UNIVERSIDADE DE SÃO PAULO
ESCOLA DE ENGENHARIA DE SÃO CARLOS

Caio Wingeter de Castilho

Otimização de Baixo Nível em Vision Transformers: Desenvolvimento de Kernels de Inferência Customizados para Eficiência de SRAM

São Carlos

2026

Caio Wingeter de Castilho

Otimização de Baixo Nível em Vision Transformers: Desenvolvimento de Kernels de Inferência Customizados para Eficiência de SRAM

Monografia apresentada ao Curso de Engenharia Aeronáutica, da Escola de Engenharia de São Carlos da Universidade de São Paulo, como parte dos requisitos para obtenção do título de Engenheiro Aeronáutico.

Orientador: Prof. Dr. Glauco Caurin

VERSÃO CORRIGIDA

São Carlos

2026

width=!,height=!,

ERRATA

A errata é um elemento opcional, que consiste de uma lista de erros da obra, precedidos pelas folhas e linhas onde eles ocorrem e seguidos pelas correções correspondentes. Deve ser inserida logo após a folha de rosto e conter a referência do trabalho para facilitar sua identificação, conforme a ABNT NBR 14724 (Associação Brasileira de Normas Técnicas, 2011).

Modelo de Errata:

CASTILHO, C. W. **Otimização de Baixo Nível em Vision Transformers: Desenvolvimento de Kernels de Inferência Customizados para Eficiência de SRAM**. 2026. 47 p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2026.

ERRATA

Folha	**Linha**	**Onde se lê**	**Leia-se**
1	10	auto-conclavo	autoconclavo

width=!,height=!, width=!,height=!,

Este trabalho é dedicado à todos que me apoiaram
e me deram o suporte durante esse percurso.

AGRADECIMENTOS

Dedico este trabalho a todos que estiveram ao meu lado ao longo dessa caminhada, oferecendo apoio, força e incentivo para que eu pudesse superar cada desafio encontrado no percurso.

Agradeço, de forma especial, à minha companheira Luana, por todo o carinho, paciência e apoio incondicional ao longo dessa jornada. Aos meus pais, André e Carolina, que, apesar de todas as dificuldades, nunca deixaram de acreditar em mim e sempre se esforçaram ao máximo para que eu pudesse alcançar meus sonhos. Ao meu irmão Breno, que esteve ao meu lado em todos os momentos, não apenas como irmão, mas como meu melhor amigo.

"Líderes são mestres em encontrar o sentido e aprendizado em todo tipo de experiência de vida."
(Jucá, 2014, p. 94)

RESUMO

CASTILHO, C. W. **Otimização de Baixo Nível em Vision Transformers: Desenvolvimento de Kernels de Inferência Customizados para Eficiência de SRAM**. 2026. 47 p. Monografia (Trabalho de Conclusão de Curso) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2026.

Este trabalho aborda a otimização de baixo nível de kernels de inferência aplicados a modelos de visão computacional, com foco específico na arquitetura Vision Transformer (ViT). O objetivo principal consiste em mitigar o gargalo de transferência de dados entre a memória de alta largura de banda (HBM) e os processadores de fluxo (SM) da GPU, fenômeno conhecido como "Memory Wall". A metodologia proposta fundamenta-se no desenvolvimento e implementação de um kernel de atenção fundido (fused attention) utilizando a linguagem de programação de baixo nível OpenAI Triton. A abordagem técnica explora estratégias de tiling e gerenciamento manual da memória estática de acesso aleatório (SRAM), permitindo que operações de produto escalar, softmax e ponderação de valores ocorram em um único ciclo de leitura da memória global. Para a validação dos resultados, realizou-se uma análise comparativa de desempenho entre o kernel customizado e a implementação padrão (Eager mode) do framework PyTorch, utilizando ferramentas de diagnóstico de hardware como o NVIDIA Nsight Compute e o Modelo Roofline. Os resultados experimentais demonstram uma redução significativa na latência de inferência e uma diminuição no tráfego de dados no barramento de memória, deslocando a execução do kernel de um estado limitado por memória (memory-bound) para um estado limitado por processamento (compute-bound). Conclui-se que a programação de kernels de baixo nível e a fusão de operadores são requisitos essenciais para a escalabilidade de modelos de inteligência artificial de larga escala, proporcionando ganhos críticos de eficiência energética e throughput necessários para sistemas de produção de alta disponibilidade.

Palavras-chave: Visão Computacional. Vision Transformers. Otimização de Kernels. OpenAI Triton. Engenharia de Performance.

ABSTRACT

CASTILHO, C. W. **Hardware-Aware Optimization of Vision Transformers: Implementing High-Performance Inference Kernels for SRAM Efficiency**. 2026. 47 p. Monograph (Conclusion Course Paper) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2026.

This work addresses the low-level optimization of inference kernels applied to computer vision models, with a specific focus on the Vision Transformer (ViT) architecture. The primary objective is to mitigate the data transfer bottleneck between High Bandwidth Memory (HBM) and the GPU's Streaming Multiprocessors (SM), a phenomenon commonly known as the "Memory Wall." The proposed methodology is based on the development and implementation of a fused attention kernel using the OpenAI Triton low-level programming language. The technical approach explores tiling strategies and manual management of Static Random Access Memory (SRAM), enabling dot-product, softmax, and value weighting operations to occur within a single global memory read cycle. For performance validation, a comparative analysis was conducted between the custom kernel and the standard PyTorch Eager mode implementation, utilizing hardware diagnostic tools such as NVIDIA Nsight Compute and the Roofline Model. Experimental results demonstrate a significant reduction in inference latency and a decrease in memory bus traffic, shifting the kernel execution from a memory-bound state to a compute-bound state. The study concludes that low-level kernel programming and operator fusion are critical requirements for the scalability of large-scale AI models, providing essential gains in energy efficiency and throughput necessary for high-availability production systems.

Keywords: Computer Vision. Vision Transformers. Kernel Optimization. OpenAI Triton. Performance Engineering.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

ABNT	Associação Brasileira de Normas Técnicas
EESC	Escola de Engenharia de São Carlos
GPU	Graphics Processing Unit
HBM	High Bandwidth Memory
HPC	High Performance Computing
ML	Machine Learning
PDF	Portable Document Format
SM	Streaming Multiprocessors
SRAM	Static Random Access Memory
TCC	Trabalho de Conclusão de Curso
USP	Universidade de São Paulo
ViT	Vision Transformer

LISTA DE SÍMBOLOS

Γ	Letra grega Gama
Λ	Lambda
ζ	Letra grega minúscula zeta
$\in$	Pertence

SUMÁRIO

1 INTRODUÇÃO

O avanço exponencial dos modelos de Aprendizado de Máquina (ML), particularmente na área de Visão Computacional, tem sido impulsionado pela transição de arquiteturas convolucionais para os Vision Transformers (ViTs). Diferente das convoluções, que operam em domínios locais, os ViTs utilizam o mecanismo de autoatenção (Self-Attention) para capturar dependências globais em uma imagem. No entanto, essa capacidade analítica superior impõe um custo computacional elevado e uma demanda intensiva de memória, apresentando desafios significativos para a implantação desses modelos em ambientes de produção de alta performance.

1.1 Contexto e delimitação do tema

Atualmente, o gargalo da computação de alto desempenho (HPC) em Inteligência Artificial não reside apenas na capacidade aritmética das Unidades de Processamento Gráfico (GPUs), mas na eficiência da movimentação de dados. Este fenômeno, conhecido como Memory Wall, ocorre quando a velocidade de transferência entre a Memória de Alta Largura de Banda (HBM) e os Processadores de Fluxo (SM) da GPU é insuficiente para manter os núcleos de processamento (Tensor Cores) ocupados.

No desenvolvimento de kernels de inferência, a maioria dos frameworks de alto nível, como PyTorch e TensorFlow, utiliza implementações genéricas. Embora versáteis, essas implementações frequentemente realizam múltiplos acessos à memória global para operações sequenciais. Esta pesquisa delimita-se à otimização de baixo nível desses processos, focando na camada de atenção de Vision Transformers, utilizando a linguagem de programação de kernels OpenAI Triton para maximizar o aproveitamento da hierarquia de memória do hardware.

1.2 Problema de pesquisa

A implementação padrão da operação de atenção em ViTs possui uma complexidade quadrática em relação ao número de patches da imagem. Durante a inferência, a necessidade de armazenar e ler matrizes intermediárias de atenção da memória global (VRAM) gera uma latência considerável e um consumo energético ineficiente. Surge, portanto, a questão: como o desenvolvimento de kernels customizados e a aplicação de técnicas de fusão de operadores podem reduzir a dependência da memória global e aproximar a performance dos modelos de visão do limite teórico do hardware?

1.3 Objetivos

O objetivo central deste trabalho é desenvolver, implementar e validar um kernel de inferência de alto desempenho para a operação de Self-Attention em Vision Transformers.

Sendo seus objetivos específicos:

- Desenvolver um kernel fundido (fused kernel) utilizando OpenAI Triton que integre as operações de produto escalar, softmax e ponderação de valores em um único ciclo de execução na SRAM;
- Implementar estratégias de tiling para otimizar o carregamento de blocos de dados nos registradores da GPU;
- Realizar um profiling comparativo entre a implementação proposta e o baseline do PyTorch, utilizando métricas de latência, throughput (TFLOPS) e utilização de banda de memória;
- Analisar a eficiência do kernel desenvolvido através do Modelo Roofline para identificar gargalos operacionais remanescentes.

2 DESENVOLVIMENTO

Neste capítulo, apresentam-se os conceitos fundamentais para a compreensão da otimização de kernels de inferência. A discussão abrange desde a arquitetura dos Vision Transformers até as minúcias da hierarquia de memória em Unidades de Processamento Gráfico (GPUs), culminando nas ferramentas de programação de baixo nível.

2.1 Vision Transformers e a Mecânica de Atenção

Os Vision Transformers (ViT) representam uma mudança de paradigma na visão computacional ao transpor a arquitetura de autoatenção, originalmente concebida para processamento de linguagem natural (Vaswani *et al.*, 2017), para o domínio de imagens. Segundo Dosovitskiy *et al.* (2021), o ViT decompõe uma imagem em patches fixos, os quais são tratados como sequências de tokens, permitindo que o modelo aprenda relações globais entre diferentes regiões da cena desde as camadas iniciais.

O cerne dessa arquitetura é o mecanismo de Scaled Dot-Product Attention, definido matematicamente pela Equação 2.1:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

Onde Q, K e V representam as matrizes de Query, Key e Value. Embora eficaz, essa operação apresenta uma complexidade computacional $O(N^2)$, onde N é o número de patches. Como apontam Dao *et al.* (2022), o gargalo dessa operação em GPUs modernas não é o poder computacional bruto, mas sim a latência de memória gerada pela leitura e escrita de matrizes intermediárias de atenção.

2.2 Arquitetura de GPU e a "Parede de Memória" (Memory Wall)

Para entender a necessidade de otimização de baixo nível, é preciso distinguir os diferentes níveis de memória em uma GPU. A memória principal, conhecida como High Bandwidth Memory (HBM), possui alta capacidade, porém latência significativamente superior à memória estática interna do chip, a SRAM.

O desequilíbrio entre a velocidade de processamento e a velocidade de transferência de dados é o que a literatura denomina como Memory Wall. Sobre este desafio, Wulf e McKee (1995) observam:

> [...] a disparidade entre a velocidade dos processadores e a velocidade da memória principal tem crescido exponencialmente. Se essa tendência continuar, a performance dos sistemas será limitada não

> pela frequência de clock do processador, mas pela taxa na qual os dados podem ser entregues a ele.

Dessa forma, otimizar um kernel significa maximizar o tempo que o dado permanece na SRAM (ou Shared Memory), minimizando acessos à HBM. Em implementações padrão de bibliotecas de alto nível, cada etapa da Equação 2.1 resulta em uma escrita e leitura na HBM, o que degrada o throughput do sistema.

2.3 OpenAI Triton

Historicamente, a escrita de kernels otimizados exigia o uso de CUDA C++, uma linguagem de alta complexidade que demanda gerenciamento manual de threads e sincronização de blocos. O OpenAI Triton surge como uma alternativa de abstração que permite a escrita de kernels altamente performáticos utilizando uma sintaxe baseada em Python.

De acordo com Tillet, Kung e Cox (2019), o Triton introduz um paradigma baseado em blocos (tiles), onde o compilador é responsável por otimizar automaticamente o agendamento de tarefas e o uso dos registradores, permitindo que o desenvolvedor foque na lógica de movimentação de dados na hierarquia de memória.

2.4 Fusão de Operadores e Flash Attention

A técnica de fusão de operadores (Operator Fusion) consiste em agrupar múltiplas operações matemáticas em um único kernel executável. No contexto de modelos de visão, isso significa realizar o cálculo do Softmax e a multiplicação por V enquanto os dados de Q ainda residem nos registradores da GPU.

Esta técnica é levada ao estado da arte pelo algoritmo Flash Attention. Como sintetizado por Dao *et al.* (2022), a fusão de operadores permite que o modelo evite a materialização da matriz de atenção $N \times N$ na memória principal, resultando em acelerações que podem chegar a 3× em relação às implementações tradicionais, reduzindo simultaneamente o consumo de memória.

3 CONCLUSÃO

Apresentar as conclusões correspondentes aos objetivos ou hipóteses propostos para o desenvolvimento do trabalho, podendo incluir sugestões para novas pesquisas.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 14724**: informação e documentação: trabalhos acadêmicos: apresentação. Rio de Janeiro, 2011. 11 p.

COMANDOS Básicos do LATEX. Juiz de Fora: UFJF, 2007. Disponível em: http://www.fisica.ufjf.br/vwfme/comandoslatex.html. Acesso em: 25 abr. 2016.

DAO, T. *et al.* FlashAttention: fast and memory-efficient exact attention with IO-awareness. **Advances in Neural Information Processing Systems**, New Orleans, v. 35, 2022.

DOSOVITSKIY, A. *et al.* An image is worth 16x16 words: transformers for image recognition at scale. **International Conference on Learning Representations**, Vienna, 2021.

TILLET, P.; KUNG, H. T.; COX, D. Triton: an intermediate language and compiler for tiled neural network computations. *In*: **Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages**. Phoenix: [*S.l.: s.n.*], 2019. p. 10–19.

UNIVERSIDADE DE SÃO PAULO. Agência USP de Gestão da Informação Acadêmica. **Diretrizes para apresentação de dissertações e teses da USP**: parte I (ABNT). 4. ed. São Paulo: AGUIA, 2020. 75 p. (Cadernos de Estudos, 9). Disponível em: http://www.livrosabertos.sibi.usp.br/portaldelivrosUSP/catalog/view/459/413/2006-1.pdf. Acesso em: 13 ago. 2021.

VASWANI, A. *et al.* Attention is all you need. **Advances in Neural Information Processing Systems**, Long Beach, v. 30, 2017.

WULF, W. A.; MCKEE, S. A. Hitting the memory wall: implications of the obvious. **ACM SIGARCH Computer Architecture News**, v. 23, n. 1, p. 20–24, 1995.

APÊNDICES

APÊNDICE A – APÊNDICE(S)

Elemento opcional, que consiste em texto ou documento elaborado pelo autor, a fim de complementar sua argumentação, conforme a ABNT NBR 14724 (Associação Brasileira de Normas Técnicas, 2011).

Os apêndices devem ser identificados por letras maiúsculas consecutivas, seguidas de hífen e pelos respectivos títulos. Excepcionalmente, utilizam-se letras maiúsculas dobradas na identificação dos apêndices, quando esgotadas as 26 letras do alfabeto. A paginação deve ser contínua, dando seguimento ao texto principal. (**??**)

APÊNDICE B – EXEMPLO DE TABELA CENTRALIZADA VERTICALMENTE E HORIZONTALMENTE

A Tabela 1 exemplifica como proceder para obter uma tabela centralizada verticalmente e horizontalmente.

Tabela 1 – Exemplo de tabela centralizada verticalmente e horizontalmente

Coluna A	**Coluna B**
Coluna A, Linha 1	Este é um texto bem maior para exemplificar como é centralizado verticalmente e horizontalmente na tabela. Segundo parágrafo para verificar como fica na tabela
Quando o texto da coluna A, linha 2 é bem maior do que o das demais colunas	Coluna B, linha 2

Fonte: Elaborada pelos autores.

APÊNDICE C – EXEMPLO DE TABELA COM GRADE

A Tabela 2 exemplifica a inclusão de traços estruturadores de conteúdo para melhor compreensão do conteúdo da tabela, em conformidade com as normas de apresentação tabular do IBGE.

Tabela 2 – Exemplo de tabelas com grade

Coluna A	**Coluna B**
A1	B1
A2	B2
A3	B3
A4	B4

Fonte: Elaborada pelos autores.

ANEXOS

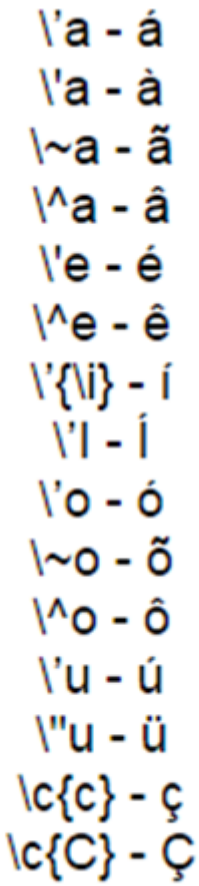
ANEXO A – EXEMPLO DE ANEXO

Elemento opcional, que consiste em um texto ou documento não elaborado pelo autor, que serve de fundamentação, comprovação e ilustração, conforme a ABNT NBR 14724. (Associação Brasileira de Normas Técnicas, 2011).

O **ANEXO B** exemplifica como incluir um anexo em pdf.

ANEXO B – ACENTUAÇÃO (MODO TEXTO - LaTeX)

Figura 1 – Acentuação (modo texto - LaTeX)



Fonte: Comandos [. . .] (**??**)

ÍNDICE