

FiCABU: A Fisher-Based, Context-Adaptive Machine Unlearning Processor for Edge AI

Eun-Su Cho^{†‡}, Jongin Choi^{†‡}, Jeongmin Jin[†], Jae-Jin Lee[‡] and Woojoo Lee^{†*}

[†]Department of Intelligent Semiconductor Engineering, Chung-Ang University, Seoul, Korea

[‡]AI Edge SoC Research Section, Electronics and Telecommunications Research Institute, Daejeon, Korea

Abstract—Machine unlearning, driven by privacy regulations and the “right to be forgotten,” is increasingly needed at the edge, yet server-centric or retraining-heavy methods are impractical under tight computation and energy budgets. We present *FiCABU* (Fisher-based Context-Adaptive Balanced Unlearning), a SW–HW co-design that brings unlearning to edge AI processors. *FiCABU* combines (i) *Context-Adaptive Unlearning*, which begins edits from back-end layers and halts once the target forgetting is reached, with (ii) *Balanced Dampening*, which scales dampening strength by depth to preserve retain accuracy. These methods are realized in a full RTL design of a RISC-V edge AI processor that integrates two lightweight IPs for Fisher estimation and dampening into a GEMM-centric streaming pipeline, validated on an FPGA prototype and synthesized in 45 nm for power analysis. Across CIFAR-20 and PinsFaceRecognition with ResNet-18 and ViT, *FiCABU* achieves random-guess forget accuracy while matching the retraining-free Selective Synaptic Dampening (SSD) baseline on retain accuracy, reducing computation by up to 87.52% (ResNet-18) and 71.03% (ViT). On the INT8 hardware prototype, *FiCABU* further improves retain preservation and reduces energy to 6.48% (CIFAR-20) and 0.13% (PinsFaceRecognition) of the SSD baseline. In sum, *FiCABU* demonstrates that back-end–first, depth-aware unlearning can be made both practical and efficient for resource-constrained edge AI devices.

I. INTRODUCTION

As machine learning models are deployed across increasingly diverse domains, the scale of training corpora continues to grow [1]. These corpora often contain sensitive personal information (e.g., biometrics), and legal as well as societal pressures around the “right to be forgotten” are mounting [2], [3]. Beyond removing samples from a dataset, a deployed model must behave *as if* specified data never influenced its parameters. This motivates *machine unlearning*—removing the effect of designated data from a pre-trained model at the parameter level [4]. Unlearning is valuable not only for compliance but also for maintaining model quality by attenuating the influence of erroneous or stale information [5], [6].

Most prior work studies unlearning under server-class assumptions—data collected on edge devices (e.g., IoT sensors, embedded cameras, or wearable health monitors) are uploaded, and unlearning is executed in the cloud [7]. However, edge compute capabilities have improved substantially [8], and—critically—the party that collects the data to be deleted is often the edge device itself. This raises a natural question: should unlearning be performed *on-device* instead of in the cloud? As illustrated in Fig. 1, cloud-based unlearning incurs communication latency/energy overheads and additional

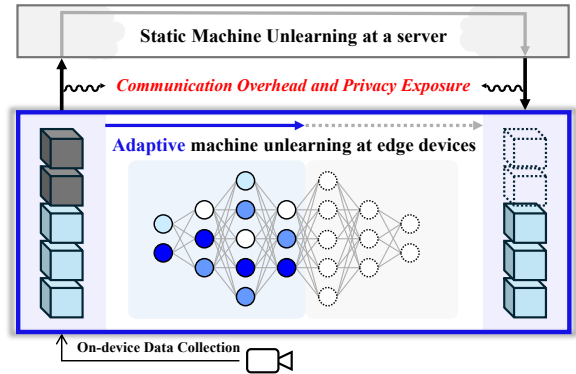


Fig. 1: Comparison of server-based unlearning, which incurs communication and privacy costs, and adaptive on-device unlearning, which performs selective updates locally.

privacy exposure, whereas executing unlearning where the data originate mitigates both concerns at their source [9]. At the same time, edge platforms remain resource-constrained relative to servers; thus, a practical on-device solution must be carefully tailored to edge budgets. Inference on the edge has long embraced *adaptive/selective processing*—doing useful work while avoiding unnecessary computation [10], [11]. We argue that on-device unlearning should adopt a similar ethos.

We therefore begin by examining algorithmic options for unlearning under edge constraints. The most definitive approach—retraining a model from scratch after excising the forget samples—guarantees removal but is prohibitively expensive for modern models [12]. Partitioned training lowers some costs yet still depends on (re)training [13]. Structure- or precision-oriented edits (e.g., pruning, quantization) simplify parameters encoding fine-grained characteristics of the forget set, but typically require additional training to restore utility and remain heavyweight on the edge [14]–[17]. More recently, retraining-free methods estimate parameter importance and update only those most attributable to the forget set. A prominent line leverages the diagonal approximation of the Fisher Information Matrix (*FIM*), which estimates sensitivity from squared gradients in a single forward-loss evaluation, avoiding repeated dataset passes [18]. This direction progressed from early ad hoc formulations [19] to post-hoc edits on pre-trained models [20] and zero-shot variants [21], alongside recent gradient- and score-based unlearning studies [22]–[24]. Selective Synaptic Dampening (*SSD*) embodies these advantages: it compares per-parameter importance for the forget set versus overall data and applies one-shot dampening only to selected parameters, achieving strong unlearning with minimal compute and memory traffic [18]. This intuition aligns with class-conditional attributions showing heterogeneous parameter contributions across classes [25].

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grants funded by the Korea government (MSIT) (No. RS-2023-00277060), and in part by the National Research Foundation of Korea (NRF) grant funded by MSIT (No. RS-2024-00345668).

[‡] Eun-Su Cho and Jongin Choi contributed equally to this work.

*Woojoo Lee is the corresponding author.

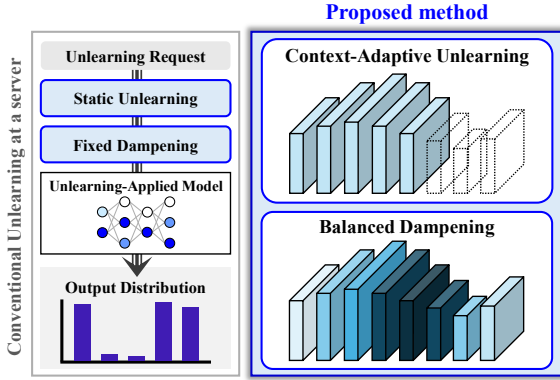


Fig. 2: Overview of FiCABU: context-adaptive unlearning, balanced dampening, and edge-oriented implementation.

Despite its promise, SSD—and existing unlearning methods more broadly—remain misaligned with edge realities. First, most techniques treat layers uniformly, even though fine-grained, class-specific features concentrate in later (back-end) layers across both Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) [26], [27]. Second, layer-agnostic hyperparameters create a tension between sufficiently suppressing back-end, class-specific detail and preserving front-end, general features, which can destabilize accuracy on the retain set. Third, the literature has not yet brought the *adaptive/selective* mindset—central to edge inference [10], [11]—into the unlearning procedure itself. As a result, unnecessary computation persists, and opportunities to save energy are left on the table.

To close these gaps, we present **FiCABU**—*Fisher-based Context-Adaptive Balanced Unlearning*—and a SW–HW co-designed processor for on-device unlearning (Fig. 2). Our approach embraces adaptive processing to remove useless work and balances layer-wise edits to protect retained performance:

- 1) **Context-Adaptive Unlearning.** Starting from the back-end, FiCABU applies unlearning layer by layer and evaluates forget accuracy at user-defined checkpoints. Once the target unlearning level (random-guess accuracy for the task) is reached, FiCABU halts further updates on front-end layers. To minimize verification overhead, activations from the initial forward pass are cached and reused for partial inference at checkpoints. This early-stop mechanism reduces Multiply–Accumulate operations (MACs)—a hardware-relevant proxy for computation [28]—by an average of 87.52%.
- 2) **Balanced Dampening.** Instead of fixed, layer-agnostic hyperparameters, FiCABU scales selection and dampening strength by a depth-dependent schedule, assigning stronger edits to back-end layers (where class-specific detail resides) while preserving front-end representations. In our experiments, this maintains unlearning efficacy while improving retain-set accuracy relative to uniform dampening.
- 3) **Edge Unlearning Processor.** We realize FiCABU as a *full-custom RISC-V edge AI processor* integrating a streaming General Matrix–Matrix Multiplication (GEMM) backbone with specialized IPs for importance estimation and dampening; these stages execute concurrently within edge budgets. The processor is implemented at RTL, prototyped on a 50 MHz Kintex-7 FPGA, and synthesized in 45 nm for power evaluation.

Experimental results across CIFAR-20 (ResNet-18, ViT) and

PinsFaceRecognition (ResNet-18) show that FiCABU achieves random-guess forget accuracy while improving retain preservation via (i) back-end-first *Context-Adaptive Unlearning* and (ii) depth-aware *Balanced Dampening*. Representative figures include MACs reduced by 87.52% on ResNet-18/CIFAR-20 and by 71.03% on ViT/CIFAR-20 (avg.). On our INT8 hardware prototype, system energy drops to 6.48% (CIFAR-20) and 0.13% (PinsFaceRecognition) of the SSD baseline. Taken together, these results demonstrate that FiCABU’s method–processor co-design makes adaptive, depth-aware unlearning practical and energy-efficient on resource-constrained edge AI processors.

II. SSD: METHOD AND LIMITATIONS

SSD [18] is a recent unlearning method that removes the influence of designated data without retraining. By leveraging the diagonal approximation of the FIM and applying a one-shot dampening only to parameters strongly associated with the forget set, SSD achieves effective unlearning with substantially reduced computation and energy, making it attractive for resource-constrained edge scenarios.

As a standard formulation in machine unlearning, the full training dataset \mathcal{D} is partitioned into a forget set \mathcal{D}_f and a retain set \mathcal{D}_r , where accuracies measured on \mathcal{D}_f and \mathcal{D}_r are termed *forget accuracy* and *retain accuracy*, respectively:

$$\mathcal{D}_f \subset \mathcal{D}, \quad \mathcal{D}_r = \mathcal{D} \setminus \mathcal{D}_f. \quad (1)$$

Parameters contribute heterogeneously to class predictions: the gradient response of a parameter to inputs from one class can differ markedly from its response to other classes [25]. This observation supports unlearning via *selective* parameter edits guided by importance scores computed on \mathcal{D}_f . In retraining-free unlearning, the diagonal Fisher is widely used because squared first-order gradients provide a lightweight sensitivity proxy and often suffice without constructing the full matrix [29]. The per-parameter diagonal Fisher on \mathcal{D}_f is

$$[\mathbb{F}_{\mathcal{D}_f}]_{ii} = \mathbb{E} \left[\left(\frac{\partial \ln p(\mathcal{D}_f | \theta)}{\partial \theta_i} \right)^2 \right], \quad \forall i \in [1, |\theta|], \quad (2)$$

and we denote the importance as $\mathbb{I}_{\mathcal{D}_f, i} \triangleq [\mathbb{F}_{\mathcal{D}_f}]_{ii}$. Because this requires only a single forward pass and the associated gradient computation to obtain the needed sensitivities, this avoids repeated full-dataset passes and hundreds of update steps, which is critical under edge compute and memory budgets.

SSD proceeds in two steps: *selection* chooses parameters to modify based on their relative importance, and *dampening* scales the chosen parameters by a factor $\beta \in (0, 1]$. Simply pruning parameters by thresholding importance at zero would reduce forget accuracy but severely harm retain accuracy and discard graded control over parameter impact. Instead, SSD compares importance scores on the forget set against those on the overall data and applies *dampening* rather than hard pruning:

$$\theta_i = \begin{cases} \beta \theta_i, & \text{if } \mathbb{I}_{\mathcal{D}_f, i} > \alpha \mathbb{I}_{\mathcal{D}, i} \\ \theta_i, & \text{if } \mathbb{I}_{\mathcal{D}_f, i} \leq \alpha \mathbb{I}_{\mathcal{D}, i} \end{cases} \quad \forall i \in [1, |\theta|], \quad (3)$$

where the threshold α controls which parameters are selected for modification. The dampening strength is

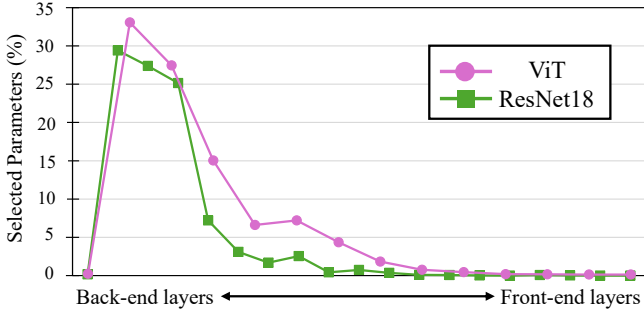


Fig. 3: Layer-wise distribution of selected parameters for ResNet-18 (RN) and ViT, highlighting concentration in back-end layers.

$$\beta = \min\left(\frac{\lambda \mathbb{I}_{\mathcal{D},i}}{\mathbb{I}_{\mathcal{D}_f,i}}, 1\right), \quad (4)$$

where λ weighs the relative importance of \mathcal{D} versus \mathcal{D}_f ; smaller λ emphasizes \mathcal{D}_f and induces stronger dampening.

In the selection step of (3), SSD uses $\mathbb{I}_{\mathcal{D}}$ rather than $\mathbb{I}_{\mathcal{D}_r}$ to avoid recomputing retain-set importance scores whenever a new forget request slightly perturbs \mathcal{D}_r . Since $\mathbb{I}_{\mathcal{D}}$ can be computed once after training and stored, large-scale access to the original training data is unnecessary during unlearning. Moreover, because typically $|\mathcal{D}_f| \ll |\mathcal{D}|$ and only about 1% of parameters are modified per unlearning event [18], the difference between $\mathbb{I}_{\mathcal{D}}$ and $\mathbb{I}_{\mathcal{D}_r}$ is practically negligible.

Empirically, SSD can reduce forget accuracy to random-guess levels while maintaining performance comparable to retraining-based approaches. At that operating point, two limitations remain relevant for edge deployment. First, selection relies on a single threshold applied uniformly across all parameters, without explicitly prioritizing back-end layers where fine-grained, class-specific information is concentrated. Second, hyperparameters are layer-agnostic: despite targeting parameters that encode the detailed traits of \mathcal{D}_f , the dampening schedule does not adapt to depth, leaving retain accuracy sensitive to over-/under-editing across layers.

In this work, we take as our operational point the setting where SSD reduces forget accuracy to random-guess levels, which we regard as the practical target for on-device unlearning. We therefore consider classes that satisfy this criterion. Our baseline SSD configuration follows the original setup on CIFAR-20 [30]: for ResNet-18 [31], $(\alpha, \lambda) = (10, 1)$; for ViT [32], $(\alpha, \lambda) = (25, 1)$. For PinsFaceRecognition [33], where inter-class similarity is relatively high, we adopt $(\alpha, \lambda) = (50, 0.1)$ as in [18]. The batch size for the forget class is fixed at $N = 64$ across all cases. All subsequent results build on these settings unless otherwise specified.

III. FiCABU: PROPOSED METHOD

The proposed FiCABU method combines two complementary techniques—*Context-Adaptive Unlearning* and *Balanced Dampening*. The former cuts computation by avoiding unnecessary edits on front-end layers, and the latter stabilizes retain-set performance by assigning depth-aware unlearning strength.

A. Context-Adaptive Unlearning

Empirical evidence indicates that fine-grained, class-specific features concentrate in later (back-end) layers. This trend is evident in Fig. 3, which shows the layer-wise distribution of parameters selected during unlearning for ResNet-18 (RN) and

Algorithm 1: Context-Adaptive Unlearning

Input : Pre-trained parameters θ ; forget mini-batch \mathcal{D}_f of size N ; checkpoint set $\mathcal{C} \subseteq \{1, \dots, L\}$; target τ ; stored global importance $\{\mathbb{I}_{\mathcal{D},i}\}$; SSD hyperparameters (α, λ) .

Output: Updated parameters $\tilde{\theta}$.

// Step 0: one forward pass on \mathcal{D}_f and cache inputs to layer l at checkpoints

for $n = 1$ **to** N **do**

run forward on sample n and cache activation $[l, n]$ for each $l \in \mathcal{C}$

// activation $[l, n]$ is the input tensor to l

Function Context_Adaptive_Unlearning(\mathcal{D}_f):

for $l = 1$ **to** L **do**

// Layer update (SSD) on layer l ((3), (4))

// Diagonal Fisher on \mathcal{D}_f : $\mathbb{I}_{\mathcal{D}_f,i} \leftarrow [\mathbb{F}_{\mathcal{D}_f}]_{ii}$

compute $\mathbb{I}_{\mathcal{D}_f,i}$ for all params in layer l ;

mark $\text{Sel}_l = \{i : \mathbb{I}_{\mathcal{D}_f,i} > \alpha \mathbb{I}_{\mathcal{D},i}\}$ // selection

for $i \in \text{Sel}_l$ **do**

$\beta_i \leftarrow \min(\lambda \mathbb{I}_{\mathcal{D},i} / \mathbb{I}_{\mathcal{D}_f,i}, 1)$;

$\theta_i \leftarrow \beta_i \theta_i$ // in-place dampening

if $l \in \mathcal{C}$ **then**

$A_{\text{forget}} \leftarrow \text{partial_inference}(\mathcal{D}_f, l)$

if $A_{\text{forget}} \leq \tau$ **then**

break // leave $l+1, \dots, L$ untouched

return $\tilde{\theta} \leftarrow \theta$

Function partial_inference(\mathcal{D}_f, l):

$A_{\text{sum}} \leftarrow 0$

for $n = 1$ **to** N **do**

$\text{act_current} \leftarrow \text{activation}[l, n]$

$A_{\text{temp}} \leftarrow \text{forward}(\text{act_current}, \text{from} = l, \text{to} = 1)$

// partial inference $l \rightarrow 1$ (toward back-end)

$A_{\text{sum}} \leftarrow A_{\text{sum}} + A_{\text{temp}}$

return $\frac{A_{\text{sum}}}{N}$ // batch-mean forget accuracy

ViT; aside from the final fully connected classifier (which has fewer parameters than convolutional layers), a clear depth-dependent pattern emerges. Unless otherwise noted, we index layers from the back-end (near the classifier) to the front-end (near the input): $l=1$ denotes the last/output layer and $l=L$ the first/input layer.

Building on this observation, we introduce *Context-Adaptive Unlearning*, an edge-friendly procedure that (i) evaluates the forget objective at intermediate checkpoints and (ii) halts further unlearning once the target is achieved, thereby saving computation on remaining front-end layers. However, verifying intermediate performance normally requires launching new forward passes, which can incur nontrivial overhead. To mitigate this, we cache activations from the initial forward pass and reuse them for partial inference when evaluating checkpoints.

Concretely, SSD performs a single forward pass on \mathcal{D}_f , then iterates from the back-end toward the front-end, computing diagonal-Fisher importance and applying dampening layer by layer. This sequential process can waste work on front-end layers once sufficient forgetting has already been achieved. Our procedure introduces a *checkpoint set* \mathcal{C} along depth, as detailed in Algorithm 1. Let layers be indexed $l = 1, \dots, L$ from back-end to front-end. Starting at $l = 1$ and increasing toward L , Context_Adaptive_Unlearning computes $\mathbb{I}_{\mathcal{D}_f,i}$ for each layer, compares it with $\mathbb{I}_{\mathcal{D},i}$, and applies in-place dampening. If $l \in \mathcal{C}$, the routine calls `partial_inference` using cached `activation[l, n]` as inputs and runs only layers $l \rightarrow 1$ current layer down to the back-end) to compute the batch-mean forget accuracy A_{forget} . If $A_{\text{forget}} \leq \tau$ (the target

TABLE I: Context-Adaptive Unlearning vs. baseline (pre-trained model without unlearning) and SSD. Rows for \mathcal{D}_r and \mathcal{D}_f report accuracy on the respective sets. All values are in percent [%].

(a) CIFAR-20 results: unlearning performance of RN and ViT on Rocket and MR classes, and the average across the remaining classes.

| Class | Metric | RN | | | ViT | | |
|--------|-----------------|----------|-------|--------------|----------|-------|--------------|
| | | Baseline | SSD | Ours | Baseline | SSD | Ours |
| Rocket | \mathcal{D}_r | 96.95 | 96.14 | 96.50 | 94.94 | 94.91 | 94.93 |
| | \mathcal{D}_f | 97.60 | 3.80 | 3.80 | 95.00 | 3.20 | 5.00 |
| | MIA | 82.00 | 9.20 | 5.40 | 65.40 | 20.40 | 20.20 |
| | MACs | – | 100 | 32.59 | – | 100 | 74.47 |
| MR | \mathcal{D}_r | 96.94 | 95.35 | 95.46 | 94.50 | 94.32 | 94.33 |
| | \mathcal{D}_f | 98.40 | 1.20 | 3.20 | 96.80 | 1.80 | 3.20 |
| | MIA | 86.20 | 6.00 | 6.00 | 61.80 | 9.60 | 5.60 |
| | MACs | – | 100 | 32.59 | – | 100 | 33.10 |
| Avg. | \mathcal{D}_r | 96.95 | 92.28 | 93.07 | 93.72 | 91.22 | 91.86 |
| | \mathcal{D}_f | 97.31 | 0.71 | 0.86 | 83.25 | 1.26 | 2.98 |
| | MIA | 79.56 | 9.44 | 5.44 | 49.33 | 25.60 | 17.42 |
| | MACs | – | 100 | 12.48 | – | 100 | 28.97 |

(b) PinsFaceRecognition results: unlearning performance of RN, reported as the average across classes.

| Model | Metric | Baseline | SSD | Ours |
|-------|-----------------|----------|-------|----------------|
| RN | \mathcal{D}_r | 98.11 | 97.92 | 98.08 |
| | \mathcal{D}_f | 97.63 | 0.00 | 0.00 |
| | MIA | 41.61 | 0.46 | 0.11 |
| | MACs | – | 100 | 0.00137 |

random-guess level), unlearning stops and layers $l+1, \dots, L$ are left untouched; otherwise, the loop continues with $l+1$. This design (i) reduces MACs by avoiding noncontributory edits, (ii) prevents unnecessary parameter changes that could harm \mathcal{D}_r performance, and (iii) allows checkpoint placement to be tuned per model and budget. For consistency, we adopt the vanilla SSD hyperparameters (α, λ) here; a depth-aware variant will be introduced in Section III-B.

Table I summarizes results. For CIFAR-20 (Table I-a), we compare a pre-trained baseline model (no unlearning applied), SSD, and our method on RN and ViT—reporting individual results for *Rocket* and *Mushroom* (MR) and the average across remaining classes. For PinsFaceRecognition (Table I-b), we report class-averaged results on RN. Checkpoints are placed at the first and last layers for both models; additionally, for RN, we insert a checkpoint every four of the 16 convolutional layers, and for ViT every three of the 12 encoder layers. Metrics include retain-set accuracy on \mathcal{D}_r , forget-set accuracy on \mathcal{D}_f (desired to reach random-guess level), Membership Inference Attack (MIA) accuracy (lower is better), and the number of MACs. MACs are reported relative to SSD (normalized to 100%), and include the overhead of checkpoint evaluation.

Overall, Context-Adaptive Unlearning matches SSD on retain accuracy and MIA while reaching random-guess forget accuracy (5% on CIFAR-20; 1% on PinsFaceRecognition) across all evaluated cases. Using MACs that include checkpoint overhead, RN on CIFAR-20 shows identical early-stop checkpoints for *Rocket* and *Mushroom* with a 67.41% reduction, and an average reduction of 87.52% over other classes. ViT achieves 25.53% (Rocket), 66.90% (MR), and 71.03% on average for the rest. For PinsFaceRecognition with RN, the average MACs reduction reaches about 99.9%, which can plausibly be attributed to the higher inter-class similarity in face data, leading to a stronger concentration of discriminative detail

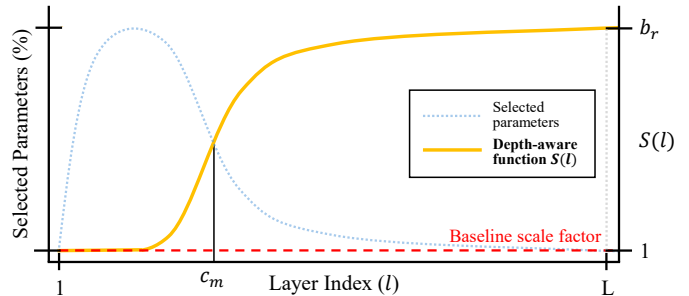


Fig. 4: Baseline uniform scaling vs. proposed sigmoid-based profile; $S(l)$ is smaller at the back-end and larger at the front-end.

TABLE II: Balanced Dampening vs. baseline and SSD. All values are in percent [%].

(a) CIFAR-20: Results for RN and ViT on Rocket, MR, and the average across remaining classes.

| Class | Metric | RN | | | ViT | | |
|--------|-----------------------|----------|-------|--------------|----------|-------|--------------|
| | | Baseline | SSD | Ours | Baseline | SSD | Ours |
| Rocket | \mathcal{D}_r | 96.95 | 96.14 | 96.25 | 94.94 | 94.91 | 94.92 |
| | \mathcal{D}_f | 97.60 | 3.80 | 3.80 | 95.00 | 3.20 | 3.60 |
| | $\Delta\mathcal{D}_r$ | – | 0.81 | 0.70 | – | 0.03 | 0.02 |
| | RPR | – | – | 13.58 | – | – | 33.33 |
| MR | \mathcal{D}_r | 96.94 | 95.35 | 95.37 | 94.50 | 94.32 | 94.35 |
| | \mathcal{D}_f | 98.40 | 1.20 | 1.20 | 96.80 | 1.80 | 3.00 |
| | $\Delta\mathcal{D}_r$ | – | 1.59 | 1.57 | – | 0.18 | 0.15 |
| | RPR | – | – | 1.26 | – | – | 16.67 |
| Avg. | \mathcal{D}_r | 96.95 | 92.28 | 92.58 | 93.72 | 91.22 | 91.61 |
| | \mathcal{D}_f | 97.31 | 0.71 | 0.73 | 83.25 | 1.26 | 2.67 |
| | $\Delta\mathcal{D}_r$ | – | 4.67 | 4.37 | – | 2.50 | 2.11 |
| | RPR | – | – | 6.42 | – | – | 15.60 |

(b) PinsFaceRecognition: Results for RN, reported as the class average.

| Model | Metric | Baseline | SSD | Ours |
|-------|-----------------------|----------|-------|--------------|
| RN | \mathcal{D}_r | 98.11 | 97.92 | 97.95 |
| | \mathcal{D}_f | 97.63 | 0.00 | 0.00 |
| | $\Delta\mathcal{D}_r$ | – | 0.19 | 0.16 |
| | RPR | – | – | 15.79 |

in later layers. In summary, the method substantially reduces computation while preserving unlearning quality—an essential property for an edge-targeted unlearning processor.

B. Balanced Dampening

While Context-Adaptive Unlearning removes unnecessary edits by early stopping, it does not address the second limitation of SSD: layer-agnostic hyperparameters. In (3), the threshold α determines selection (relative emphasis of \mathcal{D}_f vs. \mathcal{D}), and in (4), λ controls dampening strength. Smaller values induce stronger forgetting.

We generalize these scalars into a depth-aware profile so that the back-end layers receive relatively stronger edits (and the front-end layers, weaker ones):

$$(\alpha, \lambda) \rightarrow S(l) \cdot (\alpha, \lambda). \quad (5)$$

Motivated by reports that back-end layers encode more class-specific detail, we instantiate $S(l)$ as a sigmoid that is smaller at back-end layers (l small) and larger at front-end layers (l large). As illustrated in Fig. 4, the baseline scale factor (red dashed) is uniform across layers, whereas the proposed $S(l)$ (yellow) monotonically increases with l (i.e., it is smaller near the back-end and larger near the front-end), which qualitatively mirrors the observed distribution of selected parameters in reverse (blue

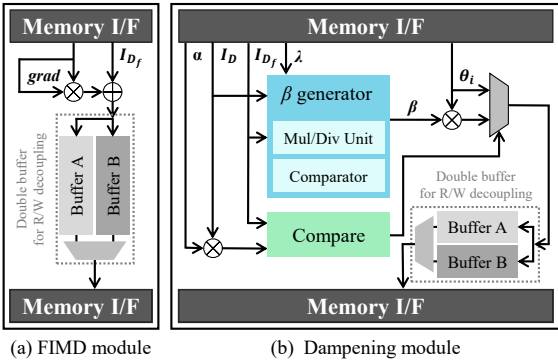


Fig. 5: Key hardware components of FiCABU: (a) FIMD module for diagonal Fisher estimation used by Context-Adaptive Unlearning, (b) Dampening module implementing Balanced Dampening updates, and (c) patch-level GEMM-FIMD-Dampening pipeline.

dotted). The midpoint c_m determines the transition region, and the bound b_r (“bound for retain”) controls the front-end scaling level. Formally, $S(l)$ is defined as

$$S(l) = 1 + (b_r - 1) \cdot \frac{\sigma(l) - \sigma(1)}{\sigma(L) - \sigma(1)}, \quad 1 \leq l \leq L, \quad (6)$$

where $\sigma(l) = \frac{1}{1 + \exp(-(l - c_m))}$. This replaces fixed scalars with a depth-aware function, enabling stronger dampening where class-specific features reside while protecting generic features in the front-end.

Table II quantitatively compares Balanced Dampening to the baseline and SSD under the same settings as Table I. For all experiments, we (i) compute the distribution of modified parameters under baseline SSD, (ii) smooth and center the sigmoid midpoint c_m at the mid-value between the smoothed extrema, and (iii) set $b_r = 10$. Metrics include forget/retain accuracy, the change in retain accuracy relative to baseline ($\Delta\mathcal{D}_r$), and the *Retain Preservation Rate* (RPR), which measures how well the retain accuracy is preserved:

$$\text{RPR} = \left(1 - \frac{\Delta\mathcal{D}_r^{\text{Ours}}}{\Delta\mathcal{D}_r^{\text{SSD}}}\right) \times 100 \quad (7)$$

where $\Delta\mathcal{D}_r^{\text{SSD}}$ and $\Delta\mathcal{D}_r^{\text{Ours}}$ denote the retain accuracy drops under SSD and under our method, respectively.

As reported in Table II-a, Balanced Dampening achieves random-guess forget accuracy comparable to SSD while consistently improving retain preservation (positive RPR across all cases). On CIFAR-20, RN yields RPR of 13.58 (Rocket), 1.26 (MR), and 6.42 on average across remaining classes; ViT yields 33.33, 16.67, and 15.60, respectively. For PinsFaceRecognition (Table II-b), RN achieves an average RPR of 15.79. In short, depth-aware scaling maintains unlearning efficacy while better safeguarding retain-set performance.

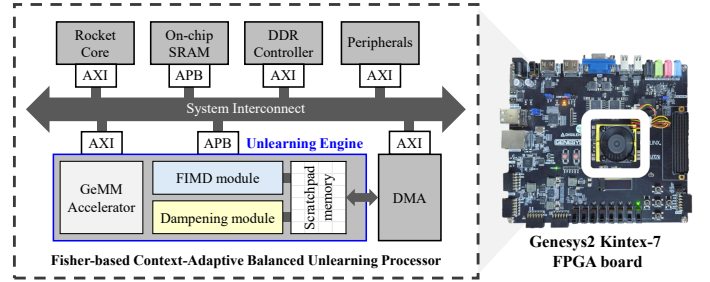


Fig. 6: FiCABU processor architecture (left) and FPGA prototype on a Genesys2 Kintex-7 board (right).

IV. IMPLEMENTATION AND EVALUATION

A. FiCABU Processor: Design and Prototyping

To realize on-device unlearning under edge constraints, we co-design FiCABU in software and hardware and integrate them into a lightweight processor. Because the dominant arithmetic in our workflow is matrix multiplication (forward-pass activations, gradient calculation), we adopt a dedicated GEMM accelerator as the backbone and attach two specialized IPs for (i) diagonal-Fisher approximation and (ii) dampening, as shown in Fig. 5.

GEMM-centric pipeline. During unlearning, the system performs a single forward pass over the forget batch to compute and cache layer-wise activations. Subsequent importance estimation and dampening reuse these cached activations together with back-propagated gradients, making GEMM the performance and energy hotspot. The GEMM engine executes matrix multiplications in fixed-size *patch* (tile) units and streams operands from memory; this streaming organization allows us to align the auxiliary IPs with the patch cadence.

Fisher Information Matrix Diagonal (FIMD) IP. The FIMD module (Fig. 5a) consumes the gradient outputs produced by the GEMM engine and implements the diagonal Fisher in (2) by squaring elements and accumulating them across the batch dimension to obtain $\mathbb{I}_{\mathcal{D}_f}$. To avoid read/write contention when accessing large parameter/gradient buffers, FIMD uses double buffering. Internally, it forms a four-stage pipeline—LOAD → SQUARE → ACCUMULATE → STORE—so that data transfers and arithmetic overlap. As a result, diagonal-Fisher computation that previously created a bottleneck when executed on the core is accelerated by $11.7\times$ on the FIMD module, hiding its latency within the GEMM patch window.

Dampening IP. The Dampening module (Fig. 5b) realizes the selection rule in (3) and the strength schedule in (4). For each parameter, it compares $\mathbb{I}_{\mathcal{D}_f,i}$ with $\alpha \mathbb{I}_{\mathcal{D},i}$, generates the per-parameter coefficient β via a dedicated β GENERATOR when selected, and updates the value by multiplication. Similar to FIMD, it employs a double-buffered datapath and a five-stage pipeline—LOAD → COMPARE → β CALC → MULTIPLY → STORE—interleaving memory traffic with computation. This organization eliminates the bottleneck of dampening observed on the core, achieving a $7.9\times$ speedup and ensuring completion within the GEMM patch window.

End-to-end streaming. By aligning the three IPs, FiCABU forms a three-stage, patch-level streaming pipeline—GEMM → FIMD → DAMPENING—that operates continuously at the GEMM patch rate (Fig. 5c). This organization hides the internal latency of the specialized IPs behind GEMM processing and ensures high throughput without enlarging on-chip buffers.

TABLE III: FPGA resource utilization of the FiCABU processor and power consumption estimated from 45 nm process technology.

| IPs | LUTs | FFs | P_{total} (mW) |
|----------------------------|---------------|--------------|------------------|
| FiCABU Processor | 71,535 | 35,059 | 185.89 |
| └ RISC-V Rocket Core | 15,246 | 9,756 | 11.2 |
| └ SRAM | 354 | 653 | 1.71 |
| └ DMA | 1,556 | 951 | 4.07 |
| └ Peripherals | 4,329 | 7,562 | 5.68 |
| └ DDR Controller | 8,102 | 7,514 | 88.62 |
| └ System Interconnect | 5,234 | 652 | 33.9 |
| └ Unlearning Engine | 36,714 | 7,971 | 40.71 |
| └ VTA | 34,529 | 7,186 | 39.9 |
| └ Specialized IPs | 2,185 | 785 | 0.81 |

Platform integration and prototyping. As shown in Fig. 6, we implemented the full-custom FiCABU processor at the RTL level on a RISC-V platform using RISC-V eXpress (RVX), an EDA environment that leverages open-source RISC-V cores for rapid processor design and customization [34]–[38]. The system comprises a 50 MHz RISC-V Rocket core [39], 64 KB on-chip SRAM, a DDR controller, peripherals, AXI DMA, and a μ NoC with AXI/APB interconnects. For GEMM, we integrate the open-source Versatile Tensor Accelerator (VTA) [40] for rapid system bring-up and reproducibility; exploring custom minimal GEMM designs is orthogonal to our contributions. The two specialized IPs (FIMD and Dampening) control a custom DMA that orchestrates bulk parameter transfers between main memory and the scratchpad, forming the *Unlearning Engine* with the GEMM accelerator. Unless noted otherwise, we target INT8 quantized models, consistent with common edge deployments. We prototype the processor on a Genesys2 Kintex-7 FPGA board and synthesize in a 45 nm technology using FreePDK45 for power estimation.

Resource and power breakdown. Table III summarizes the FPGA and ASIC-level costs. The FPGA prototype uses 71,535 LUTs and 35,059 FFs; the Unlearning Engine accounts for 36,714 LUTs and 7,971 FFs (51.3% and 22.7% of the totals, respectively). In ASIC estimates using Design Compiler, the Unlearning Engine draws 40.71 mW (21.9% of system power), while the DDR controller and VTA consume 88.62 mW (47.7%) and 39.90 mW (21.5%), respectively. Notably, the *Specialized IPs* (FIMD + Dampening) are lightweight, using only 2,185 LUTs (3.1%) and 785 FFs (2.2%), with 0.81 mW (0.44%) of power. Although off-chip memory and GEMM dominate the power budget—a typical characteristic of edge inference systems that store large models in DRAM—the proposed IPs add minimal overhead while enabling the streaming pipeline that sustains throughput at the GEMM rate. Putting these pieces together, the FiCABU processor sustains GEMM-rate throughput while incurring only negligible power/area overhead for the two specialized IPs. In the following subsection, we evaluate the developed FiCABU processor.

B. End-to-End Evaluation on the FiCABU Processor

To evaluate FiCABU end-to-end, we implemented the FiCABU processor and also prototyped a baseline processor that comprises the same components (e.g., GEMM accelerator, Rocket core, memory subsystem) but excludes the specialized unlearning IPs. The baseline method (pre-trained model without unlearning) and SSD were executed on this baseline processor for comparison. Unlike the FP32 simulations in Section III, all

TABLE IV: Unlearning performance and energy savings (ES) of the FiCABU processor running an INT8-quantized ResNet-18 on CIFAR-20 and PinsFaceRecognition. ES is measured relative to SSD on the baseline processor. All values are in percent [%].

| Metric | CIFAR-20 | | | PinsFaceRecognition | | |
|-----------------|----------|-------|--------------|---------------------|-------|---------------|
| | Baseline | SSD | FiCABU | Baseline | SSD | FiCABU |
| \mathcal{D}_r | 96.96 | 88.87 | 93.08 | 98.11 | 97.95 | 98.02 |
| \mathcal{D}_f | 96.40 | 0.41 | 1.21 | 97.70 | 0.00 | 0.00 |
| MACs | – | 100 | 11.88 | – | 100 | 0.0014 |
| RPR | – | – | 52.04 | – | – | 43.75 |
| ES | – | – | 93.52 | – | – | 99.87 |

experiments here used INT8 ResNet-18 models to reflect hardware deployment; the remaining settings followed Section III.

Table IV reports measured performance. Across both datasets, the FiCABU processor attains random-guess forget accuracy while reducing MACs by 88.11% (CIFAR-20) and 99.998% (PinsFaceRecognition) relative to SSD on the baseline processor. This confirms on hardware what Section III suggested in simulation: *Context-Adaptive Unlearning* can stop at back-end layers and still achieve the target forgetting, thereby realizing the context-adaptive processing capability essential for the edge. Retain accuracy also improves, yielding positive RPR in both cases; this indicates that applying depth-aware dampening per layer effectively preserves general performance on hardware, as in simulation. Compared to Table II, the RPR gains are stronger here because the evaluation combines *Balanced Dampening* with *Context-Adaptive Unlearning*, and the latter halts edits on front-end layers, preventing excessive dampening of general features.

Energy results are also summarized in Table IV. In addition to substantial MAC reductions (and the resulting runtime decrease), hardware assistance from the specialized IPs and patch-level pipelining, which consume only 0.44% of system power as reported in Table III, reduces system energy to 6.48% (CIFAR-20) and 0.13% (PinsFaceRecognition) of the SSD baseline. In summary, FiCABU preserves unlearning quality while driving energy to extremely low levels, firmly demonstrating that machine unlearning can be both feasible and efficient on resource-constrained edge AI processors.

V. CONCLUSION

We introduced *FiCABU*, a SW–HW co-design that enables practical unlearning on edge AI processors. FiCABU combines *Context-Adaptive Unlearning*, which begins edits from back-end layers and halts once the target forgetting is reached, with *Balanced Dampening*, which scales dampening strength by depth. These methods are realized in a full RTL design of an edge AI processor that integrates two lightweight IPs for Fisher estimation and dampening into a GEMM-centric streaming pipeline, validated on a FPGA prototype and synthesized in 45 nm for power analysis. Experiments on CIFAR-20 and PinsFaceRecognition show that FiCABU achieves random-guess forget accuracy while matching SSD on retain accuracy, with computation reduced by up to 87.52% (ResNet-18) and 71.03% (ViT). On the INT8 hardware prototype, FiCABU preserves these benefits and further improves retain preservation, while system-level energy drops to 6.48% (CIFAR-20) and 0.13% (PinsFaceRecognition) of the SSD baseline. In sum, FiCABU demonstrates that back-end–first, depth-aware unlearning can be made practical and efficient for resource-constrained edge AI devices.

REFERENCES

- [1] S. Yun, R. Masukawa, W. Y. Chung, M. Na, N. D. Bastian, and M. Imani, "Continuous gnn-based anomaly detection on edge using efficient adaptive knowledge graph learning," in *2025 Design, Automation & Test in Europe Conference (DATE)*, 2025, pp. 1–7.
- [2] P. Voigt and A. von dem Bussche, *The EU General Data Protection Regulation (GDPR): A Practical Guide. 1st Edition*. Springer International Publishing, 2017, see Chapter on Article 17: Right to erasure ('right to be forgotten').
- [3] E. Goldman, "An introduction to the california consumer privacy act (ccpa)," Santa Clara University School of Law, Legal Studies Research Paper, Tech. Rep., July 2020.
- [4] A. Huang, Z. Cai, and Z. Xiong, "A survey of machine unlearning in generative ai models: Methods, applications, security, and challenges," *IEEE Internet of Things Journal*, vol. 12, no. 16, pp. 32 563–32 580, 2025.
- [5] J. Xu, Z. Wu, C. Wang, and X. Jia, "Machine unlearning: Solutions and challenges," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 8, no. 3, pp. 2150–2168, 2024.
- [6] M. Kurmanji, P. Triantafyllou, J. Hayes, and E. Triantafyllou, "Towards unbounded machine unlearning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 1957–1987.
- [7] Y. Hu, J. Lou, J. Liu, W. Ni, F. Lin, Z. Qin, and K. Ren, "Eraser: Machine unlearning in mlaas via an inference serving-aware approach," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 3883–3897.
- [8] X. Xia, Z. Wang, R. Sun, B. Liu, I. Khalil, and M. Xue, "Edge unlearning is not "on edge"! an adaptive exact unlearning system on resource-constrained devices," in *2025 IEEE Symposium on Security and Privacy (SP)*, 2025, pp. 2546–2563.
- [9] S. P. Bayerl, T. Frassetto, P. Jauernig, K. Riedhammer, A.-R. Sadeghi, T. Schneider, E. Stapf, and C. Weinert, "Offline model guard: Secure and private ml on mobile devices," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020, pp. 460–465.
- [10] B. A. Motetti, L. Crupi, M. O. Mohammed Elamin Elshaigi, M. Risso, D. J. Pagliari, D. Palossi, and A. Burrello, "Adaptive deep learning for efficient visual pose estimation aboard ultra-low-power nano-drones," in *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2024, pp. 1–6.
- [11] G. Korol, M. G. Jordan, M. B. Rutzig, J. Castrillon, and A. C. S. Beck, "Pruning and early-exit co-optimization for CNN acceleration on FPGAs," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2023, pp. 1–6.
- [12] J. Yao, E. Chien, M. Du, X. Niu, T. Wang, Z. Cheng, and X. Yue, "Machine unlearning of pre-trained large language models," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins, and V. Srikumar, Eds. Bangkok, Thailand: Association for Computational Linguistics, Aug. 2024, pp. 8403–8419.
- [13] L. Bourtole, V. Chandrasekaran, C. A. Choquette-Choo, H. Jia, A. Travers, B. Zhang, D. Lie, and N. Papernot, "Machine unlearning," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 141–159.
- [14] J. Jia, J. Liu, P. Ram, Y. Yao, G. Liu, Y. Liu, P. SHARMA, and S. Liu, "Model sparsity can simplify machine unlearning," in *Advances in Neural Information Processing Systems*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36. Curran Associates, Inc., 2023, pp. 51 584–51 605.
- [15] Y. Tong, Y. Wang, J. Yuan, and C. Hu, "Robust machine unlearning for quantized neural networks via adaptive gradient reweighting with similar labels," 2025. [Online]. Available: <https://arxiv.org/abs/2503.13917>
- [16] J. Tu, L. Yang, and J. Cao, "Distributed machine learning in edge computing: Challenges, solutions and future directions," *ACM Comput. Surv.*, vol. 57, no. 5, Jan. 2025.
- [17] X. Wang, Z. Tang, J. Guo, T. Meng, C. Wang, T. Wang, and W. Jia, "Empowering edge intelligence: A comprehensive survey on on-device ai models," *ACM Comput. Surv.*, vol. 57, no. 9, Apr. 2025.
- [18] J. Foster, S. Schoepf, and A. Brintup, "Fast machine unlearning without retraining through selective synaptic dampening," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 38, no. 11, 2024, pp. 12 043–12 051.
- [19] C. Guo, T. Goldstein, A. Hannun, and L. Van Der Maaten, "Certified data removal from machine learning models," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. JMLR.org, 2020.
- [20] A. Golatkar, A. Achille, and S. Soatto, "Eternal sunshine of the spot-less net: Selective forgetting in deep networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [21] A. Sekhari, J. Acharya, G. Kamath, and A. T. Suresh, "Remember what you want to forget: algorithms for machine unlearning," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, Red Hook, NY, USA, 2021.
- [22] S. Cha, S. Cho, D. Hwang, and M. Lee, "Towards robust and parameter-efficient knowledge unlearning for LLMs," in *The Thirteenth International Conference on Learning Representations*, 2025.
- [23] C. Gao, L. Wang, K. Ding, C. Weng, X. Wang, and Q. Zhu, "On large language model continual unlearning," in *Proceedings of the 2024 Conference on Neural Information Processing Systems (NeurIPS)*, 2024.
- [24] C. Ding, J. Wu, Y. Yuan, J. Lu, K. Zhang, A. Su, X. Wang, and X. He, "Unified parameter-efficient unlearning for LLMs," in *The Thirteenth International Conference on Learning Representations*, 2025. [Online]. Available: <https://openreview.net/forum?id=zONMuIVCAT>
- [25] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.
- [26] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *Advances in neural information processing systems (NIPS)*, vol. 27, 2014.
- [27] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" in *Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS)*, ser. NIPS '21. Red Hook, NY, USA: Curran Associates Inc., 2021.
- [28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.
- [29] J. Shi, K. Gourgoulis, J. F. Buford, S. J. Moran, and N. Ghalyan, "DeepClean: Machine unlearning on the cheap by resetting privacy sensitive weights using the fisher diagonal," in *Computer Vision – ECCV 2024 Workshops*. Cham: Springer Nature Switzerland, 2025, pp. 1–16.
- [30] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep., 2009.
- [31] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.
- [32] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.
- [33] Burak, "Pinterest face recognition dataset," <https://www.kaggle.com/datasets/hereisburak/pins-face-recognition>, 2020, kaggle dataset, accessed 2025-09-11.
- [34] K. Han, S. Lee, K.-I. Oh, Y. Bae, H. Jang, J.-J. Lee, W. Lee, and M. Pedram, "Developing TEI-aware ultralow-power SoC platforms for iot end nodes," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4642–4656, 2021.
- [35] E. Choi, J. Park, K. Han, and W. Lee, "AESware: Developing AES-enabled low-power multicore processors leveraging open risc-v cores with a shared lightweight AES accelerator," *Engineering Science and Technology, an International Journal*, vol. 60, p. 101894, 2024.
- [36] J. Park, K. Han, E. Choi, J.-J. Lee, K. Lee, W. Lee, and M. Pedram, "Designing low-power RISC-V multicore processors with a shared lightweight floating point unit for IoT endnodes," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 9, pp. 4106–4119, 2024.
- [37] K. Lee, S. Jeon, K. Lee, W. Lee, and M. Pedram, "Radar-PIM: Developing IoT Processors Utilizing Processing-in-Memory Architecture for Ultrawideband-Radar-Based Respiration Detection," *IEEE Internet of Things Journal*, vol. 12, no. 1, pp. 515–530, 2025.
- [38] J. Choi, E. Choi, S. Choi, and W. Lee, "E-BTS: A low-power event-driven blink tracking system with hardware-software co-optimized design for real-time driver drowsiness detection," *Alexandria Engineering Journal*, vol. 128, pp. 867–877, 2025.
- [39] SiFIVE, <https://github.com/chipsalliance/rocket-chip>, accessed 11 Sep. 2025.
- [40] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy, "A hardware–software blueprint for flexible deep learning specialization," *IEEE Micro*, vol. 39, no. 5, pp. 8–16, 2019.