Utilizando a GPU ARM Mali-G31 para a execução do YOLOX com o framework Burn

Luiz F. B. de Araujo, Marcio S. Oyamada
Centro de Ciências Exatas e Tecnológicas
Universidade Estadual do Oeste do Paraná (Unioeste) – Campus de Cascavel
Cascavel-PR, Brazil
{luiz.araujo2, marcio.oyamada}@unioeste.br

***Abstract*—O presente trabalho investiga a utilização da GPU ARM Mali-G31, comum em SBCs (*single board computer*) e em TVBox de baixo custo, para o processamento de redes neurais convolucionais utilizando o *framework* Burn. O estudo foi conduzido em uma TVBox Amlogic S905X4 descaracterizada, executando o sistema Debian Bookworm, a fim de avaliar a viabilidade do uso de *hardware* reaproveitado em aplicações de computação na borda. Foram realizados testes de inferência com o modelo YOLOX-Tiny, empregando os *backends* NdArray (CPU) e WGPU (GPU) do Burn. Os resultados demonstraram que, embora o *backend* WGPU permita a execução na GPU Mali-G31 via OpenGL, o desempenho foi inferior ao da CPU, apresentando latência aproximadamente quatro vezes maior. Essa diferença é atribuída à ausência de suporte completo ao Vulkan, melhor que o OpenGL em tarefas de computação, e às limitações intrínsecas da arquitetura da GPU. Apesar disso, a execução bem-sucedida da inferência comprova a compatibilidade do Burn com os *drivers* de código aberto da GPU ARM Mali-G31, Panfrost e Mesa, e evidencia o potencial de reutilização de dispositivos descartados em aplicações sustentáveis de IA embarcada. O estudo contribui para a democratização de aplicações de aprendizado profundo em plataformas acessíveis e destaca oportunidades de otimização para trabalhos futuros.**

***Index Terms*—Computação em borda, GPU ARM Mali-G31, Burn, WGPU**

I. Introdução

A crescente demanda por aplicações de inteligência artificial em dispositivos embarcados, que em geral possuem recursos limitados, tem acelerado a necessidade de soluções eficientes e portáteis. A computação na borda, em especial, pode se beneficiar do uso de NPUs e GPUs presentes em SoCs (*System-on-Chip*) para acelerar o processamento de modelos de inferência.

Nesse sentido, existem soluções comerciais, como a linha NVIDIA Jetson ou Google Coral, que oferecem desempenho elevado para aplicações que fazem uso de redes neurais artificiais [1] [2], mas a um custo consideravelmente elevado. No entanto, entre as plataformas embarcadas disponíveis, as GPUs Arm Mali se destacam por sua ampla adoção em SBCs de baixo custo e consumo, presentes até mesmo em dispositivos do tipo TVBox que podem ser descaracterizados para executar Linux. A GPU ARM Mali-G31, por exemplo, baseado na arquitetura Bifrost, oferece suporte teórico ao OpenGL ES 3.1 e ao Vulkan 1.0 por meio dos *drivers* de código aberto Panfrost e Mesa, o que viabiliza cargas de trabalho gráficas e de computação acelerada por GPU em sistemas Linux [3].

GPUs de sistemas embarcados, como a ARM Mali-G31, por se tratarem de arquiteturas mais simples, não são compatíveis com APIs especializadas para GPGPU (computação de propósito geral em unidades de processamento gráfico), como CUDA, OpenVINO ou ROCm. Nesse contexto, o Burn, um *framework* em desenvolvimento para aprendizado profundo escrito em Rust, pode ser utilizado para aproveitar as capacidades de processamento gráfico dessas GPUs para tarefas de GPGPU utilizando APIs gráficas tradicionais em vez de APIs focadas em computação. Ele foi projetado com foco em flexibilidade, desempenho e portabilidade entre diferentes *backends*, como CUDA, ROCm, WGPU, dentre outros. Sua arquitetura permite a troca de *backends* de forma transparente, além de oferecer suporte a execução assíncrona, fusão de *kernels* e gerenciamento inteligente de memória, o que permite sua utilização em diferentes perfis de *hardware* [4].

O *backend* WGPU do Burn atua como uma camada de abstração sobre APIs gráficas multiplataforma, permitindo que o mesmo código seja executado sobre Vulkan, Metal, DirectX ou OpenGL, conforme a disponibilidade do sistema, utilizando os *shaders* de computação disponíveis nas APIs compatíveis. Em plataformas ARM com GPUs Mali, como a utilizada neste trabalho, o WGPU identifica automaticamente o suporte disponível e utiliza o OpenGL como modo alternativo (*fallback*) quando o Vulkan não está plenamente implementado [5]. Essa característica garante portabilidade, mas implica em perda de desempenho em cargas de trabalho mais intensas, com as quais o OpenGL é geralmente pior que o Vulkan [6] [7] [8].

O modelo utilizado nos experimentos foi o YOLOX-Tiny, uma variante leve da família YOLO, desenvolvida pela Megvii em 2021. O YOLOX adota uma abordagem *anchor-free*, dispensando as caixas de ancoragem tradicionais e simplificando o processo de treinamento [9]. O YOLOX oferece um bom equilíbrio entre velocidade e acurácia, sendo amplamente adotado em aplicações de detecção de objetos em tempo real. Entre suas variantes, o YOLOX-Tiny destaca-se pela eficiência computacional, o que o torna adequado para execução em dispositivos de borda com recursos limitados, como a TVBox analisada neste trabalho.

Este trabalho, portanto, faz uma investigação inicial acerca da integração do *framework* Burn com a GPU ARM Mali-G31, utilizando os *drivers* de código aberto Panfrost e Mesa.

O objetivo é avaliar a viabilidade desse conjunto de *softwares* para acelerar inferências na borda, utilizando um modelo de detecção de objetos, e indicar potenciais otimizações e destacar os desafios relacionados à maturidade dos *drivers*.

II. Preparação do Teste

Os experimentos foram realizados em uma TVBox descaracterizada, doada pela Receita Federal à Universidade Estadual do Oeste do Paraná. Trata-se de uma X Plus Pro, equipado com o SoC Amlogic S905X4. A Tabela I mostra as especificações do dispositivo, que foi descaracterizado e configurado para executar uma imagem customizada do Debian Bookworm[1] para SoC da Amlogic [10], instalada em um cartão microSD, substituindo o sistema operacional original (Android).

Table I
Especificações do dispositivo utilizado nos experimentos

Componente	Especificação
Modelo	X Plus Pro (descaracterizada)
SoC	Amlogic S905X4
CPU	4× ARM Cortex-A55 @ 2 GHz
GPU	ARM Mali-G31 MP2
Memória RAM	4 GB LPDDR4
Armazenamento interno	64 GB eMMC
Sistema operacional	Debian Bookworm (devmfc)

A Fig. 1 apresenta o diagrama de blocos do SoC Amlogic S905X4, mostrando a organização dos seus principais componentes. O processador integra quatro núcleos ARM Cortex-A55 de 64 bits, cada um com cache L1 dedicada (32 KB para instruções e 32 KB para dados) e cache L2 compartilhada, oferecendo suporte a instruções NEON e VFP para aceleração de operações vetoriais. A unidade gráfica Mali-G31 MP2, baseada na arquitetura Bifrost, é responsável pelo processamento gráfico e computacional, compartilhando a controladora de memória DDR3/DDR4/LPDDR4 e o barramento de alta largura de banda do sistema. O diagrama também mostra os blocos auxiliares dedicados a áudio e vídeo, como o HiFi4 DSP e o AVE-10 Video Engine, além do mecanismo de segurança TrustZone, que garante a execução isolada de processos sensíveis. As interfaces externas incluem controladores USB 3.0, PCIe 2.0, Ethernet, HDMI 2.1, I^{2}S, I^{2}C e SPI, permitindo conectividade e suporte multimídia. Em conjunto, esses módulos tornam o S905X4 um SoC capaz de executar aplicações de inteligência artificial e processamento de vídeo em dispositivos de baixo consumo energético.

Para a execução dos testes, foi empregado o *framework* Burn em sua versão 0.18.0. O código utilizado foi adaptado a partir dos exemplos oficiais da Tracel AI [11], incorporando medições de tempo em cada etapa do *pipeline*. O código-fonte completo encontra-se no GitHub[2].

Para facilitar os testes, a obtenção do modelo pré-treinado segue uma rotina automatizada implementada em Rust, na qual a estrutura de dados `Weights` armazena o endereço

[1]Versão 6.12.30, disponível em https://github.com/devmfc/debian-on-amlogic/releases/tag/v6.12.30

[2]Disponível em: https://github.com/lbecher/yolox-burn

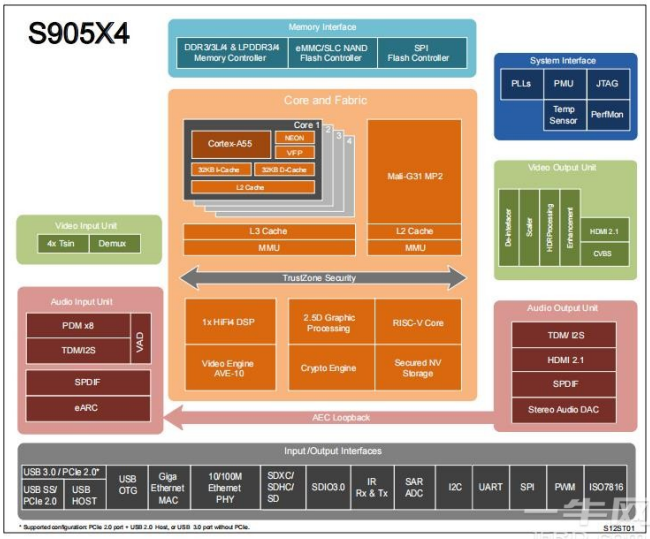

Fig. 1. Diagrama de blocos do Amlogic S905X4

do arquivo de pesos e o número de classes do conjunto COCO. Quando a funcionalidade `pretrained` é habilitada, o módulo `downloader` verifica a presença do arquivo na pasta de cache local do usuário (`~/.cache/yolox-burn`); caso o arquivo ainda não exista, ele é baixado diretamente do repositório oficial da Megvii e salvo localmente antes de ser carregado para inferência. Essa abordagem garante reprodutibilidade e dispensa a necessidade de downloads manuais, simplificando a configuração do ambiente de testes. No presente estudo, foi utilizado o modelo YOLOX-Tiny, cujos pesos pré-treinados (`yolox_tiny.pth`) foram obtidos a partir da versão 0.1.1rc0 da implementação oficial.

Quanto ao ambiente de compilação, este foi configurado diretamente no Debian Bookworm, utilizando o compilador Rust 1.90, instalado por meio do *rustup*. A compilação e execução do projeto foi realizada com os comandos:

```
cargo run -r \
    -F pretrained,wgpu-backend \
    -- dog_bike_man.jpg
cargo run -r \
    -F pretrained,ndarray-backend \
    -- dog_bike_man.jpg
```

Durante a execução, foram coletados os tempos de cada etapa do processo de inferência, incluindo inicialização do dispositivo, carregamento do modelo, pré-processamento, inferência (*forward pass*) e pós-processamento. Os testes foram repetidos para ambos os *backends* disponíveis no Burn (o NdArray, que opera exclusivamente na CPU, e o WGPU, que utiliza a GPU Mali-G31 por meio do OpenGL). Cada execução foi realizada individualmente, com medições automáticas feitas utilizando funções internas de temporização (Instant) da biblioteca padrão do Rust, garantindo precisão temporal em microssegundos. Para reduzir a variabilidade estatística e assegurar maior confiabilidade na avaliação do

desempenho, o programa foi executado cinco vezes para se obter a média simples dos tempos de execução.

O experimento teve como objetivo principal comparar o desempenho entre os dois modos de execução e verificar a viabilidade da utilização da GPU ARM Mali-G31 como aceleradora de tarefas de inferência de modelos de visão computacional.

III. Resultados e Discussões

A Fig. 2 apresenta a imagem de entrada utilizada no teste, contendo um homem, uma bicicleta e um cachorro. A Fig. 3 mostra o resultado da inferência, com as detecções sobrepostas e as respectivas classes identificadas.

Fig. 2. Imagem de entrada utilizada para inferência.

O Burn, como visto na Fig. 3, utilizando o modelo pré-treinado YOLOX-Tiny, identificou corretamente três objetos principais na Fig. 2, mantendo-se consistente entre as inferências realizadas nos dois *backends*. A primeira detecção corresponde ao homem, com confiança de 0,87 e coordenadas delimitadas em [219,56, 24,69, 433,04, 452,30]. A segunda detecção refere-se a bicicleta, com confiança de 0,96 e coordenadas [79,50, 220,71, 639,86, 567,23]. Por fim, o modelo reconheceu um cachorro, com confiança de 0,89 e coordenadas [339,69, 374,77, 463,62, 585,26].

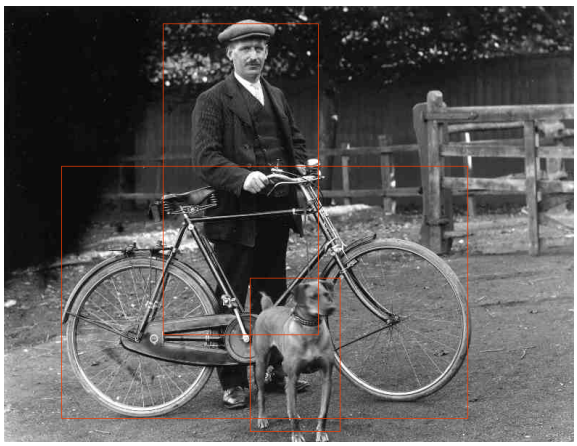

Fig. 3. Resultado da inferência gerado pelo modelo YOLOX-Tiny.

Quanto às métricas de desempenho, a Tabela II resume os tempos médios registrados para cada etapa do processo de inferência em ambos os *backends*. Observa-se que o tempo total de execução utilizando WGPU foi superior ao do NdArray, apresentando um tempo de execução cerca de quatro vezes maior. Há duas prováveis causas para essa diferença considerável. Uma delas é a falta de suporte ao Vulkan para GPUs ARM Mali no *driver* Mesa disponível nos repositórios oficiais do Debian Bookworm, forçando o WGPU a utilizar o OpenGL como *fallback*, que é menos eficiente do que o Vulkan para tarefas de computação mais exigentes. A outra é a simplicidade da arquitetura da GPU ARM Mali-G31, que pode ser incapaz de realizar tarefas de computação tão eficientemente quanto a CPU do Amlogic S905X4, constituída de quatro núcleos ARM Cortex-A55 de 2 GHz.

Além disso, ao analisar as etapas individualmente, observa-se que as fases mais críticas para o desempenho são a inferência (*forward*) e o pós-processamento. Juntas, elas representam mais de 95% do tempo total no *backend* WGPU, enquanto no NdArray a inferência sozinha já é responsável por aproximadamente 88% do tempo total. Isso indica que a otimização deve se concentrar nessas duas etapas, especialmente na execução do modelo e nas rotinas de filtragem e supressão não máxima, que envolvem operações intensivas em ponto flutuante e paralelismo irregular.

As demais etapas, como leitura de imagem, redimensionamento e conversão para tensor, possuem impacto relativamente pequeno no tempo total (menos de 2% cada), indicando que, para o cenário avaliado, o gargalo computacional não está na preparação dos dados, mas na própria execução e pós-tratamento da rede neural.

Table II
Comparativo de tempos médios de execução entre os *backends*.

Etapa	NdArray	WGPU	Speedup
Inicialização	4.7 µs	10.0 µs	2.13× mais lento
Carregamento do modelo	1.48 s	2.09 s	1.41× mais lento
Leitura da imagem	27.1 ms	51.4 ms	1.90× mais lento
Redimensionamento	120.5 ms	117.7 ms	1.02× mais rápido
Conversão para tensor	18.6 ms	45.9 ms	2.47× mais lento
Inferência (*forward*)	12.7 s	51.7 s	4.07× mais lento
Pós-processamento	37.7 ms	1.75 s	46.4× mais lento
Total	**14.42 s**	**55.80 s**	**3.87× mais lento**

IV. Considerações Finais

Os resultados obtidos neste estudo demonstram que é possível executar tarefas de inferência de modelos de visão computacional utilizando o *framework* Burn em dispositivos embarcados equipados com a GPU ARM Mali-G31, mesmo em uma TVBox Amlogic S905X4. Embora o desempenho apresentado pelo *backend* WGPU (GPU) tenha sido inferior ao do NdArray (CPU), a execução bem-sucedida da inferência confirma uma compatibilidade mínima do Burn com o ecossistema gráfico dos *drivers* Panfrost e Mesa.

A diferença de desempenho observada evidencia, entretanto, a necessidade de amadurecimento do suporte à GPGPU nas GPUs ARM Mali-G31. A ausência de suporte ao Vulkan no

driver Mesa dos repositórios oficiais do Debian Bookworm resultou em maior latência e subaproveitamento da GPU, aliada à sua arquitetura que, por si só, é bastante limitada.

Ainda assim, o experimento reforça o potencial de utilização de dispositivos descartados ou apreendidos (como TVBox) em aplicações de computação em borda e inteligência artificial distribuída, oferecendo uma alternativa sustentável e de baixo custo para ambientes acadêmicos e de pesquisa. O uso do Burn em conjunto com *drivers* e bibliotecas abertas representa um passo importante em direção à democratização do acesso a ferramentas de aprendizado profundo em *hardware* acessível, que geralmente não possuem APIs específicas destinadas a GPGPU.

Trabalhos futuros podem explorar otimizações no compilador de *shaders* do Panfrost/Mesa, completar o suporte ao Vulkan para GPUs ARM Mali-G31 e otimizações no próprio Burn/WGPU. Além disso, pretende-se investigar o desempenho do Burn em *hardwares* com suporte consolidado ao Vulkan, bem como comparar seu tempo de execução com soluções proprietário ou específicas de fabricantes de *chips*, como CUDA, OpenVINO e ROCm. Esses avanços podem contribuir para o desenvolvimento de soluções portáteis e sustentáveis de IA na borda, ampliando o escopo de aplicação do Burn em arquiteturas ARM.

REFERENCES

[1] NVidia. (2025) Nvidia jetson for robotics and edge ai. NVidia. [Online]. Available: https://developer.nvidia.com/embedded-computing

[2] Google. (2025) Coral. Google. [Online]. Available: https://www.coral.ai/

[3] Mesa3D. (2025) Panfrost. [Online]. Available: https://docs.mesa3d.org/drivers/panfrost.html

[4] TracelAI. (2025) Train and deploy ai models efficiently on any device. TracelAI. [Online]. Available: https://burn.dev/

[5] WGPU. (2025) Wgpu: portable graphics library for rust. WGPU. [Online]. Available: https://wgpu.rs/

[6] S. I. Gunadi and P. Yugopuspito, "Real-time gpu-based sph fluid simulation using vulkan and opengl compute shaders," in *2018 4th International Conference on Science and Technology (ICST)*. IEEE, 2018, pp. 1–6.

[7] O. Ferraz, P. Menezes, V. Silva, and G. Falcao, "Benchmarking vulkan vs opengl rendering on low-power edge gpus," in *2021 International Conference on Graphics and Interaction (ICGI)*. IEEE, 2021, pp. 1–8.

[8] I. Gil, "Performance improvement methods for hardware accelerated graphics using vulkan api," in *2022 VI International Conference on Information Technologies in Engineering Education (Inforino)*. IEEE, 2022, pp. 1–5.

[9] G. Jocher. (2025) Yolov8 vs. yolox: Uma análise técnica detalhada. [Online]. Available: https://docs.ultralytics.com/pt/compare/yolov8-vs-yolox/

[10] devmfc. (2025) Debian linux image for android tv boxes with amlogic soc's. [Online]. Available: https://github.com/devmfc/debian-on-amlogic

[11] TracelAI. (2025) Models. TracelAI. [Online]. Available: https://github.com/tracel-ai/models