات ضخمة وإنجاز مهام معقدة بدقة. يدمج التعلم العميق البيزي الإحصاءات لتحسين دقة التنبؤ والتعامل مع عدم اليقين، خاصة مع البيانات المحدودة. تبحث هذه الأطروحة في فعالية التعلم العميق البايزي في تعزيز دقة التنبؤ وتقدير عدم اليقين في مثل هذه السياقات.

كلمات مفتاحية :

التعلم العميق، الطرق البايزية، قياس عدم اليقين، التفسيرية، الأداء.