

VAHANA: Hardware-Aware Noise Addition Against Gradient Inversion Attacks

Arsalan Ali Malik*, Digvijay Anand*, Chau-Wai Wong, and Aydin Aysu
North Carolina State University, Raleigh, NC 27606, United States
{aamalik3, danand3, chauwai.wong, aaysu}@ncsu.edu

Abstract—Federated learning (FL) enables clients to collaboratively train a global model by sharing gradient updates instead of raw data. However, recent studies show that these shared gradients can be exploited to reconstruct private training data through gradient inversion attacks, posing a serious threat to client privacy. A common defense strategy is gradient obfuscation, which perturbs gradients before transmission; however, it relies on costly hardware components, such as Gaussian samplers and floating-point (FP) multipliers, making it unsuitable for resource-constrained edge devices. Approximate circuits—extensively studied in computer architecture and VLSI for their energy, area, and latency benefits—offer a promising defense. Yet despite its success in machine learning accelerators, it remains underexplored for privacy-preserving FL.

This work introduces variational approximate hardware-aware noise addition (VAHANA), a hardware-efficient method that integrates stochastic perturbations directly into the computation using approximate circuits and only 4 bits of uniform randomness, eliminating the need for Gaussian sampling or FP multiplication. We integrate VAHANA into the open-source RISC-V CV32E40P softcore using a hardware–software co-design approach, implementing custom instruction extensions that accelerate privacy-preserving operations with improved execution efficiency in both time and area. Evaluation on Kintex-7 FPGA shows that VAHANA reduces memory footprint by 34% and reduces latency by up to $28.1\times$, while maintaining resilience against gradient inversion attacks on Medical MNIST, CIFAR-10, and CIFAR-100 datasets.

Index Terms—Federated Learning, Approximate Computing, Privacy-Preserving Machine Learning

I. INTRODUCTION

Federated learning (FL) is a decentralized training paradigm in which multiple clients collaboratively learn a global model by exchanging gradient updates rather than raw data [1], [2]. Edge-FL deploys this paradigm on resource-constrained platforms, such as smartphones, IoT sensors, and autonomous vehicles. This makes edge-FL attractive for privacy-sensitive domains, including health diagnostics, autonomous systems, and financial fraud detection [3]–[5]. Despite numerous advantages of FL, it remains vulnerable to user privacy attacks, such as *gradient inversion*, where an adversary reconstructs private training data from shared gradient updates which encode salient properties of the underlying data distribution [6], [7]. In edge-FL, deterministic compute graphs and limited on-chip entropy further exacerbate this risk, increasing the practical effectiveness of these attacks [8].

A common practice against gradient inversion attacks is gradient perturbation [9], [10]. Scheliga *et al.* proposed privacy-enhancing module (PRECODE) that augments a model by

*Co-first authors.

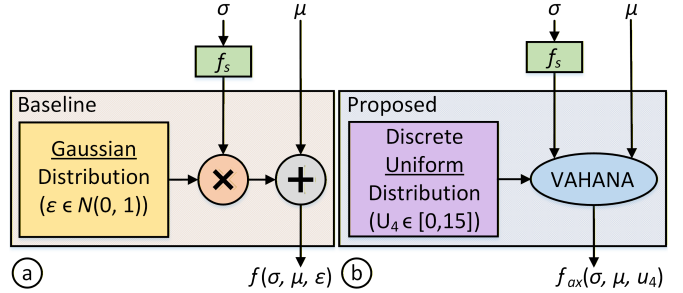


Fig. 1. (a) Baseline reparameterization in variational modeling relies on Gaussian sampling and floating-point multiplication, both of which incur additional hardware cost due to required Gaussian random number generators (RNGs) and full-precision multipliers. (b) VAHANA injects the required stochasticity using an approximate adder and 4-bits of uniform randomness to achieve the desired stochasticity, eliminating the need for Gaussian sampling and FP multiplication for privacy preservation.

inserting a variational bottleneck (VIB) layer prior to the classifier to perturb shared gradients and reduce information leakage [11]. PRECODE’s key idea is to introduce stochastic sampling in the latent representation such that resulting gradients no longer contain direct information about the original input, causing gradient-inversion optimization to fail to converge while largely preserving utility. To this end, PRECODE uses the *reparameterization trick* to draw latent features from learned parameters, mean (μ) and standard deviation (σ). Fig. 1(a) summarizes this pipeline where Gaussian noise $\epsilon \sim \mathcal{N}(0, 1)$ is multiplied by $f_s(\sigma)$ and added to μ , producing encoded training data $f(\sigma, \mu, \epsilon)$. Here, f_s is a softplus function that enforces a positive scale to σ . PRECODE implicitly assumes access to high-quality Gaussian randomness and full-precision floating-point (FP) multiplication along the training datapath, which can hinder deployment on edge-FL platforms. As privacy-preserving machine learning (PPML) shifts toward on-device learning, this gap motivates hardware-aware defenses that exploit the statistical structure of model parameters while meeting stringent delay, size, area, and power (SWaP) constraints [12], [13].

Fig. 1 (b) illustrates our proposed variational approximate hardware-aware noise addition (VAHANA¹) method, which reduces the hardware cost of PRECODE by replacing Gaussian sampling and FP multiply-add operation with a simple 4-bit uniform noise and an approximate adder. Instead of drawing $\epsilon \sim \mathcal{N}(0, 1)$, VAHANA samples an integer $U_4 \in [0, 15]$ from a discrete uniform distribution, scales it by $f_s(\sigma)$, and then combines it with the mean (μ) to produce the required

¹Source code: <https://anonymous.4open.science/r/VAHANA-3C29>

stochastic encoding $f_{\text{ax}}(\sigma, \mu, U_4)$. This design choice preserves the perturbation needed to thwart gradient inversion attacks while keeping the complexity, latency, and resource usage of hardware minimal. We claim the following contributions:

- **Hardware-efficient gradient obfuscation.** We analyze the statistical requirements for effective gradient obfuscation in FL and propose a hardware-efficient method that eliminates Gaussian sampling and floating-point multiplication while reducing area and latency, as experimentally validated on a RISC-V soft-core processor.
- **Approximate circuit for efficient realization.** We introduce an approximate circuit that embeds discrete uniform randomness into the latent feature space and empirically show that it closely preserves the statistical behavior and training stability of VIB layer used in PRECODE.
- **End-to-end privacy evaluation.** We perform a comprehensive evaluation against gradient inversion attacks on Medical-MNIST, CIFAR-10, and CIFAR-100 using three representative backbone models—SMLP, DMLP, and LeNet—showing that VAHANA preserves PRECODE’s accuracy and privacy protection.
- **Hardware–software co-design and efficiency gains.** We integrate VAHANA as a custom instruction into the open-source CV32E40P RISC-V softcore and prototype it on a Kintex-7 FPGA, achieving a 34% reduction in memory footprint and upto $28.1\times$ reduction in latency, while avoiding any additional DSP blocks.

II. BACKGROUND AND RELATED WORK

A. Federated Learning

FL trains a global model by aggregating client-side updates instead of raw data. While this improves data locality, shared gradients can still leak information about private training samples. Modern deployments utilize FL directly at the edge: mobile and wearable SoCs, XR headsets, IoT nodes, and in-vehicle or clinical devices [14]–[18]. In such platforms, FL is attractive because sensitive data (*e.g.*, images, speech, biometrics, medical scans) can remain on the device while the shared model continues to improve [3]–[5].

B. Gradient Inversion Attacks

Gradient inversion attacks reconstruct training inputs from shared gradients by optimizing dummy inputs to match observed gradients. Prior work has shown that this leakage can be severe even when raw data never leaves the client [19]–[22]. In a typical gradient inversion attack, the adversary initializes a dummy input and label, computes their corresponding gradients through the known model, and iteratively updates the dummy input until its gradients match the observed client gradients. The attack is effective because early-layer gradients encode both low-level image statistics and label-dependent information, especially when the attacker knows the model architecture, loss function, and optimization procedure. Stronger variants differ mainly in the matching objective, image prior, label handling, and regularization terms, but they share the same principle—private data are reconstructed by solving an optimization problem in input space. This motivates

defenses that perturb the gradient-bearing latent representation before gradients are shared.

C. Earlier Defense Schemes

Recent PPML defenses against gradient inversion span compression and randomization. Gradient perturbation methods, including defensive quantization (DQ) and sparsification, reduce the information revealed during training by coarsening or pruning shared gradients [10], [23]. By contrast, differentially private stochastic gradient descent (DP-SGD) and related noisy-gradient schemes inject Gaussian or Laplace noise to mask sensitive signal components [9], [20], [24]. While both classes can degrade reconstruction quality, they frequently trade privacy for reduced utility and non-trivial compute overhead.

Scheliga *et al.* proposed PRECODE, enabling private extension to models using a VIB layer to balance between privacy and model utility [11]. Fig. 4 shows PRECODE, which consists of a VIB layer comprising an encoder, a reparameterization layer, and a decoder. The encoder estimates the mean (μ) and variance-proxy (σ) for k features, where k denotes the bottleneck width. These parameters are then used by the reparameterization trick to sample latent features (z). Let k denote the latent width, *i.e.*, the number of bottleneck features produced by the encoder and consumed by the downstream classifier. For a latent vector $z = (z_1, \dots, z_k)$, the subscript $f \in \{1, \dots, k\}$ denotes one latent feature coordinate. The standard reparameterization operation is then applied independently to each coordinate as:

$$z_f = \mu_f + \sigma_f \cdot \epsilon_f, \quad f \in \{1, \dots, k\}, \quad (1)$$

where $(\cdot)_f$ represents each feature and $\epsilon_f \sim \mathcal{N}(0, 1)$ is Gaussian noise sampled independently for the f -th latent coordinate. Changing k changes the number of latent coordinates that must be reparameterized, and therefore scales the total execution latency approximately linearly, but it does not change the VAHANA datapath used for each coordinate. The training objective combines a cross-entropy loss with a Kullback-Leibler (KL) divergence term to enforce the Gaussian prior [25]:

$$\mathcal{L}(y, \hat{y}) = \mathcal{L}_F(y, \hat{y}) + \beta \cdot D_{KL}(\mathcal{N}(\mu, \sigma^2) \parallel \mathcal{N}(0, 1)) \quad (2)$$

where $\mathcal{L}_F$ is the task loss (*e.g.*, cross-entropy) and β controls the trade-off between prediction accuracy and privacy. Although PRECODE provides strong empirical protection against gradient inversion attacks, its reliance on Gaussian sampling and full-precision floating-point (FP) multiplication incurs substantial hardware overheads. In particular, PRECODE requires (i) a dedicated Gaussian random number generator (RNG), which can cost hundreds to thousands of cycles per sample, and (ii) full-precision FP multipliers and adders to implement the reparameterization. These costs hinder deployability in resource-constrained edge settings, where area, energy, and latency budgets are tightly bound.

D. Challenges with Existing Defense Frameworks

Despite progress in PPML, variational defenses remain hard to deploy on edge platforms because they assume efficient Gaussian sampling, FP-reparameterization, and privacy-aware instruction support—capabilities that edge systems rarely provide at low cost. In practice, four constraints hinder deployment on edge devices.

1. Latency, Area, and Memory Constraints: Edge-FL runs under tight power and real-time budgets, yet variational defenses add a reparameterization path (*i.e.*, $z = \mu + \sigma \cdot \varepsilon$) to every training step. Implementing this path typically consumes scarce FP or fused multiply-add (FMA) throughput, adds switching activity, and increases code/parameter footprint, which directly amplifies latency and energy. Approximate arithmetic exists, but most prior designs target *deterministic* error tolerance rather than *controlled stochastic* perturbations, limiting their use as drop-in variational units.

2. Lack of a Suitable Gaussian Noise Source: The VIB pipeline requires $\varepsilon \sim \mathcal{N}(0, 1)$, but edge SoCs commonly expose only uniform pseudo-random number generators (PRNGs) (or, at best, *limited* true random number generator entropy). Converting uniform bits into high-quality Gaussian samples is expensive: software sampling is slow, while hardware-friendly methods (*e.g.*, Box-Muller, Ziggurat) require non-trivial multipliers, transcendental support, or large lookup tables, inflating latency and on-chip memory.

3. Neglected Statistical Properties: Learned variational parameters often exhibit bounded dynamic range, sparsity, and tolerance to higher-order statistical deviations (especially in overparameterized networks). Nevertheless, many defenses default to full-precision FP representations and arithmetic, even when privacy requires only approximate zero-mean noise with bounded variance and low correlation. This ‘precision by default’ increases memory traffic and energy without proportional privacy gains.

4. Limited ISA and Accelerator Support: Even with extensible instruction set architectures (ISAs), native support for lightweight entropy generation, stochastic rounding/noise injection, and privacy-aware arithmetic is typically absent. Consequently, implementations fall back to software emulation or full-precision FP units, compounding the aforementioned costs and impeding portability across edge-class devices. Together, these gaps motivate VAHANA. Instead of emulating Gaussian sampling and FP-heavy reparameterization, we leverage a lightweight entropy source and an approximate stochastic datapath to realize privacy-preserving reparameterization within strict area and latency budgets.

E. Approximate Computing and Security

Approximate computing has been widely used to reduce latency, area, and energy in error-tolerant ML workloads. Prior work has also used approximation to improve resilience against adversarial perturbations at inference time [12], [13], [26]. Recent efforts have also explored approximation as a security primitive. Guesmi *et al.* proposed approximate floating-point multipliers (Ax-FPMs) that replace mantissa

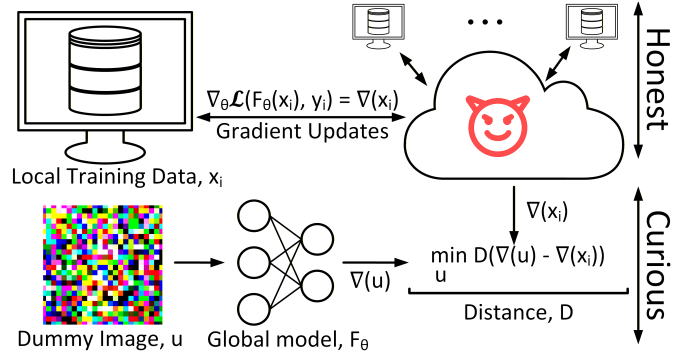


Fig. 2. Threat model for gradient inversion in federated learning. Each client i performs local training on private data x_i with label y_i using the current global model F_θ parameterized by weights θ , and transmits the gradient $\nabla_\theta \mathcal{L}(F_\theta(x_i), y_i)$ of the task loss $\mathcal{L}(\cdot)$ to the server. Under the honest-but-curious assumption, the server follows the FL protocol but attempts to reconstruct x_i from the received gradients. Following Geiping *et al.*, the attacker initializes a dummy input u and iteratively updates it to minimize a distance metric $D(\cdot)$ between gradients, *i.e.*, $\min_u D(\nabla(u), \nabla(x_i))$, where $\nabla(x_i) \triangleq \nabla_\theta \mathcal{L}(F_\theta(x_i), y_i)$ and $\nabla(u) \triangleq \nabla_\theta \mathcal{L}(F_\theta(u), y_i)$ [6]. We assume the client is the victim, the server is the attacker, and the server has sufficient compute to execute this optimization.

multiplication with an approximate mirror adder (AMA-5), injecting small, input-dependent perturbations to improve robustness to adversarial examples [27], [28]. Similarly, DA²F and DNNShield suppress adversarial noise at inference time via sparsification and approximate activation functions [29], [30]. However, these defenses are inference-only (*i.e.*, they do not mitigate privacy leakage in distributed training), and they do not address the Gaussian-sampling hardware bottleneck central to variational defenses. By contrast, VAHANA, uses approximation inside the variational reparameterization path to reduce privacy-defense cost during training.

III. THREAT MODEL

We adopt the threat model of Geiping *et al.* to perform the gradient inversion attack [6]. As illustrated in Fig. 2, the server is assumed to be *honest-but-curious*: it follows the FL protocol to uphold *honesty* but attempts to infer private client data out of *curiosity*. The attacker (server) is granted full knowledge of the learning pipeline, including the model architecture F , parameters θ , loss function $\mathcal{L}$, and the gradients $\nabla_\theta \mathcal{L}(F(x_i), y_i)$ shared by clients. The victim is a participating client whose training sample (x_i) is targeted. The attack is *remote*, the server never accesses the client device directly, and involves solving an optimization problem that searches for a dummy input u whose gradient $\nabla_\theta \mathcal{L}(F(u), y)$ matches the received gradient. Consistent with prior work, we assume the adversary has sufficient compute capabilities to execute this reconstruction procedure. Under these assumptions, gradient inversion attacks can recover high-fidelity approximations of the client’s data, motivating the need for hardware-efficient gradient obfuscation.

IV. HARDWARE DESIGN

A. Statistical Needs for Defense

Variational bottleneck (VIB) layers in PRECODE-style defenses are inherited from variational autoencoders (VAEs) and discrete representation models [31]–[33]. They enforce

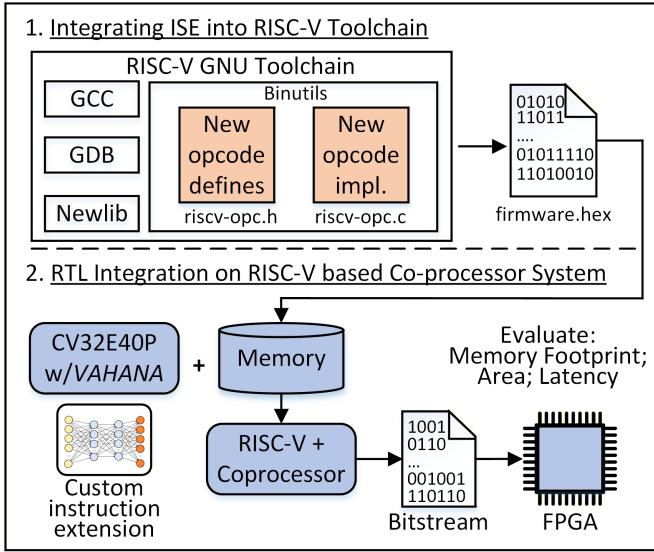


Fig. 3. Hardware-software co-design flow for integrating VAHANA into a RISC-V platform. (1) The VAHANA custom instruction is added to the RISC-V GNU toolchain by extending the opcode definitions and implementations within Binutils. Rebuilding the toolchain produces a cross-compiler capable of emitting VAHANA instructions. (2) Application code is compiled with the extended toolchain to generate firmware binaries that invoke the new instruction. (3) The CV32E40P core augmented with the VAHANA datapath is instantiated in RTL and synthesized to an FPGA, where we evaluate memory footprint, area, and cycle latency.

a (roughly) Gaussian latent space and use the reparameterization trick (eqs. (1)–(2)) to keep training differentiable while injecting stochasticity, thereby reducing the coupling between gradients and individual training examples. Our experiments indicate that, for the evaluated datasets, backbones, and attack configuration, exact continuous Gaussian samples are not required to obtain PRECODE-like empirical robustness. We therefore treat Gaussian replacement as an empirical hardware-approximation opportunity, not as a general privacy theorem: the approximate latent perturbations are designed to preserve bounded variance and sample-to-sample variability sufficient to disrupt the evaluated gradient inversion optimization. VAHANA capitalizes on this tolerance by replacing Gaussian sampling with low-bit uniform randomness and an approximate reparameterization path, achieving comparable obfuscation at substantially lower hardware cost.

B. VAHANA Method

VAHANA is a hardware-efficient replacement for the variational information bottleneck (VIB) Gaussian sampler. Rather than computing z_f via eq. (1) with full-precision Gaussian noise and a floating-point (FP) multiplier, VAHANA synthesizes an *effective* noise term directly in FP format using only low-bit uniform randomness and simple integer logic. VAHANA’s datapath is composed of four blocks: (i) a uniform randomness encoder (*randomizer*), (ii) a sign-perturbation unit, (iii) an exponent realignment unit, and (iv) a mantissa sparsifier that reuses the existing FMA pipeline.

1. Uniform Randomness Encoder (Randomizer). Let $U \in [0, 15]$ be a 4-bit unsigned random value, and let $U[j] \in \{0, 1\}$ denote its j -th bit. VAHANA first computes the centered, quantized control value:

$$q = 2 \left(\sum_{j=0}^3 U[j] - 2 \right) \quad (3)$$

which maps the Hamming weight of U to $q \in \{-4, -2, 0, 2, 4\}$. The three hardware control signals are then derived as: $ax_s = 1[q < 0]$, $ax_m = 1[|q| = 4]$, $ax_{en} = 1[q \neq 0]$. Here, sign flipper (ax_s) selects the perturbation polarity, exponent realignment (ax_m) selects the larger exponent shift, and mantissa sparsifier (ax_{en}) enables or disables perturbation injection. Thus, the randomizer does not generate a Gaussian sample; instead, it converts four uniform bits into a small set of control decisions that steer the sign, magnitude, and activation of the approximate perturbation. Higher-width implementations increase the perturbation granularity without incurring significant hardware cost, offering a tunable trade-off between model complexity and stochastic fidelity.

2. Sign-Perturbation Unit. In the classical VIB, the sign of the produced output z_f follows the sign of $\varepsilon_f, \mu_f, \sigma_f$ based on eq. (1). VAHANA instead perturbs the polarity from the encoder output and the randomizer. The perturbation sign bit (Δ) is obtained by XOR’ing ax_s with the sign of μ_f :

$$\Delta.\text{sign} = \mu_f.\text{sign} \oplus ax_s.$$

This realizes a randomized choice between additive and subtractive noise without any explicit Gaussian sampler, while still coupling the perturbation direction to model parameters.

3. Exponent Realignment Unit. Gaussian sampling scales the noise amplitude by σ_f . VAHANA approximates this effect by operating solely on the exponent field of σ_f . The exponent of the perturbation is computed as:

$$\Delta.\text{exp} = \sigma_f.\text{exp} + 1 + ax_m,$$

which shifts the scale of the injected noise by a small, random offset. This replaces a full FP mantissa multiplier with a single exponent adder and multiplexers, providing coarse-grained but effective magnitude variation at negligible hardware cost.

4. Mantissa Sparsifier and FMA Integration. To further reduce complexity, VAHANA does not propagate the full mantissa of σ_f . When $ax_{en} = 0$, the mantissa is zeroed, effectively disabling the perturbation. When $ax_{en} = 1$, only the implied leading bit is retained and the remaining mantissa bits are cleared, yielding a sparse mantissa pattern. Combined with the exponent logic, this produces a discrete set of perturbation magnitudes (e.g., $\{0, \pm 2, \pm 4\}$ in our configuration) that approximate the spread of the original Gaussian noise.

The perturbation, Δ is injected at the front end of the existing IEEE-754 FMA pipeline through multiplexers and barrel shifters and added with μ_f . All downstream stages (align-add, normalization, rounding) remain unmodified, enabling VAHANA to function as a drop-in hardware module.

Illustrative Example: Consider $U = 8$, whose 4-bit representation is 1000_2 . Since $\sum_{j=0}^3 U[j] = 1$, the centered control value becomes $q = 2(1 - 2) = -2$. Therefore, $ax_s = 1$, $ax_m = 0$, and $ax_{en} = 1$. For $(\mu_f, \sigma_f) = (1.25, 1.0)$, VAHANA flips the perturbation sign through ax_s , selects the

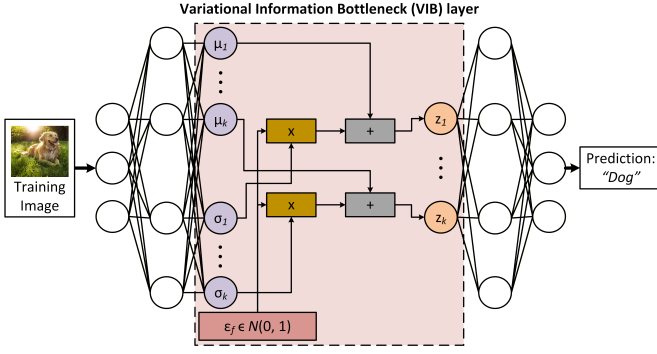


Fig. 4. PRECODE variational bottleneck module integrated into an existing unprotected model. The left simplified neural network layers extracts feature statistics, mean ($\mu_1, \dots, \mu_k$) and standard deviation $\sigma_1, \dots, \sigma_k$ for each latent dimension. The reparameterization block (shaded region) samples Gaussian noise $\varepsilon_f \sim \mathcal{N}(0, 1)$ and computes latent variables $z_i = \mu_i + \sigma_i \cdot \varepsilon_f$ using a FP multiplier and adder. Neurons in the encoder and decoder are represented by white circles. This stochastic latent vector ($z_1, \dots, z_k$) is then forwarded to the downstream classifier. PRECODE uses this variational layer to disguise the feature space and limit information leakage under gradient inversion attacks.

smaller nonzero magnitude through $ax_m = 0$, and enables perturbation injection through $ax_{en} = 1$, yielding $\Delta \approx -2$. The resulting latent value is thus $z_f = \mu_f + \Delta \approx 1.25 - 2 = -0.75$. This example shows how four uniform bits are converted into deterministic control signals that generate a stochastic latent perturbation without Gaussian sampling hardware.

C. Hardware-Software Co-design on RISC-V

We integrate VAHANA into the open-source RISC-V ecosystem by introducing a dedicated custom instruction (see Fig. 3). RISC-V is a modular ISA that explicitly reserves opcode space for vendor- and application-specific extensions, enabling tight hardware-software co-design without disrupting baseline compatibility [34]. In particular, the `inst[6:2]` major opcode field includes *custom* and *reserved* slots that can be safely repurposed to accelerate domain-specific kernels [35]. We utilize this mechanism to implement VAHANA as an in-core primitive, thereby avoiding software emulation of stochastic reparameterization and eliminating the need for costly Gaussian-sampling datapaths on edge-class processors.

For prototyping, we select the CV32E40P core [36], a 32-bit RISC-V processor from the OpenHW Group with a 4-stage in-order pipeline, RV32IM[F|Zfinx] C ISA, and support for custom extensions. CV32E40P was designed for tightly power-constrained IoT and near-threshold operation, emphasizing high computational density and reduced pressure on the shared-memory hierarchy [37]. These attributes closely match the deployment envelope of edge-FL, making CV32E40P a realistic baseline for evaluating the latency, area, and memory impact of VAHANA. **Importantly, our design is not specific to this core:** the instruction and datapath modifications can be ported to other RISC-V implementations that expose custom opcode support and an FP/FMA pipeline (including FPnew-derived units) [38]. We define a custom VAHANA instruction of R4-type with three source registers and one destination register to perform the fused operations, using the reserved opcode `5'b11010`:

```
vahana_appx_vib rd, rs1, rs2, rs3
```

Here, `rs1` and `rs2` contain the encoder outputs μ_f and σ_f , `rs3` supplies the uniform random variable U , and the randomized latent sample z_f is written to `rd`. This encapsulation maps the VAHANA reparameterization into a single architectural primitive, reducing instruction count, register pressure, and intermediate data movement compared to a software-only sequence of sign/exponent/mantissa manipulations.

Fig. 3 summarizes our end-to-end integration flow. In Step 1, we extend the RISC-V GNU toolchain to recognize `vahana_appx_vib` by adding its encoding and mnemonic to the opcode tables and implementing the corresponding assembler and disassembler hooks. Concretely, this requires updates to `riscv-opc.h` and `riscv-opc.c` in `binutils`, along with the associated decode paths, enabling the generation of correct firmware images (e.g., `firmware.hex`) for VAHANA-enabled applications. We also update the GCC RISC-V backend to expose the instruction through inline-assembly template to allow compilation.

In Step 2, we integrated the instruction into the CV32E40P RTL. We add the new opcode to the core's instruction-encoding definitions in `rv_i` and propagate the decode signal through the ID/EX control logic. We then connect the decode to the VAHANA approximate FMA front-end, ensuring that the three source operands are routed to the sign flipper, exponent re-alignment, and mantissa sparsifier sub-blocks, while keeping IEEE-754-compliant alignment/normalization/rounding backend unchanged. The writeback path is extended to commit z_f into `rd` using the standard register-file interface. This integration preserves the baseline pipeline structure while enabling native execution of variational noise injection on an edge-class core. Finally, we synthesize the modified CV32E40P with the VAHANA coprocessor on FPGA to evaluate memory footprint, area, and latency in a realistic deployment setting.

To generate U , we evaluate two entropy sources: the software-based `glibc` PRNG and the hardware Trivium32 stream cipher [39]. The `glibc` generator offers minimal overhead and reflects the default `rand()` path in C, but provides weaker statistical guarantees. Trivium32 provides a stronger cryptographic pseudo-random source at higher implementation cost. This split evaluation highlights that VAHANA's instruction-level interface is agnostic to the entropy provider, allowing system designers to trade off statistical strength and hardware overhead without changing the VAHANA datapath or the compiler-facing API. During execution, U , μ_f , and σ_f are loaded into general-purpose registers, and the VAHANA approximate FMA datapath applies the sign, exponent, and mantissa perturbations. By contrast, PRECODE's baseline hardware realization must first transform uniform samples into Gaussian noise (e.g., via inverse-transform steps) before reparameterization, incurring significantly higher latency and

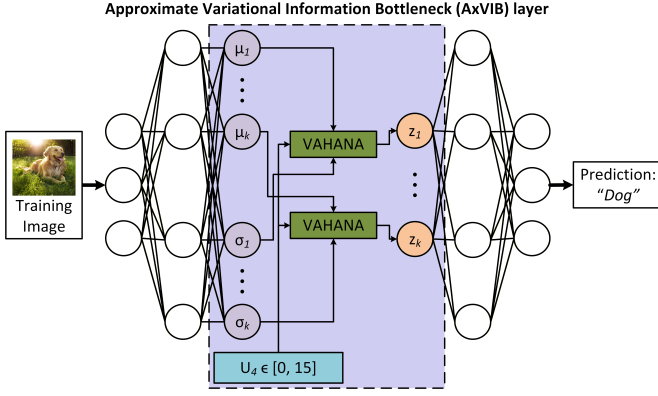


Fig. 5. VAHANA method integrated into an existing unprotected model. The left-side layers extract feature statistics, mean $(\mu_1, \dots, \mu_k)$ and standard deviation $(\sigma_1, \dots, \sigma_k)$, for a k -dimensional latent space. The VAHANA block in the shaded region replaces Gaussian sampling and FP multiplication with a lightweight stochastic datapath driven by a 4-bit uniform random variable $U_4 \in [0, 15]$. Using (μ_f, σ_f) and U_4 , VAHANA generates obfuscated latent variables $(z_1, \dots, z_k)$ to mimic the stochastic behavior of variational defenses while reducing hardware overhead. The resulting latent vector is forwarded to the downstream classifier, preserving model utility as illustrated with the label *Dog* while limiting information leakage under gradient inversion attacks.

area overhead².

V. IMPLEMENTATION RESULTS

We present VAHANA’s image-classification results in FL along two axes: (i) privacy against gradient inversion under approximate stochastic sampling, and (ii) hardware efficiency of the lightweight datapath. We first describe the privacy-evaluation setup and show that VAHANA preserves accuracy while substantially degrading input reconstruction, followed by showcasing VAHANA’s hardware efficiency.

A. Experimental Setup for Testing

We evaluate CIFAR-10, CIFAR-100, and Medical MNIST using PRECODE’s pipeline [11]. Models are trained for 300 epochs with Adam and the same objective as PRECODE. For privacy evaluation, we reconstruct a batch of 128 training images from shared gradients over 7000 attack iterations and compare under three configurations: unprotected, PRECODE, and VAHANA³. The attacker is assumed to know the architecture and training procedure, consistent with an *honest-but-curious* threat model [6]. To integrate stochastic sampling, we augment the baseline with a VAHANA-based approximate variational information bottleneck (AxVIB) layer between the final feature extractor and the classifier (Fig. 5). AxVIB embeds our approximate adder, perturbing latent codes via low-bit stochastic transformations over the sign, exponent, and mantissa fields. The randomness level is controlled by the number of mantissa bits removed.

B. Empirical Robustness Against Gradient Inversion Attacks

We empirically evaluate VAHANA’s robustness against SOTA gradient inversion attacks [6], rather than claiming a formal privacy guarantee. Our evaluation measures reconstruction degradation under the stated threat model, attack objective,

²Since PRECODE does not define a hardware implementation, we use a software-based Gaussian reparameterization path as the baseline realization of its sampling step on CV32E40P.

³All experiments are implemented in PyTorch.

TABLE I
SUMMARY OF EVALUATED BACKBONE ARCHITECTURES

Model	Latent Dim.	Backbone Layers
SMLP	1024	Flatten $\rightarrow$ FC($dim, 1024$) $\rightarrow$ ReLU $\rightarrow$ FC(1024,1024) $\rightarrow$ ReLU
DMLP	1024	Flatten $\rightarrow$ FC($dim, 1024$) $\rightarrow$ ReLU $\rightarrow$ FC(1024,1024) $\rightarrow$ ReLU $\rightarrow$ FC(1024,1024) $\rightarrow$ ReLU $\rightarrow$ FC(1024,1024) $\rightarrow$ ReLU
LeNet	768	Conv($C, 12, 5 \times 5$) $\rightarrow$ Sigmoid $\rightarrow$ Conv(12,12,5 $\times$ 5) $\rightarrow$ Sigmoid $\rightarrow$ Conv(12,12,5 $\times$ 5) $\rightarrow$ Sigmoid $\rightarrow$ Flatten

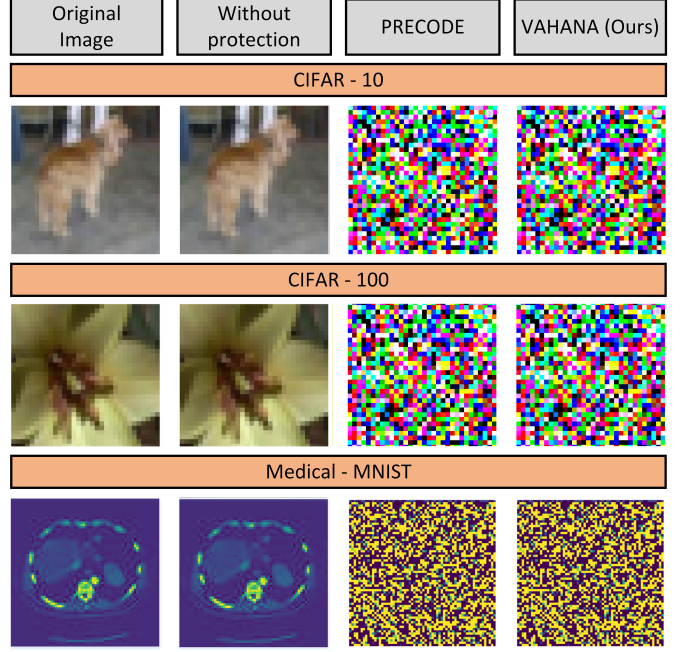


Fig. 6. Reconstruction outcomes of gradient inversion attacks on CIFAR-10, CIFAR-100, and Medical-MNIST. For each dataset, we show the original image, the reconstruction obtained without protection, and the outputs under PRECODE [11] and VAHANA. Without protection, the attacker recovers high-fidelity replicas of the private training samples. Both PRECODE and VAHANA disrupt the attack optimization, producing noise-like reconstructions that reveal no semantic content. These results demonstrate that VAHANA achieves empirical robustness comparable to PRECODE under the evaluated gradient inversion attack, while enabling a substantially more hardware-efficient implementation.

datasets, and backbone models. To enable a controlled and fair comparison, we study three baseline models with increasing capacity: a shallow Multi-Layer Perceptron (SMLP), a deeper MLP (DMLP), and a convolutional LeNet (see Table I). SMLP uses two 1024-wide hidden layers, while the DMLP extends this backbone to four 1024-wide hidden layers. LeNet baseline follows a conventional three-layer convolutional feature extractor with a 768-dimensional latent feature vector prior to classification. For PRECODE, we augment each baseline with a VIB layer and for VAHANA, we insert our proposed VAHANA module as shown in Fig. 4 and Fig. 5.

We compare VAHANA against both the unprotected baselines and PRECODE to demonstrate resilience under identical threat and training conditions. Fig. 6 provides qualitative reconstructions on CIFAR-10, CIFAR-100, and Medical MNIST.

TABLE II
PERFORMANCE AND PRIVACY METRICS ON CIFAR-10

Metric*	SMLP			DMLP			LeNet		
	w/o	PRECODE	VAHANA	w/o	PRECODE	VAHANA	w/o	PRECODE	VAHANA
trn_acc (%) ↑	96.60	90.20	93.70	98.70	96.80	96.87	76.70	73.80	73.40
tst_acc (%) ↑	52.20	48.40	48.40	53.30	51.70	52.50	59.90	58.10	61.10
MSE ↑	0.00	3.88	4.36	0.00	4.71	4.64	0.03	4.67	4.70
LPIPS ↑	0.00	0.40	0.41	0.00	0.39	0.40	0.01	0.43	0.44
PSNR ↓	75.49	6.15	5.64	87.24	5.53	5.51	27.20	5.35	5.32
SSIM ↓	0.99	0.01	0.00	0.99	0.00	0.00	0.89	0.00	0.00
ASR (%) ↓	100.00	0.00	0.00	100.00	0.00	0.00	100.00	0.00	0.00

* Metrics include training accuracy (trn_acc), testing accuracy (tst_acc), mean squared error (MSE), learned perceptual image patch similarity (LPIPS), peak signal-to-noise ratio (PSNR), structural similarity index (SSIM), and attack success rate (ASR). ↑ indicates higher is better; ↓ indicates lower is better.

TABLE III
PERFORMANCE AND PRIVACY METRICS ON CIFAR-100

Metric*	DMLP / LeNet		
	w/o	PRECODE	VAHANA
trn_acc (%) ↑	59.50 / 66.20	80.70 / 59.10	81.90 / 44.80
tst_acc (%) ↑	19.40 / 22.70	19.80 / 26.70	19.90 / 26.20
MSE ↑	0.00 / 0.03	4.01 / 4.02	4.12 / 4.03
LPIPS ↑	0.00 / 0.01	0.43 / 0.41	0.40 / 0.40
PSNR ↓	93.84 / 26.91	5.49 / 5.47	5.39 / 5.45
SSIM ↓	1.00 / 0.85	0.00 / 0.00	0.00 / 0.00
ASR (%) ↓	100.00 / 100.00	0.00 / 0.00	0.00 / 0.00

* ↑ indicates higher is better; ↓ indicates lower is better.

TABLE IV
PERFORMANCE AND PRIVACY METRICS ON MEDICAL-MNIST

Metric*	DMLP / LeNet		
	w/o	PRECODE	VAHANA
trn_acc (%) ↑	99.80 / 99.90	99.90 / 99.90	99.60 / 99.80
tst_acc (%) ↑	99.70 / 99.90	99.70 / 99.90	99.40 / 99.80
MSE ↑	0.04 / 0.40	8.50 / 6.99	8.90 / 7.79
LPIPS ↑	0.43 / 0.24	1.20 / 1.14	1.19 / 1.16
PSNR ↓	28.38 / 28.53	3.34 / 4.20	3.14 / 3.72
SSIM ↓	0.24 / 0.26	0.00 / 0.00	0.00 / 0.00
ASR (%) ↓	100.00 / 100.00	0.00 / 0.00	0.00 / 0.00

* ↑ indicates higher is better; ↓ indicates lower is better.

The unprotected models yield visually recognizable reconstructions, indicating substantial leakage of private training samples. By contrast, PRECODE and VAHANA suppress semantic content and degrade reconstructions into noise-like outputs, preventing meaningful information recovery.

We further quantify reconstruction fidelity using standard complementary metrics [11]. Across all datasets and backbones, the unprotected baselines exhibit structural similarity index (SSIM) values near 1.0 and 100% attack success rate (ASR), confirming high susceptibility to gradient inversion. Both PRECODE and VAHANA reduce SSIM to near zero and drive ASR to 0%, demonstrating effective empirical protection under the evaluated attack setting. On CIFAR-10 (Table II), VAHANA achieves slightly higher mean squared error (MSE) and learned perceptual image patch similarity (LPIPS) than PRECODE, suggesting stronger perceptual disruption. On CIFAR-100 (Table III), VAHANA matches or improves MSE and peak signal-to-noise ratio (PSNR) while

consistently maintaining 0% ASR. For Medical MNIST (Table IV), VAHANA remains comparable to PRECODE with marginal gains in MSE and LPIPS. Importantly, model utility is preserved: test accuracy remains stable across CIFAR-10, CIFAR-100, and Medical MNIST. These results indicate that VAHANA strengthens privacy against gradient inversion without sacrificing learning performance.

C. Hardware Performance Analysis

We implement VAHANA on a Kintex-7 FPGA (XC7K160T-1FBG676C) to quantify the hardware efficiency of our approximate reparameterization relative to the PRECODE baseline. All designs are synthesized and implemented using Xilinx Vivado 2022.2 with identical tool settings. We use Verilator⁴ for cycle-accurate profiling of the reparameterization path and to extract software memory footprints of RISC-V benchmarks [40].

Table V summarizes the cycle cost of the reparameterization pipeline under two entropy sources: the software `glibc` PRNG and the Trivium32 stream cipher. For the baseline Gaussian VIB, latency is dominated by RNG, resulting in 1,264 and 3,887 cycles per reparameterization step for `glibc` and Trivium32, respectively. By contrast, VAHANA replaces Gaussian sampling with a 4-bit uniform source and lightweight sign/exponent/mantissa logic, cutting reparameterization latency to 45 cycles (`glibc`) and 188 cycles (Trivium32). This corresponds to $28.1\times$ and $20.7\times$ end-to-end speedups at latent width $k=256$. The slight increase in FMA latency stems from using an R4-type custom instruction instead of the baseline R3-type FMA, but this overhead is negligible relative to the savings from eliminating Gaussian generation.

Table VI reports the memory costs of deploying Gaussian versus approximate reparameterization on CV32E40P. The Gaussian baseline requires larger Flash/ROM budgets due to the sampling routines and supporting arithmetic, totaling 28.59 KiB with `glibc` and 29.05 KiB with Trivium32. VAHANA's approximate path reduces total ROM to 18.45 KiB and 19.12 KiB, corresponding to savings of 10.14 KiB (35.5%) and 9.93 KiB (34.2%), respectively. RAM usage is effectively unchanged across configurations, indicating that VAHANA's gains are driven by a smaller code footprint rather than shifting

⁴An open-source Verilog/SystemVerilog simulation and compilation framework that translates RTL designs into executable C++/SystemC models.

TABLE V
LATENCY BREAKDOWN FOR VAHANA AND VIB USING GLIBC AND TRIVIUM32 AS ENTROPY SOURCES

Configuration	RNG	FMA	Latency ($k=1$)	Latency ($k=256$)
VAHANA / glibc	26	19	45	11,520
VIB / glibc	1,249	14	1,264	323,584
Savings ($\times$)	48.04	0.73	28.09	28.09
VAHANA / Trivium	169	19	188	48,128
VIB / Trivium	3,873	14	3,887	995,072
Savings ($\times$)	22.92	0.73	20.67	20.67

TABLE VI
MEMORY FOOTPRINT ANALYSIS FOR VAHANA AND VIB (KiB)

Configuration (Benchmark / RNG)	Flash/ROM		RAM	
	Code	Data	Shared	Heap/Stack
VAHANA / glibc	13.71	4.74	4,077.47	
VAHANA / Trivium	14.36	4.76	4,076.75	
VIB / glibc	23.46	5.13	4,067.33	
VIB / Trivium	23.92	5.14	4,066.75	

overhead to runtime memory. A software⁵-only VAHANA implementation would retain the 26-cycle glibc PRNG call and baseline floating-point addition, while adding scalar operations for Hamming-weight computation, control generation, sign/exponent/mantissa handling, and floating-point packing/unpacking. Thus, the 45-cycle latency in Table VI reflects the measured hardware realization of the full perturbation pipeline, not merely the replacement of Gaussian sampling.

Table VII lists the resource footprint of the CV32E40P core on Kintex-7. VAHANA is integrated by reusing the existing floating-point pipeline, introducing only lightweight front-end randomizer and control logic, avoiding a standalone Gaussian RNG datapath or full-precision mantissa multiplier. This design choice aligns with the constraints of edge-FL accelerators, where dedicated DSP-heavy sampling hardware is often unavailable or prohibitively expensive. Taken together, the FPGA and cycle-accurate results show that VAHANA delivers order-of-magnitude latency reductions and ~ 34 -36% ROM savings over PRECODE while adhering to the structural constraints of an edge-oriented RISC-V core. These measurements isolate the hardware cost of reparameterization itself and motivate VAHANA as a practical, hardware-friendly primitive for gradient obfuscation on resource-constrained platforms.

VI. CONCLUSION AND FUTURE WORK

This work presented VAHANA, a hardware-efficient alternative to PRECODE’s Gaussian reparameterization for defending FL against gradient inversion. By replacing Gaussian sampling and FP mantissa multiplication with a 4-bit uniform entropy source and lightweight approximate logic, implemented as a custom instruction on a CV32E40P-class RISC-V core, VAHANA preserves accuracy and matches PRECODE’s empirical robustness against the evaluated gradient inversion attacks across CIFAR-10, CIFAR-100, and Medical-MNIST. On a Kintex-7 FPGA prototype, VAHANA achieves

⁵A cycle-accurate software-only VAHANA implementation and baseline comparison are left as future work.

TABLE VII
AREA CONSUMPTION FOR VAHANA AND VIB ON KINTEX-7 FPGA

Configuration	LUT	FF	BRAM	DSP
VIB (no pipeline)	723	0	0	0
VAHANA (4-stage pipeline)	126	41	0	2
CV32E40P core	13,348	6,492	261.5	7

up to $28.1\times$ latency reduction and a 34% smaller ROM footprint, demonstrating that approximate computing can serve as a practical primitive for privacy-preserving edge learning. Future work includes developing formal privacy guarantees for VAHANA, evaluating scalability on larger architectures, studying adaptive attacks that exploit its discrete perturbation and hardware structure, and reporting power/energy readings.

VII. ACKNOWLEDGMENT

This work was funded through the Office of Naval Research (ONR) grant N00014-23-1-2103 and the National Science Foundation (CNS-1943245, SaTC 2340856). The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. ChatGPT-5.5 was used to assist with editorial improvements.

REFERENCES

- [1] P. Kairouz, H. B. McMahan, B. Avent *et al.*, “Advances and open problems in federated learning,” *Found. Trends Mach. Learn.*, vol. 14, no. 1–2, p. 1–210, Jun. 2021.
- [2] K. A. Bonawitz, H. Eichner, W. Grieskamp *et al.*, “Towards federated learning at scale: system design,” in *SysML*, 2019.
- [3] B. McMahan, E. Moore, D. Ramage *et al.*, “Communication-efficient learning of deep networks from decentralized data,” in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.
- [4] W. Li, F. Milletari, D. Xu *et al.*, “Privacy-preserving federated brain tumour segmentation,” in *Machine Learning in Medical Imaging, Shenzhen, China*, 2019, pp. 133–141.
- [5] Q. Yang, Y. Liu, T. Chen *et al.*, “Federated machine learning: concept and applications,” *Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.
- [6] J. Geiping, H. Bauermeister, H. Dröge *et al.*, “Inverting gradients - how easy is it to break privacy in federated learning?” in *34th International Conference on Neural Information Processing Systems*, 2020.
- [7] B. Hitaj, G. Ateniese, and F. Perez-Cruz, “Deep models under the GAN: information leakage from collaborative deep learning,” in *Conference on Computer and Communications Security*, 2017, pp. 603–618.
- [8] K. Gupta and T. Ajanthan, “Improved gradient based adversarial attacks for quantized networks,” 2021.
- [9] K. Wei, J. Li, M. Ding *et al.*, “User-level privacy-preserving federated learning: analysis and performance optimization,” *Transactions on Mobile Computing*, vol. 21, no. 09, pp. 3388–3401, 2022.
- [10] J. Lin, C. Gan, and S. Han, “Defensive quantization: when efficiency meets robustness,” in *International Conference on Learning Representations*, 2019.
- [11] D. Scheliga, P. Mäder, and M. Seeland, “PRECODE - a generic model extension to prevent deep gradient leakage,” in *Winter Conference on Applications of Computer Vision*, 2022, pp. 1849–1858.
- [12] J. Han and M. Orshansky, “Approximate computing: an emerging paradigm for energy-efficient design,” in *2013 18th IEEE European test symposium (ETS)*. IEEE, 2013, pp. 1–6.
- [13] S. Venkataramani, A. Ranjan, K. Roy *et al.*, “AxNN: energy-efficient neuromorphic systems using approximate computing,” in *International Symposium on Low Power Electronics and Design*, 2014, pp. 27–32.
- [14] Microsoft Corporation, “About HoloLens 2,” <https://learn.microsoft.com/en-us/hololens/hololens2-hardware>, 2023, accessed: 2025-12-12.
- [15] L. Rao, N. Argaman, J. Zhuang *et al.*, “Display and optics architecture for Meta’s AR/VR development,” *IEEE Open Journal on Immersive Displays*, vol. 1, pp. 71–78, 2024.

- [16] P. Hübner, A. Hu, I. Peng *et al.*, “Apple vs. oranges: evaluating the Apple silicon M-series SoCs for HPC performance and efficiency,” 2025.
- [17] Qualcomm Technologies, Inc., “Qualcomm AI engine direct SDK,” 2024, accessed: 2025-12-12. [Online]. Available: <https://www.qualcomm.com/developer/software/qualcomm-ai-engine-direct-sdk>
- [18] E. Talpes, D. D. Sarma, G. Venkataramanan *et al.*, “Compute solution for Tesla’s full self-driving computer,” *IEEE Micro*, 2020.
- [19] M. Nasr, R. Shokri, and A. Houmansadr, “Comprehensive privacy analysis of deep learning: passive and active white-box inference attacks against centralized and federated learning,” in *Symposium on Security and Privacy (SP)*, 2019, pp. 739–753.
- [20] M. Abadi, A. Chu, I. Goodfellow *et al.*, “Deep learning with differential privacy,” in *Conference on Computer and Communications Security*, 2016, pp. 308–318.
- [21] R. L. Rivest, L. Adleman, M. L. Dertouzos *et al.*, “On data banks and privacy homomorphisms,” *Foundations of Secure Computation*, vol. 4, no. 11, pp. 169–180, 1978.
- [22] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *International Conference on the Theory and Applications of Cryptographic Techniques*, 1999, pp. 223–238.
- [23] A. F. Aji and K. Heafield, “Sparse communication for distributed gradient descent,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, Sep. 2017. [Online]. Available: <https://aclanthology.org/D17-1045>
- [24] C. Dwork, F. McSherry, K. Nissim *et al.*, “Calibrating noise to sensitivity in private data analysis,” in *Third Theory of Cryptography Conference*, New York, NY, USA, 2006, pp. 265–284.
- [25] A. A. Alemi, I. Fischer, J. V. Dillon *et al.*, “Deep variational information bottleneck,” in *International Conference on Learning Representations*, 2017. [Online]. Available: <https://openreview.net/forum?id=HyxQzBceg>
- [26] V. Mrazek, S. S. Sarwar, L. Sekanina *et al.*, “Design of power-efficient approximate multipliers for approximate artificial neural networks,” in *International Conference on Computer-Aided Design*, 2016, pp. 1–7.
- [27] A. Guesmi, I. Alouani, K. N. Khasawneh *et al.*, “Defending with errors: approximate computing for robustness of deep neural networks,” *arXiv preprint arXiv:2211.01182*, 2022.
- [28] V. Gupta, D. Mohapatra, A. Raghunathan *et al.*, “Low-power digital signal processing using approximate adders,” *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2013.
- [29] Y. C. Bian, J. Wang, W. gong Zhang *et al.*, “DA2F: research on robustness of deep learning models using approximate activation function,” *Conference on High Performance Computing & Communications (HPCC)*, pp. 412–419, 2022.
- [30] M. H. Samavatian, S. Majumdar, K. Barber *et al.*, “DNNShield: dynamic randomized model sparsification, a defense against adversarial machine learning,” *CoRR*, 2022.
- [31] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” in *2nd International Conference on Learning Representations, Canada*, 2013.
- [32] D. J. Rezende, S. Mohamed, and D. Wierstra, “Stochastic backpropagation and approximate inference in deep generative models,” in *International Conference on Machine Learning*, 2014, pp. 1278–1286.
- [33] A. van den Oord, O. Vinyals, and K. Kavukcuoglu, “Neural discrete representation learning,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6309–6318.
- [34] A. Waterman and K. Asanović, “The RISC-V instruction set manual, volume I: unprivileged specification,” <https://github.com/riscv/riscv-isa-manual>, RISC-V International, 2024.
- [35] A. Dubey, R. Cammarota, A. Varna *et al.*, “Hardware-software co-design for side-channel protected neural network inference,” in *2023 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2023, pp. 155–166.
- [36] M. G. et.al, “OpenHW group CORE-V CV32E40P RISC-V IP.” [Online]. Available: <https://github.com/openhwgroup/cv32e40p>
- [37] M. Gautschi, P. D. Schiavone, A. Traber *et al.*, “Near-threshold RISC-V Core with DSP extensions for scalable IoT endpoint devices,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 10, pp. 2700–2713, 2017.
- [38] S. Mach, F. Schuiki, F. Zaruba *et al.*, “FPnew: An open-source multi-format floating-point unit architecture for energy-proportional transprecision computing,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 774–787, 2021.
- [39] C. De Cannière and B. Preneel, *Trivium*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 244–266. [Online]. Available: <https://doi.org/10.1007/978-3-540-68351-3%5F18>
- [40] W. Snyder, “Verilator and systemperl,” in *North American SystemC Users’ Group, Design Automation Conference*, vol. 79, 2004.