Accelerating Complex-valued Uncertainty Estimation via Sparsity Exploration and Exploitation

***Abstract*—Magnetic resonance fingerprinting (MRF) estimates multiple tissue parameters from complex-valued signals and is increasingly explored for quantitative imaging in oncology and radiotherapy. Complex-valued neural networks (CVNNs) process these signals to predict tissue parameters while preserving the coupling between their real and imaginary parts, but their predictions are deterministic and cannot flag unreliable outputs. Bayesian complex-valued neural networks (BayesCVNN) provide uncertainty estimates at the cost of repeated stochastic inference. This work presents an algorithm-hardware co-design framework that exploits two complementary kinds of sparsity in such models: Bayesian-layer sparsity, from making only a subset of layers Bayesian, and activation sparsity, from the dropout operations. A sparsity-aware exploration framework jointly searches Bayesian-layer selection and per-part dropout rates. A customized FPGA accelerator employs column-wise sparse matrix-vector multiplication to turn the explored sparsity into latency reduction, with runtime-adaptable scheduling to accommodate hybrid layer types. Experiments conducted on the AMD VCK190 demonstrate the design achieves 1.64× to 5.77× speedup and 14.6× to 49.9× lower energy costs compared with GeForce RTX 3090 Ti baselines, while using 11.3 percent of its power.**

***Index Terms*—Bayesian complex-valued neural networks, Monte Carlo dropout, uncertainty estimation, FPGA acceleration, magnetic resonance fingerprinting, sparsity exploitation**

I. Introduction

Magnetic resonance fingerprinting (MRF) is a quantitative technique that enables the estimation of multiple tissue-specific parameters [1], [2]. This capability is particularly important in oncology and radiotherapy, where quantitative tissue characterization can support tumor assessment and treatment-response monitoring [3]–[5]. MRF signals are inherently complex-valued, with the real and imaginary parts, or equivalently the magnitude and phase, jointly capturing information relevant to tissue properties [6], [7]. Treating such data as purely real-valued features weakens their intrinsic structure, leading to unavoidable accuracy loss. To mitigate this drawback, complex-valued neural networks (CVNNs) [8] have been proposed specifically to handle tasks in complex-valued domains, and they process MRF signals to predict tissue parameter estimates. By keeping the real and imaginary parts coupled, they preserve the phase information that purely real-valued models discard, avoiding notable accuracy loss.

Although CVNNs have achieved promising performance on complex-valued tasks, their predictions are typically deterministic and cannot quantify prediction confidence and reliability, which are particularly important when MRF supports quantitative tissue characterization in oncology and radiotherapy. In practical scenarios, medical measurements may be corrupted by noise [9]–[11], yet conventional deterministic models give no measure of confidence and can stay overconfident even when the predictions are degraded. The uncertainty estimation assesses the reliability and trustworthiness of model-assisted results, thereby minimizing harmful risks [11]–[13]. To achieve this, Bayesian neural networks (BayesNNs) [9], [14] provide a principled probabilistic framework to perform uncertainty estimates of model predictions.

Among existing Bayesian approximations, Monte-Carlo dropout is particularly attractive due to its high efficiency for hardware deployment [15]. However, uncertainty quantification in BayesNNs requires multiple stochastic sampling, incurring substantial computation and memory overheads. To reduce this cost, several BayesNN hardware accelerators have been explored [16]–[22]. Most of these designs primarily target real-valued neural networks and cannot be directly extended to complex-valued domains, due to the distinct arithmetic rules between real-valued and complex-valued domains.

A recent pioneering work introduces dropout-based Bayesian complex-valued neural networks (BayesCVNNs), enabling uncertainty estimation for CVNNs and demonstrating a design flow for identifying model configurations and generating accelerators [23]. This prior work establishes the feasibility of constructing and accelerating BayesCVNNs, suggesting their potential for MRF applications. It further shows that layer-mixing and part-mixing configurations can outperform uniform configurations. Nevertheless, dropout operations introduce sparse computation in layer operations, and the sparsity pattern of each layer is determined by the dropout rate. But [23] adopts a fixed dropout rate and does not explicitly consider the incurred sparsity for identifying model configurations. Moreover, the proposed hardware design lacks tailored support for sparsity exploitation. Therefore, considerable optimization potential remains underexplored.

To bridge this research gap, several issues need to be addressed. On the algorithmic side, the exploration space considering the sparsity is huge. Dropout introduces activation sparsity, but in the complex domain the real and imaginary parts may favor non-uniform dropout configurations. Moreover, prior works [22], [24], [25] suggest that sparsity can also exist in layer settings, where only a subset of layers needs to be Bayesian. It is therefore challenging to determine model configurations that satisfy diverse application requirements and scenario constraints by manual efforts in such huge design space. On the hardware side, dropout requires random dropping operations, introducing unstructured sparsity patterns that can vary across real and imaginary parts, and across layers. Developing tailored hardware architecture to exploit

the varied sparsity is difficult.

To address these issues, we propose an algorithm-hardware co-design framework for **sparse exploration** and **sparse exploitation** in dropout-based BayesCVNNs. We identify two major types of sparsity: activation sparsity, which is determined by dropout rates, and Bayesian-layer sparsity, which is determined by the number of Bayesian layers. Different sparsity settings lead to varied amounts of computation and also affect algorithmic performance, including accuracy and uncertainty quality. To identify an effective model configuration for the target scenario, this work introduces an automatic sparsity-aware exploration framework for efficient search.

A customized FPGA accelerator, equipped with efficient compute engines and runtime-adaptable scheduling, is then proposed to exploit the explored sparsity. At the kernel level, a column-wise matrix-vector multiplication (MVM) engine is adopted to skip redundant operations and translate sparsity into actual latency reduction. The compute engine is reused across the sub-operations of complex-valued layers, significantly mitigating resource underutilization caused by imbalanced workloads between the real and imaginary parts. At the scheduling level, an optimized pipeline hides the weight-loading cost of complex-valued computation and the generation overhead of dropout masks, while enabling compatible execution of both standard layers with dense computation, and Bayesian layers with sparse computation involving multiple sampling.

The main contributions of this work are summarized as follows:

- Propose an automatic sparsity-aware exploration framework that searches model configurations, considering Bayesian-layer sparsity and activation sparsity.
- Develop a customized FPGA accelerator with a column-wise computation engine for complex-valued layer operations, with the run-time adaptable scheduling to accommodate both standard layers and Bayesian layers.
- Conduct comprehensive experiments to demonstrate the effectiveness of sparsity explorations and the efficiency of sparsity exploitation.

II. Related Work

A. Magnetic resonance fingerprinting

MRF is a quantitative imaging framework that estimates multiple tissue parameters from a single acquisition [1], [2]. In MRF, each tissue produces a characteristic temporal signal trajectory, often referred to as a fingerprint, which reflects its underlying physical properties. These quantitative results provide a more objective description of tissue properties and may improve tissue characterization and disease assessment [3]. MR fingerprints are inherently complex-valued signals, containing both magnitude and phase information [6], [7]. Treating complex data simply as magnitude images or as two independent real-valued channels may weaken the physical structure of the signal [6], [7]. Therefore, complex-valued neural networks have been introduced as a more natural method for processing MR fingerprints [6]–[8].

B. Complex-Valued Neural Networks

CVNNs extend real-valued neural networks by preserving and transforming real and imaginary parts throughout the model. Compared with representing complex data as two independent real-valued channels, CVNNs can better maintain the algebraic structure of complex-valued signals and the coupling between magnitude and phase. Prior work has developed complex-valued convolutions, nonlinearities, normalization layers, and recurrent structures for signal-processing and imaging applications [8].

For ease of implementation, complex-valued layer operations are usually performed in rectangular form. Let the input activation be $A = A_r + iA_i$ and the weight be $W = W_r + iW_i$, where A_r and W_r denote the real parts and A_i and W_i denote the imaginary parts. The real and imaginary outputs, denoted as C_R and C_I, are computed as $C_r = A_r W_r - A_i W_i$, $C_i = A_r W_i + A_i W_r$. Therefore, one complex-valued operation is decomposed into four real-valued sub-operations, followed by addition and subtraction. This decomposition allows CVNNs to be implemented using standard real-valued arithmetic units, but it also increases computation, memory traffic, and scheduling complexity compared with real-valued networks. These overheads become more challenging when Monte Carlo dropout introduces sparse and potentially imbalanced activations, where the real and imaginary parts can take different dropout rates and hence leave different numbers of active values.

C. Bayesian Neural Networks

Bayesian neural networks estimate predictive uncertainty by treating model parameters or activations probabilistically. Monte Carlo dropout offers a practical approximation by enabling dropout during inference and aggregating multiple stochastic forward passes [15]. This approach is attractive because it can be integrated into existing models. Quantifying the uncertainty requires repeated stochastic forward pass, which increases computation and memory traffic. Therefore, hardware accelerators have been proposed [16]–[20], [20]. These designs demonstrate the potential of hardware acceleration for uncertainty estimation, but they target real-valued models and do not address the arithmetic and scheduling challenges introduced by complex-valued computation. The prior BayesCVNN work [23] establishes the feasibility of dropout-based uncertainty estimation for complex-valued neural networks and considers heterogeneous real/imaginary dropout configurations; however, its exploration is relatively coarse-grained, uses fixed dropout rates, and does not explicitly exploit dropout-induced activation sparsity in hardware.

In contrast, this work targets efficient BayesCVNN inference by jointly exploring Bayesian-layer configurations and real/imaginary dropout rates. The resulting hybrid sparsity patterns are then mapped onto a runtime-adaptable FPGA accelerator with sparse column-wise execution and mode-aware scheduling.

III. Sparsity-aware exploration framework

An overview of the proposed sparsity-aware exploration framework is illustrated in Fig. 1, consisting of three phases. Targeting the specific scenarios, the input contains a standard CVNN model and dropout-rate ranges, along with datasets and application-specific constraints. Given the inputs, the framework aims at exploring BayesCVNN model configurations delivering high performance, considering different types of sparsity. The outputs of the framework produces the optimized settings, including the number of Bayesian layers, dropout rates of real and imaginary parts for each Bayesian layer. Although this work targets MRF, the framework is general and can be adapted to wide applications, such as robotics and autonomous driving. Note that the search methodology is similar to [21], [23], our novelty resides in explicitly considers Bayesian-layer sparsity and activation sparsity in Phase 1 and Phase 3.

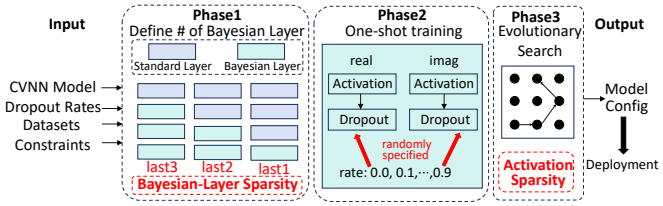

Fig. 1. Sparsity-aware exploration framework.

A. Bayesian-Layer Sparsity

Phase 1 specifies the last N layers to be converted into Bayesian layers, where N is provided by the design constraint. For a given model architecture and N, multiple variants can be constructed by setting the number of Bayesian layers to $N-1$, ... 2, 1 according to practical demands. For instance, $N = 3$ in Fig. 1, there are totally three variants. This is inspired by prior observations [22], [24], [25] that making only part of the network Bayesian may even improve algorithmic performance while avoiding the cost of making the full network Bayesian, which has been demonstrated in real-valued domains [22]. Under this configuration, only a subset of layers performs multiple sampling during Bayesian inference, while the remaining layers stay as standard deterministic layers. We refer to this as *Bayesian-layer sparsity*, which determines the number of layers requiring dropout operations and sampling.

B. One-Shot Training

After the Bayesian layers are specified, Phase 2 performs one-shot training for all model variants similar to prior paradigms [21], [26]. The purpose of this phase is to train a supernet once under stochastic dropout, so that substantial dropout rate configurations can be evaluated without retraining the model for each configuration. During training, dropout is applied to the activations of the selected Bayesian layers, and the real and imaginary parts are followed by independent dropout settings. One-shot training makes fine-grained dropout search practical by decoupling weight learning from configuration evaluation.

C. Dropout-Rate Search

After one-shot training, Phase 3 searches dropout rate configurations for the selected Bayesian layers in each model variant. The dropout rates applied to real and imaginary activations affect not only accuracy and uncertainty estimation, but also the activation sparsity that can be exploited by hardware.

For each Bayesian layer, the framework searches two dropout rates, one for the real activation and one for the imaginary activation. Both rates are selected from 0.0 to 0.9. Thus, each Bayesian layer has $10 \times 10 = 100$ possible real or imaginary dropout-rate choices. For n ($n = N, N-1, ..., 2, 1$) Bayesian layers, the total search space is $100^n = 10^{2n}$. When the number of Bayesian layers becomes large, exhaustive enumeration becomes expensive. This framework uses an evolutionary algorithm to search the configuration space. During the search, a candidate is encoded as a chromosome of $2n$ integers, where each pair represents the real and imaginary dropout rates of one Bayesian layer. The EA initializes a population of candidate chromosomes, evaluates and selects high-quality candidates, and then applies crossover and mutation to generate new candidates. Mutation changes one or more dropout rates, while crossover exchanges layer-wise real and imaginary dropout settings between two parent candidates.

IV. Hardware Accelerator

A. Architecture Overview

Fig. 2 illustrates the architecture of the proposed FPGA accelerator. The processing system (PS) feeds inputs, receives the processed results, and controls the launch and termination of the accelerator design on the programmable logic (PL) side. The core computational modules are complex-valued processing engines. Model weights are stored in off-chip memory and loaded into on-chip buffers for each layer operation. The whole BayesCVNN is executed layer by layer using the same processing engines. Since the proposed framework explores the layer sparsity, the produced model configuration may contain hybrid layers, including standard layers and Bayesian layers. Standard layers do not require applying dropout masks on the inputs, thus involving deterministic and dense operations. Bayesian layers, in contrast, activate dropout during inference, leading to sparse execution patterns. The accelerator provides runtime adaptability to support tailored execution modes, as elaborated in Section V.

B. Column-Wise Matrix-Vector Multiplication

For each input, layer operations mainly involve matrix-vector multiplication. A conventional implementation adopts a row-wise mapping scheme, which multiplies the input vector with one matrix row to yield one output element at a time, as illustrated by the upper-left example in Fig. 3. In Bayesian layers, the elements in activation vectors are dropped randomly. To skip redundant computation, the row-wise engine first identifies the valid activations according to the dropout mask, then fetches the corresponding valid weights from the current matrix row, and finally accumulates these products to

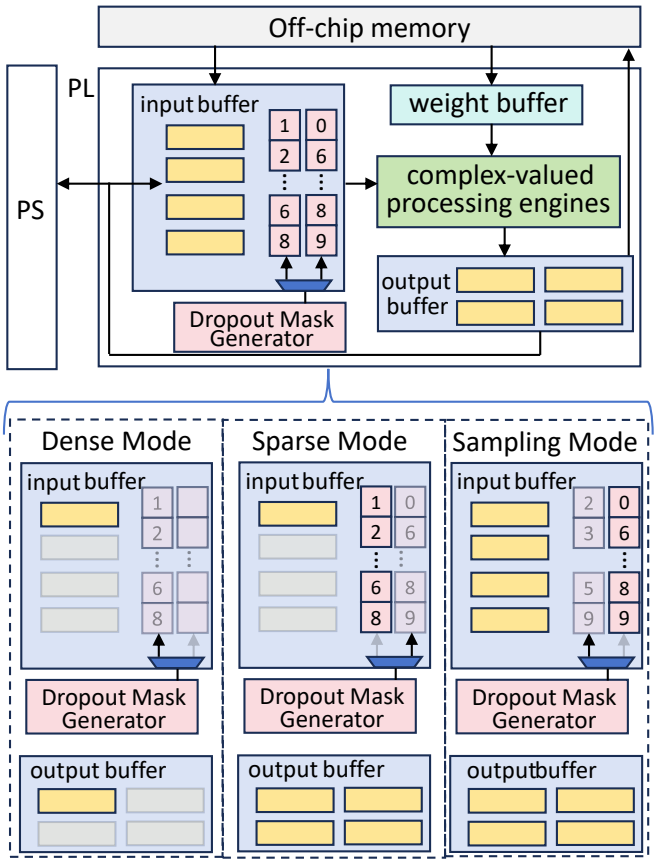

Fig. 2. Overview of the accelerator design with runtime adaptable modes.

produce one output element. This process is repeated for all matrix rows to produce resultant vectors.

Although row-wise execution can skip some arithmetic operations, it does not guarantee translating activation sparsity into latency reduction under a fixed parallelism. The latency of the row-wise execution is determined by the number of parallel execution rounds required for each output row. If the number of valid activations after dropout still falls into the same number of rounds, the latency remains unchanged even though fewer products are computed. As shown in Fig. 3, the dense case and the case with a 0.3 dropout rate require the same number of row-wise execution rounds; similar observations also hold for the cases with 0.6 and 0.8 dropout rates. This is due to the fact that the remaining valid elements are gathered from non-contiguous positions and mapped to the available parallel lanes. Therefore, since dropout rates vary across layers and dropout masks introduce unstructured sparsity, the row-wise execution scheme cannot guarantee efficient sparsity exploitation for Bayesian layer operations. Moreover, insufficient concurrency after dropout also leads to underutilization of the computational resources.

To address this issue, the proposed engine employs column-wise MVM inspired by [27]. Instead of computing one output row at a time, the engine takes one valid activation element and multiplies it with the corresponding valid weight column. The partial products are accumulated into the output vector, and this process is repeated until all valid activation columns are processed. If an activation is dropped, the entire column update associated with that activation is redundant, and thus skipped. As a result, with column-wise MVM, the hardware parallelism determines the processing latency of each valid column, while the dropout mask determines the number of valid columns to be processed. Reducing the number of valid activations directly reduces the number of column update operations. In the same example shown in Fig. 3, the actual latency reductions are achieved compared with the row-wise computational scheme. Consequently, unstructured sparsity introduced by dropout translates into latency reduction more reliably across different dropout rates, exhibiting effective sparsity exploitation.

C. Shared Complex Sub-Operation Engine

As introduced in Section II, each complex-valued layer operation consists of four sub-operations followed by addition and subtraction on the intermediate results. Dropout further complicates this computation. Since the real and imaginary activations undergo separate dropout masks and therefore exhibit distinct sparsity patterns, the sub-operations involving real and imaginary activations can incur imbalanced workloads across the complex-valued layer operation.

A naive implementation for complex-valued layer operations allocates four separate compute engines, one for each sub-operation for concurrent execution in Stage 1. In Stage 2, the intermediate results are fed to the addition and subtraction logic. However, this design scheme suffers from inefficiency under distinct sparsity patterns, as shown in Fig. 3. Distinct sparsity patterns lead to imbalanced workloads and inconsistent latency among the sub-operations. Since the subsequent addition and subtraction stage must wait until all four sub-operations have completed, engines that finish earlier remain idle while waiting for the unfinished sub-operations, which causes resource underutilization and weakens the benefit of latency reductions introduced by the sparsity.

To address this issue, the proposed design employs a single shared compute engine to perform complex-valued layer operations. Rather than statically partitioning the available compute resources into four dedicated engines, the design consolidates these resources into a unified engine. The four sub-operations are therefore executed in a scheduled sequential order, and their intermediate results are accumulated through the corresponding addition and subtraction datapath. Since no separate engines complete asynchronously, the design avoids idle waiting caused by workload imbalance and maintains more stable resource utilization under distinct real and imaginary sparse patterns.

D. Scheduling within the Shared Engine

Regarding the detailed execution order of the four sub-operations, since model weights reside in off-chip memory and are loaded into on-chip buffers during layer execution, minimizing weight traffic is critical for lowering latency and improving energy efficiency in complex-valued layer

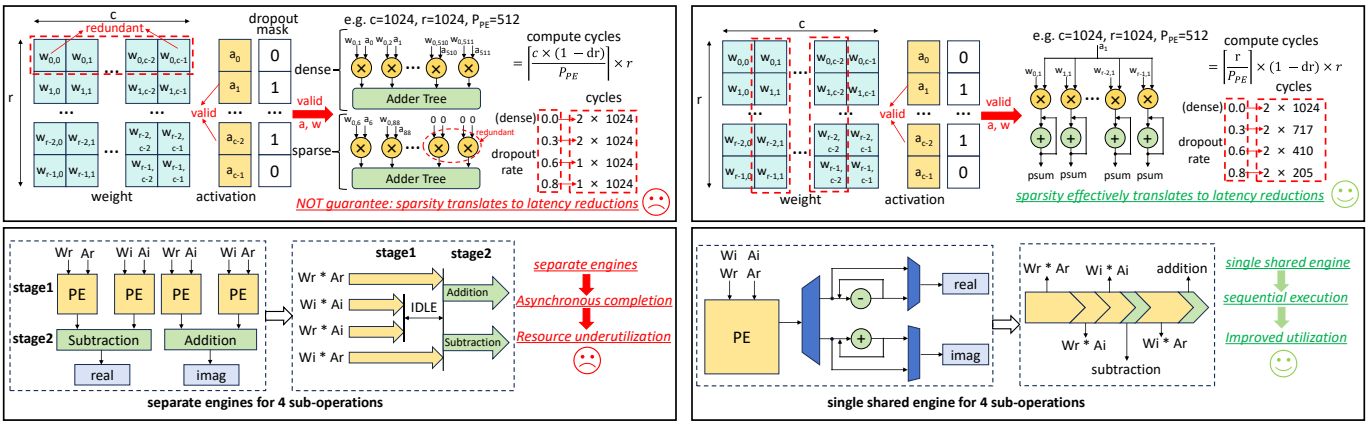

Fig. 3. Comparison between row-wise and column-wise sparse MVM execution.

operations. To minimize memory access cost, the two sub-operations, $A_r{*}W_r$, $A_i{*}W_r$, are executed first, followed by the remaining two sub-operations, $A_r * W_i$, $A_i * W_i$, which takes advantage of the data reusability of the real and imaginary parts of model weights.

In addition, the addition and subtraction required by complex arithmetic can be integrated by controlling accumulator updates through lightweight control logic, rather than being implemented as a separate second stage. As shown in Fig. 3, the products from A_rW_r and A_iW_r are written directly to the corresponding real and imaginary result buffers as the initial partial results. The remaining products are then combined through the addition and subtract datapath. A_rW_i is added to the imaginary result, while A_iW_i is subtracted from the real result. For subsequent partial results, the same addition and subtraction datapath accumulates them into the existing output buffers. In this way, the proposed scheme reduces weight traffic and enables streaming complex-valued layer execution with lightweight control logic.

V. Scheduling with Runtime-Adaptable Modes

The sparsity-aware framework in Section III explores the layer sparsity, producing BayesCVNN configurations that may contain both standard layers and Bayesian layers. Performing model prediction with uncertainty estimation requires multiple stochastic sampling. Each sampling can produce different dropout masks, resulting in varied sparse execution patterns. Repeatedly executing the whole network incurs expensive costs. This design exploits the layer sparsity to skip redundant computation. Since the intermediate results of the deterministic layers during the forward pass remain unchanged across sampling, the associated computation is performed only once for each input. Moreover, to support the hybrid layers, the proposed scheduler can switch the datapath according to the layer type and sampling state, operating in different modes with runtime adaptability. As shown in Fig. 2, the accelerator supports the dense mode for deterministic computation, and the sparse and sampling modes for sparse computation with multiple sampling. For each layer, the scheduler uses layer-level configuration metadata, including layer dimensions, input and weight addresses, dropout rates, and the number of samples. Based on this metadata, the scheduler configures the operating mode during runtime.

A. Runtime-Adaptable Execution Modes

1) Dense Mode: For standard layers, the accelerator uses the dense mode, disabling the sparse-control path and bypassing the dropout operations. Standard layers are executed once without multiple sampling, and their outputs are written to the on-chip input buffer without being written back to off-chip memory. Furthermore, the dense mode is also applied to the first Bayesian layer. As dropout operations are performed on the output activations of the first Bayesian layer, the computation is still dense, and the results are reused as the input activations in the second Bayesian layer, where dropout masks are then applied to these buffered activations. Therefore, all computation under dense mode is performed only once per input, significantly eliminating redundant computation and reducing the overall latency.

2) Sparse Mode: After the standard layers and the first Bayesian layer have been processed, the scheduler enables the sparse mode for the second Bayesian layers by activating dropout masks. In the sparse mode, the input activations are cached in a single input buffer and reused across different sampling. For each sampling, dropout masks are generated according to the given real and imaginary dropout rates of the current layer. Since the real and imaginary parts of activations use different dropout masks, sparse mode maintains separate valid-index streams for both parts, which are consumed by the shared complex processing engines described in the previous section. Different from the dense mode, the sparse mode produces multiple stochastic outputs for the same cached input activation. Therefore, the design uses multiple output buffers to temporarily store a batch of sampling results, which are generated sequentially, and then transferred back to off-chip memory in batches.

3) Sampling Mode: For the remaining Bayesian layers, the design is configured as Sampling mode. As the outputs from the second Bayesian layers are stored in off-chip memory, the input samples are loaded from the off-chip memory. Each input sample undergoes sparse computation using newly generated dropout masks, producing one stochastic output, which will be written back to off-chip memory as well.

In summary, this mode-level control ensures that the required modules are enabled adapting to demands. The same datapath can therefore support hybrid BayesCVNN configurations without instantiating separate hardware modules for dense layers, sparse layers, and sampling. This runtime-adaptable scheduling allows the accelerator to exploit sparsity in Bayesian layers while avoiding unnecessary overhead in standard layers. By executing different modes for different layers, the accelerator skips redundant computation in non-Bayesian layers while preserving the ability to produce uncertainty estimates.

B. Pipeline Hiding and Mode Transitions

Sparse execution introduces control overhead from dropout-mask generation, valid-index compaction, and mode switching. To prevent these overheads from dominating the benefit of sparsity, the scheduler optimizes pipeline to overlap data preparation with computation.

For weight loading, the design adopts ping-pong buffer to overlap data transfer with computation. Weight loading is overlapped between the real and imaginary parts of weights within a layer and across consecutive layers, thereby hiding part of the off-chip weight-loading latency.

The scheduler also overlaps dropout-mask preparation with the previous layer execution. If the next layer switches to sparse mode, the mask-generation logic is activated in advance to generate dropout masks and compact them into valid-index streams. Before the next sparse computation begins, the valid-index streams have already been prepared and buffered. Similarly, in sampling mode, dropout-mask generation is enabled ahead of each stochastic run to prepare the sparse valid-index streams for the next sampling. In this way, all of the index streams are prepared before they are consumed by the compute engine, reducing the runtime overhead of sparse execution.

In addition, sparse and sampling modes process a batch of stochastic samples to reduce repeated weight-loading overhead. Samplings within the same batch can reuse the loaded weights, improving the weight locality. The batch size is determined by the available on-chip buffer resources. In our design, the batch size in the sampling mode is four.

After all samples are completed, the outputs are aggregated to obtain the predictive result and uncertainty estimate. The predictive mean can be used as the final output, while predictive variance or entropy can be used as the uncertainty score.

VI. Experiments

A. Experimental Setup

We implement the proposed FPGA accelerator targeting the AMD VCK190 evaluation platform. Vitis HLS 2024.1 is used for synthesis and implementation. The accelerator is synthesized as PL kernel and is controlled by the PS for input transfer, execution launch, and result collection. Model weights are stored in off-chip memory and loaded into on-chip buffers during layer-by-layer execution. The final design is clocked at 200MHz. The post-implementation design consumes 251,581 LUTs (27.96%), 167,429 FFs (9.30%), 631 DSPs (32.06%), and 427 BRAMs (44.16%).

B. Dataset and Model Configurations

We evaluate the proposed framework on a dataset made from Bloch simulations using State Transition Matrices [28], which are carried out in Julia [29] to generate complex-valued signal evolutions for 2048 parameter combinations randomly drawn from physiologically relevant ranges. The simulations use a MRF sequence with an adiabatic inversion pulse followed by a fast radial gradient echo readout with a sinusoidal flip angle pattern and gradient-spoiling [30]. The dataset is split into training, validation and test sets in a ratio of 7:1:2. The framework is run separately for each of four prediction targets: the relaxation-related tissue parameters R_1, R_2 and R_2', together with B_1, which is not a tissue property but a factor accounting for the inhomogeneous strength of the RF transmit field.

The model consists of a six-layer complex-valued backbone followed by a task-specific output head. Five configurable dropout layers are placed after the first five backbone layers. The active dropout sites and their dropout rates are determined by the proposed search framework. 8-bit linear quantization is adopted to improve the hardware performance. The relative quantization error of the searched configurations stays below 3% for both MAE and ECE, demonstrating that quantization preserves algorithmic quality. During inference, 128 sampling is performed to estimate the predictive mean and uncertainty.

C. Evaluation Metrics

For algorithmic performance, we evaluate parameter-estimation accuracy using the mean absolute error (MAE) of each predicted parameter. Uncertainty calibration is measured using the expected calibration error (ECE) [31]. Considering the practical scenarios, we add a constraint that predictive uncertainty responds meaningfully to input degradation. We introduce an uncertainty-noise response metric, which evaluates whether the predictive uncertainty increases as the input signal becomes more corrupted. For each candidate configuration, we evaluate the model under different relative input noise levels: 0 (clean), 0.1, 0.3, and 0.5. At each level, zero-mean complex Gaussian noise is added to the clean input signal, with the real and imaginary components sampled independently and scaled according to the target noise level. Predictive uncertainty is estimated from 128 Monte Carlo dropout samples. We define relative uncertainty as the predictive standard deviation divided by the absolute predictive mean, averaged over the evaluation samples. The uncertainty-noise response is quantified using the Spearman rank correlation between the noise level and the relative uncertainty. We further impose a strict monotonicity constraint, requiring the relative uncertainty to increase along with increased noise levels.

For hardware evaluation, we report latency, power, and energy efficiency. Latency is measured for BayesCVNN inference, including the deterministic complex-valued layers, Monte Carlo dropout-enabled layers, and multiple sampling.

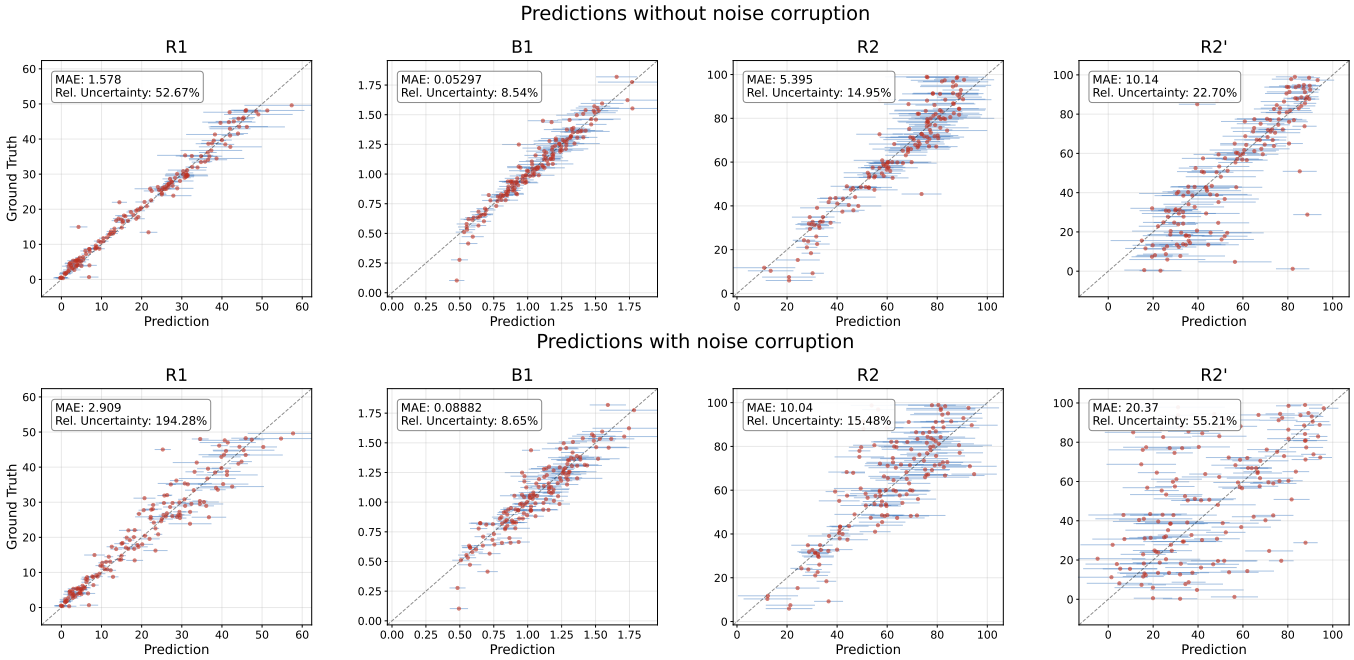

Fig. 4. BayesCVNN prediction with and without noise corruption.

Power is estimated using the Xilinx's BEAM tool on-board. The energy efficiency is derived from the measured latency and consumed energy.

D. Sparsity-Aware Search Results

1) Results Analysis: We set different optimization aims for the sparsity-aware framework to profile the searched configurations, as shown in Table I. For each target parameter, the table reports the best configuration statistics under three aims: minimizing MAE, minimizing ECE, and maximizing the uncertainty-noise response. The uncertainty-noise aim selects the configuration whose relative predictive uncertainty increases most consistently with the input noise level, measured by the Spearman rank correlation between noise level and relative uncertainty. The Number of Bayesian Layers reports the enabled dropout layers, where "last$k$" denotes that only the last k candidate dropout sites are enabled for Monte Carlo sampling, while the earlier layers remain deterministic. The Imbalanced Layers denotes the percentage of enabled Bayesian layers where the real and imaginary branches use different dropout rates. The Dropout rate ≥ 0.5 reports the percentage of active non-zero dropout rates that are no smaller than 0.5.

Three observations drawn from the results back up our assumption and motivation. First, the optimal Bayesian-layer sparsity is target- and objective-dependent. The selected configurations range from last1 to last5, indicating that a fixed number of Bayesian layers cannot consistently satisfy all accuracy, calibration, and uncertainty objectives. Incorporating Bayesian-layer sparsity to the search framework is justified. Morever, dropout rates no smaller than 0.5 frequently appear in the best configurations, suggesting that the searched models expose substantial activation sparsity. Developing customized hardware support to exploit the activation sparsity can bring performance gain. In addition, real and imaginary dropout imbalance is consistently selected across all targets and objectives. Mitigating resource underutilization caused by imbalanced workload in hardware design is critical.

TABLE I
SPARSITY-AWARE SEARCH RESULTS FOR TARGET PARAMETERS.

Prediction	Configuration Statistic	MAE Best	ECE Best	Unc.-Noise Best
R_1	Number of Bayesian Layers	last5	last5	last1
	Imbalanced Layers	100.0%	100.0%	100.0%
	Dropout Rate ≥ 0.5	50.0%	50.0%	50.0%
B_1	Number of Bayesian Layers	last2	last5	last1
	Imbalanced Layers	100.0%	100.0%	100.0%
	Dropout Rate ≥ 0.5	100.0%	66.7%	100.0%
R_2	Number of Bayesian Layers	last4	last5	last5
	Imbalanced Layers	100.0%	100.0%	100.0%
	Dropout Rate ≥ 0.5	50.0%	60.0%	60.0%
R_2'	Number of Bayesian Layers	last3	last2	last2
	Imbalanced Layers	100.0%	100.0%	100.0%
	Dropout Rate ≥ 0.5	66.7%	100.0%	100.0%

2) Practical Applications: For practical medical fingerprint scenarios, we search model configurations considering MAE and ECE with constraints requiring relative predictive uncertainty to increase monotonically with input noise, ensuring that uncertainty estimates are noise-sensitive, aligning with practical demands. Among configurations satisfying this constraints, candidates are further ranked according to MAE and ECE. Therefore, the selected configuration strikes a balance between prediction accuracy and calibration quality.

Fig. 4 illustrate the practical applications of prediction with uncertainty estimates in MRF. In real clinical or experimental deployments, the acquired fingerprints can be corrupted by noise, while the ground-truth parameters are generally unavailable at inference time. In such cases, a deterministic model may produce an overconfident parameter estimate, but the user has limited information about whether the prediction should be trusted. By contrast, BayesCVNNs provide predic-

tive uncertainty through multiple stochastic forward passes without requiring ground-truth labels during inference. Therefore, high-uncertainty outputs can be treated as unreliable cases that require human expert inspection, repeat acquisition, or downstream correction, while low-uncertainty outputs can support more automated estimation.

E. Comparison with GPU and Related Works

Table II compares the proposed FPGA accelerator with GPU implementations under the same searched model configurations. The batch size is set as 1 for all models on the evaluated hardware platforms following [16], [18]. We measure the performance of both naive GPU implementations and GPU implementations exploiting Bayesian-layer sparsity, denoted with the _L suffix. The latency for 128 samplings is reported, and the energy cost is calculated as the energy consumed per input. In terms of speed, the proposed FPGA design achieves 1.82×∼5.77× speedup over the naive GPU implementations and 1.64×∼2.20× speedup over the sparsity-exploiting GPU implementations across the four prediction target parameters. Moreover, the FPGA accelerator consumes only 11.3% of the GPU power, representing a substantial power reduction. Consequently, compared with the naive GPU implementations, the proposed design reduces the energy cost by 16.1×∼49.9×. When compared with the sparsity-exploiting GPU implementations, it still achieves energy cost reductions of 14.6×∼19.3×. These performance gains are attributed to efficient sparsity exploration and exploitation. The automated search framework identifies Bayesian-layer sparsity and activation sparsity, which are exploited in the customized hardware via skipping a substantial amount of redundant computation.

TABLE II
FPGA DESIGN VS. GPU IMPLEMENTATION PERFORMANCE.

	GPU				Ours			
Platform	GeForce RTX 3090 Ti				AMD VCK190			
Frequency	1.40 GHz				200 MHz			
Technology	8 nm				7 nm			
Power (W)	128.3				14.5			
Target	R_1	B_1	R_2	R_2'	R_1	B_1	R_2	R_2'
Latency (ms)	855.0	874.1	866.8	862.1	148.1	211.1	476.7	219.1
Latency_L (ms)	316.2	461.4	783.6	481.6				
EnergyCost (J/input)	109.7	112.1	111.2	110.6	2.2	3.1	6.9	3.2
EnergyCost_L (J/input)	40.6	59.2	100.5	61.8				

To further demonstrate the benefits of the proposed hardware architecture and optimization framework as a whole, we also compare this work against the prior BayesCVNN accelerator [23] as shown in Table III. The optimization in [23] primarily targets complex-valued convolutions and only considers heterogeneity of real and imaginary configurations, without fine-grained sparsity exploitation or customized hardware support. Since the targeted model is different, it is difficult to make a direct comparison. [23] reports the hardware performance of the complex-valued LeNet, we therefore use effective throughput, defined as FLOPs divided by latency, as a metric for comparison, suggesting that the proposed algorithm-hardware co-design achieves a higher realized compute rate.

TABLE III
COMPARISON WITH PRIOR BAYESCVNN ACCELERATION.

Design	Model Type	Sparse-Aware	Effective Throughput (GFLOPs/s)
[23]	BayesCV-Conv	No	12.3
Ours	BayesCV-FC	Yes	53.2

F. Ablation Study

To evaluate the effect of exploiting Bayesian-layer sparsity and activation sparsity, we conduct ablation experiments as shown in Fig. 5. The naive baseline executes all of model layers, involving fully dense computation. With *T1: Bayesian-layer sparsity* enabled, only the selected Bayesian layers are executed with multiple sampling, while the remaining front layers are executed once in deterministic mode. This reduces redundant layer computation and achieves an average 3.07× speedup. With *T2: activation sparsity exploitation* further enabled, the hardware skips redundant operations introduced by dropout operations in the Bayesian layers. This provides an average 1.22× speedup over the T1-only design. The improvement is target-dependent because different searched configurations use different numbers of Bayesian layers and varied dropout rates for real and imaginary parts, leading to different amounts of exploitable sparsity. Overall, the ablation study demonstrates the effectiveness of both sparsity exploitation, which reduce the number of layers requiring stochastic sampling, and accelerate the sparse computation for Bayesian layer operations.

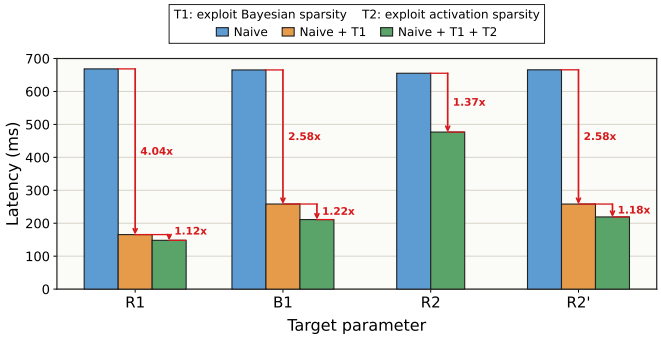

Fig. 5. Ablation study on Bayesian-layer and activation sparsity exploitation.

VII. CONCLUSION

This paper presents an framework for acceleration of dropout-based BayesCVNNs via sparsity exploration and exploitation for MRF. On the algorithmic side, the proposed framework explicitly explores Bayesian-Layer sparsity and activation sparsity, selecting target-dependent configurations under specific accuracy and uncertainty objectives. On the hardware side, the FPGA accelerator employs column-wise sparse MVM with runtime-adaptable scheduling to exploit the searched sparsity in hybrid standard and Bayesian layers. The experimental results show that the searched configurations select different configurations across different tasks. Compared with GPU execution, the proposed FPGA accelerator achieves faster speed while consuming less power. These results demonstrate that jointly exploring and exploiting sparsity can make BayesCVNN inference more efficient.

REFERENCES

[1] D. Ma, V. Gulani, N. Seiberlich, K. Liu, J. L. Sunshine, J. L. Duerk, and M. A. Griswold, "Magnetic resonance fingerprinting," *Nature*, vol. 495, no. 7440, pp. 187–192, 2013.

[2] A. Panda, B. B. Mehta, S. Coppo, Y. Jiang, D. Ma, N. Seiberlich, M. A. Griswold, and V. Gulani, "Magnetic resonance fingerprinting–an overview," *Current opinion in biomedical engineering*, vol. 3, pp. 56–66, 2017.

[3] H. Ding, C. Velasco, H. Ye, T. Lindner, M. Grech-Sollars, J. O'Callaghan, C. Hiley, M. D. Chouhan, T. Niendorf, D.-M. Koh *et al.*, "Current applications and future development of magnetic resonance fingerprinting in diagnosis, characterization, and response monitoring in cancer," *Cancers*, vol. 13, no. 19, p. 4742, 2021.

[4] P. J. Van Houdt, Y. Yang, and U. A. Van der Heide, "Quantitative magnetic resonance imaging for biological image-guided adaptive radiotherapy," *Frontiers in oncology*, vol. 10, p. 615643, 2021.

[5] A. M. Aldawsari, B. Al-Qaisieh, D. A. Broadbent, D. Bird, L. Murray, and R. Speight, "The role and potential of using quantitative mri biomarkers for imaging guidance in brain cancer radiotherapy treatment planning: A systematic review," *Physics and Imaging in Radiation Oncology*, vol. 27, p. 100476, 2023.

[6] P. Virtue, X. Y. Stella, and M. Lustig, "Better than real: Complex-valued neural nets for mri fingerprinting," in *2017 IEEE international conference on image processing (ICIP)*. IEEE, 2017, pp. 3953–3957.

[7] E. Cole, J. Cheng, J. Pauly, and S. Vasanawala, "Analysis of deep complex-valued convolutional neural networks for mri reconstruction and phase-focused applications," *Magnetic resonance in medicine*, vol. 86, no. 2, pp. 1093–1109, 2021.

[8] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," *arXiv preprint arXiv:1705.09792*, 2017.

[9] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" *Advances in neural information processing systems*, vol. 30, 2017.

[10] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. Dillon, B. Lakshminarayanan, and J. Snoek, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," *Advances in neural information processing systems*, vol. 32, 2019.

[11] K. Zou, Z. Chen, X. Yuan, X. Shen, M. Wang, and H. Fu, "A review of uncertainty estimation and its application in medical imaging," *Meta-Radiology*, vol. 1, no. 1, p. 100003, 2023.

[12] S. Seoni, V. Jahmunah, M. Salvi, P. D. Barua, F. Molinari, and U. R. Acharya, "Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013–2023)," *Computers in Biology and Medicine*, vol. 165, p. 107441, 2023.

[13] S. Faghani, M. Moassefi, P. Rouzrokh, B. Khosravi, F. I. Baffour, M. D. Ringler, and B. J. Erickson, "Quantifying uncertainty in deep learning of radiologic images," *Radiology*, vol. 308, no. 2, p. e222217, 2023.

[14] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *International conference on machine learning*. PMLR, 2015, pp. 1613–1622.

[15] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*. PMLR, 2016, pp. 1050–1059.

[16] H. Fan, M. Ferianc, M. Rodrigues, H. Zhou, X. Niu, and W. Luk, "High-performance fpga-based accelerator for bayesian neural networks," *arXiv preprint arXiv:2105.09163*, 2021.

[17] M. Ferianc, Z. Que, H. Fan, W. Luk, and M. Rodrigues, "Optimizing bayesian recurrent neural networks on an fpga-based accelerator," in *2021 International conference on field-programmable technology (ICFPT)*. IEEE, 2021, pp. 1–10.

[18] H. Fan, M. Ferianc, Z. Que, S. Liu, X. Niu, M. R. Rodrigues, and W. Luk, "Fpga-based acceleration for bayesian convolutional neural networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 12, pp. 5343–5356, 2022.

[19] H. Fan, M. Chen, L. Castelli, Z. Que, H. Li, K. Long, and W. Luk, "When monte-carlo dropout meets multi-exit: Optimizing bayesian neural networks on fpga," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2023, pp. 1–6.

[20] Z. Zhang, M. Genci, H. Fan, A. Wetscherek, and W. Luk, "Accelerating mri uncertainty estimation with mask-based bayesian neural network," in *2024 IEEE 35th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*. IEEE, 2024, pp. 107–115.

[21] Z. Zhang, H. Fan, H. Chen, L. Dudziak, and W. Luk, "Hardware-aware neural dropout search for reliable uncertainty prediction on fpga," in *Proceedings of the 61st ACM/IEEE Design Automation Conference*, 2024, pp. 1–6.

[22] H. Fan, M. Ferianc, Z. Que, X. Niu, M. Rodrigues, and W. Luk, "Accelerating bayesian neural networks via algorithmic and hardware optimizations," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 12, pp. 3387–3399, 2022.

[23] Z. Zhang, M. Chen, H. Li, and W. Luk, "Algorithm and hardware co-design for efficient complex-valued uncertainty estimation," in *Proceedings of the 63st ACM/IEEE Design Automation Conference*, 2026, pp. 1–6.

[24] A. Kendall, V. Badrinarayanan, and R. Cipolla, "Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding," *arXiv preprint arXiv:1511.02680*, 2015.

[25] A. Kristiadi, M. Hein, and P. Hennig, "Being bayesian, even just a bit, fixes overconfidence in relu networks," in *International conference on machine learning*. PMLR, 2020, pp. 5436–5446.

[26] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European conference on computer vision*. Springer, 2020, pp. 544–560.

[27] Z. Que, H. Nakahara, E. Nurvitadhi, A. Boutros, H. Fan, C. Zeng, J. Meng, K. H. Tsoi, X. Niu, and W. Luk, "Recurrent neural networks with column-wise matrix–vector multiplication on fpgas," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 30, no. 2, pp. 227–237, 2021.

[28] N. Scholand, X. Wang, V. Roeloffs, S. Rosenzweig, and M. Uecker, "Quantitative mri by nonlinear inversion of the bloch equations," *Magnetic Resonance in Medicine*, vol. 90, no. 2, pp. 520–538, 2023.

[29] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing," *SIAM review*, vol. 59, no. 1, pp. 65–98, 2017.

[30] T. Bruijnen, O. van der Heide, M. P. Intven, S. Mook, J. J. Lagendijk, C. A. Van den Berg, and R. H. Tijssen, "Technical feasibility of magnetic resonance fingerprinting on a 1.5 t mri-linac," *Physics in Medicine & Biology*, vol. 65, no. 22, p. 22NT01, 2020.

[31] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *International conference on machine learning*. PMLR, 2017, pp. 1321–1330.