CISC 352 Group 4 Report

Max Hao, Kanice leung

Queen's University
CISC 352

Table of Contents

Empirical Study

Cancer remains a critical global health issue, with approximately 19.3 million new cases and 10 million deaths reported in 2020. [1] Cancer is a disease characterized by the uncontrolled proliferation of certain cells within the body, which may subsequently disseminate to other regions. These abnormal cells can aggregate to form tumors, defined as tissue lumps. Tumors can be classified into two categories: benign and malignant. Malignant tumors possess the ability to invade adjacent tissues through a process known as metastasis, potentially resulting in pain, abnormal bleeding, and severe complications, including organ failure, if not adequately treated. In contrast, benign tumors do not invade surrounding tissues and typically do not recur after surgical removal. [2] Consequently, malignant tumors are generally considered more life-threatening than benign tumors. Early, accurate diagnosis significantly improves treatment outcomes, yet current diagnostic methods often suffer from subjectivity, interpretation variability, and difficulty in accurately identifying early-stage or borderline malignancies. Artificial Intelligence (AI) provides effective solutions by efficiently analyzing large patient datasets, thereby enhancing early cancer detection and facilitating more accurate clinical decision-making.

Particularly significant among AI approaches are Bayesian Neural Networks (BNNs), which integrate probabilistic inference directly into neural network architectures. This integration allows robust uncertainty quantification, critically influencing clinical decision-making and treatment planning.

Moreover, integrating Planning Domain Definition Language (PDDL) with BNNs optimizes clinical workflows through automated decision-making. PDDL systematically sequences and selects diagnostic tests, providing parameters to BNN and significantly reducing computational costs and enhancing diagnostic accuracy. From the patient's perspective, preliminary AI-driven assessments reduce anxiety by providing early insights. Clinicians benefit from structured diagnostic pathways, streamlined decision-making, and improved resource utilization, ultimately enhancing patient care quality.

Abstract

This report explores the integration of three distinct methodologies—planning, deep learning, and probability inference—for classifying tumors as benign or malignant. Specifically, it focuses on combining the Planning Domain Definition Language (PDDL) with Bayesian Neural Networks (BNNs)[3]. Using PDDL streamlines interactions with APIs, particularly through a Python-based plan manager that evaluates feedback from the BNN model within a normalized range ($0 \leq x \leq 1$). The planning framework assesses uncertainty levels and dynamically updates diagnostic actions by excluding scenarios that demonstrate either high uncertainty or prohibitive computational demands. Structured plans generated by PDDL subsequently guide neural network training.

While traditional PDDL provides efficiency, probabilistic PDDL would better manage uncertainties inherent in medical diagnostics. However, due to computational constraints and the lack of robust support for probabilistic PDDL extensions in software such as VSCode, standard PDDL was selected. Among conventional planners, LAMA proved especially effective, rapidly evaluating potential scenarios to identify plans, thus maximizing computational efficiency. Although traditional PDDL lacks native support for conditional effects, this limitation is circumvented by explicitly defining all possible scenarios within the planning framework along with plan extension. The fundamental knowledge of planning is also implemented across the entire project.

Bayesian inference proves especially relevant due to conditional dependencies present in tumor datasets. Because tumor characteristics are not completely conditionally independent, Bayesian inference facilitates more nuanced diagnostic assessments. Initial use of Naive Bayes could offer preliminary probability weighting; however, constructing detailed inference graphs that combine posterior and prior probabilities achieves superior accuracy. These graphs enable a comprehensive exploration of data relationships, resulting in more precise diagnostics and richer insights for training.

Incorporating deep learning techniques, specifically Bayesian Neural Networks, enhances the Bayesian inference process. Effective BNN implementation requires careful definition of evaluation and loss functions, optimization algorithms, , normalization techniques, and different variation approximations. Furthermore, strategically designing and adjusting hidden layers within BNN architectures ensures scalability, adaptability, and reusability for future research. [3]

This report provides a robust, reusable methodology and foundational codebase that future researchers can leverage to improve tumor classification accuracy. The proposed integrated approach notably reduces computational costs, enhances predictive accuracy, and delivers richer informational insights. This blend of efficiency, precision, and detailed interpretability constitutes a valuable contribution to ongoing advancements in medical diagnostics and artificial intelligence.

Introduction

This project supports both manual and automated execution modes, each producing identical outcomes. The manual mode allows users to interact directly with diagnostic sequences, offering deeper insight into how the Plan Extension simulates planner behavior and how the Plan Manager coordinates with downstream components. The core objective of this work is to integrate state-of-the-art methodologies—automated planning, deep learning, and probabilistic inference—into a unified framework for early tumor classification. At the outset, patient data is uploaded in CSV format and passed into a planning module, which uses predefined knowledge to generate an optimal diagnostic plan. This plan is then processed by the Plan Manager, which extracts relevant features and forwards them to the Bayesian Neural

Network (BNN). The BNN produces a prediction along with an uncertainty estimate. These uncertainty scores are provided to clinicians for further interpretation, while patients receive a recommendation if the predicted malignancy probability exceeds a threshold (e.g., 0.5). In this way, both clinicians and patients gain valuable insight into the diagnostic outcome. Each component in the framework plays a critical role: the Planning Domain Definition Language (PDDL) supplies the symbolic foundation for generating diagnostic pathways; BNNs offer predictive models that include uncertainty quantification essential for clinical decision-making; and probabilistic inference techniques, such as variational approximations and Monte Carlo methods, enable the system to account for both model and data uncertainty. The Plan Manager and Plan Extension modules act as a bridge between these components, enabling real-time feedback and adaptive decision-making. Together, this architecture mirrors the logic of clinical reasoning—progressing from raw data to actionable diagnosis—thereby enhancing accuracy, interpretability, and efficiency in cancer diagnostics. The result is a robust, intelligent system designed to improve early detection, streamline clinical workflows, and build trust through transparency and probabilistic rigor.

Methodology:

Automated Planning section:

Initially, our research intended to employ Probabilistic Planning Domain Definition Language (PPDDL) to effectively manage uncertainties inherent in diagnostic scenarios. PPDDL allows explicit modeling of probabilistic effects, enabling planners to dynamically select diagnostic tests based on evolving probabilistic assessments. Specifically, PPDDL facilitates adaptive planning through probabilistic effects, allowing tests to be dynamically incorporated or excluded depending on probabilistic outcomes during execution.

Adapting method and API implementation in supporting PDDL

However, PPDDL introduces substantial complexity, particularly through probabilistic effects, which significantly amplify grounding complexity. Grounding complexity arises because each probabilistic effect generates multiple state transitions that must be explicitly enumerated, resulting in exponential growth in the number of grounded actions. Consequently, computational resources required for grounding and solving PPDDL domains quickly become impractical for real-world-sized problems.

Software tools capable of managing Probabilistic Planning Domain Definition Language (PPDDL) include probabilistic planners such as Probabilistic-FF and Markov Decision Process (MDP)-based planners like RDDLsim. However, due to constraints such as limited planner availability, resource limitations, and substantial computational overhead inherent to PPDDL, we opted for standard PDDL version 2.1. Specifically, we employed the LAMA planner, a highly efficient and widely recognized planner known for its heuristic-based search methods and proficiency in handling complex deterministic planning scenarios via STRIPS and ADL specifications. While LAMA effectively addresses deterministic planning problems, it does not natively support probabilistic or conditional effects.

The preference for LAMA over planners such as A* or other cost-driven methods was strategically determined by our diagnostic objectives. Cost-driven planners, like A*, emphasize finding the minimal-cost solution path, which is not aligned with our goal of achieving clear and decisive diagnostic classifications for now. In our context and the priority of tasks, the priority is rapidly identifying a valid and straightforward path to a conclusive diagnosis (benign or malignant). Pursuing the lowest-cost solutions can blur diagnostic clarity, complicate feedback interactions, and obscure diagnostic endpoints. Consequently, LAMA's capacity to quickly deliver the shortest valid planning paths, prioritizing diagnostic clarity over absolute cost optimization, significantly streamlined our diagnostic planning process and improved the overall computational cost of BNN.

Standard PDDL 2.1, despite lacking direct probabilistic capabilities, offers comprehensive deterministic planning constructs. It includes support for explicit definitions of actions, preconditions, effects, and advanced logical formulations, such as disjunctive preconditions and conditional logic through carefully structured predicate relationships.

To address the inherent limitations of standard PDDL—particularly its inability to represent probabilistic uncertainty directly—we introduced Bayesian Neural Networks (BNNs) to replicate PPDDL's intended dynamic decision-making. BNNs provided uncertainty estimates for each diagnostic decision, thus simulating probabilistic reasoning. This integration involved enhanced API interactions between the planning domain and BNN models, allowing real-time adjustment of diagnostic actions based on computed uncertainty. Conceptually, this hybrid approach closely mirrors the adaptive functionality initially envisioned with PPDDL.

However, several limitations remained. Due to the restricted extensibility of PDDL, we were unable to implement more advanced AI-based cost optimization strategies within the planning formalism itself. Additionally, the planner could not directly interact with scripts, and components of the plan solver were embedded within the machine. These factors made it challenging to capture the evolving nature of diagnosis and to dynamically adapt plans in response to uncertainty during execution.

To address these issues, we designed and developed a **Plan Manager and a Plan Extension** that work in close coordination with the AI planning logic. This system operates alongside the standard PDDL planner, interfacing directly with outputs from the Bayesian Neural Network (BNN) and domain feedback to support real-time plan adaptation. The Plan Extension simulates the intended behavior of the original plan, especially in cases where the plan cannot be executed via scripts and requires manual intervention. In such scenarios, the Plan Extension replicates the planned sequences and actions, allowing it to interact with the Plan Manager. The Plan Manager then communicates relevant patient information to the BNN model, which returns predictions along with uncertainty estimates. These predictions are used either as outputs or as feedback within an inner loop to support continuous learning and plan refinement.

The modified standard PDDL 2.1 domain explicitly defined several components:
Requirements: Included essential PDDL features such as STRIPS, typing, negative preconditions, equality, action-costs, and disjunctive preconditions. STRIPS provided foundational action representation while typing facilitated categorization, ensuring semantic clarity. Negative preconditions and equality enabled precise logical constraints, and action-costs managed decision-making priorities effectively. [7]

Types and Constants: Types categorized data explicitly into "measurements" and "values," systematically enumerating constants based on attributes derived directly from the dataset, ensuring accurate semantic representation.

Predicates: Essential predicates explicitly captured conditional logic:

- has-value predicates ensured clear attribute dependency representation (e.g., radius measurements were necessary for area calculations).
- test-performed predicates tracked diagnostic test progression explicitly.
- initialized-weights predicates ensured synchronization between BNN outputs and planning actions, providing coherent decision-making across models.

Action-Cost Management: Action costs explicitly influenced decision-making by prioritizing necessary diagnostic actions and reducing redundant tests. This explicit management of costs also supported iterative refinement by providing valuable secondary insights for researchers and enhancing neural network accuracy.

Given the absence of conditional effects support in standard PDDL using LAMA, we manually enumerated all possible diagnostic scenarios. Each diagnostic attribute was discretized into three discrete states, resulting in 27 ($3\times3\times3$) explicitly defined combinations per diagnostic scenario. Through experiment, although it can be reduced to the lower computation cost by reducing parts of the pddl planning. But we believe the existing framework allows scalable projects in the future in implementing with costs search methods. Or different planning if needed. We could,in exchange, not use all parts of the planning to reduce the computation cost for this project.

Implementation Improved from Draft

Following the comprehensive definition and verification of all planning actions, a dynamic feedback mechanism was developed via the API-driven **Plan Manager**. This system continuously integrates execution feedback to refine diagnostic plans dynamically. Python scripts were implemented to iteratively adjust plans, ensuring comprehensive diagnostic coverage and continuous refinement based on real-time outcomes.

Despite the structural rigidity of PDDL, we continued to extract meaningful planning patterns from the domain and commented out alternate diagnostic paths as better plans were discovered. This manual curation process—although limited—revealed that the overall planning structure and ideology remain highly scalable and promising. Our work demonstrates that even under formal constraints, adaptive AI planning can be meaningfully integrated through hybrid architectures and customized planning infrastructures like the Plan Manager.

And while our framework extends beyond traditional PDDL usage, each element—whether defined through logic-based actions, probabilistic modeling via Bayesian Neural Networks, or external control mechanisms like the Plan Manager—is conceptually rooted in foundational principles of AI planning. At its core, AI planning concerns itself with generating a sequence of actions that transition a system from an initial state to a desired goal state. In our case, this manifests as producing diagnostic action sequences that lead from uncertain patient measurements to a definitive diagnosis.

PDDL serves as the symbolic and logical substrate for defining planning operators, domain constraints, and goal formulations, aligning directly with classical STRIPS-based planning models. These logical formulations underpin AI planning by enabling explicit state representations, goal-driven search, and deterministic planning heuristics, as seen in planners like LAMA.

However, the introduction of Bayesian Neural Networks (BNNs) into the planning loop reflects a shift toward decision-theoretic planning, a subfield of AI planning that deals with uncertainty and probabilistic reasoning. This is conceptually tied to models like Partially Observable Markov Decision Processes (POMDPs) and deterministic planning, where planners must select actions not just for progress but to reduce uncertainty and adapt to evolving knowledge. While we did not formally model our domain as a POMDP due to tooling limitations, the role of the BNN in providing uncertainty estimates serves an analogous function to belief state updates in such models—bridging symbolic planning and probabilistic inference.

The Plan Manager embodies another core tenet of AI planning: plan monitoring and re-planning. It functions similarly to execution monitoring systems in AI planning literature, where planners dynamically react to execution-time feedback and update the plan accordingly. By integrating real-time data from BNN predictions and adjusting future planning actions, the Plan Manager acts as an intelligent controller, enhancing plan robustness and enabling a feedback loop—central to closed-loop planning systems often seen in robotics and autonomous decision-making contexts.

Moreover, the way we refined plans over time through feedback and cost-based evaluations aligns with meta-reasoning in planning—the idea that systems should not only plan actions but also evaluate and refine their planning strategies. This reflects broader AI planning goals, especially in hybrid systems that combine symbolic and sub-symbolic reasoning.

Thus, even though our approach utilizes auxiliary tools and external mechanisms, it remains firmly grounded in AI planning theory. The combination of symbolic planning (PDDL), probabilistic estimation (BNNs), and dynamic plan adaptation (Plan Manager) aligns with modern trends in AI planning, particularly those that seek to integrate deep learning with symbolic reasoning to overcome the brittleness of purely logic-driven systems.

BNN Section :

A Bayesian Neural Network (BNN) fundamentally reimagines a neural network by treating its weights not as fixed values, but as probability distributions — typically initialized with a Gaussian prior such as $w \sim N(0,1)$. Unlike standard neural networks that optimize point estimates via backpropagation, BNNs seek to learn posterior distributions over weights, reflecting both model uncertainty (epistemic) and data noise (aleatoric). This is achieved by applying Bayes' rule, where the posterior $P(w|D) \propto P(D|w) \cdot P(w)$ [17] balances the likelihood (fit to data) with the prior (preference for simplicity). As a result, BNNs inherently penalize over-parameterized or overly complex models — a property known as Occam's Razor — without the need for manually added regularization like L2 or dropout. This regularization emerges naturally through the KL divergence term in variational inference (VI), a scalable approximation method that optimizes the Evidence Lower Bound (ELBO) to estimate the posterior. Each training step in a BNN typically requires multiple forward passes, gradient computations through stochastic layers, and optimization of both mean and variance parameters, leading to significantly greater computational complexity compared to standard networks. Nevertheless, the trade-off is worthwhile in safety-critical tasks like medicine, autonomous systems, and active learning, where uncertainty quantification is crucial. Foundational work by MacKay (1992) introduced the practical Bayesian framework for backpropagation, showing how model evidence peaks at optimal complexity, while Gal and Ghahramani (2016) later proposed Monte Carlo Dropout as a lightweight Bayesian approximation for deep learning. Today, modern derivations of BNNs appear in Bayes by Backprop (Blundell et al., 2015), Stochastic Gradient Langevin Dynamics (Welling & Teh, 2011), and Bayesian Deep Learning libraries like Pyro, TensorFlow Probability, and Edward, pushing scalable and practical inference methods forward.

Issue Addressed # 1 = Loss Curves

In the draft version of our project. Training neural networks involves the process of minimizing a loss function over a dataset by adjusting parameters using gradient-based optimization methods. In practice, the trajectory of this loss function over training epochs often exhibits fluctuations—a phenomenon known as oscillating loss. While such oscillations can be expected and even benign in traditional stochastic optimization, their nature becomes more complex and problematic in probabilistic models such as Bayesian

Neural Networks (BNNs). The contrast between these two cases lies not only in the cause of the oscillation but also in its impact on model performance and the strategies required for stabilization.

In conventional neural networks, model parameters are treated as fixed values, and optimization seeks the best point estimates that minimize an objective function such as cross-entropy or mean squared error. Using methods like Stochastic Gradient Descent (SGD) or Adam, weights are updated iteratively based on gradients computed from mini-batches of data. Oscillations in this setting typically arise when the learning rate is too high, leading to overshooting of the optimal region, or when mini-batch gradients vary significantly due to noise or poor data shuffling. These instabilities are usually mitigated by tuning hyperparameters, using momentum to smooth updates, or employing adaptive learning rates that automatically scale step sizes. As a result, while oscillations may be present, they tend to dampen as the model approaches convergence.

By contrast, Bayesian Neural Networks introduce a more intricate source of loss oscillation due to their probabilistic nature. Rather than learning single-point values for each weight, BNNs aim to learn distributions over weights, most commonly by approximating the intractable posterior distribution with a simpler, tractable family such as diagonal Gaussians. This is typically achieved through variational inference, where the training objective is the Evidence Lower Bound (ELBO), which balances the likelihood of the data given sampled weights and a regularization term that penalizes divergence from the prior distribution. Every forward pass through a BNN samples a new set of weights from these distributions, causing the model's output to be inherently stochastic—even when evaluating on the same input multiple times.

This repeated sampling from the posterior introduces randomness not just into the predictions but also into the gradient estimates. As a result, the optimization trajectory becomes noisier and more volatile. What makes this worse is that the loss function itself is composed of two fundamentally different parts: the negative log-likelihood, which measures how well the model fits the data, and the KL divergence, which penalizes deviation from the prior and governs uncertainty regularization. While the likelihood term behaves similarly to losses in traditional networks and often converges smoothly, the KL divergence term can oscillate dramatically, especially in early training when the approximated posterior is poorly initialized or overly uncertain. The combined effect of this architecture and objective structure is a loss curve that fluctuates sharply and unpredictably[17].

Comparing standard neural networks and BNNs in terms of oscillating behavior reveals important distinctions. In standard models, oscillations generally reflect issues with learning rate or gradient variance and are often localized to early training. With appropriate tuning, these models typically stabilize. In contrast, the oscillations in BNNs can persist indefinitely unless specific measures are taken to control the stochastic effects of posterior sampling and regularization. These fluctuations have a more severe impact in BNNs because they can distort uncertainty estimates, lead to unstable predictions, and, in some cases, completely derail convergence.

In our experiments with BNNs, we observed particularly sharp oscillations when training models using both KL-based variational inference and α-divergence loss functions. Interestingly, while the likelihood component of the loss stabilized as expected, the KL divergence fluctuated wildly across epochs. This strongly suggested that the instability was not due to how the model was fitting the data, but instead stemmed from how it was learning to represent uncertainty. The posterior sampling mechanism, combined with the optimizer's sensitivity to noisy gradients, amplified these effects.

To counteract this, we implemented several effective solutions, beginning with switching from Adam to SGD with momentum. Adam, while efficient and adaptive, updates each parameter based on moment estimates of gradients, making it highly responsive to sudden changes in the gradient landscape. In the context of BNNs, where these changes are often due to sampling variability rather than meaningful learning signals, Adam can overcorrect and cause the loss to spike. SGD with momentum, by accumulating gradients over time and updating weights in a more smoothed manner, provided a stabilizing influence, leading to a notable reduction in oscillatory behavior[16].

Interestingly, when the learning rate was increased from 0.01 to 0.1, Adam began to outperform SGD with momentum. This improvement can be attributed to Adam's adaptive learning rate mechanism, which adjusts updates for each parameter based on the first and second moments of the gradients. At higher learning rates, SGD with momentum can become unstable or get trapped in suboptimal regions of the loss landscape, especially in noisy or non-convex settings common to BNNs[15]. In contrast, Adam's per-parameter scaling allows it to maintain more balanced and controlled updates, even at elevated learning rates. This prevents it from overshooting or stagnating, enabling it to escape shallow minima or flat regions more effectively than SGD. As a result, Adam proved to be more resilient and efficient under these conditions, suggesting that its performance can be sensitive not only to the model and data characteristics but also to specific training hyperparameters like learning rate.

Last key method we used was tuning the β parameter in the ELBO [8][12]. This hyperparameter controls the weight of the KL divergence in the total loss. When set too high, the network prioritizes staying close to the prior over fitting the data, leading to underfitting. When too low, the network overfits by ignoring uncertainty. We found that setting β within a carefully chosen range helped maintain a balance between accurate data modeling and coherent uncertainty estimation, resulting in smoother convergence and more reliable training curves.

We also explored alternative approaches to Bayesian inference that inherently mitigate the sampling-induced instability. Monte Carlo Dropout (MC Dropout) is one such method. It uses dropout not just during training but also at inference time, treating each dropout mask as a sample from a distribution over models. This approximates Bayesian posterior sampling without explicitly learning distributions over each weight, greatly reducing the variance of gradients and improving both convergence and generalization. MC Dropout achieved high accuracy in our tests with far more stable loss curves than full variational methods.

In conclusion, while standard neural networks exhibit oscillating loss due to classic optimization factors like learning rate and gradient noise, Bayesian Neural Networks face deeper and more persistent challenges. Their probabilistic nature, reliance on posterior sampling, and dependence on two-part loss functions introduce dynamic and often severe fluctuations during training. However, through careful experimentation—switching optimizers, tuning regularization, and leveraging alternative inference techniques—we effectively addressed these issues. Our work illustrates not only how to stabilize BNN training but also how to maintain the benefits of Bayesian modeling, such as uncertainty estimation, without sacrificing convergence quality. These insights contribute meaningfully to the broader effort of building robust and reliable probabilistic learning systems.

Issue Addressed 2 = Other inference models other than KL & is KL is most suitable

Bayesian Neural Networks (BNNs) represent a powerful fusion of Bayesian inference and deep learning, offering not only predictive performance but also calibrated uncertainty estimates[3]. Unlike traditional neural networks, which learn point estimates for weights, BNNs model weights as probability distributions, enabling principled reasoning under uncertainty. However, the true posterior distribution over the weights given data is intractable in most real-world settings, requiring approximation techniques to make Bayesian learning feasible. This report, as well as our final work, provides a comprehensive analysis of four common posterior approximation methods—Variational Inference using KL divergence, α-divergence, and Monte Carlo Dropout—by dissecting their theoretical foundations, practical implementations, and observed empirical behaviors in our work.

$$p(w|D) = \frac{p(D|w)p(w)}{p(D)}$$

w represents the network weights, D the observed dataset, p(w) the prior over weights, and p(D|w) the likelihood. The denominator p(D), known as the marginal likelihood or evidence, integrates over all possible weights and is often computationally intractable in high-dimensional, non-convex spaces like those encountered in deep learning. As such, posterior approximation becomes necessary, leading to a variety of strategies, each with distinct trade-offs in expressiveness, computational cost, and practical stability.

1. <u>Variational Inference (KL Divergence Minimization)</u>

One of the most commonly used techniques is variational inference (VI), where we approximate the true posterior p(w|D) with a tractable distribution q(w). Typically, q(w) is chosen to be a diagonal Gaussian:

$$q(w)=N(w;\mu,\sigma 2)$$

We then minimize the Kullback-Leibler (KL) divergence between q(w) and p(w|D), which results in optimizing the Evidence Lower Bound (ELBO):

LELBO=Eq(w)[logp(D|w)]−β·KL(q(w)‖p(w))

The parameter β balances data fit and regularization. This formulation is a stochastic regularized objective where we optimize not only the predictive capability but also penalize divergence from a prior belief. The theoretical roots of this method trace back to David MacKay (1992) and were later popularized for deep networks by Blundell et al. (2015) in *Weight Uncertainty in Neural Networks*.

In practice, we observed that KL-VI performed reasonably well, with stable likelihood loss but erratic KL divergence during training. This is a known behavior: KL tends to favor mode-seeking, which can lead to underestimation of uncertainty and overly confident predictions.

2. α-Divergence: Generalizing KL with Adjustable Bias

KL divergence is just one member of a broader family of divergences. The α-divergence is a generalization that allows greater flexibility in how the approximation behaves. By adjusting the parameter α, we can bias the divergence toward mode-seeking (like KL) or mass-covering, which better represents the full posterior.

$$D_{\alpha}(q\|p) = \frac{1}{\alpha(\alpha-1)}(1 - \int p(w)^{\alpha} q(w)^{1-\alpha} dw)$$

For α approaching 0, it recovers the reverse KL (mass-covering); for α approaching 1, it becomes the standard KL. This flexibility is what motivated Yingzhen Li and Yarin Gal (2017) to propose its use in dropout-based BNNs, showing that tuning α can correct the overconfidence issue in variational methods.

In our implementation, however, α-divergence exhibited increased loss oscillations and a noticeable drop in predictive accuracy. This highlights a known drawback: α-divergence, while more expressive, is harder to optimize and is sensitive to α and initialization. It requires careful tuning and may not stabilize easily in high-dimensional spaces, particularly when the model is deep or the dataset is complex.

3. Monte Carlo Dropout (MC Dropout): A Scalable Bayesian Approximation

Perhaps the best prediction approximation we have implemented is Monte Carlo Dropout, a method that was made widely known by Yarin Gal and Zoubin Ghahramani (2016). Their key insight was that standard dropout, if applied during both training and testing, can be interpreted as approximate Bayesian inference in a deep Gaussian process.

In this framework, the model is trained with dropout as usual, but predictions are made by averaging multiple forward passes, each using a different random dropout mask:

$$Y = \frac{1}{T}\sum_{t=1}^{T} f(x; w_t) \text{ as the w is the dropoutMask}$$

This method approximates a variational posterior without explicitly learning distributions over each parameter. The simplicity and compatibility with existing models make MC Dropout very appealing in applied settings.

In fact, in our experiment, the MC Dropout achieved the highest accuracy, with a smooth loss curve and minimal training instability. This is not surprising, as MC Dropout combines the regularization benefits of dropout with a lightweight method of uncertainty estimation, offering a practical solution where full variational inference may be too complex or unstable. But it would be way over the extent of the course content; therefore, we would not implement this as our model.

Our comparative evaluation of posterior inference techniques in Bayesian Neural Networks (BNNs) reveals the fundamental trade-offs between mathematical expressiveness and computational practicality. Approximation methods are essential in BNNs because the exact posterior distribution over weights—denoted as p(w|D), where w are the network parameters and D is the dataset—is intractable due to the high dimensionality and non-linearity of deep models. And we could use normal forward passes functions like sigmoid or ReLU to incorporate uncertainty in BNNs. And it provides us not just a prediction but a distribution over predictions, incorporating both epistemic (model) uncertainty and aleatoric (data) uncertainty. Each of the four methods we implemented—KL-based Variational Inference (VI), α-Divergence, Monte Carlo Dropout—offers a different lens through which to approximate this posterior and apply Bayesian reasoning in practice. KL-VI assumes a tractable parametric form, typically a Gaussian, and minimizes the divergence from this approximation to the true posterior. This leverages Bayes' theorem directly through the Evidence Lower Bound (ELBO), which balances data fit and prior regularization. α-Divergence, on the other hand, generalizes the KL objective to allow control over the approximation's behavior—whether it should focus on the mode of the posterior (mode-seeking) or spread across its full support (mass-covering), a vital distinction in safety-critical tasks where underestimating uncertainty could be catastrophic. MC Dropout sidesteps explicit posterior modeling by interpreting dropout at test time as implicit variational inference, sampling different subnetworks and averaging predictions. It makes use of Bayesian rules through the stochastic nature of prediction aggregation, allowing robust uncertainty estimation with minimal changes to the architecture. MC Dropout is often ideal; for post-hoc uncertainty layering, KL-VI or α-divergence offer depth at the cost of complexity. Our empirical evaluations underscore this balance—between inference fidelity and usability—and affirm that approximation is not just a necessity, but a design decision that reflects the goals and constraints of each Bayesian application.

In the conclusion of BNN, the knowledges and methods that are being used is a fusion of deep learning and probabilistic inference that reframes neural network training by modeling weights as probability distributions rather than deterministic values, thereby extending traditional deep learning into the probabilistic domain. Rooted in Bayes' theorem, the posterior over weights $P(w|D) \propto P(D|w) \cdot P(w)$ becomes central to training, where the likelihood P(D|w) reflects data fit (a deep learning concern), while the prior P(w) acts as a regularizer (a probabilistic mechanism) that penalizes model complexity—embodying Occam's Razor through the KL divergence. Unlike standard neural networks that minimize pointwise loss functions using methods like stochastic gradient descent (SGD) or Adam, BNNs optimize the Evidence Lower Bound (ELBO)—a probabilistic objective function combining deep learning's log-likelihood loss with a KL divergence term that enforces probabilistic regularization. The optimization of ELBO links directly to variational inference, a method from probability theory that approximates the intractable true posterior with a simpler distribution q(w), typically a diagonal Gaussian, and minimizes their divergence, often using reparameterization to preserve differentiability during sampling—connecting probabilistic theory with deep learning backpropagation. However, this probabilistic sampling introduces stochasticity into the model's predictions and gradients, leading to oscillating loss curves during training—a phenomenon rarely seen in standard deep learning models where deterministic weights enable more stable convergence. These instabilities arise primarily from the KL term, especially early in training when the approximated posterior is uncertain, highlighting a purely probabilistic cause that directly impacts deep learning performance metrics like convergence and generalization. Through experimentation, we observed that while the log-likelihood component (deep learning signal) stabilized normally, the KL divergence (probabilistic regularizer) fluctuated wildly, distorting uncertainty estimates and model calibration. To mitigate this, we used the Adam optimizer using learning rate of 0.1—commonly used in deep learning for its adaptive learning rate. We also fine-tuned the β parameter in the ELBO, which adjusts the weight of the KL divergence, effectively balancing deep learning's desire for accurate prediction with the probabilistic need for credible uncertainty modeling. In addition to optimizer and loss balancing strategies, we evaluated four posterior approximation methods that stem from different theoretical corners of Bayesian inference but impact deep learning training and deployment differently. First, KL-based Variational Inference (VI), derived from information theory and probability, assumes a Gaussian form for q(w) and minimizes the KL divergence between q(w) and p(w|D, producing efficient but often overconfident models due to its mode-seeking nature. Second, α-divergence generalizes KL by introducing a parameter α to interpolate between mode-seeking and mass-covering behaviors, adding expressiveness from a probabilistic standpoint but at the cost of optimization stability in deep architectures, especially in high-dimensional parameter spaces. Finally, Monte Carlo (MC) Dropout emerges from deep learning practice—originally a regularization technique—reinterpreted through a probabilistic lens by Gal and Ghahramani (2016) as approximate Bayesian inference; here, stochastic subnetworks are sampled using dropout masks during both training and inference, producing calibrated uncertainty without learning full posterior distributions, making it computationally efficient and stable. Across our evaluations, KL-VI and α-divergence offered deeper posterior modeling fidelity but demanded careful probabilistic calibration to avoid instability, while MC Dropout and Laplace approximation favored practical deployment by blending deep learning routines with lightweight uncertainty modeling. Thus, every concept we explored—loss curves, optimizer behavior, ELBO structure, divergence selection, and inference techniques—is deeply tied to either deep learning (through architecture, optimization, and prediction accuracy) or probabilistic inference (through uncertainty modeling, posterior estimation, and Bayesian regularization). Together, they demonstrate how modern BNNs exist at the intersection of these two fields: requiring deep learning's computational frameworks to scale and optimize while drawing on probabilistic inference to reason under uncertainty and ensure model robustness in the alignment with our proposal to detect cancer patients early.

References

1. Sung, H., Ferlay, J., Siegel, R. L., Laversanne, M., Soerjomataram, I., Jemal, A., & Bray, F. (2021). *Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries*. CA: A Cancer Journal for Clinicians, 71(3), 209–249. https://doi.org/10.3322/caac.21660

2. National Cancer Institute. (2021, October 11). *What is cancer?* https://www.cancer.gov/about-cancer/understanding/what-is-cancer

3. Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). *Hands-on Bayesian neural networks: A tutorial for deep learning users*. IEEE Computational Intelligence Magazine, 17(2), 29–48. https://doi.org/10.1109/MCI.2022.3155327

4. Younes, H. L. S., & Littman, M. L. (2004). *PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects* (CMU-CS-04-167). School of Computer Science, Carnegie Mellon University. http://www.cs.rutgers.edu/~mlittman/papers/ppddl1.pdf

5. Littman, M. L. (1997). *Probabilistic propositional planning: Representations and complexity*. In *Proceedings of the 14th National Conference on Artificial Intelligence* (pp. 748–754). AAAI Press. https://cdn.aaai.org/AAAI/1997/AAAI97-116.pdf

6. Richter, S., & Westphal, M. (2010). *The LAMA planner: Guiding cost-based anytime planning with landmarks*. Journal of Artificial Intelligence Research, 39, 127–177. https://doi.org/10.1613/jair.2972

7. Planning.wiki. (n.d.). *What is PDDL?* https://planning.wiki/guide/whatis/pddl

8. van de Schoot, R., Kaplan, D., Denissen, J., Asendorpf, J. B., Neyer, F. J., & van Aken, M. A. G. (2014). *A gentle introduction to Bayesian analysis: Applications to developmental research*. Child Development, 85(3), 842–860. https://doi.org/10.1111/cdev.12169

9. Terven, J. R., Cordova-Esparza, D. M., Ramirez-Pedraza, A., Chavez-Urbiola, E. A., & Romero-Gonzalez, J. A. (2024). *Loss functions and metrics in deep learning* [Preprint]. https://arxiv.org/pdf/1811.00639

10. Kullback, S., & Leibler, R. A. (1951). *On information and sufficiency*. The Annals of Mathematical Statistics, 22(1), 79–86. https://doi.org/10.1214/aoms/1177729694

11. Tran, D., Dusenberry, M. W., Hafner, D., & van der Wilk, M. (2019). *Bayesian layers: A module for neural network uncertainty*. In *Advances in Neural Information Processing Systems* (NeurIPS 2019). https://proceedings.neurips.cc/paper_files/paper/2019/file/154ff8944e6eac05d0675c95b5b8889d-Paper.pdf

12. Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization*. arXiv. https://arxiv.org/abs/1412.6980

13. MacKay, D. J. C. (1992). *A practical Bayesian framework for backpropagation networks*. Neural Computation, 4(3), 448–472. https://doi.org/10.1162/neco.1992.4.3.448

14. 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). https://proceedings.mlr.press/v48/gal16.html

15. Blundell, C., Cornebise, J., Kavukcuoglu, K., & Wierstra, D. (2015). *Weight uncertainty in neural networks*. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1613–1622). https://proceedings.mlr.press/v37/blundell15.html

16. Li, Y., & Gal, Y. (2017). *Dropout inference in Bayesian neural networks with α-divergences*. In *Proceedings of the 34th International Conference on Machine Learning* (pp. 2052–2061). https://proceedings.mlr.press/v70/li17a.html

17. Duvenaud, D. (n.d.). *Distill Bayesian networks tutorial*. University of Toronto. https://www.cs.toronto.edu/~duvenaud/distill_bayes_net/public/