

Eficiência Energética de Arquiteturas *In-Order* e *Out-of-Order* para Aprendizado Federado de Borda na Detecção de Intrusão em CAN de Veículos Autônomos

Luís Fillipe M. C. Pereira, Matheus A. Souza, Henrique C. Freitas

Computer Architecture and Parallel Processing Team (CArT)
Programa de Pós-graduação em Informática (PPGINF)
Pontifícia Universidade Católica de Minas Gerais (PUC Minas)
Belo Horizonte, Minas Gerais - Brasil

lfmcpereira@sga.pucminas.br, {matheusalcantara, cota}@pucminas.br

Abstract. *Machine-learning intrusion detection systems protect in-vehicle CAN networks, yet their architectural cost on constrained edge cores is poorly characterized. Using gem5, McPAT, and CACTI at 22 nm, we characterize four IDS workloads (Random Forest, XGBoost, MLP, and linear SVM) on in-order (MinorCPU) and out-of-order (O3CPU) cores, including the local training round of federated learning. The out-of-order core reduces EDP for all workloads, but the gain ranges from $1.3\times$ to $5.1\times$ depending on the memory and branch profile. Enlarging caches worsens EDP for most workloads, since leakage dominates at 22 nm. Architecture selection for vehicular IDS must follow the algorithm's computational profile.*

Resumo. *Sistemas de detecção de intrusão baseados em aprendizado de máquina protegem redes CAN veiculares, mas seu custo arquitetural em núcleos de borda restritos é pouco caracterizado. Com gem5, McPAT e CACTI em 22 nm, é realizada a caracterização de quatro cargas de IDS (Random Forest, XGBoost, MLP e SVM linear) em núcleos in-order (MinorCPU) e out-of-order (O3CPU), incluindo o round de treino local do aprendizado federado. O núcleo out-of-order reduz o EDP em todas as cargas, mas o ganho varia de $1,3\times$ a $5,1\times$ conforme o perfil de memória e de desvios. Ampliar as caches piora o EDP da maioria das cargas, pois o leakage domina em 22 nm. A escolha de arquitetura para IDS veicular deve seguir o perfil computacional do algoritmo.*

1. Introdução

A crescente conectividade dos veículos autônomos ampliou de forma significativa a superfície de ataque dos sistemas embarcados automotivos. A *Controller Area Network* (CAN), barramento de comunicação predominante entre as unidades eletrônicas de controle de um veículo, foi projetada priorizando confiabilidade e baixo custo, e não incorpora mecanismos nativos de autenticação ou criptografia. Essa ausência a torna vulnerável

Os autores agradecem ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq; códigos 311697/2022-4, 402837/2024-0 e 309494/2026-5), à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES; código de financiamento 001), à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG; códigos APQ-05058-23 e APQ-01863-26) e ao FIP PUC Minas (código 2026/35258).

a ataques de injeção, negação de serviço e falsificação de mensagens, cujos efeitos podem comprometer diretamente a segurança física dos ocupantes [Liu et al. 2017]. Nesse cenário, sistemas de detecção de intrusão (IDS) baseados em aprendizado de máquina tornaram-se uma linha de defesa relevante, capazes de identificar padrões anômalos de tráfego diretamente na borda veicular.

Executar esses modelos de forma distribuída, entre múltiplos veículos autônomos, esbarra em restrições de privacidade e de largura de banda: enviar o tráfego CAN bruto para um servidor central é inviável em escala e sensível do ponto de vista de dados. O Aprendizado Federado (*Federated Learning*, FL) responde a esse problema treinando os modelos localmente em cada veículo e compartilhando apenas parâmetros do modelo, e não os dados. Contudo, o treino e a inferência *on-device* impõem um custo computacional e energético que é crítico em plataformas de borda com orçamento térmico e de potência limitado e que permanece pouco caracterizado do ponto de vista arquitetural.

De fato, a literatura de IDS veicular concentra-se majoritariamente em métricas de detecção (acurácia, F1-score), enquanto o custo de *hardware* de executar esses algoritmos, como ciclos, taxa de faltas de cache, potência e energia, raramente é quantificado, mesmo em trabalhos recentes que investigam detectores sofisticados [Souza et al. 2025]. Trabalhos que caracterizam cargas de aprendizado de máquina do ponto de vista de sistema evidenciam que algoritmos distintos apresentam perfis de acesso à memória e de computação profundamente heterogêneos [Hazelwood et al. 2018, Adolf et al. 2016], o que impede assumir, sem medição, qual configuração de processador é mais eficiente para uma dada carga de IDS federado.

Este trabalho aborda essa lacuna por meio da caracterização arquitetural de um conjunto de cargas representativo de IDS veicular federado. A dimensão federada é contemplada pelo custo do *round* de treino local executado em cada veículo, e não pela simulação do ciclo completo de agregação. As principais contribuições são: (i) a proposta de um conjunto de quatro cargas de trabalho (*Random Forest*, *XGBoost*, *Multilayer Perceptron* (MLP) e *Support Vector Machine* (SVM) linear) treinadas sobre um mesmo *dataset* de intrusão CAN, cobrindo perfis computacionais distintos; (ii) a caracterização dessas cargas com a metodologia gem5 + McPAT + CACTI a 22 nm, contrastando um núcleo *in-order* (MinorCPU) e um *out-of-order* (O3CPU); (iii) a evidência de que o ganho do *out-of-order*, embora presente em todas as cargas, varia de $1,3 \times$ a $5,1 \times$ conforme o perfil de acesso à memória, de modo que o dimensionamento não pode ser inferido do tipo de núcleo isoladamente; e (iv) uma análise de sensibilidade dos recursos de *cache* e de especulação, e do custo do *round* de treino local do aprendizado federado.

As Seções 2 e 3 apresentam a fundamentação e a metodologia; a Seção 4 discute os resultados e a Seção 5 conclui o trabalho.

2. Fundamentação e Trabalhos Relacionados

2.1. Detecção de intrusão em redes CAN

O barramento CAN transmite quadros identificados por um campo de arbitragem (*arbitration ID*) seguido de até oito *bytes* de dados. Como não há autenticação de origem, um atacante com acesso ao barramento pode injetar quadros arbitrários, sobrescrever mensagens legítimas ou saturar a rede. Os ataques comumente estudados incluem negação

de serviço (DoS), injeção de quadros aleatórios (*fuzzy*) e falsificação de sinais específicos, como rotação do motor (*RPM spoofing*) e marcha (*gear spoofing*). A detecção baseada em aprendizado de máquina modela o comportamento normal do tráfego e sinaliza desvios, sendo a escolha do algoritmo determinante tanto para a qualidade de detecção quanto para o custo de execução na borda.

Neste trabalho é utilizado o AV-IDS-Dataset [Aguilar 2026], gerado no simulador de veículos autônomos CARLA e composto por múltiplos cenários que combinam tipos de via, condições climáticas e perfis de ataque. Como atributos de entrada empregam-se o *arbitration ID*, o código de tamanho (DLC) e os oito *bytes* de *payload*. O *arbitration ID* é altamente informativo para a detecção, pois identifica a ECU transmissora e é particularmente relevante para ataques direcionados a identificadores específicos [Song et al. 2016]; sua inclusão reflete a configuração de um IDS embarcado realista. Como o objeto deste estudo é o impacto arquitetural das cargas, e não a proposta de um detector, esse conjunto de atributos favorece modelos com qualidade de detecção representativa do que se executaria em um veículo real.

2.2. Aprendizado Federado na borda veicular

No paradigma de FL, cada cliente, aqui representado por cada veículo, treina um modelo local sobre seus próprios dados e envia apenas os parâmetros resultantes a um agregador, que os combina em um modelo global [Meneguet et al. 2023]. O esquema clássico de agregação, *FedAvg*, opera pela média ponderada dos pesos dos modelos locais e pressupõe, portanto, modelos parametrizados por pesos contínuos. Dentre as quatro cargas deste estudo, apenas o MLP é diretamente compatível com esse esquema: modelos baseados em árvores (*Random Forest*, *XGBoost*) e o SVM federam por mecanismos distintos, como o compartilhamento de árvores ou de vetores de suporte, que não correspondem à média de pesos. Por essa razão, a caracterização do *round* de treino local (Seção 4) é conduzida sobre o MLP, o único representante *FedAvg*-nativo do conjunto.

2.3. Caracterização arquitetural com gem5, McPAT e CACTI

A avaliação de desempenho e energia por simulação é prática consolidada em arquitetura de computadores. O simulador gem5 [Lowe-Power et al. 2020] modela o comportamento de ciclo do processador e da hierarquia de memória, enquanto o McPAT [Li et al. 2009] e o CACTI [Wilton and Jouppi 1996] estimam potência e energia a partir das estatísticas de execução e de um nó tecnológico-alvo. A combinação gem5 + McPAT tem sido amplamente aplicada à exploração de espaço de projeto, inclusive em contexto nacional [Tashiro et al. 2016], e a plataformas heterogêneas voltadas à exploração de arquiteturas com métricas de desempenho e energia [Qureshi et al. 2021]. Em trabalho recente, o gem5 foi empregado para avaliar como a posição na hierarquia de memória restringe aceleradores matriciais RISC-V, evidenciando o caráter limitado por banda dessas cargas [Aquino et al. 2025]. Especificamente no domínio de borda, essa metodologia foi validada contra *hardware* real com erro inferior a 4% [Yassin et al. 2021], o que respalda seu uso para estimar o custo de cargas embarcadas.

2.4. Posicionamento e contribuição

A identificação dos trabalhos correlatos baseou-se em buscas no Google Scholar com palavras-chave que combinam inferência de modelos baseados em árvores e micro-arquitetura (hierarquia de memória, predição de desvios), simulação de aprendizado

de máquina com gem5 e McPAT e custo de *hardware* de IDS para redes CAN, seguidas de triagem por título e resumo. Os trabalhos recuperados organizam-se em três grupos, nenhum dos quais une a carga de IDS veicular à análise do custo de *hardware*. O primeiro, de IDS veicular, compara algoritmos predominantemente por métricas de detecção [Aguilar et al. 2025, Souza et al. 2025]. O segundo caracteriza cargas de aprendizado de máquina em centros de dados ou sobre redes neurais profundas [Hazelwood et al. 2018, Adolf et al. 2016, Sze et al. 2017], cujo padrão regular de acesso difere do percurso de árvores; mesmo quando avaliadas em dispositivos embarcados, seja na inferência [Alf et al. 2023] ou no treino [Bernardo et al. 2021], a medição ocorre no nível da placa e do *framework*, sem isolar o efeito da microarquitetura. O terceiro trata da eficiência de modelos baseados em árvores, em geral em substratos distintos do processador de propósito geral, como FPGA [Silva et al. 2024] e GPU [Penha et al. 2023]. Nenhum deles propõe um conjunto de cargas para IDS veicular federado nem compara *Random Forest*, XGBoost, MLP e SVM sobre um mesmo *dataset* de intrusão CAN sob a ótica da microarquitetura de CPU, lacuna que o presente trabalho ocupa; o trabalho anterior dos autores restringiu-se à inferência de *Random Forest* [Pereira et al. 2026].

3. Metodologia

3.1. As quatro cargas de trabalho

O conjunto reúne quatro algoritmos clássicos de aprendizado supervisionado escolhidos por apresentarem perfis computacionais deliberadamente distintos, cobrindo um espectro de comportamento em relação ao processador e à memória. O *Random Forest* (RF) e o XGBoost são *ensembles* baseados em árvores de decisão, cuja inferência percorre estruturas irregulares e dependentes de dados, tendendo a estressar a hierarquia de memória. O MLP é uma rede neural densa, dominada por multiplicações de matriz e, portanto, intensiva em ponto flutuante. O SVM linear representa o extremo mais leve: sua inferência resume-se a um produto interno seguido de comparação, funcionando como o piso de custo do *benchmark*.

Todos os modelos foram treinados sobre o AV-IDS-Dataset utilizando os mesmos dez atributos (o *arbitration ID*, o DLC e os oito *bytes* de *payload*), com divisão treino/teste por cenário operacional, agrupando por tipo de via e clima, de modo a evitar vazamento entre contextos. A inclusão do *arbitration ID* reflete o cenário realista de um IDS embarcado, que dispõe do identificador para caracterizar a ECU transmissora e ataques direcionados a identificadores específicos. O desbalanceamento entre classes foi tratado no conjunto de treino por uma combinação de sobreamostragem (SMOTE) e subamostragem aleatória. Os hiperparâmetros foram otimizados por busca bayesiana (Optuna, amostrador TPE) com validação cruzada de cinco dobras e F1-*macro* como objetivo, seguindo a metodologia de trabalhos correlatos sobre o mesmo domínio [Aguilar et al. 2025], com o *scikit-learn* 1.9.0, o XGBoost 3.3.0 e o Optuna 4.9.0. A Tabela 1 resume a estrutura resultante de cada modelo e sua qualidade de detecção no conjunto de teste.

A qualidade de detecção é coerente com a reportada para o mesmo *dataset* [Aguilar 2026], o que atesta a validade das cargas; o SVM linear, por sua natureza, fica atrás dos demais. Cabe destacar que, para elevar a qualidade de detecção, o processo de otimização produziu um *Random Forest* substancialmente maior (499 árvores,

Tabela 1. Estrutura dos modelos após otimização por Optuna e qualidade de detecção no conjunto de teste.

Carga	Estrutura do modelo	Acurácia	F1-macro
Random Forest	499 árvores, profundidade 27 ($\approx 14,4$ M nós)	0,884	0,825
XGBoost	1375 árvores, profundidade 11 (≈ 722 k nós)	0,884	0,825
MLP	camadas $10 \rightarrow 75 \rightarrow 64 \rightarrow 5$	0,878	0,813
SVM linear	hiperplano de 10 atributos ($C = 9,2$)	0,857	0,785

$\approx 14,4$ M de nós), cujo *working set* excede em muito a capacidade das *caches* de borda, característica cujo impacto arquitetural é analisado na Seção 4.

3.2. Do modelo ao binário: inferência e treino em C

Para executar as cargas no simulador, cada modelo treinado foi exportado para código C puro, representando as árvores como vetores de nós percorridos iterativamente, o MLP como matrizes de pesos e o SVM como o vetor de pesos do hiperplano, e compilado para ARM *aarch64*. A equivalência entre a inferência em C e a implementação original em *scikit-learn* foi verificada, com concordância entre 99% e 100% das predições.

O custo do *round* de treino local, associado ao ciclo de FL, foi caracterizado por um *micro-benchmark* em C que executa as operações dominantes de uma iteração de treino do MLP, isto é, a propagação direta (multiplicação de matrizes, ativação ReLU e *softmax*), a retropropagação (as multiplicações transpostas) e a atualização dos pesos por gradiente descendente, sobre um *minibatch*, empregando os pesos reais do modelo treinado. O objetivo não é a convergência, mas a reprodução fiel do perfil arquitetural do treino (intensivo em ponto flutuante e em multiplicação de matrizes), coerente com a proposta de caracterização de custo. Conforme discutido na Seção 2.2, essa análise restringe-se ao MLP por ser o único modelo *FedAvg*-nativo.

3.3. Configuração de simulação e métricas

As cargas foram simuladas no *gem5* em modo de emulação de chamadas de sistema (SE), que isola o custo computacional intrínseco da carga sem a interferência de efeitos do sistema operacional e de interrupções, prática consolidada na caracterização arquitetural de cargas embarcadas [Yassin et al. 2021]. Em contrapartida, o modo SE não captura aspectos determinantes para um IDS embarcado sujeito a prazos, como o escalonamento de tempo real, a latência de interrupções e a variação temporal (*jitter*), cuja avaliação em modo de sistema completo é discutida como trabalho futuro na Seção 5. As simulações foram realizadas para a arquitetura ARM *aarch64*, sob dois modelos de núcleo: o MinorCPU, *in-order*, e o O3CPU, *out-of-order*. O MinorCPU emite até duas instruções por ciclo, enquanto o O3CPU dispõe de largura de emissão de oito, ROB de 192 entradas e filas de *load* e *store* de 32 posições; essas diferenças de recursos, para além do escalonamento dinâmico, são consideradas na leitura dos resultados. Ambos operam a 2,76 GHz e compartilham a mesma hierarquia de memória de referência: *caches* L1 de dados e de instruções de 64 KiB (associatividade 4), L2 privada de 2 MiB (associatividade 8) e L3 compartilhada de 16 MiB (associatividade 16). A região de interesse (ROI) medida corresponde ao laço de 5.000 inferências sobre o conjunto de teste, isolada por marcadores

de *dump* de estatísticas do gem5 e precedida por 100 inferências de aquecimento (*warm-up*) das *caches* e do preditor. As estatísticas de execução do gem5 (instruções por tipo, acessos e faltas por estrutura e ciclos ativos e ociosos) alimentam o McPAT, configurado com os parâmetros dos núcleos simulados, que por sua vez invoca o CACTI para modelar a potência e a área das estruturas de memória (*caches* e preditor de desvios) a partir de sua organização; a potência e a energia foram estimadas ao nó tecnológico de 22 nm, escolhido por ser o mais avançado com modelos públicos validados simultaneamente suportados por ambas as ferramentas. Foram empregados o gem5 v25.1.0.1, o McPAT v1.3 e o CACTI v7.0, com os binários compilados pelo GCC 15.2.0 para *aarch64* com otimização `-O2` e ligação estática. A métrica integradora adotada é o produto energia-atraso (*Energy-Delay Product*, EDP), definido como

$$\text{EDP} = P_{\text{chip}} \times T^2, \quad (1)$$

em que P_{chip} é a potência do *chip* de processamento (núcleos e hierarquia de *caches* on-chip, somando as componentes dinâmica e de fuga) e T o tempo de execução da ROI; ficam fora de P_{chip} o controlador de memória e as interfaces de E/S modelados pelo McPAT e a memória externa, cujo modelo de LPDDR5 no gem5 não especifica as correntes de alimentação necessárias à estimativa de energia. A ponderação quadrática do tempo penaliza soluções lentas, sendo adequada a cenários de borda sensíveis à latência.

Para além do EDP, foram coletadas as seguintes métricas: instruções por ciclo (IPC), faltas de cache por mil instruções (MPKI) na L1D e na L2, taxa de erro de predição de desvios, potência dinâmica e de fuga, e o tempo de execução. A exploração de espaço de projeto totalizou 102 simulações, organizadas em: uma configuração principal de referência (8 execuções: 4 cargas $\times$ 2 núcleos); varreduras de tamanho de L2 (24) e L1D (24); de *prefetcher* (8) e de preditor de desvios (16); escalabilidade multinúcleo com 1, 2 e 4 núcleos (16); e a análise do *round* de treino do MLP variando o *minibatch* em 32, 64 e 128 (6). Cada varredura altera um único parâmetro a partir da configuração de referência, o que isola o efeito de cada recurso, embora não capture as interações entre eles; os recursos variados são os que respondem diretamente aos dois regimes de acesso que distinguem as cargas, irregular nas árvores e regular nas cargas densas.

4. Resultados e Discussão

4.1. A amplitude de custo entre algoritmos e a ineficiência do in-order

O primeiro resultado que emerge da caracterização é a magnitude da disparidade de custo entre algoritmos que, do ponto de vista de detecção, são intercambiáveis. A Figura 1a apresenta o EDP de cada carga nos dois núcleos, em escala logarítmica. Executando na mesma CPU de borda, o custo de um lote de inferência varia em mais de cinco ordens de grandeza: o SVM linear consome $1,1 \times 10^{-6}$ Js, enquanto o *Random Forest* atinge $3,4 \times 10^{-1}$ Js. Essa amplitude, raramente quantificada em conjunto sob uma métrica energética unificada, evidencia que a seleção do algoritmo de detecção é, por si só, uma decisão de projeto arquitetural de primeira ordem, e não um detalhe de implementação a ser resolvido após a escolha do *hardware*.

Scripts, configurações e resultados disponíveis em: <https://github.com/cart-pucminas/gem5-FL-AV-IDS>.

Sobre esse cenário, confirma-se que o núcleo *out-of-order* é mais eficiente em energia para todas as quatro cargas, contrariando a premissa recorrente na literatura de que arquiteturas simples *in-order* seriam a escolha natural para inferência de aprendizado de máquina na borda [Lin et al. 2020, Sze et al. 2017]. O ganho decorre da métrica EDP depender do quadrado do tempo: embora o O3 consuma mais potência, sua conclusão mais rápida dilui o custo energético no fator T^2 . O resultado estende ao conjunto de algoritmos clássicos de IDS uma observação antes restrita à inferência de *Random Forest* [Pereira et al. 2026].

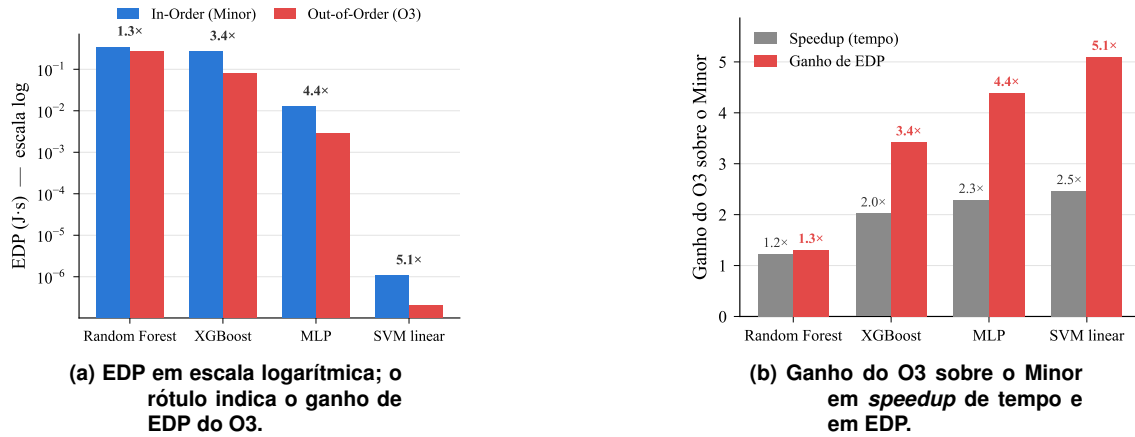


Figura 1. Comparação entre os núcleos Minor (*in-order*) e O3 (*out-of-order*), por algoritmo.

4.2. O ganho do out-of-order depende do perfil de memória da carga

O resultado central deste trabalho não é qual núcleo vence, mas o quanto vence: a vantagem do *out-of-order* sobre o *in-order* é ditada pelo perfil de acesso à memória de cada carga. A Figura 1b mostra que o ganho de EDP do O3 cresce de $1,3\times$ no *Random Forest* a $5,1\times$ no SVM linear, acompanhado pelo respectivo *speedup* de tempo.

O *Random Forest* é o pior caso para o *out-of-order* por ser a carga mais *memory-bound*: MPKI de 214 na L1D e de 28 na L2, com IPC de apenas 0,28 no O3, reflexo do *pointer-chasing* e dos desvios dependentes de dados de *ensembles* de árvores profundas, custo de indireção também observado em GPU [Penha et al. 2023]. O núcleo *out-of-order* passa a maior parte do tempo à espera de memória, sem converter sua capacidade especulativa em desempenho. O XGBoost fica no meio da faixa ($3,4\times$): também percorre árvores, com MPKI de 211 na L1D, mas seu modelo, de dimensão próxima à da L3, produz MPKI de apenas 7,9 na L2. Nas cargas densas e regulares, ao contrário, a especulação se converte em ganho: o MLP atinge IPC de 2,07 e *speedup* de $2,3\times$ sobre o Minor, e o SVM, de $2,5\times$.

Duas evidências independentes sustentam esse diagnóstico. O *prefetcher* de passo constante (*stride*) não reduz a MPKI da L2 do *Random Forest* e chega a piorar seu EDP em 1,4% no Minor, pois não há passo regular em um percurso de árvore guiado pelos dados; o único ganho relevante ocorre no SVM, de acesso sequencial (13,9% de redução de EDP no O3). O tráfego com a DRAM, não incluído em P_{chip} , segue o mesmo contraste: na ROI do O3, o *Random Forest* lê 318 MB (cerca de 62 KiB por inferência), pois seu modelo, de aproximadamente 347 MB, excede em muito a L3; o XGBoost lê 9,3 MB, e o MLP e

o SVM, apenas os 0,2 MB das amostras de entrada, pois seus modelos permanecem nas *caches*. A inclusão da energia da DRAM elevaria, portanto, o custo das cargas baseadas em árvores, em particular do *Random Forest*.

A implicação de projeto, principal contribuição deste trabalho, é que a escolha de microarquitetura para um IDS veicular não pode ser genérica, mas deve ser guiada pelo perfil computacional do algoritmo de detecção.

4.3. Escalabilidade multinúcleo

A mesma dicotomia entre cargas limitadas por memória e por computação governa o comportamento multinúcleo. A Figura 2 apresenta o *speedup* de *throughput* de inferência ao distribuir o fluxo de mensagens CAN em partições disjuntas entre 2 e 4 núcleos; trata-se, portanto, de um limite superior de escalabilidade, adequado ao cenário de inferência independente por mensagem, que não contempla o custo de sincronização de uma implementação real. O MLP escala de forma quase ideal (até $4,0\times$ em quatro núcleos), pois seu *working set* reduzido cabe nas *caches* privadas e cada núcleo opera sem contenção. Já o *Random Forest* satura ($2,6\times$ no O3) e o XGBoost ocupa posição intermediária ($3,1\times$), limitados pela disputa por largura de banda na hierarquia de memória compartilhada. O SVM foi excluído desta análise: sua ROI, inferior a 10^6 instruções, é curta demais para que a janela de medição seja consistente entre configurações, e a execução em quatro núcleos registra 26% menos instruções que a de um núcleo para o mesmo trabalho, o que invalida a razão de tempos como *speedup*. Nas três cargas mantidas na análise, a contagem de instruções da ROI varia menos de 2,6% entre um e quatro núcleos (RF 2,5%, XGBoost 0,4% e MLP abaixo de 0,01%), o que preserva a razão de tempos como medida de *speedup*; o SVM é o único caso em que a carga de trabalho efetiva não se conserva. O resultado reforça, por uma dimensão independente, o diagnóstico de que o custo de memória, e não a capacidade de processamento, é o fator determinante para as cargas de árvore.

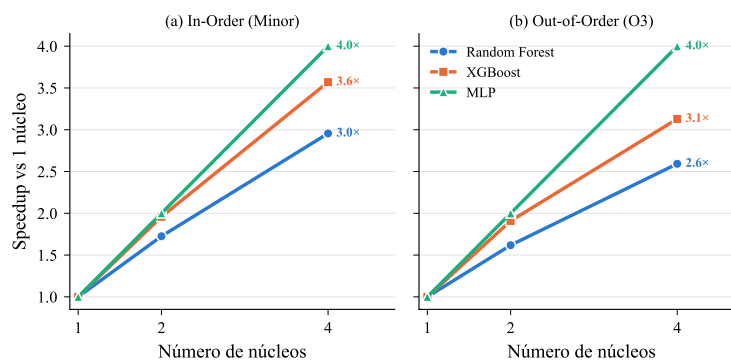


Figura 2. *Speedup* de *throughput* de inferência em 1, 2 e 4 núcleos, nos núcleos (a) Minor e (b) O3.

4.4. A vantagem do out-of-order é condicionada à qualidade da especulação

Se o ganho do *out-of-order* provém de sua capacidade de especular, é de esperar que ele seja mais sensível do que o *in-order* à qualidade do preditor de desvios. A varredura de três preditores (BiMode, Tournament e LTAGE) sobre os dois núcleos confirma essa dependência de forma sistemática. O LTAGE apresenta o menor EDP em todas as oito

combinações de carga e núcleo, mas o que distingue os dois núcleos é a *magnitude* do benefício: substituir o BiMode pelo LTAGE reduz o EDP do O3 em 9,2% no *Random Forest* e em 36,7% no XGBoost, contra apenas 4,1% e 6,8% nas mesmas cargas sobre o Minor. O mecanismo é direto: com *pipeline* mais profundo e janela de execução mais larga, cada desvio mal predito no O3 descarta consideravelmente mais trabalho útil, de modo que a mesma redução de taxa de erro, de 5,0% para 4,0% no XGBoost, converte-se em economia muito maior.

O MLP é a exceção que reforça a interpretação: por ser dominado por laços regulares de multiplicação de matrizes, sua taxa de erro de predição já é praticamente nula com o LTAGE (0,03%), e a escolha do preditor torna-se irrelevante (2,1% de variação de EDP no O3). Isso qualifica o resultado da Seção 4.2: a superioridade do *out-of-order* não é propriedade isolada do escalonamento dinâmico, mas depende de os recursos que a sustentam estarem adequados ao perfil de desvios da carga. Em um IDS baseado em árvores o preditor é componente de primeira ordem no orçamento energético; em um IDS baseado em redes densas, é praticamente indiferente.

4.5. Um resultado contraintuitivo: ampliar a cache pode piorar o EDP

A exploração do tamanho da *cache* L2 revela um comportamento que contraria a intuição de que mais *cache* é sempre benéfico. A Figura 3 mostra o EDP de cada carga, normalizado pela configuração de referência de 2 MiB, ao variar a L2 de 512 KiB a 4 MiB. O XGBoost é a única carga que se beneficia de capacidade adicional, por ser genuinamente *memory-bound* e converter a redução de faltas em redução de tempo [He et al. 2022]: ampliar a L2 de 512 KiB para 2 MiB reduz seu EDP em 38% no O3. O benefício, porém, não é ilimitado: de 2 MiB para 4 MiB o EDP volta a *piorar* (+5,8%), de modo que mesmo na carga que melhor aproveita a *cache* o ótimo é interior ao espaço de projeto.

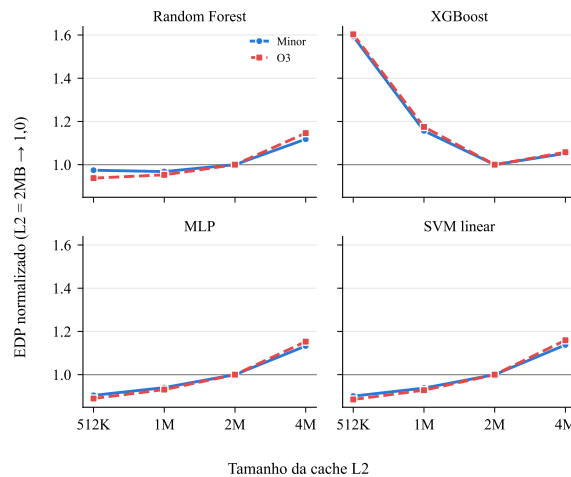


Figura 3. EDP ao variar a *cache* L2, normalizado pela configuração de referência (L2 = 2 MiB → 1,0).

Nas outras três cargas esse ponto de inflexão ocorre antes do menor tamanho avaliado, e o EDP cresce de forma praticamente monotônica com a L2 (no O3, RF +22%, MLP +30% e SVM +31% de 512 KiB a 4 MiB): seus *working sets* não aproveitam a capacidade extra, de modo que a *cache* maior apenas acrescenta potência estática de fuga (*leakage*), dominante no nó de 22 nm [Kim et al. 2003, Kaxiras et al. 2001, Albonesi 1999].

O dimensionamento da *cache* para IDS na borda não pode, portanto, ser desacoplado do algoritmo.

A varredura da L1 de dados, de 16 KiB a 128 KiB, mostra o mesmo mecanismo no nível mais próximo do núcleo e permite generalizá-lo: o tamanho ótimo acompanha o *working set* da carga. O MLP exhibe um degrau abrupto, com a MPKI da L1D no Minor caindo de 82,7 com 16 KiB para praticamente zero com 32 KiB, quando as matrizes de pesos passam a caber na *cache*, o que reduz o EDP em 15,1%; além desse ponto, ampliá-la apenas degrada. Nas demais cargas a menor L1D é a mais eficiente, por razões opostas: o *working set* do SVM é pequeno demais para exigir capacidade (MPKI estável em 14,8) e o percurso de árvores do XGBoost é irregular demais para aproveitá-la (MPKI em torno de 210). O *Random Forest* é o contraste mais informativo: ampliar a L1D de 16 KiB para 128 KiB reduz sua MPKI no O3 em 9,5% e ainda assim *piora* o EDP.

A decomposição de potência da Figura 4 sustenta esse comportamento: o *leakage* responde por 90% a 98% da potência total nas configurações de referência, dominado pela L3 compartilhada de 16 MiB, que sozinha representa de 61% a 76% de P_{chip} ; como as cargas exercitam pouco a hierarquia, a componente dinâmica é minoritária. Onde a potência estática domina, capacidade de *cache* ociosa traduz-se diretamente em desperdício energético. Cabe delimitar a validade desse mecanismo: as plataformas embarcadas mais recentes voltadas a veículos autônomos empregam nós abaixo de 10 nm, para os quais não há modelos públicos validados suportados simultaneamente pelas ferramentas utilizadas. Como o achado depende da fração de potência estática, que a 22 nm responde por 90% a 98% do total e depende também do tipo de transistor adotado para as *caches*, o ponto ótimo de capacidade de *cache* deve ser reavaliado em nós mais avançados, ainda que a dependência entre esse ótimo e o *working set* da carga permaneça.

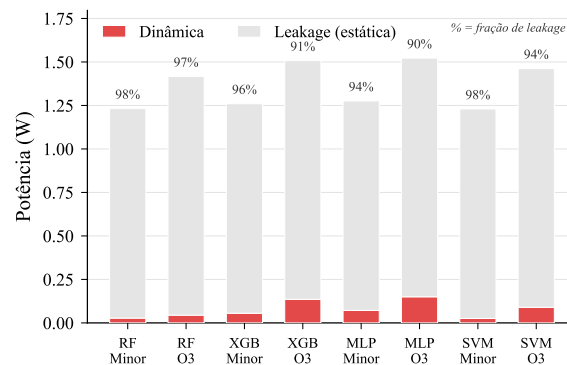


Figura 4. Decomposição da potência em componentes dinâmica e de fuga (*leakage*), por carga e núcleo.

4.6. O custo arquitetural de elevar a qualidade de detecção

Um resultado que conecta diretamente as dimensões de detecção e de arquitetura diz respeito ao custo de melhorar a qualidade do modelo. Elevando o F1-*macro* médio de 0,72 para 0,83 por meio da inclusão do *arbitration ID* e da reotimização dos modelos, observa-se que o impacto sobre o custo arquitetural é fortemente *não uniforme entre algoritmos*, ao contrário do que uma relação monotônica entre acurácia e custo faria supor. A Figura 5 contrasta o EDP das duas configurações. No *Random Forest*, a busca por maior F1 produziu um modelo muito maior (de $\approx 5,3$ M para $\approx 14,4$ M de nós), elevando o EDP em

4,7 $\times$. No XGBoost, ao contrário, a configuração de maior F1 é três vezes *mais barata*, por ter convergido para um modelo menor (de 2045 para 1375 árvores); no MLP o custo é inalterado e no SVM cresce 20%.

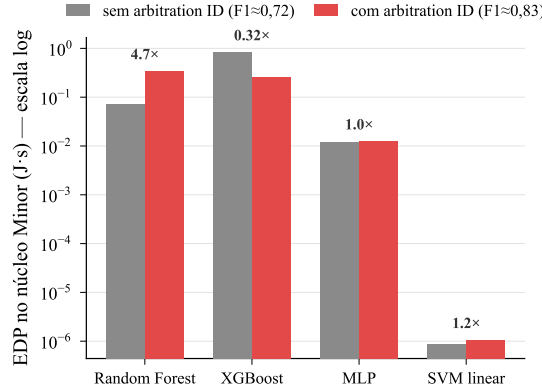


Figura 5. EDP no núcleo Minor sem e com o *arbitration ID* (F1-*macro* médio de 0,72 e 0,83); o rótulo indica o fator de variação.

4.7. Custo do round de treino local para Aprendizado Federado

A leitura arquitetural é que a relação entre qualidade de detecção e custo de recursos é mediada pela estrutura que o algoritmo assume para atingir aquela qualidade, não pela acurácia em si. Um mesmo ganho de F1 pode ser obtido de forma barata (XGBoost) ou proibitivamente cara (RF), dependendo de como o processo de otimização molda o *working set*. Nesse sentido, a preferência por muitas árvores rasas em lugar de poucas profundas, já observada na avaliação de árvores de decisão em GPU [Penha et al. 2023], encontra aqui uma justificativa energética. Para o projeto de IDS na borda, isso implica que a co-otimização de modelo e *hardware* deve considerar o custo arquitetural como critério explícito, sob pena de adotar um modelo de alta acurácia, porém inviável no orçamento de recursos de uma ECU. As conclusões arquiteturais mantiveram-se consistentes nas duas configurações de qualidade avaliadas, sendo portanto robustas ao ponto de operação de detecção escolhido.

Por fim, a caracterização estende-se ao *round* de treino local, etapa que define o custo do Aprendizado Federado *on-device* e que raramente é avaliada em nível arquitetural. A Figura 6 apresenta o EDP do treino do MLP para três tamanhos de *minibatch*. O núcleo *out-of-order* também domina o treino, com EDP cerca de 2,9 $\times$ menor que o *in-order* de forma consistente nos três tamanhos, pois o treino é intensivo em ponto flutuante e regular no acesso à memória, o regime em que a execução especulativa se paga. O tempo e a energia crescem linearmente com o tamanho do *minibatch* e o EDP de forma quadrática, pela ponderação em T^2 , o que fornece uma base quantitativa para dimensionar o custo de uma rodada de FL em função da carga de treino atribuída a cada veículo. A etapa de comunicação não é simulada, mas sua ordem de grandeza pode ser estimada a partir da estrutura do modelo: os 6.014 parâmetros do MLP em precisão simples correspondem a cerca de 24 KB transmitidos por veículo a cada *round* de treinamento, e não a cada inferência. A modelagem do custo energético dessa transmissão em uma federação completa é tratada como trabalho futuro.

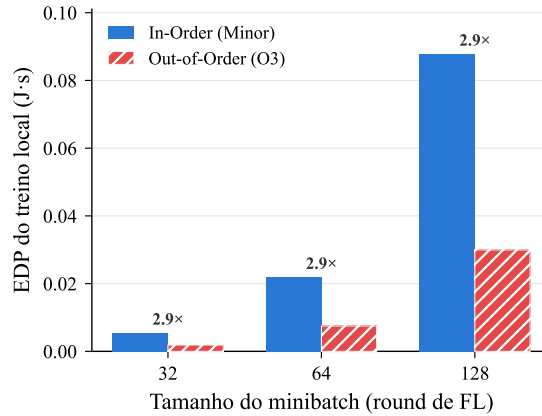


Figura 6. EDP do *round* de treino local do MLP em função do tamanho do *mini-batch*, nos dois núcleos; o rótulo indica o ganho de EDP do O3.

5. Conclusão e Trabalhos Futuros

A adoção de sistemas de detecção de intrusão baseados em aprendizado federado na borda de veículos autônomos exige compreender não apenas a qualidade de detecção dos modelos, mas também o custo de executá-los em processadores com orçamento de energia restrito. Este trabalho apresentou a caracterização arquitetural de quatro cargas representativas desse cenário (*Random Forest*, XGBoost, MLP e SVM linear) treinadas sobre um mesmo *dataset* de intrusão em rede CAN e avaliadas em dois modelos de núcleo de borda, *in-order* e *out-of-order*, por meio da metodologia gem5 + McPAT + CACTI a 22 nm.

Os resultados evidenciam que, embora o *out-of-order* reduza o EDP em todas as cargas, o ganho varia de $1,3\times$ a $5,1\times$ conforme o perfil de acesso à memória, sendo menor justamente nas cargas mais irregulares e limitadas por memória. A análise da hierarquia de *cache* mostrou que o tamanho ótimo de L2 e de L1D acompanha o *working set* da carga: capacidade adicional só reduz o EDP quando se converte em redução de tempo, como no XGBoost, e mesmo nele o ótimo é interior ao espaço de projeto; nas demais cargas, apenas eleva a potência estática, dominante a 22 nm. A varredura de preditores de desvios mostrou ainda que a vantagem do *out-of-order* é condicionada à qualidade da especulação, com a substituição do preditor BiMode pelo LTAGE reduzindo o EDP em até 36,7% nas cargas baseadas em árvores e sendo praticamente indiferente nas cargas densas. Esses achados reforçam que a escolha de arquitetura para IDS veicular deve ser guiada pelo perfil computacional da carga, e não por regras genéricas de dimensionamento. Cabe registrar que os dois núcleos avaliados diferem em mais de um recurso além do escalonamento de instruções (largura de emissão, profundidade de *pipeline*, entre outros); as conclusões referem-se, portanto, às configurações MinorCPU e O3CPU efetivamente avaliadas.

Como trabalhos futuros, pretende-se avaliar as cargas em simulação de sistema completo, com o efeito do sistema operacional e de interrupções; incluir modelos sequenciais e convolucionais unidimensionais, frequentes em IDS de séries temporais; validar as estimativas de energia em plataformas de borda reais, como processadores ARM Cortex e módulos embarcados para veículos autônomos; e investigar o ciclo completo do aprendizado federado, incluindo a comunicação com o agregador e o treino local dos modelos baseados em árvores e do SVM.

Referências

- Adolf, R., Rama, S., Reagen, B., Wei, G.-Y., and Brooks, D. (2016). Fathom: Reference workloads for modern deep learning methods. In *IEEE International Symposium on Workload Characterization (IISWC)*, pages 1–10. IEEE.
- Aguilar, I., Nobre, C., and Freitas, H. (2025). Analyzing car-hacking dataset to understand cyberattacks and intrusion detection in autonomous vehicles. In *Proceedings of the LVII Brazilian Symposium on Operations Research (SBPO)*, pages 1–12. SOBRAPO.
- Aguilar, I. R. (2026). Bases de dados com cenários heterogêneos para análise e avaliação de intrusão em *Controller Area Networks* de veículos autônomos. Dissertação (mestrado em informática), Pontifícia Universidade Católica de Minas Gerais, Belo Horizonte. Disponível em: https://bib.pucminas.br/teses/Informatica_IsabelaReginaAguilar_32153_TextoCompleto.pdf. Acesso em: 16 jul. 2026. AV-IDS-Dataset disponível em: <https://github.com/cart-pucminas/AV-IDS-Dataset>.
- Albonesi, D. H. (1999). Selective cache ways: On-demand cache resource allocation. In *Proceedings of the 32nd Annual ACM/IEEE International Symposium on Microarchitecture (MICRO)*, pages 248–259. IEEE.
- Alf, L., Hoffmann, R., Müller, C., and Griebler, D. (2023). Análise da execução de algoritmos de aprendizado de máquina em dispositivos embarcados. In *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 61–72, Porto Alegre, RS, Brasil. SBC.
- Aquino, I. C., Krebs, C. P., Wanner, L., and Rigo, S. (2025). Evaluating memory constraints of risc-v matrix accelerators using gem5. In *Anais do XXVI Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 193–204. SBC.
- Bernardo, F., Yokoyama, A., Schulze, B., and Ferro, M. (2021). Avaliação do consumo de energia para o treinamento de aprendizado de máquina utilizando single-board computers baseadas em arm. In *Anais do XXII Simpósio em Sistemas Computacionais de Alto Desempenho*, pages 60–71, Porto Alegre, RS, Brasil. SBC.
- Hazelwood, K., Bird, S., Brooks, D., Chintala, S., Diril, U., Dzhulgakov, D., Fawzy, M., Jia, B., Jia, Y., Kalro, A., et al. (2018). Applied machine learning at Facebook: A datacenter infrastructure perspective. In *IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 620–629. IEEE.
- He, M., Thottethodi, M., and Vijaykumar, T. N. (2022). Booster: An accelerator for gradient boosting decision trees training and inference. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 1051–1062. IEEE.
- Kaxiras, S., Hu, Z., and Martonosi, M. (2001). Cache decay: Exploiting generational behavior to reduce cache leakage power. In *Proceedings of the 28th Annual International Symposium on Computer Architecture (ISCA)*, pages 240–251. ACM/IEEE.
- Kim, N. S., Austin, T., Blaauw, D., Mudge, T., Flautner, K.-s.-t., Hu, J. S., Irwin, M. J., Kandemir, M., and Narayanan, V. (2003). Leakage current: Moore’s law meets static power. *Computer*, 36(12):68–75.
- Li, S., Ahn, J. H., Strong, R. D., Brockman, J. B., Tullsen, D. M., and Jouppi, N. P. (2009). McPAT: An integrated power, area, and timing modeling framework for multicore and

- manycore architectures. In *Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 469–480. ACM.
- Lin, J. et al. (2020). MCUNet: Tiny deep learning on IoT devices. *Adv. Neural Inf. Process. Syst. (NeurIPS)*, 33:11711–11722.
- Liu, S., Tang, J., Zhang, Z., and Gaudiot, J.-L. (2017). Computer architectures for autonomous driving. *Computer*, 50(8):18–25.
- Lowe-Power, J., Ahmad, A. M., Akram, A., et al. (2020). The gem5 simulator: Version 20.0+. In *arXiv preprint*. arXiv:2007.03152.
- Meneguette, R. I., De Grande, R. E., Ueyama, J., Rocha Filho, G. P. R., and Madeira, E. R. M. (2023). Vehicular edge computing: Architecture, resource management, security, and challenges. *ACM Computing Surveys*, 55(2):1–46.
- Penha, J., Silva, A. K. C. d., Barros, O., Moreira, I., Nacif, J. A. M., and Ferreira, R. (2023). Avaliação de estilos de código para árvores de decisão em gpu com microbenchmarks. In *Anais do XXIV Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 277–288. SBC.
- Pereira, L. F. M. C., Souza, M. A., and Freitas, H. C. (2026). Eficiência energética de arquiteturas *In-Order* e *Out-of-Order* para inferência de *Random Forest* em borda veicular. In *Anais da Escola Regional de Alto Desempenho da Região Sudeste (ERAD-SE)*. SBC.
- Qureshi, Y. M., Simon, W. A., Zapater, M., Olcoz, K., and Atienza, D. (2021). gem5-X: A many-core heterogeneous simulation platform for architectural exploration and optimization. *ACM Transactions on Architecture and Code Optimization (TACO)*, 18(4):1–27.
- Silva, A., Silva, O., Moreira, I., Nacif, J. A., and Ferreira, R. (2024). Implementações eficientes de random forest em fpga de baixo custo para internet das coisas e computação de borda. In *Anais do XXV Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 49–60. SBC.
- Song, H. M., Kim, H. R., and Kim, H. K. (2016). Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network. In *International Conference on Information Networking (ICOIN)*, pages 63–68. IEEE.
- Souza, M. T. d., Souza, M. A., and Freitas, H. C. d. (2025). Quantum machine learning with enhanced autoencoders for intrusion detection. In *Anais do XXVI Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 217–228. SBC.
- Sze, V. et al. (2017). Efficient processing of deep neural networks: A tutorial and survey. *Proc. IEEE*, 105(12):2295–2329.
- Tashiro, G. et al. (2016). An environment for design space exploration using gem5-mcpat. In *VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pages 220–225. IEEE.
- Wilton, S. J. E. and Jouppi, N. P. (1996). CACTI: An enhanced cache access and cycle time model. *IEEE Journal of Solid-State Circuits*, 31(5):677–688.

Yassin, Y., Jahre, M., Kjeldsberg, P. G., Aunet, S., and Catthoor, F. (2021). Fast and accurate edge computing energy modeling and DVFS implementation in GEM5 using system call emulation mode. *Journal of Signal Processing Systems*, 93:33–48.