Trustworthy Deep Learning Acceleration with Customizable Design Flow Automation

Zhiqiang Que
Computing
Imperial College London
London, United Kingdom
z.que@imperial.ac.uk

Hongxiang Fan
Imperial College London
London, United Kingdom
h.fan17@imperial.ac.uk

Gabriel Figueiredo
Imperial College London
London, United Kingdom
gabriel.figueiredo@imperial.ac.uk

Ce Guo
Imperial College London
London, United Kingdom
c.guo@imperial.ac.uk

Wayne Luk
Imperial College London
London, United Kingdom
w.luk@imperial.ac.uk

Ryota Yasudo
Kyoto University
Kyoto, Japan
yasudo@i.kyoto-u.ac.jp

Masato Motomura
Institute of Science Tokyo
Tokyo, Japan
motomura@artic.iir.titech.ac.jp

Abstract

In recent years, deep learning has brought the development of accurate and complex models across various domains. Deploying these models efficiently on resource-constrained platforms while maintaining high accuracy and trustworthiness, however, remains a critical challenge. This paper introduces an automated framework for optimizing trustworthy deep learning by enabling trade-off between three metrics: computational efficiency, trustworthiness, and predictive accuracy. Traditional compression techniques such as pruning and scaling reduce computational complexity but can compromise model calibration and uncertainty quantification, which is critical for safety-critical applications. To address this challenge, we integrate Monte Carlo Dropout (MCD) for Bayesian Convolutional Neural Networks (BayesCNNs) and propose an automated Design Space Exploration (DSE) approach driven by Bayesian Optimization to identify Pareto-optimal configurations. Our framework dynamically tunes pruning rates, dropout probabilities, and other parameters to achieve Pareto-optimal trade-offs between accuracy, efficiency, and uncertainty estimation. Two BayesCNN architectures are evaluated to demonstrate that our approach can systematically optimize deep learning models for trustworthiness and efficiency. Our results show that no single configuration is optimal across all metrics, demonstrating the need to automate and customize co-optimization strategies. Compared to state-of-the-art FPGA implementations, our optimized design achieves up to 7.67× faster inference and 12.8× higher energy efficiency while maintaining well-calibrated uncertainty estimates.

This work is licensed under a Creative Commons Attribution International 4.0 License.

HEART 2025, Kumamoto, Japan

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-1432-0/25/05
https://doi.org/10.1145/3728179.3728198

ACM Reference Format:
Zhiqiang Que, Hongxiang Fan, Gabriel Figueiredo, Ce Guo, Wayne Luk, Ryota Yasudo, and Masato Motomura. 2025. Trustworthy Deep Learning Acceleration with Customizable Design Flow Automation. In *The International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies 2025 (HEART 2025), May 26–28, 2025, Kumamoto, Japan.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3728179.3728198

1 Introduction

The continuous advancement of deep learning has allowed increasingly accurate and complex models for various applications, from computer vision [4, 13] to natural language processing [10]. However, efficiently deploying these models on edge and embedded systems is still challenging due to limited computational power, memory, and energy constraints [22]. To address these constraints, model compression techniques, such as pruning and quantization, are commonly used to reduce computational complexity while preserving accuracy.

At the same time, trustworthiness in AI is becoming an increasingly important requirement, particularly in safety-critical applications like autonomous driving [9], healthcare [23], and finance. Conventional deep learning models tend to be overly confident in their predictions and struggle to express uncertainty, which may result in errors in high-risk decision-making environments. To address this, Bayesian Convolutional Neural Networks (BayesCNNs) are utilized to estimate uncertainty using techniques such as Monte Carlo Dropout (MCD). However, these Bayesian approaches can introduce a significant computational overhead, since multiple stochastic forward passes are required for reliable uncertainty estimates.

The key challenge is to enable a trade-off between compression, trustworthiness, and accuracy, which are three competing metrics that are difficult to optimize simultaneously. Each metric affects the other ones, and requires a careful trade-off to support both efficiency and reliability.

- **Compression**: reducing model size through techniques such as pruning and scaling reduces memory usage, lowers energy consumption, and speeds up inference. This is especially important for Bayesian CNNs where Monte Carlo Dropout (MCD) requires multiple forward passes, increasing computational costs. However, aggressive pruning may remove essential network parameters, leading to accuracy loss and making the model behave more deterministically, diminishing its ability to estimate uncertainty.
- **Trustworthiness**: Bayesian methods such as MCD can improve model reliability by generating predictive distributions rather than a single-point result. However, each input leads to multiple forward passes and increases inference time as well as energy consumption. While compression techniques can reduce per-pass computation overhead, excessive pruning – as mentioned above – can disrupt Bayesian uncertainty calibration, resulting in unreliable confidence estimates.
- **Accuracy**: The key objective of a deep learning model is to keep high predictive accuracy, but compression and trustworthiness mechanisms can introduce competing constraints. Pruning can remove critical representations with information loss which degrades accuracy, while MCD introduces variance in predictions due to randomized dropout masks, which affects stability.

Given the above considerations, we believe that manually tuning the trade-offs between compression, uncertainty, and accuracy is impractical in both large-scale models and edge AI applications due to the vast design space. So a naive approach, such as grid search, would be computationally expensive and inefficient.

To address these challenges, this paper introduces an automated framework for developing Bayesian neural networks by optimizing trade-offs between these three metrics. We propose Design Space Exploration (DSE) driven by Bayesian Optimization to dynamically select pruning rates, scaling factors, dropout probabilities, and Bayesian layer configurations. Our approach supports efficient navigation of this large design space, identifying configurations with effective trade-offs between compression, trustworthiness, and predictive accuracy.

Although prior work has individually explored techniques such as model compression [21], uncertainty estimation with Monte Carlo Dropout [8], and Bayesian Optimization, our work is the first to automate co-optimization of these techniques in a unified framework for acceleration. Moreover, we extend existing methods – which primarily focus on static configurations and manual tuning [8] – by using DSE driven by Bayesian Optimization to dynamically adjust optimization parameters. This enables the discovery of Pareto-optimal configurations across compression, trustworthiness and accuracy dimensions while eliminating the need for manual tuning.

Our contributions are as follows:

(1) An automated DSE for Bayesian CNNs (**Section 3**) which:
   - explores optimal configurations for compression techniques, such as pruning rates (β_p) and scaling factors (β_s) dynamically;
   - adjusts MCD settings including dropout rate p, number of Bayesian layers B to improve uncertainty estimates;
   - optimizes the trade-offs between compression, trustworthiness and accuracy, generating a set of Pareto-optimal configurations.
(2) Custom optimization strategies tailored to different deployment scenarios, such as reducing computation for efficiency but maintaining trustworthiness (**Section 4**).
(3) Comprehensive evaluation of our approach (**Section 5**). Our designs achieve up to 7.67× faster inference and 12.8× higher energy efficiency when compared to state-of-the-art FPGA-based implementations, demonstrating significant improvements in performance.

2 Related Work

Co-optimization for Deep Learning Accelerators. Efficient deployment of deep learning models often requires co-optimization of both software and hardware, especially in resource-constrained platforms such as FPGAs and edge devices.

Conventional approaches for deep learning acceleration typically rely on predefined hardware templates, which limits flexibility in design space exploration [6, 12, 15, 24, 28]. Recent work has introduced hardware-aware Neural Architecture Search (NAS) [1, 18] and mixed-integer geometric programming-based frameworks [5] to automate deep learning accelerator design, optimizing both deep learning architectures and hardware efficiency.

Some approaches, such as MetaML [21], advance this field by integrating multi-level deep learning accelerator optimization with metaprogramming techniques [25] to enable automated source-to-source transformations for high-level synthesis (HLS), optimizing hardware efficiency without fixed hardware templates.

Uncertainty Estimation in Bayesian Deep Learning. Bayesian deep learning has been widely studied to enhance model trustworthiness, particularly through MCD and Bayesian Neural Networks (BayesNNs).

Recent efforts have sought to optimize Bayesian CNN acceleration using hardware-aware techniques. For example, FPGA-based Bayesian CNN architectures [8] utilize MCD with intermediate-layer caching to reduce redundant computations. Besides, [8] introduces structured sparsity techniques which exploit channel, layer, and sample sparsity to skip redundant computations. Bayesian modeling techniques have also been extended to Graph Neural Networks (GNNs), improving robustness against adversarial attacks and enhancing predictive uncertainty [27].

Despite advances in Bayesian deep learning, prior studies lack an automated method for jointly optimizing compression, uncertainty estimation, and accuracy. They rely on manual tuning or fixed configurations. The proposed framework leverages Bayesian Optimization to automate the tuning of pruning rates, dropout settings, and Bayesian layers, in order to achieve optimal trade-offs between efficiency and trustworthiness based on design requirements. Unlike previous static approaches which tie to specific hardware constraints, our method is adaptive and can generalize across diverse deployment scenarios.

Key novelties include:

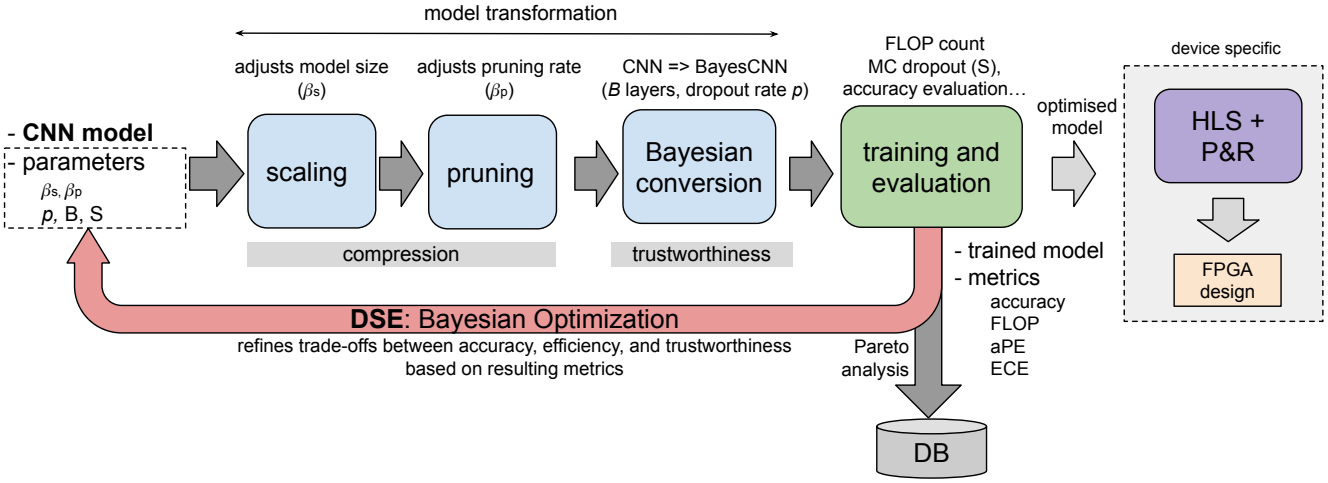

Figure 1: Our Bayesian Optimization process is platform-agnostic, enabling trade-off between compression, uncertainty estimation, and efficiency through scaling, pruning, and Bayesian conversion. It iteratively refines model parameters based on performance metrics such as accuracy, FLOP, aPE, and ECE. Identified optimized models are prime candidates for hardware acceleration on devices such as FPGAs (see Section 5.6).

(i) systematic exploration of the design space, eliminating the need for manual tuning;
(ii) modeling the complex trade-offs between efficiency, accuracy, and trustworthiness;
(iii) applicability across different platforms, including CPUs, GPUs, and FPGAs.

3 Approach

This section presents a platform-agnostic approach to optimize deep learning models and ensure they remain small, efficient, and trustworthy. We achieve this through an automated framework that enables optimizing trade-offs between compression, trustworthiness, and accuracy. This framework navigates these trade-offs using a design space exploration (DSE) to minimize computational cost while identifying high-performance configurations. Its modular architecture allows the integration of diverse optimization techniques even under challenging conditions such as out-of-distribution (OoD) inputs.

3.1 Workflow

Figure 1 shows the workflow of our optimization process. It follows a sequence of transformations to progressively fine-tune the model in order to identify optimal trade-offs between different metrics.

The process begins with the compression stage, where the model is optimized for computational efficiency through *scaling* and *pruning*. Scaling modifies the network size based on **scaling factor** β_s (Table 1). Given an original CNN with L_0 length and W_0 width per layer, the scaled network has:

$$L = \lfloor \beta_s L_0 \rfloor, \quad W = \lfloor \beta_s W_0 \rfloor \tag{1}$$

where $\lfloor \cdot \rfloor$ denotes the floor function. A smaller β_s reduces computational complexity but risks removing important features. Pruning, on the other hand, removes unimportant weights from the model based on a pruning rate β_p, defined as:

$$W_{\text{pruned}} = (1 - \beta_p) W_{\text{orig}} \tag{2}$$

where W_{orig} is the original number of weights. Higher pruning rates ($\beta_p \rightarrow 1$) reduce computational complexity but may degrade accuracy.

Following compression, the trustworthiness stage transforms the model into a Bayesian Convolutional Neural Network (BayesCNN) by integrating Monte Carlo Dropout (MCD) and Bayesian layers. This enables the model to quantify predictive uncertainty. MCD supports controlled stochasticity during the evaluation phase, and it enables uncertainty estimation by incorporating dropout distributions rather than fixed values. The dropout rate (p) and number of Bayesian layers (B) can be fine-tuned to optimize the trade-offs between uncertainty estimation and computational cost(Table 1).

After the trustworthiness stage, models are trained and then evaluated to calculate key performance metrics, including model accuracy, computational efficiency, and uncertainty estimations. Computational efficiency is measured in terms of floating-point operations (FLOP), a widely used proxy metric for estimating the computational cost. Please note that this does not refer to floating-point operations per second. This provides a hardware-agnostic measure of model complexity and efficiency. In our approach, FLOP corresponds to the total number of floating-point operations required to execute all forward steps, taking into account the pruning rate. In addition, uncertainty quantification is measured using Average Predictive Entropy (aPE) and Expected Calibration Error (ECE), as presented in Table 2.

Finally, this framework employs Bayesian Optimization to dynamically refine hyperparameters, including compression factors (β_s, β_p) and uncertainty settings (p, B). This process systematically explores trade-offs among model accuracy, efficiency, and trustworthiness, ultimately converging toward Pareto-optimal solutions.

3.2 Uncertainty Estimation

Bayesian Conversion. Traditional Convolutional Neural Networks (CNNs) are inherently deterministic and lack the ability to

Table 1: Optimization Parameters

Parameter	Description	Range	Impact on Optimization
β_p	**pruning rate:** determines the percentage of network weights removed to reduce model size and computation, potentially affecting accuracy and uncertainty estimation.	5% - 95%	higher pruning rates reduce memory and computational costs but can degrade accuracy and uncertainty estimation.
β_s	**scaling rate:** adjusts the proportion of network layers or neurons to shrink the model, balancing efficiency with accuracy retention.	5% - 95%	decreasing the scaling rate decreases model size and computational demands but may negatively impact accuracy and uncertainty quantification.
p	**dropout rate for Monte Carlo Dropout (MCD):** controls the probability of dropping neurons during inference, improving uncertainty estimation at the cost of feature richness.	5% - 95%	higher dropout rates enhance uncertainty estimation but reduce feature representation quality, which can affect model stability.
B	**number of Bayesian layers:** specifies the number of layers incorporating Bayesian inference, enhancing uncertainty quantification while increasing computational overhead.	1 to N (Total CNN layers)	increasing the number of Bayesian layers improves uncertainty estimation but adds computational overhead, influencing model efficiency.
S	**number of Monte Carlo samples:** a higher number of stochastic forward passes for uncertainty estimation, improving reliability but slowing inference.	10	a higher number of samples improves uncertainty quantification but slows inference, impacting real-time deployment feasibility. For simplicity, this is fixed to 10 in this work, but the proposed approach can support varying values.

quantify uncertainty, which is critical for decision-making in safety-critical applications. A model's confidence in its predictions should align with its accuracy; otherwise, it may make overconfident yet incorrect decisions. To support trustworthiness, we automatically transform a CNN into a Bayesian CNN (BayesCNN) by replacing deterministic dropout with Monte Carlo Dropout (MCD).

Monte Carlo Dropout (MCD). Monte Carlo Dropout (MCD) provides an alternative approximation to Bayesian inference. It introduces stochasticity at inference time by randomly dropping neurons with a probability p. The final prediction is obtained by averaging multiple stochastic forward passes:

$$\hat{y} = \frac{1}{S}\sum_{s=1}^{S} f(x; W_s, p) \tag{3}$$

where:

- S is the number of Monte Carlo samples (forward passes).
- W_s represents a different set of randomly sampled weights from the Bayesian layers at each forward pass due to dropout.
- p is the dropout probability (see Table 1), which controls the fraction of neurons randomly deactivated at each inference step to introduce uncertainty.
- $\hat{y}$ represents the final predictive output, serving as the best estimate of the model's output.

MCD allows models to quantify uncertainty through aPE and ECE. It mitigates overconfidence by providing well-calibrated probability estimates, ensuring accuracy (Acc) aligns with confidence. In addition, it helps detect out-of-distribution (OoD) inputs, making it valuable for high-risk decision tasks.

However, its effectiveness depends on dropout rate (p), the number of Monte Carlo samples (S) and the number of Bayesian layers (B), as too few samples degrade uncertainty estimation while too many add computational overhead. MCD introduces computational overhead since multiple forward passes (S) increase floating-point operations (FLOP). This will slow inference. While higher dropout rates (p) improve uncertainty estimation, they will reduce the effective feature representation and impact classification performance. Additionally, pruning rate (β_p) and scaling factor (β_s) influence both efficiency and calibration; excessive pruning or downscaling can disrupt Bayesian inference, degrading ECE and aPE. The number of Bayesian layers (B) also affects uncertainty quantification – it requires tuning to optimize the trade-off between trustworthiness, accuracy, and efficiency.

Expected Calibration Error (ECE) [11]. ECE quantifies how well a model's predicted confidence correlates with its actual accuracy. A well-calibrated model should satisfy:

$$P(Y = \hat{Y} \mid \hat{P}) \approx \hat{P} \tag{4}$$

where $\hat{P}$ is the model's predicted probability for its top class. ECE is computed by binning predictions into M intervals $[B_1, B_2, ..., B_M]$ and measuring the average absolute difference between accuracy and confidence within each bin:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\text{acc}(B_m) - \text{conf}(B_m)\right| \tag{5}$$

where:

- N is the total number of predictions.

Table 2: Platform-Agnostic Design Metrics

Metric	Description	Typical Range	Impact on Optimization
Accuracy	**model accuracy:** measures the classification performance of the model on test data	80% - 99%	higher accuracy is essential for reliable predictions.
aPE	**average predictive entropy:** quantifies model uncertainty by evaluating the entropy of the predictive distribution	$0.001 - 2.0$	a higher aPE suggests greater uncertainty, which helps identify ambiguous or out-of-distribution inputs.
ECE	**expected calibration error:** assesses how well the predicted confidence aligns with actual correctness.	$0.01 - 0.2$	a lower ECE indicates better-calibrated confidence scores, reducing the risk of overconfident incorrect predictions.
FLOP	**floating-point operations:** represents the total number of floating-point operations required for running S forward passes of the model, taking into account pruning rate (β_p). Note that this does not refer to floating-point operations per second.	model-specific	lower FLOP values improve computational efficiency and reduce inference latency, but excessive reductions may degrade model accuracy.

- $\text{acc}(B_m)$ is the accuracy within bin B_m.
- $\text{conf}(B_m)$ is the average confidence in bin B_m.

Consider a model that predicts class probabilities for test images. Suppose it classifies several images and, within a specific confidence bin (80%-90%), assigns an average confidence of 85%, but the actual accuracy is only 75%. This means overconfidence and results in a high ECE. Such miscalibration is often worsened by extreme compression settings, such as high pruning rates (β_p) and scaling (β_s), which remove network parameters and degrade calibration. On the other hand, a well-calibrated model with a lower ECE has confidence values that closely align with actual accuracy across all bins and ensures more reliable decision-making.

Average Predictive Entropy (aPE) [8]. aPE measures the uncertainty in the predictions of a model by evaluating the entropy of the probability distribution:

$$\text{aPE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} P(y_c \mid x_i)\log P(y_c \mid x_i) \tag{6}$$

where:

- C is the number of classes.
- $P(y_c \mid x_i)$ is the predicted probability of class c for input x_i.

To understand aPE, consider an image classification model where uncertainty quantification is important for detecting ambiguous or OoD inputs. Suppose a model trained on CIFAR-10 is evaluated on two test samples: a clean image of an animal and a noisy variant generated with the same mean and variance as the training data. For the clean image, a well-calibrated Bayesian model should express strong confidence, resulting in a low aPE. While for the noisy image, which deviates from the original distribution, the same model should express higher uncertainty, yielding a higher aPE. This effect is influenced by the dropout rate, where a lower p often results in lower uncertainty, while a higher value increases uncertainty. Ensuring that aPE rises for noisy or OoD inputs helps the model signal unreliable predictions, which is critical for risk-sensitive applications. Note that we use noisy inputs in our evaluation to specifically assess aPE in Section 5.

3.3 Bayesian Optimization

Bayesian Optimization (BO) is a powerful technique which can efficiently optimize functions that are expensive to evaluate. This makes it particularly useful for Design Space Exploration (DSE) in deep learning. Unlike conventional optimization methods, which may require exhaustive searches or gradient-based techniques, BO builds a surrogate model of the objective function, which allows it to predict promising configurations without evaluating every possibility. This surrogate model, often based on a Gaussian Process (GP), estimates the relationship between different parameter choices and their outcomes, enabling the optimization process to be uncertainty-aware. Instead of blindly testing multiple configurations, BO strategically selects new points to evaluate, improving efficiency while maintaining accuracy.

To guide this selection, BO uses an acquisition function to determine where to sample next by balancing exploration (trying uncertain regions to discover better solutions) and exploitation (refining known good solutions). We use Expected Improvement (EI) function, which prioritizes evaluating configurations that are likely to outperform the best-known result. This approach ensures that BO efficiently converges toward the best configurations while avoiding wasted computations on suboptimal regions.

In our approach, BO is used to optimize hyperparameters governing deep learning model compression, trustworthiness, and accuracy. Let $x = (\beta_p, \beta_s, p, B)$ represent the configuration parameters:

- β_p: pruning rate
- β_s: scaling factor
- p: dropout rate for Monte Carlo Dropout (MCD)
- B: number of Bayesian layers

The optimization problem is formulated as:

$$x^* = \arg\max_x f(x) \tag{7}$$

subject to model performance constraints (e.g., accuracy A, average predictive entropy aPE, expected calibration error ECE). The objective function combines different performance metrics using a

weighted sum:

$$f(x) = \sum_{m \in Acc, FLOP, aPE, ECE} W_m \cdot \text{Norm}_m(x) \tag{8}$$

where W_m are user-defined weights that identify the relative importance of each metric, and $\text{Norm}_m(x)$ normalizes the metric values. Infeasible configurations are penalized using an infinite negative score in order to enforce constraints:

$$f(x) = -\text{sys.maxsize}, \quad \text{if constraints not met} \tag{9}$$

This ensures that configurations exceeding hardware limits or violating user-supplied constraints are automatically discarded. The BO algorithm iterates over several rounds, refining hyperparameters until convergence or a predefined budget is exhausted.

We find that the proposed formulation allows our framework to efficiently navigate the design space, identifying Pareto-optimal configurations with effective trade-offs between efficiency, accuracy, and trustworthiness in deep learning model deployment.

4 Optimization Strategies

To explore the trade-offs between compression, trustworthiness, and accuracy, we define multiple optimization strategies tailored to various deployment scenarios. We encode different strategies by revising the weights in Eq. 8 and enforce constraints using Eq. 9 to penalize configurations exceeding computational limits or violating accuracy constraints.

4.1 Minimizing Computational Cost while Preserving Accuracy (S1)

This strategy aims to prune redundant parameters and also scale down model size to reduce the computational cost while maintaining model accuracy.

The pruning rate β_p and the scaling factor β_s are tuned to ensure that the resulting model remains lightweight without sacrificing model accuracy. However, aggressive FLOP reduction can lead to a loss of essential features and impact the model's ability to generalize. It can also disrupt the uncertainty calibration. Therefore, this strategy ensures that when FLOP is reducing, the accuracy drop is constrained within acceptable margins to avoid severe performance degradation.

4.2 Maximizing Accuracy while Constraining Computational Cost (S2)

For applications where model accuracy is critical, this strategy prioritizes accuracy while considering computational costs. The pruning and scaling rates as well as Monte Carlo Dropout (MCD) configurations are fine-tuned to keep critical network parameters, so that the model can maintain high accuracy. While this strategy aims for high accuracy, it has the cost of increased computation. Thus, the number of FLOP is constrained in our experiments to prevent excessive computational overhead.

4.3 Minimizing Calibration Error under Constraints (S3)

This strategy aims to minimize ECE to ensure that the model produces well-calibrated predictions that align with true confidence levels. This strategy tunes both the dropout rate β_d and the number of Bayesian layers B to achieve reliable uncertainty estimation. However, minimizing ECE introduces additional challenges that aggressive dropout may reduce accuracy by eliminating valuable feature representations. To address the challenge, we adjust ECE reduction taking into account both FLOP and accuracy constraints, leading to a well-calibrated and computationally efficient model. This strategy is useful for applications where uncertainty-awareness is essential, such as medical diagnosis and autonomous decision-making.

4.4 Balancing Accuracy and Trustworthiness (S4)

This strategy focuses on identifying trade-offs between accuracy, aPE, and ECE. Unlike previous strategies which optimize for a single objective, this strategy ensures that no individual metric is disproportionately prioritized at the cost of others. Bayesian optimization explores Pareto-optimal solutions and searches for the models with sufficient accuracy while maintaining well-calibrated uncertainty estimates. This strategy is well-suited for real-world applications since maintaining both accuracy and trustworthiness is critical, particularly for high-risk environments.

4.5 Balancing Computational Cost and Trustworthiness (S5)

This strategy focuses on balancing FLOP, aPE, and ECE to achieve a lightweight but reliable model. The pruning and scaling rates are adjusted to reduce model complexity while the MCD configurations are adjusted to maintain acceptable uncertainty calibration. This strategy ensures that computational efficiency remains a key consideration but the challenge is how to prevent over-pruning and over-down-scaling which could degrade uncertainty estimation. The proposed Bayesian optimization based framework mitigates this by dynamically adjusting hyperparameters to find configurations that offer the best compromise between efficiency and trustworthiness.

4.6 Balancing Accuracy, Computational Cost and Trustworthiness (S6)

This strategy optimizes all four metrics to ensure that a model remains compact, well-calibrated, and highly accurate under resource constraints. Unlike S4 (balancing accuracy and trustworthiness) and S5 (balancing computational cost and trustworthiness), this strategy jointly optimizes pruning rate β_p, scaling rate β_s, dropout rate β_d, and Bayesian layer number B while imposing constraints on both FLOP and accuracy. To achieve this trade-off, Bayesian Optimization dynamically tunes hyperparameters to identify Pareto-optimal configurations to prevent over-pruning which can degrade accuracy and uncertainty estimation, and excessive Bayesian inference overhead which increases latency and energy consumption. This strategy is particularly useful for deployments on resource-limited devices that require reliable decision-making, such as real-time healthcare monitoring, autonomous robotics, and embedded AI systems.

Table 3: The resultant configurations, and the corresponding performance of the Bayes-LeNet5.

Opt-Mode	β_p	β_s	p	B	ECE (%) ↓	aPE (nats) ↑	Accuracy (%) ↑	FLOP (10^6) ↓
Opt-Confidence	0.35	0.95	0.15	1	**0.55**	0.5718	99.07	28.2
Opt-Uncertainty	0.95	0.70	0.05	3	0.95	**1.124**	97.76	1.26
Opt-Accuracy	0.65	0.95	0.05	1	0.56	0.4013	**99.10**	15.2
Opt-Efficiency	0.95	0.60	0.30	1	1.53	0.8458	96.51	**0.96**
Balance-Top1	0.95	0.70	0.05	3	0.95	1.124	97.76	1.26
Balance-Top2	0.95	0.85	0.05	1	0.82	0.9247	98.14	1.77
Balance-Top3	0.95	0.95	0.05	3	0.89	0.9195	98.11	2.17

5 Evaluation

To evaluate the effectiveness of the proposed framework, we conduct experiments on Bayesian Convolutional Neural Networks (BayesCNNs) across two datasets and architectures. This section provides details of our experimental setup and results.

5.1 Experimental Setup

As presented in Table 2, we consider model predictive accuracy (Acc) for assessing classification performance, FLOP as a measure of computational efficiency, average predictive entropy (aPE) [8] for evaluating the quality of quantified uncertainty and expected calibration error (ECE) [11] to measure the calibration of the confidence in the predictions. This evaluation focuses on the **S6** strategy (balancing accuracy, computational cost and trustworthiness) and assigns an equal weight of 0.25 to each of the four metrics. This ensures a comprehensive trade-off among key performance metrics.

To evaluate the effectiveness of our framework, we use the BayesCNNs on image classification datasets. To stay consistent with previous designs [2, 3, 26], we consider MNIST [17], and CIFAR-10 [16]. We utilize two widely used CNN architectures: LeNet-5 [17], a lightweight model suitable for small-scale datasets like MNIST, and ResNet-18 [14], a deeper network designed for more complex datasets such as CIFAR-10. These models are implemented using TensorFlow and Keras. For training and evaluation (see green box in Fig. 1), we use the NVIDIA RTX 3090 GPU to speed-up training and compute model predictive accuracy after the model transformation phase.

To assess the quality of uncertainty estimation for inputs that should inherently confuse the network, we evaluate the model's response to random Gaussian noise generated with the same mean and variance as the training data. As previously mentioned, we quantify uncertainty using aPE, computed over a dataset of size N with C classes as shown in Eq. 6. This metric captures the entropy of the model's predictive distribution, providing insight into how well the model expresses uncertainty when encountering ambiguous or out-of-distribution inputs. Hence, for our evaluation a higher aPE value is desirable.

In addition, we evaluate the calibration of the BayesCNN's confidence using ECE on unmodified test data. ECE is computed as a weighted average of the absolute difference between accuracy and confidence across bins as shown in Eq. 5. Lower ECE values indicate better-calibrated models, ensuring that confidence scores reliably reflect true predictive performance.

The evaluation results presented in Sections 5.2–5.5 are platform-agnostic. In contrast, the results in Section 5.6 are platform-specific, based on the most efficient DNN model (Opt-Efficiency) and the most confident model (Opt-Confidence) derived from our approach, which are subsequently synthesized to hardware. Our FPGA workflow employs HLS4ML to convert selected models to HLS for AMD Xilinx Kintex UltraScale KU115 FPGAs, operating at a default frequency of 181 MHz. The bit width is set to 16-bit fixed-point precision with 6 integer bits. Power consumption is reported by Vivado after place and route.

5.2 Trade-Off between Accuracy, Efficiency, and Trustworthiness

This section leverages Bayesian Optimization-driven design space exploration to automatically identify Pareto-optimal configurations.

To illustrate the effectiveness of our approach, we present both Table 3 and Figure 2 to conduct Pareto frontier analysis for the Bayes-LeNet5 model.

Table 3 presents the optimized configurations for Bayes-LeNet5 with different metrics. It highlights the impact of key hyperparameters, including pruning rate (β_p), scaling rate (β_s), dropout probability (p), and the number of Bayesian layers (B), on model performance. The Opt-Confidence design achieves the lowest ECE (0.55%) and produces the most trustworthy confidence estimates. However, this configuration is computationally expensive with a high FLOP count (28.2E+6). The Opt-Uncertainty prioritizes uncertainty estimation by maximizing average predictive entropy (aPE = 1.124 nats). The Opt-Accuracy achieves the highest model accuracy (99.10%) in our search and it shows that accuracy optimization alone does not always lead to well-calibrated uncertainty estimates. Moreover, the Opt-Efficiency design has a small FLOP count, just 0.96E+6, nearly 30× smaller than the Opt-Confidence design. However, this efficiency comes at a trade-off in accuracy (96.51%). We also have balanced designs (Balance-Top1, Top2, Top3) which represent the top-3 solutions found by Bayesian Optimization search

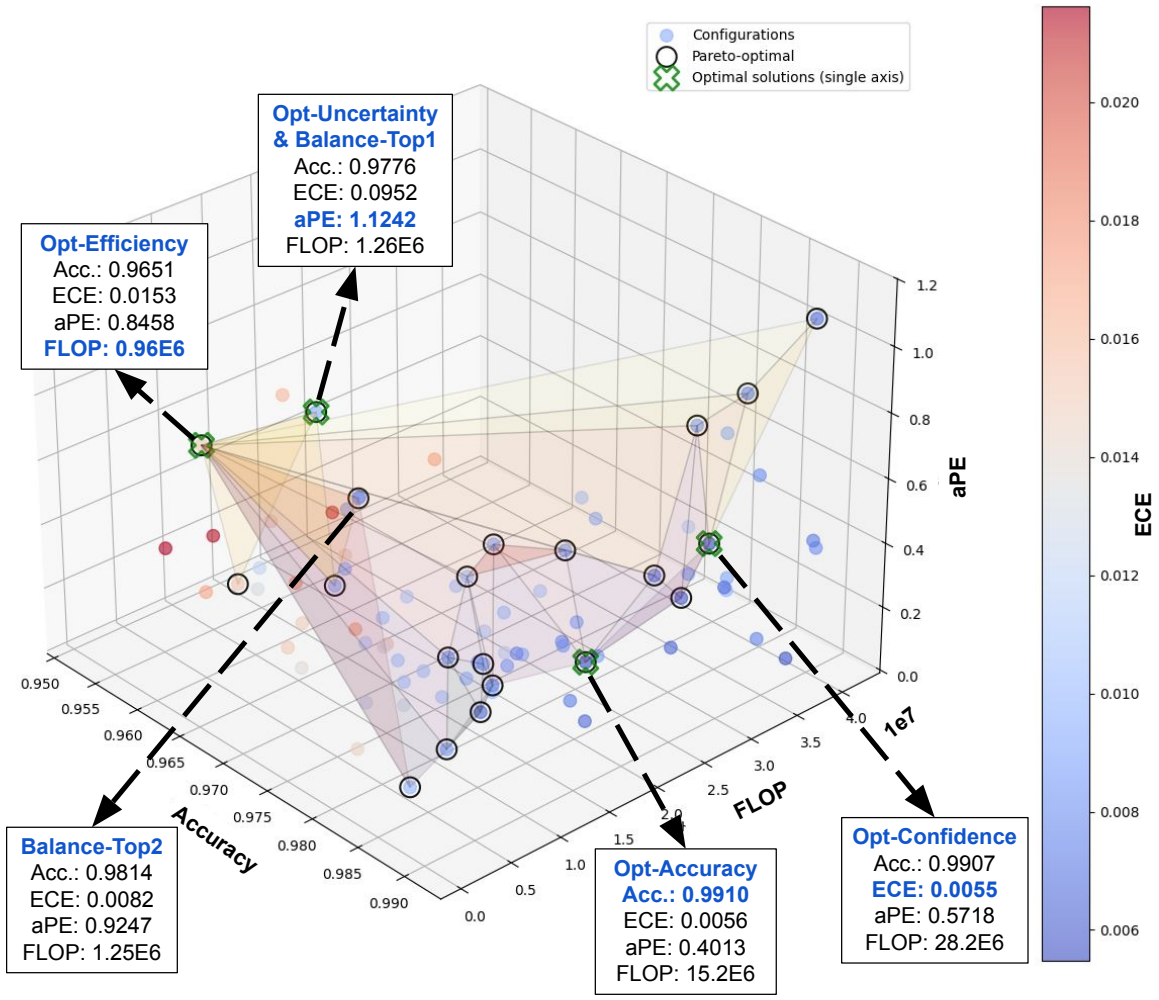

Figure 2: Pareto frontier analysis of Bayesian LeNet5 on MNIST, illustrating the trade-offs between accuracy, FLOP, Expected Calibration Error (ECE), and Average Predictive Entropy (aPE). Each point represents a model candidate.

under the balanced strategy. They achieve good trade-offs across all key metrics.

While Table 3 shows the key design candidates, Figure 2 shows all the designs and demonstrates the effectiveness of our framework by plotting the Pareto-optimal solutions for Bayes-LeNet5. This figure highlights the trade-offs between Accuracy, FLOP, aPE, and ECE. Opt-Efficiency is positioned in the low-computation region and it demonstrates extreme efficiency with significantly reduced FLOP. However, this comes at the cost of low accuracy and weak uncertainty estimate. Opt-Accuracy and Opt-Confidence are positioned at a high-computation region with increased computational burden. Balance-Top1 shows good trade-offs to provide well-calibrated uncertainty estimates while maintaining reasonable efficiency. Finally Balance-Top2, which is on the Pareto frontier, offers good compromises between FLOP, accuracy, and uncertainty metrics, making it a practical choice for real-world deployment where both efficiency and trustworthiness are required.

Both Table 3 and Figure 2 show that no single configuration is optimal across all dimensions, which means that accuracy, uncertainty estimation, and computational efficiency must be co-optimized. Our experimental results demonstrate that the proposed framework can generate balanced solutions, achieving compact, accurate, and trustworthy BayesCNNs.

5.3 Extending the Analysis to ResNet-18 on CIFAR-10

Following our analysis of Bayes-LeNet5, we extend our evaluation to a more complex model, Bayes-ResNet18, trained on the CIFAR-10 dataset. Using the proposed DSE, we analyze trade-offs between FLOP, accuracy, and uncertainty estimation. The results are summarized in Table 4, and the Pareto frontier analysis (Figure 3) illustrates trade-offs across different options.

Opt-Confidence achieves the lowest ECE (7.47%), which shows well-calibrated confidence estimates, as shown in Table 4. However, similar to Bayes-LeNet5, this design candidate has a high FLOP count as 10.6E+8. This means improved calibration comes at a computational cost.

Table 4: The resulting configurations, and the corresponding performance of the Bayes-ResNet18 on CIFAR10.

Opt-Mode	β_p	β_s	p	B	ECE (%) ↓	aPE (nats) ↑	Accuracy (%) ↑	FLOP (10^8) ↓
Opt-Confidence	0.85	0.80	0.15	1	**7.47**	0.2638	87.61	10.6
Opt-Uncertainty	0.80	0.60	0.05	3	7.64	**1.6464**	88.07	7.87
Opt-Accuracy	0.05	0.30	0.40	2	8.81	0.4711	**88.68**	9.41
Opt-Efficiency	0.95	0.45	0.05	3	7.92	1.1512	86.03	**1.07**
Balance-Top1	0.95	0.60	0.05	3	7.65	1.5959	87.03	1.97
Balance-Top2	0.80	0.60	0.05	3	7.64	1.6464	88.07	7.87
Balance-Top3	0.80	0.55	0.05	3	7.50	1.5908	88.17	6.68

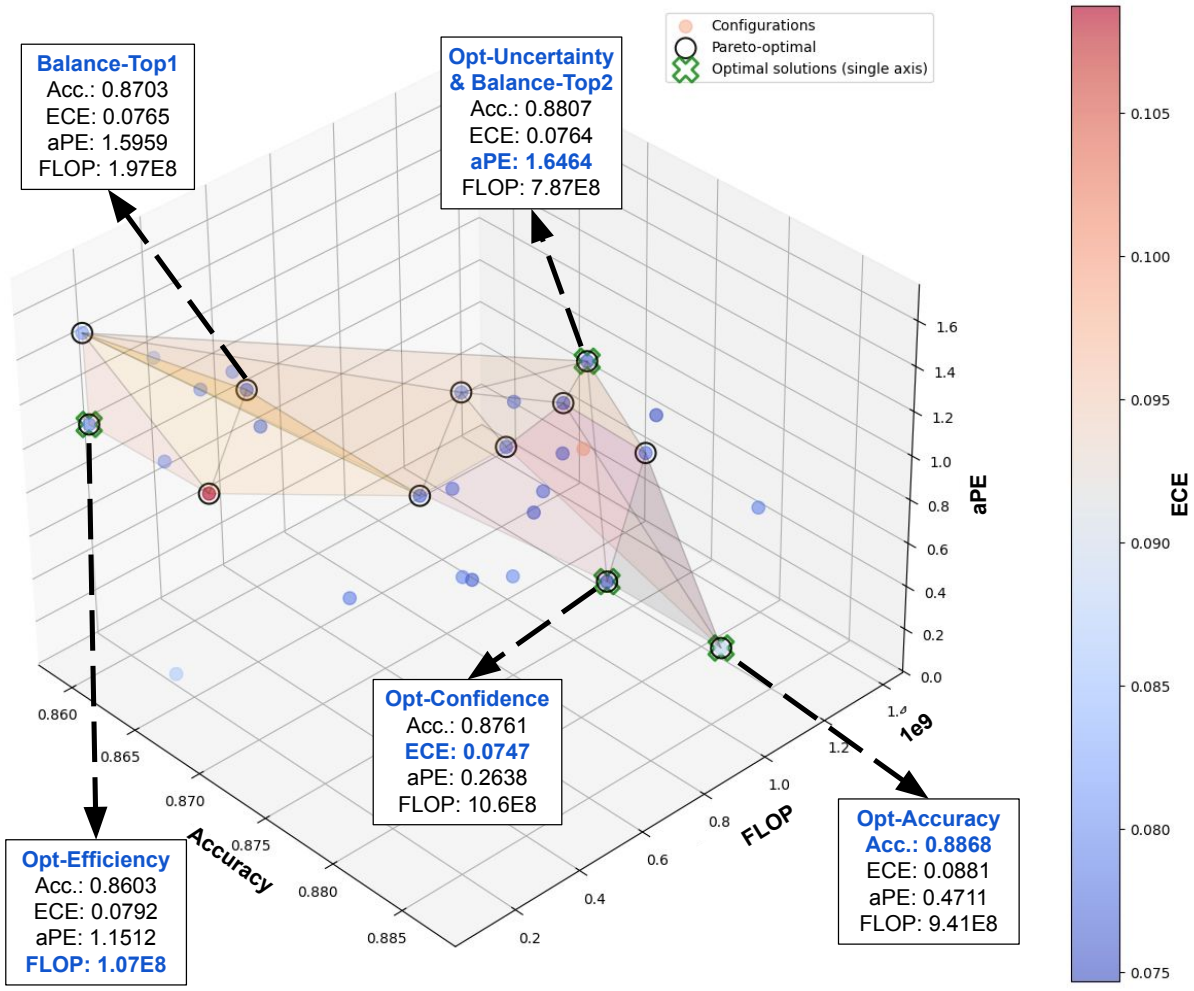

Figure 3: Pareto frontier analysis of Bayesian ResNet18 on CIFAR10, illustrating the trade-offs between accuracy, FLOP, ECE, and aPE. Each point represents a model candidate.

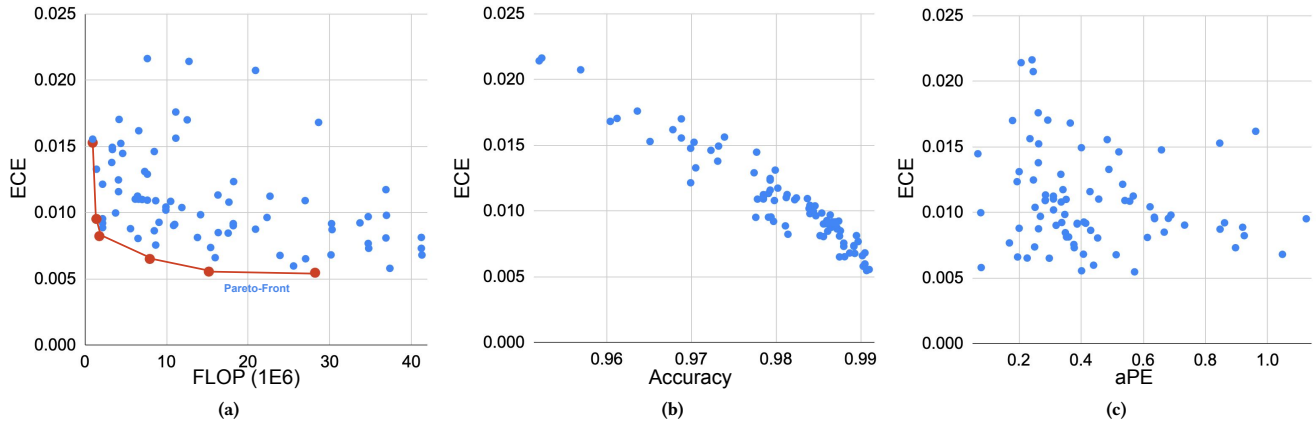

Figure 4: (a) Trade-off between model calibration (ECE) and computational efficiency (FLOP). (b) Trade-off between ECE and Accuracy. (c) Trade-off between ECE and aPE, taking into account confidence calibration and uncertainty estimation.

Opt-Uncertainty has the largest uncertainty estimation with aPE = 1.6464 nats, making it well-suited for scenarios where reliable uncertainty representation is essential.

Opt-Accuracy achieves the highest accuracy (88.68%), slightly outperforming other designs. However, similar to LeNet5, optimizing for accuracy alone does not always lead to the best uncertainty calibration, as ECE is higher, which is 8.81%, and aPE is relatively low, which is 0.4711 nats. This shows overconfident predictions.

Opt-Efficiency severely reduces computational overhead and achieves only 1.07E+8 FLOP. It is around 10 times smaller than Opt-Confidence. However, this efficiency gain comes at the cost of low accuracy and uncertainty estimation. Balance-Top1/2/3 explore Pareto-optimal trade-offs across accuracy, efficiency, and uncertainty metrics. Figure 3 further illustrates these design candidates in a Pareto frontier.

As with LeNet-5, the evaluation of ResNet-18 shows that no single design is universally optimal, which highlights the need to balance trade-offs between all these metrics. By utilizing Bayesian Optimization for DSE, our framework automates the search for Pareto-optimal configurations, potentially allowing models to be tailored based on deployment needs, from small devices such as edge platforms to larger computing systems.

5.4 ECE-Focused Analysis

As previously mentioned, Expected Calibration Error or ECE is a critical metric in Bayesian CNNs, which reflects how well a model's predicted confidence aligns with actual correctness. Unlike accuracy, which measures classification performance, and FLOP, which quantifies computational efficiency, ECE evaluates the trustworthiness of predictions. Figure 4 illustrates the complex relationships between ECE and FLOP, Accuracy as well as aPE, which shows on how different optimization strategies impact calibration quality.

In particular, Figure 4(a) presents the trade-off between ECE and FLOP, highlighting how reducing computational complexity impacts model calibration. It shows that models with lower FLOP tend to have higher ECE. This indicates that aggressive model compression can weaken calibration. This trend is also shown in the Opt-Efficiency designs discussed in the previous sections, where reducing FLOP leads to less reliable confidence estimates. Opt-Confidence designs, which achieve the lowest calibration error, retain significantly higher FLOP, demonstrating that well-calibrated Bayesian CNNs require sufficient computational capacity to maintain robust uncertainty estimation. We also have well-balanced designs which can effectively reduce FLOP while maintaining acceptable calibration quality. These results highlight the importance of compression strategies to preserve uncertainty quantification. Minimizing FLOP too aggressively can weaken calibration, so one needs to optimize the trade-off between efficiency and model trustworthiness in Bayesian CNNs.

Figure 4(b) shows the relationship between ECE and model accuracy, illustrating whether improving model calibration necessarily enhances classification performance. It shows that better-calibrated models tend to have higher classification performance. However the lowest ECE values do not always correspond to the highest accuracy, for instance, Opt-Confidence achieves superior calibration but does not yield the best accuracy.

Figure 4(c) investigates the relationship between ECE and aPE. It shows that larger aPE (uncertainty estimation) does not guarantee good calibration. This reinforces the need for fine-tuning of Monte Carlo dropout strategies.

These three figures clearly show that model compression degrades calibration, so careful compression strategies to retain model trustworthiness are necessary. Besides, accuracy and ECE do not always correlate, which means that high-performance models may still be poorly calibrated. Furthermore, aPE must be optimized alongside calibration to ensure that models remain reliable in real-world deployments.

5.5 Understanding the Impact of Hyperparameters on Model Trustworthiness

To evaluate how different optimization parameters influence Bayesian CNNs' efficiency, trustworthiness, and predictive performance, we present a correlation matrix (Figure 5) which shows

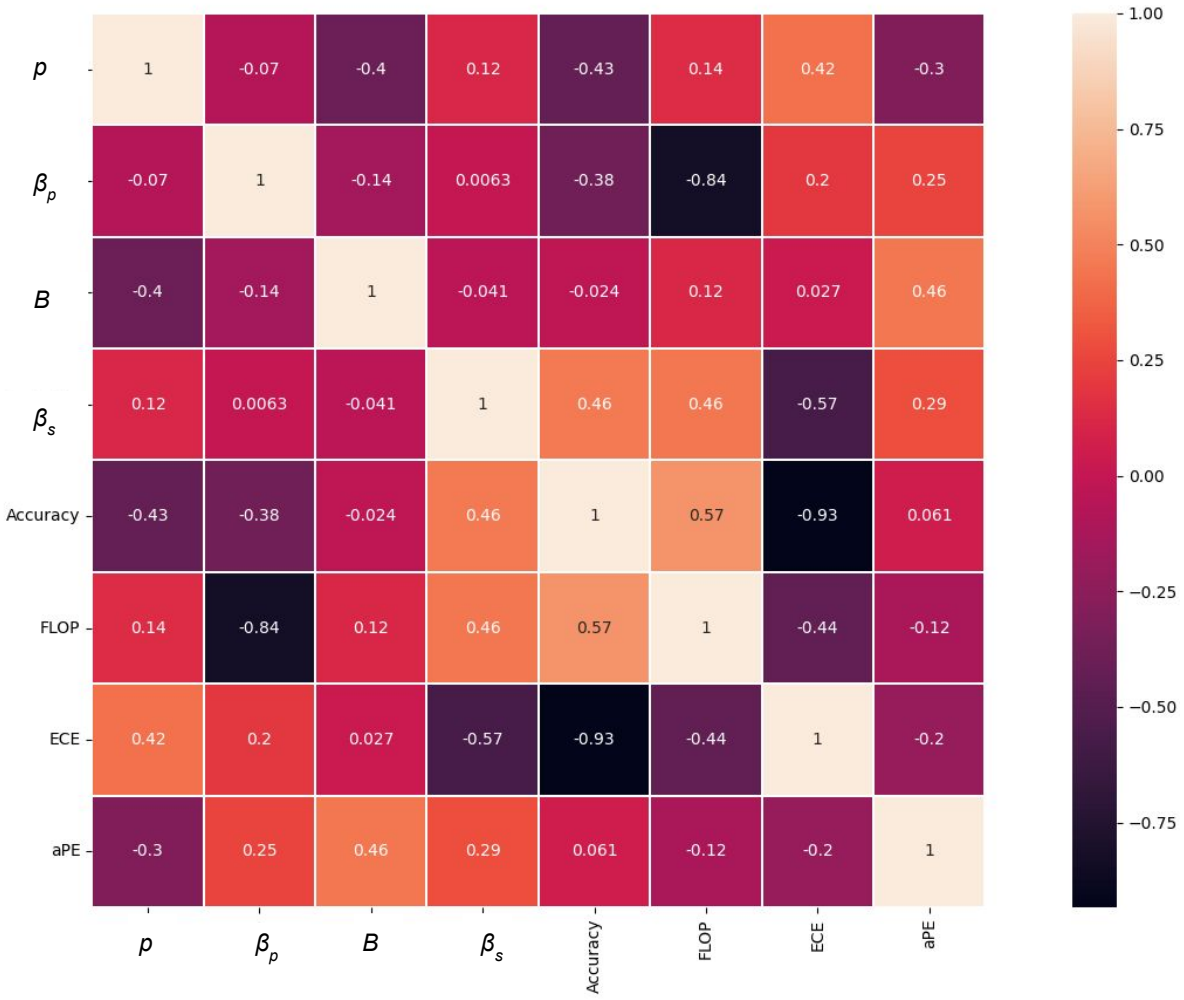

Figure 5: Correlation matrix illustrating the relationships between key hyperparameters including dropout rate (p), pruning rate (β_p), number of Bayesian layers (B), and scaling rate (β_s), and performance metrics including Accuracy, FLOP, Expected Calibration Error (ECE), and Average Predictive Entropy (aPE).

the relationships between four key hyperparameters and four performance metrics.

Regarding the ECE, the Figure 5 shows that dropout rate (p) has a positive correlation with ECE (0.42), indicating that increasing dropout rate introduces uncertainty but at the cost of degraded calibration. Pruning rate has a moderate positive correlation with ECE, suggesting that severe pruning leads to poor calibration. The scaling rate has a negative correlation with ECE (-0.57), showing that proper scaling improves calibration quality. Accuracy strongly correlates with lower ECE (-0.93), confirming that good-calibrated models tend to have high classification accuracy as shown in Figure 4(b). In summary, low ECE is best achieved through structured dropout and proper scaling, rather than aggressive pruning.

Regarding the aPE, the correlation matrix shows that the number of Bayesian layers (B) has a strong positive correlation with aPE (0.46), reinforcing that more Bayesian layers enhance uncertainty estimation. Pruning rate (β_p) has a moderate positive correlation with aPE (0.25), suggesting that proper pruning can help retain uncertainty representation. Dropout rate has a mild negative correlation with aPE (-0.30), indicating that excessive dropout may suppress useful uncertainty information. And the scaling rate has a mild positive correlation with aPE (0.29), showing that proper scaling improves uncertainty quantification. In summary, the number of Bayesian layers is the primary driver of aPE, while pruning and scaling also influence uncertainty estimation.

5.6 Comparison with Other FPGA designs

While the previous subsections covered a platform-agnostic evaluation, this section focuses on our framework's ability to support efficient FPGA design. In particular, we select two optimized BayesLeNet5 models automatically identified by our approach: (1) Calibration Optimized (**Opt-Confidence**) and (2) Efficiency Optimized (**Opt-Efficiency**). We adopt the HLS4ML workflow from [7] with

Table 5: Performance comparison of our final Bayesian LeNet5 designs with CPU, GPU and other FPGA-based implementations. The Opt-Efficiency design has been partially unrolled to improve latency due to its compact size.

	CPU [7]	GPU [7]	ASPLOS 2018 [3]	DATE 2020 [2]	TPDS 2022 [8]	DAC 2023 [7]	This work Opt-Confid.	This work Opt-Effici.
Platform	Intel i9-9900K	NVIDIA RTX 2080	Altera Cyclone V	Zynq XC7Z020	Arria 10 GX1150	Kintex XCKU115	Kintex XCKU115	Kintex XCKU115
Freq. (MHz)	3600	1545	213	200	220	181	181	181
Technology	14 nm	12 nm	28 nm	28 nm	20 nm	20 nm	20 nm	20 nm
Accuracy	-	-	-	-	99.3%	98.9%	99.07%	96.51%
Power (W)	205	236	6.11	2.76	43.6	4.6	4.48	2.72
Latency (ms)	1.26	0.57	5.5	4.5	0.32	0.89	0.855	0.116
Energy Effi. (mJ/Image)	258	134	33	12	14	4.1	3.8	0.32

MCD supported. With a custom HLS4ML configuration, as described in Section 5.1, we generate two corresponding FPGA designs and we compare them against CPU, GPU, and other FPGA-based designs, as presented in Table 5. The comparison uses Bayes-LeNet5 on the MNIST dataset since they are the most common network and dataset across different work [2, 3, 7, 8]. The results of our Calibration Optimized (Opt-Confidence) and Efficiency Optimized (Opt-Efficiency) designs demonstrate significant trade-offs between accuracy, latency, power, and energy efficiency, as discussed next.

Our design optimized for calibration utilizes the same parallelism configurations as [7]. It has a strong model accuracy of 99.07%, outperforming the [7] (98.9%). Besides, it consumes 4.48W, making it more power-efficient than other high-accuracy designs, such as [8] (43.6W). Its latency is 0.855ms, slightly better than the state-of-the-art [7], and it achieves an energy efficiency of 3.8 mJ/Image, better than previous FPGA-based implementations.

The Opt-Efficiency design, focused on FLOP reduction, results in a small design that allows more unrolling to improve latency with a cost of hardware resources. It achieves not only lowest power consumption, but also the lowest latency (0.116ms), making it the fastest and most efficient design among all evaluated implementations. However, this efficiency comes at a slight accuracy drop (96.5%), compared to Opt-Confidence (99.07%) and previous designs (99.3%). Nevertheless, this design achieves the highest energy efficiency, making it an ideal solution for ultra-low-power and real-time embedded applications. Compared to a state-of-the-art implementation [7], our Opt-Efficiency design is 7.67 times faster and 12.8 times more energy-efficient. Compared with a CPU implementation [7], our FPGA designs are 1.5~10.9 times faster. In terms of the energy efficiency, our designs are 67.9~806 times higher than the CPU implementation. When compared to a GPU implementation [7], our FPGA designs achieve 35.3~419 times higher power efficiency than the GPU implementation.

In addition, our approach is automatic. In contrast to their manual implementations with expert optimizations, it significantly enhances design productivity and scalability. These evaluation results confirm the effectiveness of the proposed framework which enables FPGA-based designs that are faster, more power-efficient, and automatically optimized. This pushes the boundaries of real-time energy-efficient Bayesian CNN accelerators.

6 Conclusion

Deploying deep learning models in real-world applications requires a careful balance among a number of factors, including computational efficiency, predictive accuracy, and trustworthiness. This paper introduces automated design space exploration driven by Bayesian Optimization to improve BayesCNNs for efficient and reliable deployment.

Our approach dynamically tunes key hyperparameters to identify Pareto-optimal trade-offs across multiple performance metrics. Through an extensive evaluation, we demonstrate that our framework enables optimizing trade-offs between key metrics. This ensures that models remain lightweight, accurate, and well-calibrated. Our results show that compared to state-of-the-art FPGA implementations, our optimized designs achieve up to 7.67× faster inference and 12.8× higher energy efficiency, while maintaining high classification performance and robust uncertainty estimation.

Our future work includes framework extension to cover larger-scale deep learning models. In addition, we will integrate advanced model compression techniques, including Supermask-based pruning [19, 20]. Moreover, we plan to explore optimization approaches tailored for transformer architectures to further optimize their deployment on various hardware platforms.

Acknowledgments

The support of the United Kingdom EPSRC (grant number UKRI256, EP/V028251/1, EP/N031768/1, EP/S030069/1, and EP/X036006/1), Intel, and AMD is gratefully acknowledged.

References

[1] Mohamed S Abdelfattah, Łukasz Dudziak, Thomas Chau, Royson Lee, Hyeji Kim, and Nicholas D Lane. 2020. Best of both worlds: Automl codesign of a cnn and its hardware accelerator. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[2] Hiromitsu Awano and Masanori Hashimoto. 2020. BYNQNet: Bayesian neural network with quadratic activations for sampling-free uncertainty estimation on FPGA. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1402–1407.

[3] Ruizhe Cai, Ao Ren, Ning Liu, Caiwen Ding, Luhao Wang, Xuehai Qian, Massoud Pedram, and Yanzhi Wang. 2018. VIBNN: Hardware acceleration of Bayesian neural networks. *ACM SIGPLAN Notices* 53, 2 (2018), 476–488.

[4] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. 2016. R-fcn: Object detection via region-based fully convolutional networks. *Advances in neural information processing systems* 29 (2016).

[5] Yuhao Ding, Jiajun Wu, Yizhao Gao, Maolin Wang, and Hayden Kwok-Hay So. 2023. Model-Platform Optimized Deep Neural Network Accelerator Generation through Mixed-Integer Geometric Programming. In *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 83–93.

[6] Zhen Dong, Yizhao Gao, Qijing Huang, John Wawrzynek, Hayden KH So, and Kurt Keutzer. 2021. Hao: Hardware-aware neural architecture optimization for efficient inference. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 50–59.

[7] Hongxiang Fan, Mark Chen, Liam Castelli, Zhiqiang Que, He Li, Kenneth Long, and Wayne Luk. 2023. When Monte-Carlo Dropout meets multi-exit: Optimizing Bayesian neural networks on FPGA. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 1–6.

[8] Hongxiang Fan, Martin Ferianc, Zhiqiang Que, Xinyu Niu, Miguel Rodrigues, and Wayne Luk. 2022. Accelerating Bayesian neural networks via algorithmic and hardware optimizations. *IEEE Transactions on Parallel and Distributed Systems* 33, 12 (2022), 3387–3399.

[9] Alessandro Frigerio, Bart Vermeulen, and Kees GW Goossens. 2021. Automotive architecture topologies: Analysis for safety-critical autonomous vehicle applications. *IEEE Access* 9 (2021), 62837–62846.

[10] Yoav Goldberg. 2016. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research* (2016).

[11] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International conference on machine learning*. PMLR, 1321–1330.

[12] Cong Hao, Jordan Dotzel, Jinjun Xiong, Luca Benini, Zhiru Zhang, and Deming Chen. 2021. Enabling design methodologies and future trends for edge AI: specialization and codesign. *IEEE Design & Test* 38, 4 (2021), 7–26.

[13] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. 2018. Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and Imagenet?. In *Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 6546–6555.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[15] Weiwen Jiang, Lei Yang, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Shouzhen Gu, Sakyasingha Dasgupta, Yiyu Shi, and Jingtong Hu. 2020. Hardware/software co-exploration of neural architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39, 12 (2020), 4805–4815.

[16] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[18] Jonas Ney, Dominik Loroch, Vladimir Rybalkin, Nico Weber, Jens Krüger, and Norbert Wehn. 2021. HALF: Holistic auto machine learning for FPGAs. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 363–368.

[19] Yasuyuki Okoshi, Ángel López García-Arias, Kazutoshi Hirose, Kota Ando, Kazushi Kawamura, Thiem Van Chu, Masato Motomura, and Jaehoon Yu. 2022. Multicoated Supermasks Enhance Hidden Networks.. In *ICML*. 17045–17055.

[20] Hikari Otsuka, Daiki Chijiwa, Ángel López García-Arias, Yasuyuki Okoshi, Kazushi Kawamura, Thiem Van Chu, Daichi Fujiki, Susumu Takeuchi, and Masato Motomura. [n. d.]. Partially Frozen Random Networks Contain Compact Strong Lottery Tickets. In *Workshop on Machine Learning and Compression, NeurIPS 2024*.

[21] Zhiqiang Que, Jose GF Coutinho, Ce Guo, Hongxiang Fan, and Wayne Luk. 2025. MetaML-Pro: Cross-Stage Design Flow Automation for Efficient Deep Learning Acceleration. *arXiv preprint arXiv:2502.05850* (2025).

[22] Markus Rognlien, Zhiqiang Que, Jose GF Coutinho, and Wayne Luk. 2022. Hardware-aware optimizations for deep learning inference on edge devices. In *International Symposium on Applied Reconfigurable Computing*. Springer, 118–133.

[23] Divya Saxena and Vaskar Raychoudhury. 2017. Design and verification of an NDN-based safety-critical application: A case study with smart healthcare. *ieee transactions on systems, man, and cybernetics: systems* 49, 5 (2017), 991–1005.

[24] Shikhar Tuli, Chia-Hao Li, Ritvik Sharma, and Niraj K Jha. 2023. CODEBench: A neural architecture and hardware accelerator co-design framework. *ACM Transactions on Embedded Computing Systems* 22, 3 (2023), 1–30.

[25] Jessica Vandebon, Jose Coutinho, and Wayne Luk. 2022. Meta-Programming Design-Flow Patterns for Automating Reusable Optimisations. In *Proceedings of the 12th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies* (Tsukuba, Japan) *(HEART '22)*. Association for Computing Machinery, New York, NY, USA, 42–50. doi:10.1145/3535044.3535050

[26] Qiyu Wan and Xin Fu. 2020. Fast-BCNN: Massive neuron skipping in Bayesian convolutional neural networks. In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 229–240.

[27] Fangxin Wang, Yuqing Liu, Kay Liu, Yibo Wang, Sourav Medya, and Philip S Yu. 2024. Uncertainty in graph neural networks: A survey. *arXiv preprint arXiv:2403.07185* (2024).

[28] Yifan Yang, Qijing Huang, Bichen Wu, Tianjun Zhang, Liang Ma, Giulio Gambardella, Michaela Blott, Luciano Lavagno, Kees Vissers, John Wawrzynek, et al. 2019. Synetgy: Algorithm-hardware co-design for convnet accelerators on embedded fpgas. In *Proceedings of the 2019 ACM/SIGDA international symposium on field-programmable gate arrays*. 23–32.