Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Uma Análise sobre Gasto de Energia e Emissão de Poluentes nas Aplicações de Aprendizado de Máquina

Luca Diogo da Silva

Monografia Final

mac 499 — Trabalho de Formatura Supervisionado

Supervisor: Prof. Dr. Alfredo Goldman vel Lejbman

São Paulo
2025

Resumo

Luca Diogo da Silva. **Uma Análise sobre Gasto de Energia e Emissão de Poluentes nas Aplicações de Aprendizado de Máquina**. Monografia (Bacharelado). Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2025.

A atual expansão do mercado de modelos de aprendizado de máquina (ML — machine learning) esteve ancorada não só no desenvolvimento de modelos mais sofisticados para resolver problemas complexos, como o uso de Transformers para problemas de processamento de linguagem natural e geração de imagens, mas também no aumento quantitativo no número de parâmetros dos modelos. Para servir esses grandes modelos e atender a cada vez mais usuários, novos *datacenters* estão sendo construídos no mundo todo, tornando necessário analisar o impacto ecológico dessa expansão. Essa análise não é trivial porque os dados de consumo das maiores aplicações de ML — geralmente modelos de linguagem em aplicações comerciais — raramente são divulgados de forma detalhada, assim como configurações do modelo e de *hardware* que permitiriam a replicação dos processos de treinamento e inferência. Neste trabalho, avaliou-se o gasto de energia e consequente emissão de gases poluentes pelas implementações de modelos de ML, explorando modelos para o cálculo dessas emissões de forma independente, e métodos para reduzí-las.

Palavras-chave: energia. emissões de poluentes. dióxido de carbono. aprendizado de máquina. grandes modelos de linguagem. inteligência artificial.

Abstract

Luca Diogo da Silva. **Uma Análise sobre Gasto de Energia e Emissão de Poluentes nas Aplicações de Aprendizado de Máquina**. Capstone Project Report (Bachelor). Institute of Mathematics and Statistics, University of São Paulo, São Paulo, 2025.

The current expansion of the machine learning model market has been based not only on the development of more sophisticated models capable of solving complex problems, such as the use of Transformers for natural language processing problems and image generation, but on a rise of the models' numbers of parameters. To serve these larger models to an increasing amount of users, new datacenters are being built all over the worlds, which makes analysing the ecological impact of this expansion a necessary effort. This analysis is far from trivial because usage and resource consumption data of the major applications of ML — usually language models in commercial applications — are hardly ever published with appropriate detail; neither are the model and hardware settings that would make replicating the training and inference processes possible. In this work, we study the energy consumption and the resulting emission of polluting gases by implementations of ML models, exploring ways to calculate those emissions independently, and also ways to reduce them.

Keywords: energy. pollutant emissions. carbon dioxide. machine learning. large language models. artificial intelligence.

Lista de abreviaturas

ML	Aprendizado de Máquina (*Machine Learning*)
IA	Inteligência Artificial
LLM	Grande Modelo de Linguagem (*Large Language Model*)
ACV	Avaliação de ciclo de vida
NLP	Processamento de Linguagem Natural (*Natural Language Processing*)
GPT	*Transformers* generativos pré-treinados (*Generative Pre-Trained Transformers*)
MoE	Mistura de Especialistas (*Mixture of Experts*)
Wh	Watts-hora
IC	Intensidade de Carbono
REC	Certificado de Energia Renovável (*Renewable Energy Certificate*)
EUA	Estados Unidos da América
ISO	*International Organization for Standardization*
TSMC	*Taiwan Semiconductor*
CPU	Unidade de Processamento Central (*Central Processing Unit*)
GPU	Unidade de Processamento Gráfico (*Graphics Processing Unit*)
TPU	Unidade de Processamento de Tensores (*Tensor Processing Unit*)
FPGA	Matriz de Portas Programáveis em Campo (*Field Programmable Gate Array*)
EPA	Energia por Área
GPA	Gases por Área
MPA	Emissões de Matéria-Prima por Área
HDD	Unidade de Disco Rígido (*Hard Disk Drive*)
SSD	Unidade de Estado Sólido (*Solid State Drive*)
RAM	Memória de Acesso Arbitrário (*Random Access Memory*)
STEC	Carbono Incorporado Espacial-Temporal (*Spatial-Temporal Embodied Carbon*)
MAC	Multiplicação-Acumulação (*Multiply-Accumulate*)
DSPs	Processador de Sinais Digitais (*Digital Signal Processor*)
SLC	Células de Nível Único (*Single Level Cells*)
MLC	Células de Nível Múltiplo (*Multiple Level Cells*)

FLOP	Operação de Ponto Flutuante (*Floating-Point Operation*)
ReLU	Unidade Linear Retificada (*Rectified Linear Unit*)
MFU	Utilização de FLOPs pelo Modelo (*Model FLOP Utilization*)
CNN	Rede Neural Convolucional (*Convolutional Neural Network*)
SMD	*Stochastic Mini-Batch Dropping*
SLU	Atualização Seletiva de Camadas (*Selective Layer Update*)
PSG	Gradiente Descendente de Sinal Preditivo (*Predictive Sign Gradient Descent*)
PEP	Produto de Perplexidade e Gasto Energético (*Perplexity-Energy Product*)
FFT	Transformada Rápida de Fourier (*Fast Fourier Transform*)
ONU	Organização das Nações Unidas
MIG	GPUs de múltiplas instâncias (*multi-instance GPU*)

Lista de figuras

Lista de programas

Sumário

Introdução

A inteligência artificial (IA), em particular as técnicas de aprendizado de máquina (*machine learning* — ML), têm sido uma das principais áreas de pesquisa na computação dos últimos anos, pois o uso das redes neurais, com sua capacidade de aproximar qualquer função contínua, é o melhor método encontrado até agora para a resolução computacional de vários problemas, como o reconhecimento de imagem e áudio, predição da estrutura tridimensional de proteínas e o processamento de linguagem natural (*natural language processing* — NLP).

Essa última aplicação recebeu um salto qualitativo em 2017 com a publicação do artigo *Attention Is All You Need*, em que pesquisadores da Google Brain descreveram pela primeira vez os modelos *Transformers*, cujo mecanismo de "atenção" (*attention*) se mostrou mais eficiente na codificação de relações sintáticas e semânticas entre as palavras — ou *tokens* — de um texto quando comparados aos modelos com recorrência, até então considerados estado-da-arte (Vaswani *et al.*, 2023). Esses modelos, combinados aos modelos de difusão usados na geração de imagens e vídeos, são a base da chamada "IA generativa" e estão no centro do último *boom* mercadológico da IA que se iniciou ao final de 2022, quando a organização OpenAI disponibilizou o ChatGPT, um *chatbot* baseado nos modelos proprietários GPT (*generative pre-trained Transformers* — *Transformers* generativos pré-treinados).

Desde então, segundo pesquisa anual da empresa de consultoria McKinsey, a porcentagem de empresas que relataram, nas pesquisas anuais da consultoria, usar IA em pelo menos um de seus processos internos cresceu de 50% a 88% entre 2022 e 2025, e, especificamente, de 33% para 79% no uso de IA generativa (Singla *et al.*, 2025), como mostra a Figura 1. Já a OpenAI relatou que entre 2024 e 2025, o número de *prompts* diários aumentou mais de cinco vezes, de 451 milhões para 2,63 bilhões *prompts* feitos por 700 milhões de usuários (Chatterji *et al.*, 2025), enquanto o Gemini, concorrente do ChatGPT desenvolvido pela Google, atinge 350 milhões de usuários por mês, como descrito em documentos internos da empresa publicados durante o julgamento antitruste em curso nos Estados Unidos (Diaz, 2025).

Além do aumento do número de usuários, e, portanto, de inferências, os modelos que compõem os mais populares produtos de IA também estão cada vez maiores. Em 2015, antes do advento dos *Transformers*, engenheiros da Digital Reasoning já haviam divulgado a criação de um modelo de 160 bilhões de parâmetros para NLP (Trask *et al.*, 2015). O GPT-3, de 2020, base das primeiras versões do ChatGPT, foi o último modelo da OpenAI a ter a sua quantidade de parâmetros publicada: 175 bilhões de parâmetros. Estima-se que

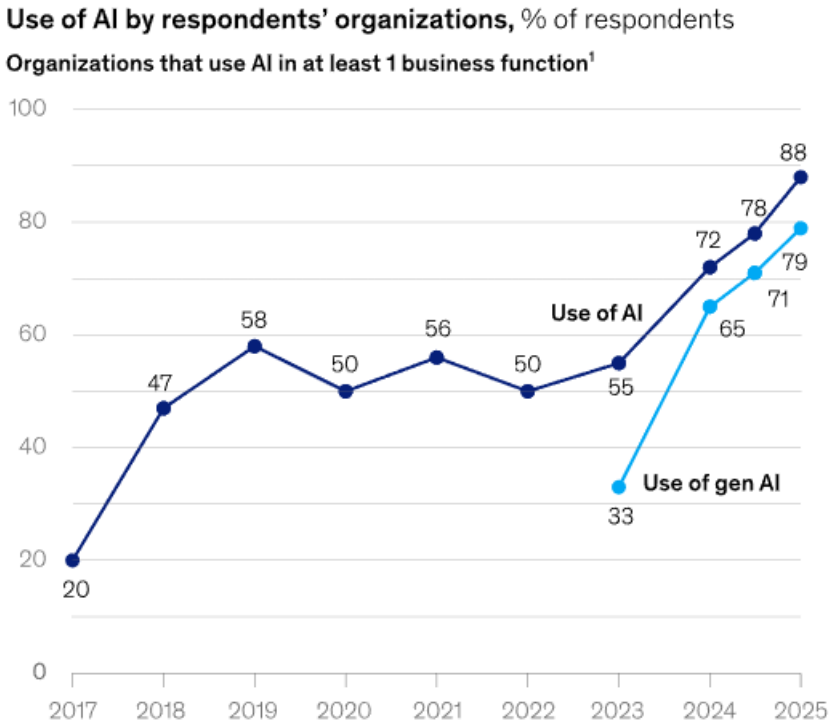


Figura 1: *Uso de ferramentas de IA por empresas entrevistadas pela McKinsey (Singla et al., 2025)*

tanto os GPTs 4 e 5, quanto seus concorrentes de performance similar como o Google Gemini e Anthropic Claude Opus, tenham parâmetros na ordem de trilhões, tomando como referência os modelos DeepSeek v3, com 671 bilhões de parâmetros (DeepSeek-AI *et al.*, 2025), e Qwen3-Max, da Alibaba Cloud, com "mais de 1 trilhão" (Team, 2025). Esses modelos estado-da-arte, por via de regra, usam apenas uma parte desses parâmetros em cada inferência, lançando mão do mecanismo de mistura de especialistas (*mixture-of-experts* - MoE), onde — como mostra a Figura 2 — na camada final dos Transformers, a grande rede neural que gera o próximo *token* é substituída por um conjunto de redes menores, das quais apenas uma é selecionada pelo modelo a cada *token*, a partir de uma outra rede neural anterior — chamada de roteadora (*router*) — geradora de probabilidades para cada especialista. Ainda assim, o número de parâmetros ativos em uma inferência é provavelmente crescente entre os modelos comerciais.

Para servir esses modelos cada vez mais computacionalmente caros para mais usuários, há uma grande expansão na quantidade e tamanho de *datacenters* pelo mundo, resultando num aumento no consumo total de energia. De acordo com outra pesquisa da McKinsey, a demanda global por potência energética de *datacenters* pode crescer de 55 gigawatts para 171 a 298 gigawatts entre 2025 e 2030, ou seja, crescimentos anuais entre 19% e 27% (Srivathsan *et al.*, 2024). Já o Lawrence Berkeley National Laboratory estimou em 2024 que o consumo total de energia por instalações nos EUA ir de 176 TWh (terawatts-hora) por ano para 580 TWh, representando 12% do consumo no país (Shehabi *et al.*, 2024).

O alto consumo energético pode ter um grande impacto ecológico caso a fonte energética utilizada tenha alta intensidade de carbono (IC), isto é, emitem mais dióxido de carbono (CO_2) por watt-hora de energia produzida. A região norte do estado da Virginia,

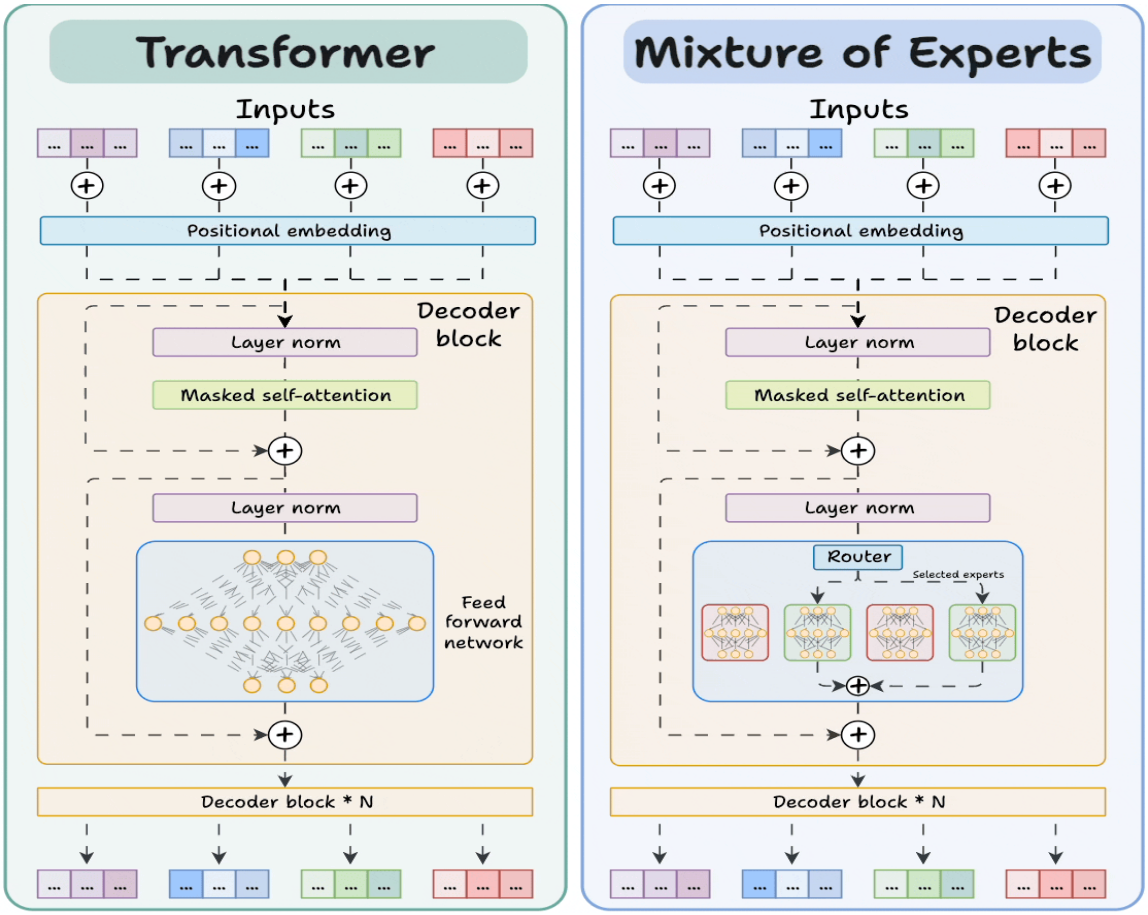


Figura 2: *Diferença entre blocos de decodificação de* Transformers *tradicionais e com MoE (*Chawla, 2025*)*

onde, de acordo com a Synergy Research Group, está localizado 14% dos *datacenters* de hiperescala, medido em potência energética, é bastante dependente de fontes de alta IC: por volta de 62% da energia utilizada no estado vem do gás natural (*Synergy Identifies the World's Top 20 Locations for Hyperscale Data Centers* 2024).

É comum empresas tentarem compensar essas emissões pela compra de certificados de energia renováveis (*renewable energy certificate* — REC), que comprovam um investimento e uso de energia renovável nas instalações da empresa. Contudo, como um REC pode ser emitido em qualquer parte dos EUA, a sua emissão não significa uma mudança na provisão de energia nos grandes pólos de *datacenters*, logo o impacto ecológico ainda existe e é fortemente concentrada em pequenas áreas. Uma reportagem do The Guardian apurou que, em 2023, quando se retira as compensações por RECs, as emissões de CO_2 por *datacenters* da Meta aumentam em 3100 vezes, enquanto as da Apple aumentam em 402 vezes e as da Microsoft em 21, ou seja, políticas compensatórias como RECs e créditos de carbono, apesar de valiosas, são insuficientes para lidar com emissões bastante localizadas (O'Brien, 2024).

Diante deste cenário de crescimento acelerado e incertezas em relação ao impacto ecológico presente e futuro das grandes implementações de IA/ML, este trabalho se propõe a analisar formas de estimar o gasto energético das aplicações e diminuir suas emissões em duas principais frentes:

- o carbono incorporado, isto é, emissões de gases poluentes durante o processo produtivo do *hardware*
- o carbono operacional, ou seja, emissões a partir do uso do *hardware*, equivalente às emissões das fontes da energia fornecida a ele

Capítulo 1

Carbono Incorporado

Iniciaremos nossa análise pelo processo de produção do *hardware* utilizado nas aplicações de ML, com um foco particular nos equipamentos estado-da-arte utilizados nos maiores *datacenters*, chamados de hiperescaladores ou *hyperscalers*. Segundo um estudo da McKinsey de 2024, até 2030, essas instalações concentrarão cerca de 85% das implementações de IA e ML, sendo aproximadamente 25% de modelos próprios e 60% de modelos de terceiros a partir do fornecimento de ambientes em nuvem. (Srivathsan *et al.*, 2024)

Para isso, usa-se o conceito de **carbono incorporado** (*embodied carbon*), que corresponde ao cálculo de todo o gasto energético, na forma de emissão de CO_2 e outros gases provocadores de efeito-estufa, na produção de certo objeto, desde a extração das matérias-primas necessárias, manufatura, montagem e transporte da fábrica para os consumidores. Cálculos mais robustos incluem também os gastos no descarte do objeto após o fim de sua vida útil (Hammond e Jones, 2009).

O cálculo para peças de *hardware* tende a ser bastante complexo, devido à dificuldade de acesso aos dados de todas as empresas envolvidas no processo de fabricação de uma peça. Dessa forma, ao invés de desenvolver análises totalmente independentes, geralmente se utiliza os dados produzidos pelos próprios fabricantes, como as avaliações de ciclo de vida (ACV) feitas por empresas como Dell e Seagate, para seus produtos. Evidentemente, isso torna o estudo do carbono incorporado suscetível a subnotificação. Para evitar esse problema, a organização ISO (*International Organization for Standardization*) criou a série de padrões 14040, definindo metodologias para o ACV. No entanto, ainda não há um padrão específico para *hardware*. A Figura 1.1 ilustra a gama diversa de informações utilizadas para estimar o carbono incorporado, por meio do modelo ACT, que será explorado na seção seguinte.

1.1 Modelos de Cálculo e Estimativa

Dadas as limitações do método ACV, novos métodos foram desenvolvidos na academia e indústria para obter informações mais corretas sobre o carbono incorporado no *hardware* computacional.

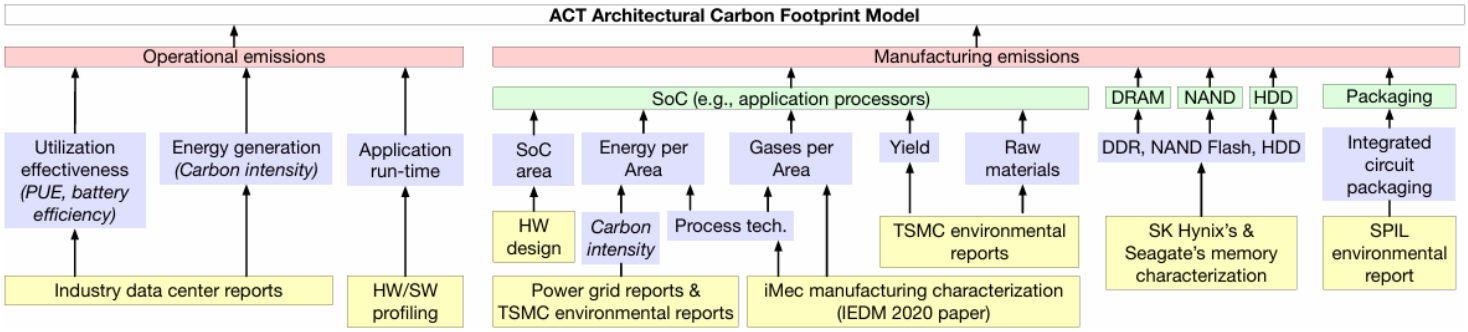


Figura 1.1: *Diagrama de dados que compõem o modelo ACT (Gupta et al., 2022)*

O primeiro método que analisaremos é o ACT (Gupta *et al.*, 2022), publicado em 2022 a partir de uma parceria de pesquisadores da Universidade de Harvard e da Meta. Com o intuito de promover a projeção e desenvolvimento de *hardware* menos poluente, o modelo busca utilizar dados mais especializados do que aqueles divulgados pelo produtor final, como dados publicados pela *Taiwan Semiconductor* (TSMC) — uma das maiores produtoras de semicondutores do mundo, que tem empresas como Apple, Nvidia, Qualcomm e Sony como clientes.

As emissões são divididas em três tipos:

- emissões de energia, relacionadas à geração da energia elétrica consumida nos processos produtivos (portanto, dependentes da intensidade de carbono das fontes).
- emissões de gases, relacionadas a gases resultantes da queima de produtos químicos (exceto combustíveis, que se encaixariam na categoria anterior).
- emissões de matéria-prima, originadas na extração e transporte destas.

Para unidades de processamento como CPUs, GPUs e FPGAs, compostas dos mesmos semicondutores organizados de forma diferente, considera-se que as emissões E são proporcionais à àrea de processamento da peça:

$$E = \frac{1}{Y} * \text{área} * (IC * EPA + GPA + MPA)$$

Onde:

- Y — a eficiência (*yield*) da fábrica (um número entre 0 e 1 representativo da proporção de semicondutores que não são descartados durante a produção)
- IC — intensidade de carbono da fonte de energia
- EPA — energia por área do semicondutor
- GPA — emissões de gases por área
- MPA — emissões de matéria-prima por área

Já para unidades de armazenamento (HDDs, SSDs e memória RAM), as emissões E são proporcionais à capacidade de armazenamento, em *bytes*, por meio da medida carbono por byte (CPB):

$$E = CPB * \text{capacidade em } bytes$$

Com essas definições, além de estimar o carbono incorporado em um computador ou *datacenter*, é possível fazer previsões sobre o impacto da substituição de um certo componente, ou de mudanças no próprio processo produtivo. Por exemplo, os autores estimam que, comparada às fábricas da TSMC, bastante dependentes de combustíveis fósseis, uma fábrica de semicondutores utilizando energia 100% renovável teria uma redução de aproximadamente 66% nas emissões de energia por área de chip ao produzir pastilhas de 3nm (o menor tamanho comercializado hoje), e de aproximadamente 60% para pastilhas de 10nm, ainda utilizadas em processadores Intel Raptor Lake (Figura 1.2). Ou ainda, pastilhas de 3nm e 5nm gastam quase a mesma quantidade de energia por cm² produzido, porém gastam 28% a mais que pastilhas de 7nm feitas com a mesma tecnologia (litografia ultravioleta extrema).

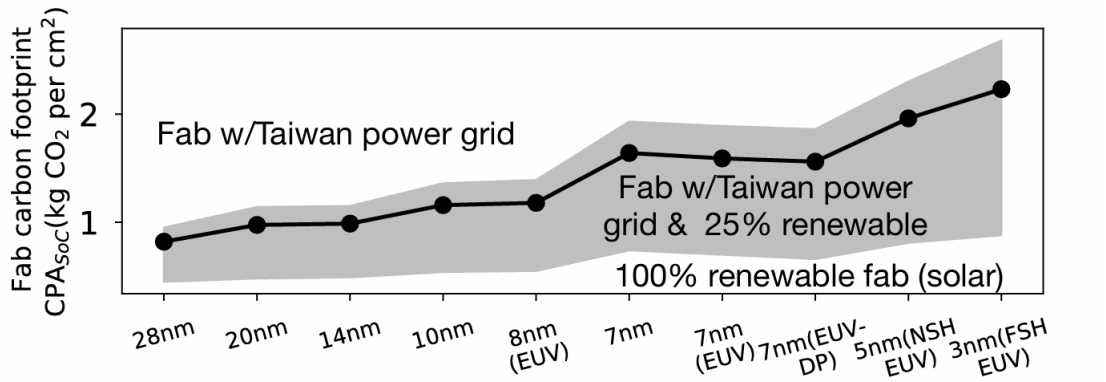


Figura 1.2: *Emissão de CO_2 por tecnologia de semicondutor, considerando variações no fornecimento de energia renovável (Gupta et al., 2022)*

O modelo *Spatial-Temporal Embodied Carbon* (STEC) (Zhang *et al.*, 2024) estende ACT ao considerar o tempo e espaço onde certo componente foi produzido, já que a disponibilidade de energia de fontes diferentes é fortemente relacionada ao local geográfico, assim como certas fontes de energia renovável, como solar e eólica, também variam suas efetividades conforme as estações do ano.

Sendo assim, o modelo STEC propõe análises em três níveis de granularidade:

- ano de fabricação e zona do planeta, baseado em grupos de países que compartilham um mesmo tratado energético (STEC-ZY)
- estação do ano e país de fabricação (STEC-CS)
- dia do ano e país de fabricação (STEC-CD)

Essa divisão dá enfoque às emissões de energia, pois, segundo os autores, as outras duas formas de emissão são menos suscetíveis à variação de espaço e tempo.

A maior granularidade em relação ao ACT gera medições de carbono incorporado bastante distintas. Analisando o processo produtivo de 6 dos maiores fabricantes de semicondutores, com produção em Taiwan, EUA, Itália, Irlanda, China e Coreia do Sul, a divergência média entre STEC-CD e o ACT para o ano de 2021 foi de 10,88%, e a máxima de 40,54%. Para os modelos menos granulares, a variação média aumenta, chegando a 18,01% entre STEC-ZY e ACT. A figura 1.3 ilustra essa divergência mostrando emissões na produção de CPUs em fábricas italianas e irlandesas. Ambos os locais possuem fontes de energia limpa, mas com disponibilidade variável ao longo do ano: na Itália, a energia solar é mais forte durante o verão, enquanto na Irlanda, há potencial de energia eólica, mais forte nos invernos. Mesmo com essa variação, a média resultante do carbono incorporado nas duas localidades é menor do que a média encontrada por ACT — aqui representado por STEC-GY (*global-year* — ano de fabricação e global).

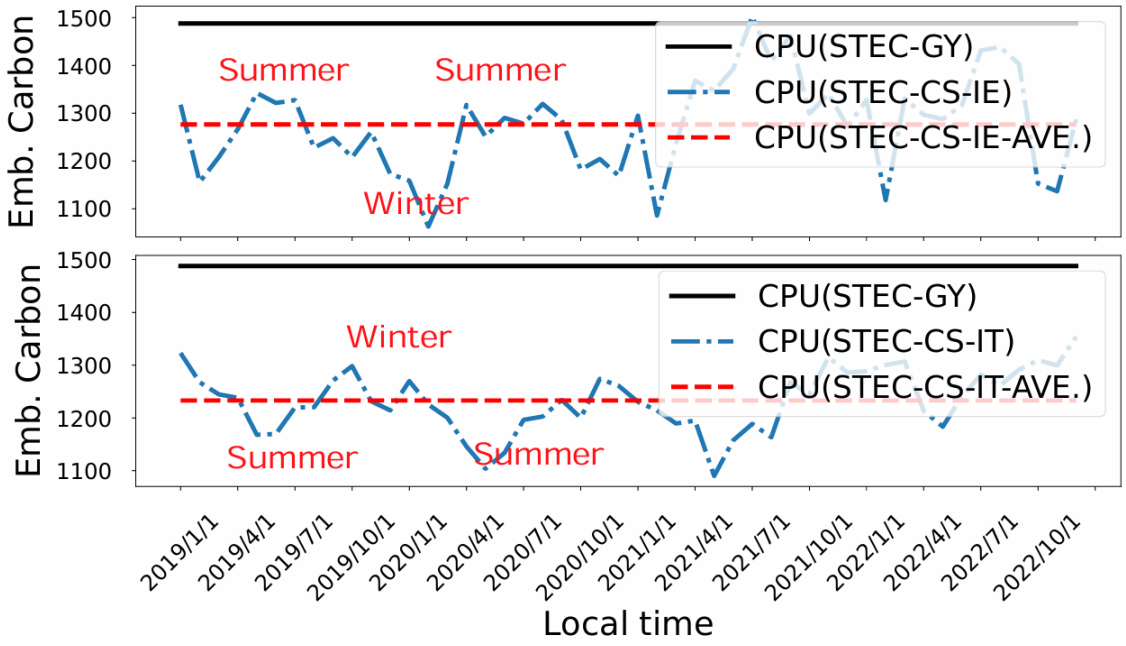


Figura 1.3: *Variação no carbono incorporado na produção de CPUs em fábricas na Irlanda (IE) e Itália (IT) durante o ano (Zhang et al., 2024)*

Além de maior precisão na análise do presente, STEC pode auxiliar no planejamento desses fabricantes na distribuição de sua produção e instalação de novas fábricas. Por exemplo, a já citada abundância de energia eólica no inverno irlandês faz com que fábricas lá instaladas, como o campus fabril da Intel Leixlip, tenha menos emissões energéticas que fábricas em Taiwan, onde, em grande parte, se utiliza o carvão como fonte de energia.

Segundo Li et al., a maior parte do carbono incorporado nos grandes *datacenters* costuma ser o das GPUs. (B. Li, Basu Roy *et al.*, 2023) Os servidores destinados a operações de ML um número maior de GPUs comparada às CPUs, em proporções 2:1 como no NVIDIA GB200 NVL72, base do *datacenter* Fairwater da Microsoft, ou 4:1 como no JUPITER Booster,

datacenter considerado o quarto mais eficiente do mundo em termos de operações de ponto flutuante por segundo (FLOPS), na lista TOP500 de Junho de 2025. (TOP500, 2025)

A preferência por esses componentes se justifica pelas suas maiores eficiências nos cálculos matemáticos simples, como multiplicações de matrizes e de tensores, essenciais nos processos de ML, quando comparados às tradicionais CPUs. Quando comparamos uma GPU e uma CPU estados-da-arte, como, respectivamente, a AMD MI250X e a AMD EPYC 7763, a primeira apresenta aproximadamente duas vezes mais carbono incorporado que a segunda (B. Li, Basu Roy *et al.*, 2023). Porém, ao dividirmos essas quantidades pelo número máximo de operações de ponto flutuante (FLOP) por segundo — uma maneira de avaliar o equilíbrio entre performance e emissões no processo produtivo — a GPU passa a ser nove vezes mais eficiente nesse aspecto.

No entanto, os autores destacam que componentes de armazenamento de dados, como HDDs e SSDs, também representam uma parte significativa dessas emissões. Por exemplo, 36% do carbono incorporado ao Frontier, número 2 da TOP500, está nas suas GPUs, enquanto 30% está nos HDDs e 12% nos SSDs. Pelo estudo de Zhang et al., as emissões de ambos os componentes são similares em termos brutos, sendo que HDDs são vantajosos em relação à capacidade de armazenamento e os SSDs vantajosos na velocidade na transferência (leitura e escrita) de dados. (Zhang *et al.*, 2024)

1.2 Formas de Redução

Além de apresentar o método ACT, Gupta et al. também explana algumas formas de reduzir a emissão do carbono incorporado, com foco em sistemas móveis, mas que são aplicáveis para casos gerais. (Gupta *et al.*, 2022)

Primeiramente, os autores propõem um método de desenvolvimento de *hardware* especializado onde se fixam as métricas de desempenho e busca-se, como num problema de otimização, minimizar a emissão de carbono incorporado. Usando como base a arquitetura de *hardware* aberto NVDLA, desenvolvido pela NVIDIA, fixado o problema de processar 30 imagens de mesma resolução por segundo, os autores exploram os possíveis números de unidades de computação aritmética básica (unidades MAC) resolvem o problema, ao mesmo tempo que otimizam métricas de carbono.

Nesse exemplo, como mostra a figura 1.4, 256 MACs são suficientes para atingir o processamento necessário, e é o valor ótimo, entre potências de dois, encontrado pelos autores. 2048 MACs é o valor que otimiza a performance, aproximadamente nove vezes a mais que o objetivo de 30 imagens por segundo, mas a um custo de emissão incorporada 3.3 vezes maior.

O conceito proposto de desenvolvimento de *hardware* com objetivos de performance bastante específicos parece promissor, e é pouco abordado na literatura acadêmica, porém sua viabilidade no contexto atual pode ser limitada pois as indústrias que circundam o mercado de ML têm operado em uma lógica de corrida tecnológica, tratando a performance como o principal atrativo comercial. Dessa forma, essa mudança de estratégia por parte dos fabricantes de *hardware* pode ser bastante custosa, e caso não seja adotada pelos consumidores desses produtos — empresas que implementam modelos de ML e os

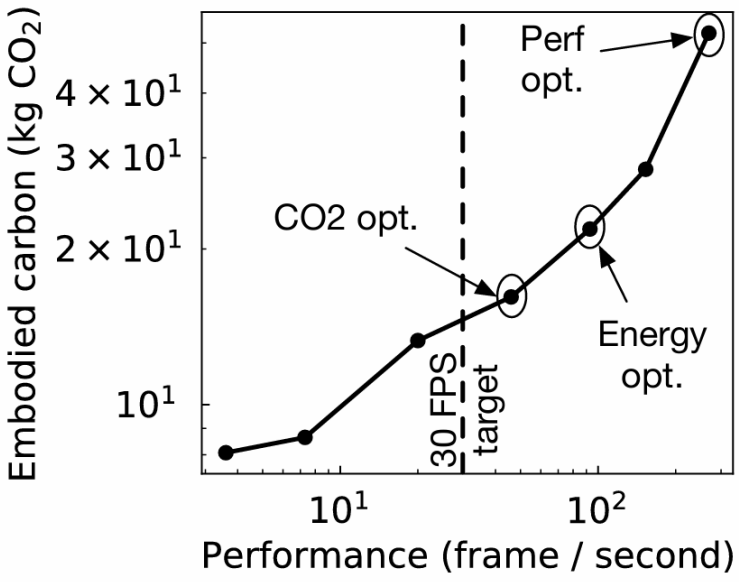


Figura 1.4: *Relação entre carbono incorporado e performance das unidades MAC. Os pontos destacados são quantidades de MACs em potências de dois, de 32 a 2048 (Gupta et al., 2022)*

incorporam em seu *software* —, a tendência é que os componentes sejam substituídos com maior frequência, anulando os benefícios dessa forma de redução.

Os autores também propõem duas formas de reutilização de *hardware*: balancear o uso de CPUs e *hardware* especializado, como GPUs, FPGAs e DSPs, e estender a vida útil dos componentes.

Já foi descrito que GPUs carregam mais carbono incorporado nos seus processos produtivos que as CPUs, e que se entende, do ponto de vista do consumidor, que tanto essas emissões quanto as operacionais são compensadas pela maior performance nas tarefas de ML. No entanto, essa interpretação é bastante limitada: os benefícios dos *hardware* específicos não são inerentes à sua existência, mas surgem da sua utilização correta e bem-projetada. Assim, são necessários experimentos e cálculos para definir quais componentes usar, e como usá-los, de forma a evitar a subutilização e minimizar as emissões.

Por exemplo, supondo uma GPU e uma CPU com mesmas quantidades de carbono incorporado, um servidor com 4 dessas GPUs utilizado em 50% possui 67% mais carbono incorporado do que o necessário, ou seja, que um servidor com 2 GPUs utilizado em 100%, sem considerar as emissões operacionais das placas subutilizadas. O mesmo princípio de balanço pode ser aplicado a SSDs e HDDs: SSDs apresentam mais carbono incorporado que HDDs de mesma capacidade, mas possuem menor carbono operacional e maior velocidade nas operações de leitura e escrita. A figura 1.5 demonstra a diferença nas emissões dessas peças, supondo uma potência média de 4.2 W para um HDD e 1.3 W para um SSD.

Ademais, Wadenstein e Vanderbauwhede explicam que substituir um SSD de 500 GB por outro de 10TB incorre em um aumento de 842 $kgCO_2$, 70% do total do servidor utilizado de exemplo. Nesse contexto, é necessário que a utilização do novo armazenamento seja maior que 8,5% para que a substituição seja compensatória ecologicamente. (Wadenstein e Vanderbauwhede, 2025)

Storage	Energy (KWh)		OPEX CO2e (Kg)		CAPEX CO2e (Kg)		Total CO2e (Kg)	
Exp. Life	5yr	10yr	5yr	10yr	5yr	10yr	5yr	10yr
HDD (1TB)	183.9	367.9	79.6	159	20	40	99.6	199
SSD (1TB)	56.9	113.8	24.6	49.2	160	320	184	369.2

Figura 1.5: *Carbono incorporado em HDDs e SSDs de 1TB (Tannu e Nair, 2023)*

Mais adiante, é necessário refletir sobre a extensão da vida útil dos componentes, de forma a diminuir a demanda, e a consequente produção, de novas peças. Os dispositivos de armazenamento (HDDs e SSDs) são os mais críticos nesse aspecto, pois o limitante de suas vidas úteis não é apenas a lentidão em relação ao início desse período ou às peças de tecnologia mais recente, mas também a corrupção dos dados armazenados.

Uma maneira de estender a vida útil de um SSD é aumentar o seu provisionamento excessivo, isto é, capacidade de armazenamento que não é utilizada diretamente pelo usuário, mas pelo *firmware* do dispositivo para reduzir a quantidade de erros de leitura e escrita. Supondo que há manutenção no espaço útil do SSD, aumentar seu provisionamento excessivo é, evidentemente, aumentar o carbono incorporado. Por isso, busca-se um equilíbrio entre esses dois fatores. Gupta et al. encontram que o valor ótimo para uma vida útil de 2 anos é 16%, e 34% para 4 anos. (Gupta *et al.*, 2022)

Tannu e Nair propõem outras maneiras de extensão, como a implementação de algoritmos mais potentes de correção de erros, protegendo o sistema contra corrupção de dados, ao custo de maior latência — e, portanto, gasto energético — na leitura e escrita; o uso conjunto de células de nível único (SLC), excelentes na durabilidade e velocidade, e de nível múltiplo (MLC), excelentes na capacidade de armazenamento e a transformação de dispositivos MLC em SLC para tarefas menos exigentes, quando erros se tornarem frequentes. (Tannu e Nair, 2023)

Capítulo 2

Carbono Operacional

2.1 Modelos de Cálculo e Estimativa

Recentemente, duas das grandes empresas do mercado de ML divulgaram o custo médio da inferência de suas aplicações. Em junho de 2025, Sam Altman, CEO da OpenAI, afirmou em um post de seu *blog* pessoal que o processamento médio de um *prompt* pelo ChatGPT gasta 0,34 Wh (Altman, 2025), enquanto a Google afirmou em um artigo que o custo médio por *prompt* do Google Gemini é de 0,24 Wh, com emissão de 0,02 gCO_2e (Elsworth *et al.*, 2025).

A publicação de Altman não contém metodologia ou dados que embasem a estatística publicada, mas o artigo publicado pela Google detalha a metodologia de cálculo, medindo a energia gasta pelos "aceleradores de IA" (TPUs e GPUs, provavelmente), CPU e DRAM, além da energia gasta pelos sistemas externos aos computadores, como os de resfriamento. Apesar disso, o artigo não apresenta as medidas capturadas ou as configurações dos *datacenters* para a replicação e verificação dos resultados. Ou seja, como ainda há muita limitação de dados e de acesso aos ambientes dessas grandes aplicações, é preciso desenvolver formas de estimar o carbono operacional de forma independente.

Schwartz et al. argumentam que a melhor métrica para calcular as emissões de CO_2 de um modelo de ML é o número de operações em ponto flutuante (FLOPs) realizadas durante o treinamento, ou durante uma inferência. (Schwartz *et al.*, 2020) A partir desse valor, podemos, a partir das especificações de *hardware*, determinar a quantidade de energia gasta, em watts-hora (Wh) e sabendo a intensidade de carbono das fontes de energia utilizadas, determinar a quantidade de CO_2 emitido.

Os níveis de suporte para esse cálculo nos *frameworks* de ML são variados. Além de pacotes independentes desenvolvidos pela comunidade, como (Sovrasov, 2024) e (He, 2022), o PyTorch (Paszke *et al.*, 2019), principal biblioteca de ML em Python, oferece a opção "with_flops" em seu módulo *profiler*, para calcular os FLOPs durante a execução. No entanto, as únicas operações analisadas, até agora, são soma e multiplicação de matrizes, e convoluções bidimensionais, sem suporte para funções de ativação tal qual Softmax e ReLU, ou para convoluções tridimensionais, bastante usadas em análises de vídeo, por exemplo.

A biblioteca JAX (Bradbury *et al.*, 2018), também bastante popular, oferece estimativas de custo pela função "cost_analysis" antes da compilação dos modelos.

Muitos autores também se dedicaram a calcular, de forma prévia, o número de operações de modelos de MLs. Para *Transformers*, por exemplo, usa-se a aproximação de $6N$ operações por *token*, sendo P o número de parâmetros do modelo. O fator escalar 6 vem do fato de uma inferência (ou *forward-pass*) custar aproximadamente $2p$, e uma atualização de parâmetros (*backward-pass*) custar o dobro de um *forward-pass*, logo $4p$. Hoffmann et al. oferece um cálculo mais detalhado da inferência para esses modelos.

Para os *embeddings*, temos

$$FLOP_{\text{embed}} = 2 * n_{\text{ctx}} * n_{\text{vocab}} * d_{\text{modelo}}$$

onde n_{ctx} expressa o tamanho da janela de contexto, n_{vocab} o tamanho do vocabulário —*tokens* passíveis de serem gerados pelo modelo — e d_{modelo} a dimensionalidade da representação vetorial (o *embedding*) dos *tokens*.

As camadas de atenção podem ser decompostas em cinco operações:

- a projeção da entrada nos espaços Q, K e V (*query, key, value*) tem custo

$$FLOP_{\text{QKV}} = 2 * 3 * n_{\text{ctx}} * d_{\text{modelo}} * d_{\text{key}} * n_{\text{camadas}}$$

- o produto QK tem custo

$$FLOP_{\text{prodQK}} = 2 * n_{\text{ctx}}^2 * d_{\text{key}} * n_{\text{camadas}}$$

- o Softmax após o produto QK tem custo

$$FLOP_{\text{softQK}} = 3 * n_{\text{ctx}}^2 * n_{\text{camadas}}$$

- a redução (multiplicação pela matriz V) custa

$$FLOP_{\text{prodV}} = 2 * n_{\text{ctx}}^2 * d_{\text{key}} * n_{\text{camadas}}$$

- por fim, a projeção de volta às dimensões do modelo custa

$$FLOP_{\text{proj}} = 2 * n_{\text{ctx}} * d_{\text{modelo}} * d_{\text{key}} * n_{\text{camadas}}$$

onde n_{camadas} representa o número de camadas de atenção (ou *attention heads*) e d_{key} é a dimensão da matriz-chave (*key* ou K).

Depois, a camada de rede neural tem suas operações expressas por

$$FLOP_{\text{neural}} = 2 * 2 * n_{\text{ctx}} * d_{\text{modelo}} * d_{\text{neural}}$$

sendo d_{neural} a dimensão da saída da rede neural. O último passo é realizar um outro

Softmax de custo

$$FLOP_{\text{softmax}} = 2 * n_{\text{ctx}} * n_{\text{vocab}} * d_{\text{modelo}}$$

Comparando essas fórmulas com a estimativa $6p$, Hoffmann et al. encontra que, para modelos reais, a diferença é pequena, como mostra a figura 2.1.

Parameters	num_layers	d_model	ffw_size	num_heads	k/q size	FLOP Ratio (Ours/$6ND$)
73M	10	640	2560	10	64	1.03
305M	20	1024	4096	16	64	1.10
552M	24	1280	5120	10	128	1.08
1.1B	26	1792	7168	14	128	1.04
1.6B	28	2048	8192	16	128	1.03
6.8B	40	3584	14336	28	128	0.99

Figura 2.1: *Comparação entre o método de Hoffmann et al. para cálculo de FLOPs em* Transformers *e a aproximação* $6P$ *(Hoffmann et al., 2022)*

Apesar de facilitar a comparação entre modelos, a métrica de FLOPs é bastante vulnerável às variações nas implementações dos modelos, principalmente na questão do uso de memória, o maior fator para a subutilização das GPUs, como ilustrado por Ivanov et al., afirmando que para uma implementação padrão do modelo BERT no Pytorch, 25,5% do tempo de execução é gasto em operações de normalização de resultados, apesar de essas operações representarem apenas 0,17% dos FLOPs, o que demonstra o alto custo da transmissão de dados entre as unidades computacionais. (Ivanov *et al.*, 2021)

O problema da subutilização dessas unidades motivou a Google a desenvolver a métrica de eficiência de modelos de linguagem MFU (*model FLOPs utilization* — utilização de FLOPs pelo modelo), definida pela razão $\frac{F*T}{M}$, onde F é a quantidade de FLOPs por *token* gerado, T é o número de *tokens* gerados por segundo e M o valor máximo de FLOPs por segundo realizados pelo computador. A métrica foi divulgada pela primeira vez em 2022 junto com o modelo PaLM (Chowdhery *et al.*, 2023), que atingia cerca de 46% de MFU. Segundo Casson, as implementações mais populares de modelos baseados em Transformers até 2023 tinham MFU entre 10% e 65%. (Casson, 2023)

Faiz et al. desenvolveram o modelo LLMCarbon, que estima a emissão de CO_2 a partir da contagem de FLOPs e do cálculo do carbono incorporado. Exemplificaremos o funcionamento do modelo supondo o treinamento de um LLM baseado no modelo Transformer:

Primeiro, determina-se o número de FLOPs executados durante o treinamento. Como foi visto anteriormente, em Transformers esse valor é próximo de $6PT$, onde P é o número de parâmetros do modelo e T o número de *tokens* presentes no conjunto de dados de treinamento.

Depois, é necessário obter o MFU do modelo. Caso não seja possível obtê-lo empiricamente, uma opção viável é estimá-lo a partir do número de peças de *hardware* especializado (GPUs, TPUs, FPGAs e semelhante), utilizando uma fórmula inspirada pelo trabalho de Narayanan et al. Essa fórmula supõe que existe um número N de peças que obtêm eficiência

ótima por meio de técnicas de paralelismo de dados (peças diferentes recebem entradas diferentes, e os parâmetros aprendidos pelos modelos são agregados periodicamente), de tensores (cada camada é particionada entre as peças) e *pipeline* (cada peça trata uma camada diferente).

Essas N peças atingem MFU mfu_N, e estimamos o MFU de M peças mfu_M por:

$$\begin{cases} \gamma_0 * \frac{M}{N} * mfu_N, \text{ se } M < N \\ \gamma_1 * \frac{N}{M} * mfu_N + \gamma_2, \text{ se } M > N \end{cases}$$

onde $\gamma_0, \gamma_1, \gamma_2$ são constantes a serem obtidas empiricamente. Com mfu_M, o tempo t de execução do treinamento nas N peças pode ser calculado como

$$t = \frac{total\ de\ FLOPs\ realizados}{M * mfu_M * MAXFLOPS}$$

onde *MAXFLOPS* representa o máximo de FLOPs por segundo que a peça consegue realizar.

O gasto de energia de um conjunto de M peças de hardware especializado iguais é, portanto, expressa por

$$energia = W * mfu_M * t * M * PUE$$

onde W representa a potência máxima da peça, em watts, e *PUE* é a eficiência de uso energético da infraestrutura do servidor. Para obter a quantidade de carbono emitido, basta multiplicar a quantidade de energia gasta pela intensidade de carbono da matriz energética que alimenta o servidor.

2.2 Formas de Redução

2.2.1 Redução do Número de Operações

Como vimos na seção anterior, a emissão de carbono operacional tem forte relação com o tempo de execução das operações de inferência e treinamento dos modelos. Sendo assim, otimizações que permitem reduzir o tempo de execução tendem a reduzir também o gasto energético e as consequentes emissões de CO_2.

Wang et al. propõem três técnicas, ilustradas no diagrama da figura 2.2 para a redução do número de operações, durante o treinamento e inferência de modelos, com um foco em redes neurais convolucionais (*convolutional neural networks* — CNNs), modelos utilizados principalmente em problemas de visão computacional (Wang *et al.*, 2019).

A primeira dessas técnicas (*stochastic mini-batch dropping* — SMD) consiste em definir uma probabilidade (a princípio, 50%) para que, durante uma iteração do treinamento, um pequeno conjunto de exemplos seja ignorada. Dessa forma, o número de operações é reduzido em aproximadamente 50%, porém, segundo os experimentos dos autores, o impacto nas métricas de corretude do modelo são pequenas, e pode até ser positivo, pois

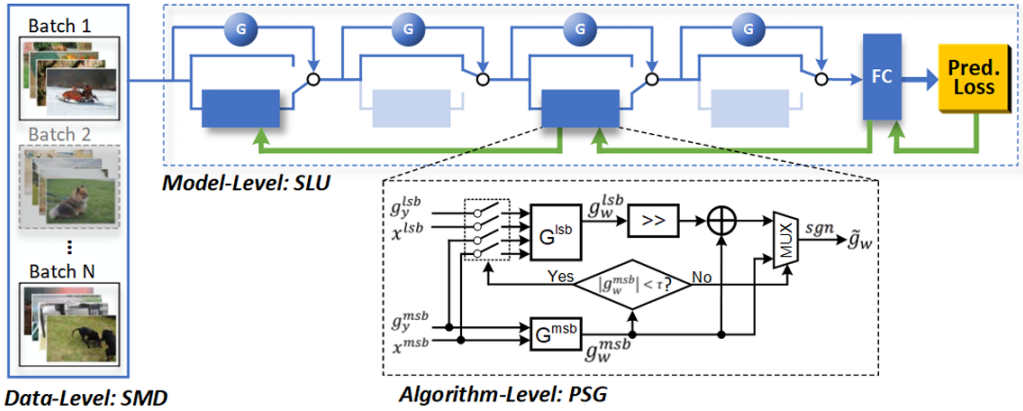


Figura 2.2: *Diagrama dos métodos de Wang et al. para otimização de CNNs (WANG et al., 2019)*

pular exemplos introduz certa variabilidade durante o treinamento que pode ser útil para evitar pontos de sela e mínimos locais insatisfatórios na função a ser minimizada.

Treinando o modelo ResNet-74 na base de dados CIFAR-10, que contém 60000 imagens coloridas de baixa resoluções a serem classificadas em 10 categorias, utilizando entre 64 e 128 mil iterações, a figura 2.3 mostra que empregar SMD resulta consistentemente em maior acurácia que utilizar todas as amostras em todas as iterações, com uma diferença máxima de 0.86% em 75 mil iterações, e mínima de 0.39% em 128 mil iterações.

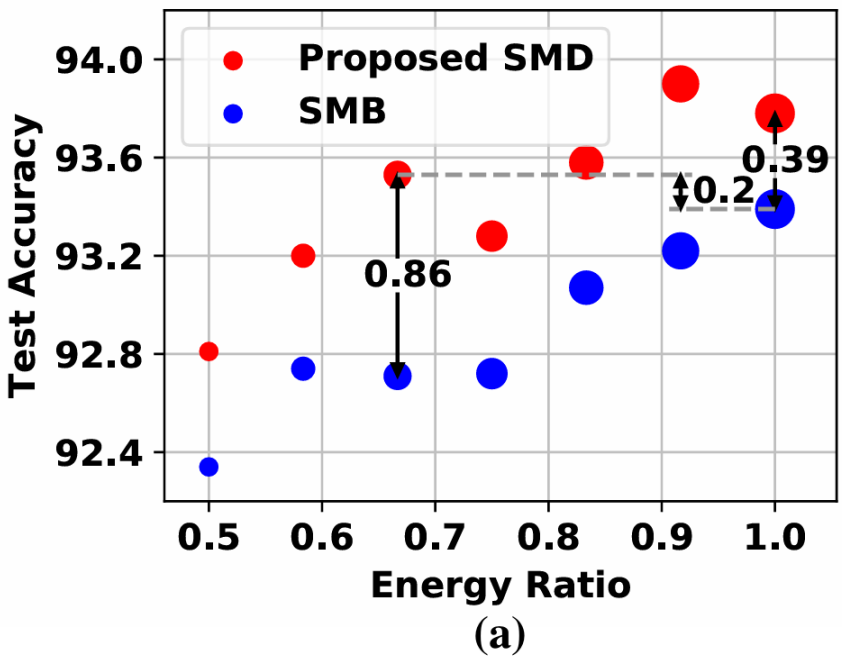


Figura 2.3: *Resultados do SMD em comparação à estratégia padrão de* mini-batches *(SMB* — standard mini-batch*) (WANG et al., 2019)*

A segunda técnica proposta, aplicada tanto no treinamento quanto na inferência, é chamada de atualização seletiva de camadas baseado na entrada (*input-dependent selective*

layer update — SLU). Os principais modelos baseados em CNNs, como as ResNets, frequentemente apresentam dezenas de camadas, provocando o desenvolvimento de métodos para determinar quais camadas são essenciais para garantir a corretude da classificação de uma entrada e quais camadas são pouco importantes ou podem ser ignoradas, em um mecanismo parecido à combinação de especialistas (*mixture of experts* — MoE) nos Transformers.

Em particular, o SLU introduz pequenas redes neurais recorrentes antes de cada camada, que recebem a saída da camada anterior e retornam um valor entre 0 e 1 que corresponde à probabilidade da camada ser utilizada para aquele exemplo. Essas redes neurais podem ter seus parâmetros treinados junto do restante do modelo, porém a um custo baixo devido a serem redes pequenas.

Como na figura 2.4, esse método rende resultados melhores que o método de profundidade estocástica proposto por Huang et al., que ignora camadas aleatoriamente com probabilidades que aumentam linearmente conforme a profundidade, priorizando as primeiras camadas, que tendem a identificar as características mais cruciais da entrada. Ajustando os parâmetros da profundidade estocástica para que o número de camadas utilizadas seja próximo do SLU a cada passo, logo com número de operações similares, SLU obtém acurácia consideravelmente melhor.

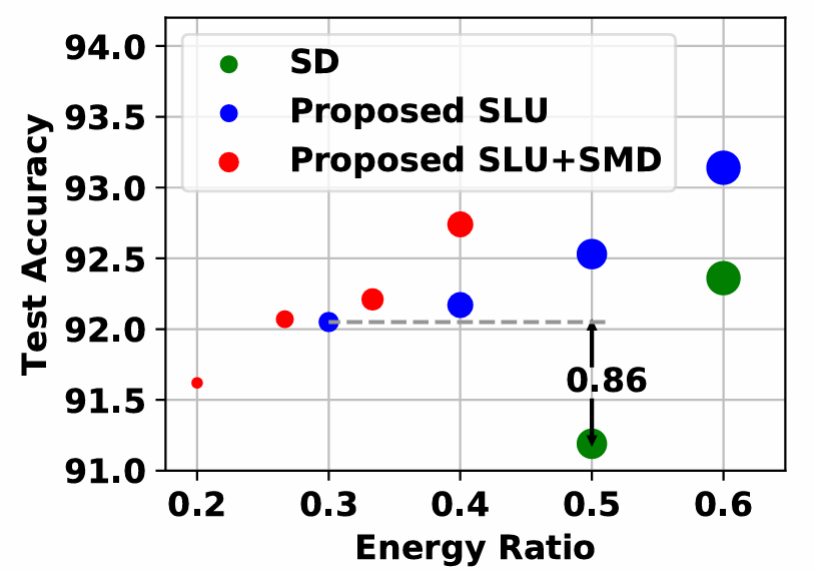


Figura 2.4: *Comparação entre SLU e profundidade estocástica (SD* —stochastic depth*) (*Wang et al., 2019*)*

A acurácia é comparável até quando SLU utiliza menos camadas: uma implementação de SLU utilizando 30% das camadas tem acurácia 0.86% maior que uma implementação de profundidade estocástica com probabilidade inicial de 50%, ou seja, realizando, em média, 20% mais operações e consumindo mais energia.

A terceira técnica é o gradiente descendente de sinal preditivo (*predictive sign gradient descent* — PSG). Essa técnica estende métodos anteriores, que, durante a retropropagação, usam apenas o sinal dos gradientes para atualizar os parâmetros, para reduzir o custo computacional das atualizações. No entanto, esses métodos ainda calculam os gradientes com precisão máxima.

Por sua vez, PSG considera apenas os bits mais significativos para o gradiente da função de erro da saída do modelo e para a entrada (os autores sugerem 10 e 4 bits, respectivamente) para aproximar ou "prever"o sinal do gradiente do parâmetro a ser otimizado. Os autores explicam que a probabilidade de erro tem relação exponencial com o número de bits dos gradientes, porém também determinam um híperparâmetro τ com o seguinte propósito: se ao calcular o gradiente G de um parâmetro p utilizando o método PSG, encontramos que $|G_p^{PSG}| < \tau$, então calculamos G_p com precisão máxima, para minimizar a chance de erro ao determinar o sinal de G_p.

Comparando o método PSG ao método de adotar parâmetros de 8 bits porém gradientes de 32 bits, PSG economizou 60% de energia nos experimentos do artigo, com uma perda absoluta de apenas 0.65% na acurácia (de 93.24% de classificações corretas a 92.59%). Comparado ao método de utilizar parâmetros e gradientes de 32 bits, atualizando-os apenas com os sinais, os resultados de eficiência são parecidos, mas com um pequeno aumento na acurácia.

A escolha de hiperparâmetros pode também ter grande impacto no consumo energético. Puvis de Chavannes et al. analisaram a relação entre vários hiperparâmetros — listados na figura 2.5 — de modelos Transformer, o consumo energético e a perplexidade, métrica de incerteza na geração de uma sequência de *tokens* (Puvis de Chavannes *et al.*, 2021).

Parameter	Interval
vocab_size	[1,30522]
hidden_size_multiplier	[1,100]
num_hidden_layers	[1,12]
num_attention_heads	[1,18]
intermediate_size	[1,3072]
hidden_act	(relu, silu, gelu, gelu_new)
hidden_dropout_prob	[0.1,1]
attention_probs_dropout_prog	[0.1,1]
max_position_embeddings	[512,512]
type_vocab_size	[1,1]
initializer_range	[0.02,0.02]
layer_norm_eps	[1.00E-12,1.00E-12]
position_embedding_type	(absolute, relative_key, relative_key_query)

Figura 2.5: *Lista de hiperparâmetros explorados por Puvis de Chavannes et al. (Puvis de Chavannes et al., 2021)*

Buscando minimizar o produto entre a perplexidade e o gasto energético (PEP - *perplexity-energy product*), em kWh, para o modelo RoBERTa, um aprimoramento desenvolvido pelo Facebook ao modelo BERT, os autores realizaram uma busca bayesiana pela combinação de hiperparâmetros como o número de camadas de atenção (*attention*

heads), funções de ativação e tamanho do vocabulário, usando a ferramenta HyperOpt.

O estudo encontrou que, para as melhores configurações em termos de PEP, a correlação entre a perplexidade e o custo de energia é alta e negativa (-0.88), porém, considerando todas as 154 configurações, a correlação é de 0.5, ou seja, em muitos casos, é possível reduzir ambas as métricas. As demais correlações podem ser observadas na figura 2.6.

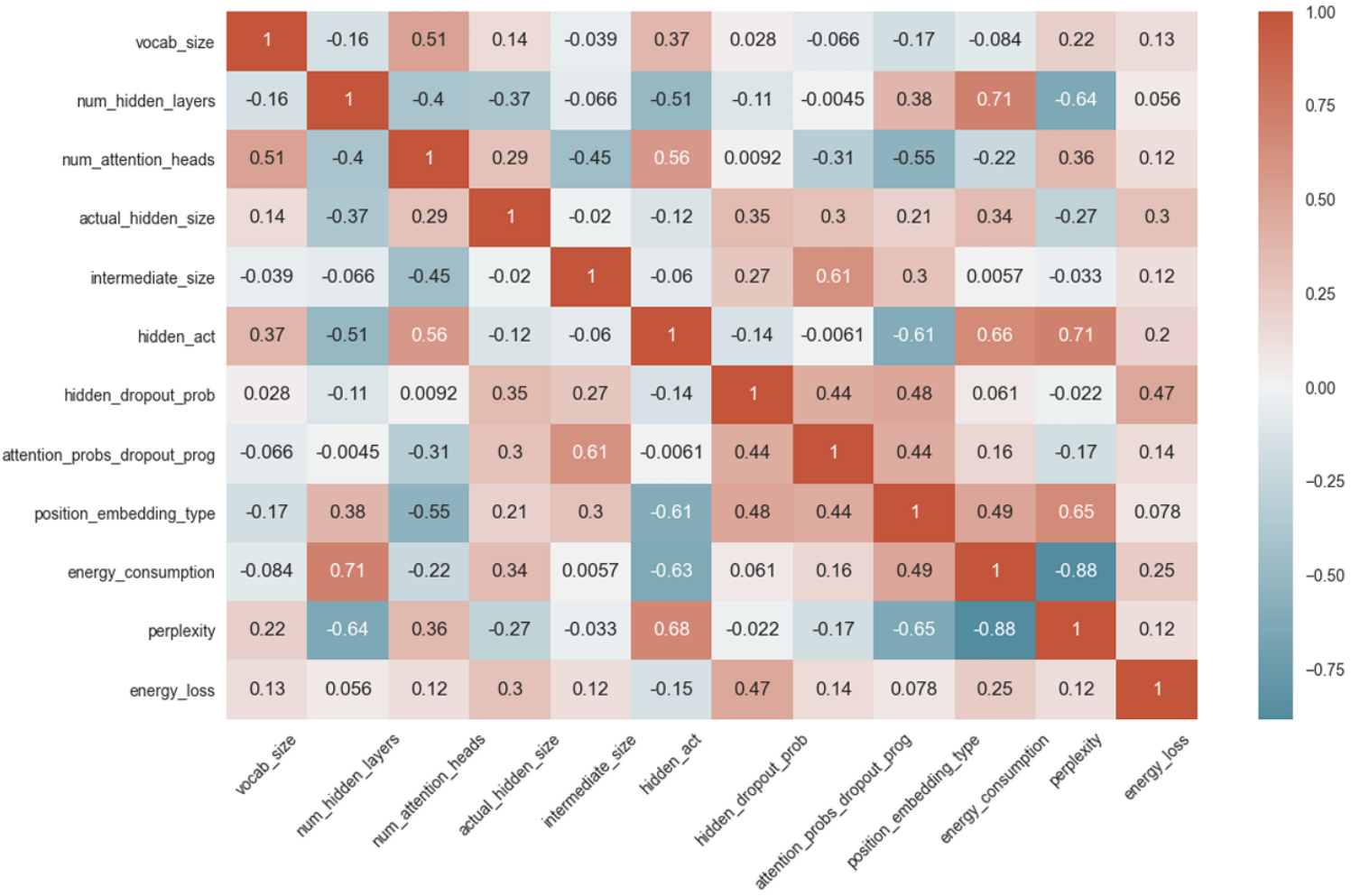


Figura 2.6: *Correlações entre hiperparâmetros e métricas encontradas por Puvis de Chavannes et al. para os 15% melhores modelos em PEP (Puvis de Chavannes et al., 2021)*

O hiperparâmetro de maior impacto é o número de camadas das redes neurais que compõem a parte final dos Transformers, com correlação de 0,71 e -0,64 em relação à energia e perplexidade, respectivamente, para os melhores modelos. A média de camadas é de 1,91, com desvio padrão de 0,92. Esse valor é baixo comparado às 12 camadas padrão do modelo RoBERTa, incorrendo em um custo de maior perplexidade que, segundo os autores, poderia ser compensado com o uso de mais dados de treinamento, pois RoBERTa foi treinado com aproximadamente 10 vezes mais exemplos.

A escolha da função de ativação é a segunda de maior impacto, com correlação de 0,66 para o consumo energético e 0,71 para a perplexidade, nos melhores modelos. Nesse cenário,

a função GELU é a mais frequente, superando SiLU e a tradicional RELu, porém ela aparece também em algumas das piores configurações, sendo a função de maior custo energético.

Curiosamente, o número de camadas de atenção tem correlação mais baixa quando comparado ao número de camadas das redes neurais, sendo -0,22 para a energia e 0,36 para a perplexidade entre as melhores configurações. No caso geral, a correlação com a perplexidade é quase nula: 0,057.

Os autores destacam que a probabilidade de *dropout*, ou seja, a probabilidade de anular um certo parâmetro do modelo durante o treinamento, para evitar o sobreajuste, é um dos únicos hiperparâmetros que tem impacto na perplexidade mas quase nenhum no consumo energético, portanto pode ser otimizado buscando apenas a maior chance de acerto. Nos modelos de maior PEP, a probabilidade de *dropout* média foi de 0,18, com desvio padrão de 0,06 (figura 2.7).

	Best 15% Mean	Best 15% Std. Dev	Worst 15% Mean	Worst 15% Std. Dev
vocab_size	21187.73	6426.71	18545.04	9141.88
actual_hidden_size	116.43	84.20	727.78	27.33
num_hidden_layers	1.52	0.77	7.82	3.26
num_attention_heads	8.08	5.27	12.78	4.48
intermediate_size	890.47	554.19	1149.65	739.19
hidden_dropout_prob	0.37	0.23	0.40	0.15
attention_probs_dropout_prog	0.28	0.14	0.47	0.28
energy comsumption	0.99	0.17	6.18	1.93
perplexity	338.51	576.74	1236.04	962.74

Figura 2.7: *Médias e desvios padrões encontrados por Puvis de Chavannes et al. para os hiperparâmetros nos 15% melhores modelos em PEP (Puvis de Chavannes et al., 2021)*

2.2.2 Otimização do Uso do *Hardware*

Outra importante forma de reduzir o carbono operacional é desenvolver maneiras para utilizar o hardware mais eficientemente. As GPUs têm sido o foco dos estudos dessa vertente pois as suas arquiteturas e funcionamento são essenciais para a viabilidade da implementação de grandes modelos de ML, porém necessitam de estratégias específicas para maximizar seus desempenhos.

Como vimos anteriormente, o principal fator a ser otimizado nas GPUs é a administração da memória, pois a busca de dados na memória global ou compartilhada da GPU é uma operação custosa que frequentemente causa interrupções nos programas.

Li et al. busca essa otimização especificamente para redes neurais convolucionais em dois aspectos: escolher a ordem das dimensões dos tensores que mais acelera o acesso aos dados pelas *threads* da GPU e reduzir chamadas de comando de leitura de memória externa à GPU (C. Li *et al.*, 2016).

Os tensores convolucionais geralmente possuem quatro dimensões:

- o número N de imagens processadas por iteração

- o número C de mapas de atributos (*feature maps*)
- o comprimento W e altura H das imagens

Assim, existem 16 formas de representar esses tensores em memória, sendo que o artigo analisa duas delas: a CHWN, utilizada pela biblioteca cuda-convnet com o algoritmo tradicional de convolução, e o NCHW, aplicado pelo Caffe e cuDNN, que possuem implementações de convolução por meio da multiplicação de matrizes ou pela transformada rápida de Fourier (*fast Fourier transform* — FFT). As demais formas foram descartadas ou por terem resultados piores, como é o caso de HWCN e NHWC, ou por dificultarem a aplicação das convoluções, como as transposições que separam as duas dimensões das imagens (CHNW, por exemplo).

Para os testes, os autores apresentam uma adaptação da biblioteca Caffe com implementações próprias da mudança de disposição de CHWN e NCHW, e vice-versa, e da convolução tradicional, testando-a para diferentes CNNs conhecidas, como a AlexNet e a ZFNet, medindo o desempenho em algumas camadas dessas redes, cujas configurações estão listadas na figura 2.8.

No que tange às convoluções, a disposição CHWN, com a convolução tradicional, tem desempenho melhor quando C é pequeno, como na primeira camada de convolução de cada rede, onde C corresponde ao número de canais da imagem (3 para imagens coloridas RGB, 1 para imagens em escala de cinza). Aumentar o número de imagens processadas também tende a favorecer essa configuração. Em média, a disposição e algoritmos ideais para cada camada são 2,48 vezes mais eficientes que a outra opção, 2,08 vezes quando consideramos o custo das transposições. Para a primeira camada da LeNet (CV1 na figura 2.9), processando 128 imagens 28x28 em escala de cinza com 16 mapas de atributos, CHWN é mais de 6 vezes mais rápido que NCHW (4 vezes considerando a transposição).

Após a aplicação das convoluções, CNNs tipicamente possuem camadas de agregação ou *pooling*, que reduzem a dimensionalidade das características calculadas nas camadas anteriores por meio de operações de complexidade linear como médias ou seleção do maior valor. Por isso, essas camadas, assim como a camada final de classificação, têm custo computacional menor que as convolucionais, aumentando o impacto das operações de leitura e escrita na memória.

Para essas camadas, CHWN se mostrou muito mais eficiente em todos os exemplos, pois nessa disposição, as *threads* da GPU acessam, a cada leitura da memória global, endereços consecutivos de memória, cada uma representando uma imagem processada. Quando *threads* diferentes acessam endereços adjacentes de memória, dizemos que o acesso é coalescente. Já para a disposição NCHW, os acessos não são consecutivos, visto que as *threads* aqui representam os mapas de atributos mas a dimensão contínua da memória é a coluna do mapa. Logo, o número de *bytes* lidos por operação é fixo, mais operações de leitura são necessárias em uma iteração.

Layer	Ni	Co	H/W	F_w/F_h	Ci	S	Description
CONV1 (CV1)	128	16	28	5	1	1	LeNet[17]: Model Error rate: 0.18% (epoch 200)
CONV2 (CV2)	128	16	14	5	16	1	
POOL1 (PL1)	128	-	28	2	16	2	
POOL2 (PL2)	128	-	14	2	16	2	
CLASS1	128 images and 10 categories						
CONV3 (CV3)	128	64	24	5	3	1	Cifar10[15]: Model Error rate:14.04% (epoch 100)
CONV4 (CV4)	128	64	12	5	64	1	
POOL3 (PL3)	128	-	24	3	64	2	
POOL4 (PL4)	128	-	12	3	64	2	
CLASS2	128 images and 10 categories						
POOL5 (PL5)	128	-	55	3	96	2	ImageNet With AlexNet[12] Model
POOL6 (PL6)	128	-	27	3	192	2	
POOL7 (PL7)	128	-	13	3	256	2	
CLASS3	128 images and 1000 categories						
CONV5 (CV5)	64	96	224	3	3	2	ImageNet with ZFNet Model[25]
CONV6 (CV6)	64	256	55	5	96	2	
CONV7 (CV7)	64	384	13	3	256	1	
CONV8 (CV8)	64	384	13	3	384	1	
POOL8 (PL8)	64	-	110	3	96	2	
POOL9 (PL9)	64	-	26	3	256	2	
POOL10 (PL10)	64	-	13	3	256	2	
CLASS4	64 images and 1000 categories						
CONV9 (CV9)	32	64	224	3	3	1	ImageNet with VGG Model [22]
CONV10 (CV10)	32	256	56	3	128	1	
CONV11 (CV11)	32	512	28	3	256	1	
CONV12 (CV12)	32	512	14	3	512	1	
CLASS5	32 images and 1000 categories						

Legenda:

- **Ni**: número de imagens ou *batch size*
- **Co**: número de *kernels* de convolução
- **H/W**: altura e comprimento, respectivamente
- **Fw/Fh**: comprimento e altura dos *kernels*
- **Ci**: canais de entrada
- **S**: *stride*, ou o tamanho do passo entre operações de *pooling* consecutivas em uma mesma entrada

Figura 2.8: *Configurações de camadas convolucionais (CV) e de* pooling *(PL) explorados por Li et al. (C. Li et al., 2016)*

Na camada de classificação, há uma outra otimização específica que pode ser empregada: a fusão de *kernels* em CUDA. Nas três bibliotecas exploradas, cada passo do algoritmo de classificação Softmax (programa 2.1) é implementado em um *kernel* diferente, ou seja, a cada passo, os resultados são transferidos para a memória externa à GPU para serem lidos no início do próximo passo, junto com as instruções do novo *kernel*. Assim, se fosse possível executar o algoritmo em um único *kernel*, menos transações de memória externa seriam necessárias, otimizando o processo.

Programa 2.1 Softmax (sequencial)

```
1   FUNCAO softmax(v)
2       max ← v[0]
3       r ← vetor nulo de mesmo tamanho de v
4       para todo elem em v
5           se elem > m
6               max ← elem
7           fim
8       fim
9       soma ← 0
10      para todo i em 0..tamanho(v)
11          x ← exp(v[i] − max)
12          r[i] ← x
13          soma += x
14      fim
15
16       para todo elem em r
17          elem /= soma
18       fim
19       devolva r
```

No artigo, apresenta-se um *kernel* onde cada bloco de *threads* representa uma imagem processada, e cada *thread* do bloco representa um pequeno conjunto de categorias. Nesse contexto, as *threads* podem operar em paralelo, consultando a memória compatilhada da GPU apenas para ler e escrever *max* e *soma* (linhas 10–14 e 16–18 do programa 2.1). Essa otimização surte grande efeito quando o número de categorias é alto. Para uma configuração de 128 imagens sendo classificadas em 10 mil categorias, o novo *kernel* atingiu 94% da largura de banda da GPU utilizada, sendo 3,58 vezes mais rápido que a implementação do cuDNN, a melhor entre as bibliotecas analisadas.

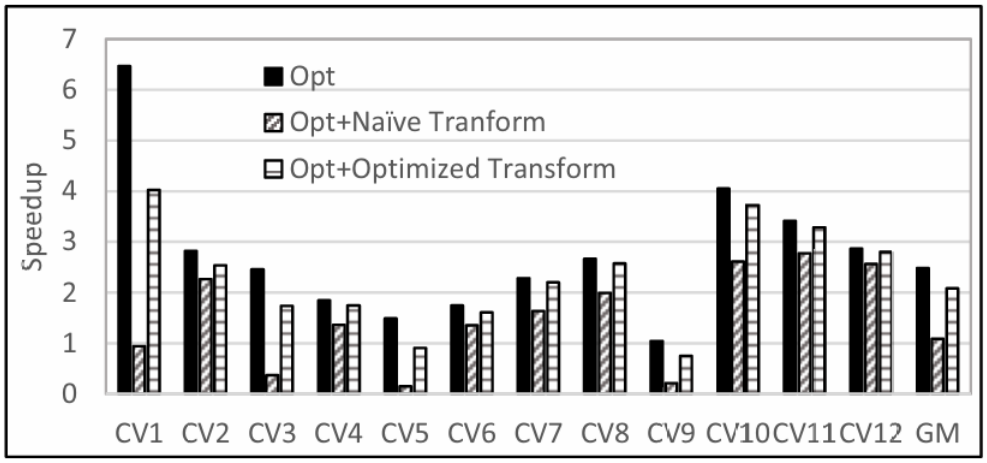


Figura 2.9: *Otimização ao usar a transposição e algoritmo ideias para cada camada convolucional em Li et al., considerando algoritmos simples (*naive*) ou mais elaborado (*optimized*) de transposição (C. Li et al., 2016)*

A fusão de *kernels* é também um dos métodos apresentados em Ivanov et al., que busca aprimorar o uso da memória por modelos *Transformers* (Ivanov *et al.*, 2021). Aplicando-a para operações simples como funções de ativação, aplicação de viéses e *dropouts*, foi possível acelerar algumas operações em até 90%, como é o caso da sequência adição do viés–ReLU–*dropout* na primeira camada neural do modelo BERT. Para a segunda camada, o mesmo processo, incluindo a normalização dos resultados, é 70% mais rápida com a fusão de *kernels*. A estratégia também é empregada para a atualização destes mesmos parâmetros durante o algoritmo de retropropagação, com otimizações de 64% e 58% respectivamente.

Após a fusão, os autores propõem a otimização das disposições dos tensores representando-as por um grafo direcionado, onde cada vértice representa uma configuração possível para determinada operação, com arcos conectando operações subsequentes e cujos pesos são o tempo de execução da operação e transposição posterior (se necessária), obtidos empiricamente. Executando um algoritmo de caminho mais curto, é possível encontrar a configuração ótima. A configuração ótima encontrada é 30% mais rápida no treinamento da camada de *encoding* do modelo BERT quando comparado ao Pytorch, com uma redução de 23% na movimentação de dados entre as memórias, e 20% quando comparado a uma implementação com TensorFlow com otimizações do compilador XLA.

Li et al. apresenta uma outra estratégia de otimização, denominada Clover: o particionamento de GPUs entre modelos de tamanhos variáveis. (B. Li, Samsi *et al.*, 2023) A partir da geração Ampere com a tecnologia MIG (*multi-instance GPU* — GPU de múltiplas instâncias), as GPUs NVIDIA podem ser divididas em até sete partições independentes, em cinco tamanhos: 1, 2, 3, 4, 7, tais que a soma dos tamanhos deve ser sempre sete. Ou seja, é possível dividir a placa em cinco partições de tamanho 1 e uma de 2, ou dividí-la em duas partições de tamanho 3 e uma de tamanho 1. As 19 possibilidades de particionamento podem ser encontradas na figura 2.10.

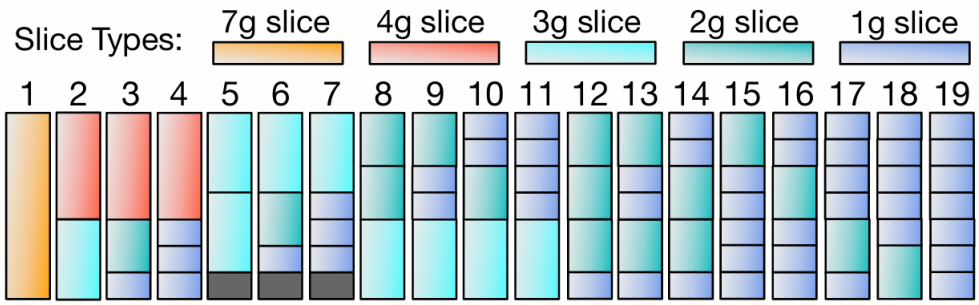


Figura 2.10: *Formas de particionamento disponíveis com a tecnologia MIG em placas NVIDIA (B. Li, Samsi et al., 2023)*

Quando consideramos um modelo de um mesmo tamanho, adequado para partições pequenas, usar partições menores já pode incorrer em menor gasto de energia como mostra a figura 2.11, com um certo custo de latência devido à necessidade de disputar recursos da GPU com outras partições, porém utilizar modelos de tamanhos variados simultaneamente permite a implementação de modelos mais complexos gastando ainda menos energia.

Os autores propõem encontrar a melhor configuração a partir de um problema de otimização definido da seguinte maneira:

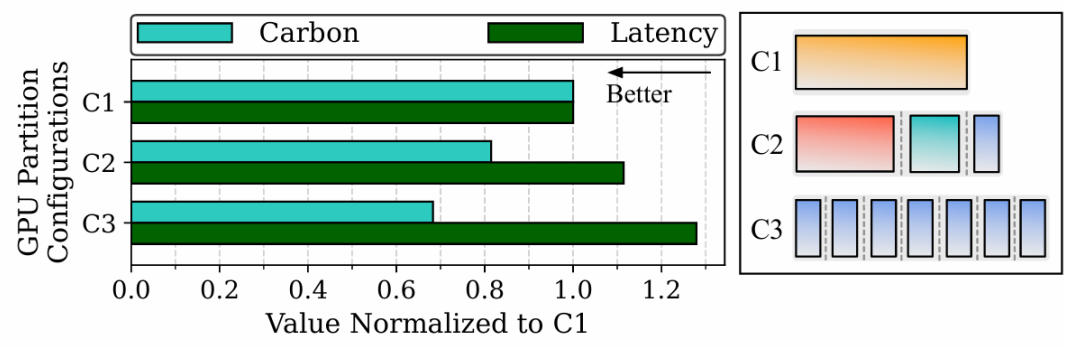


Figura 2.11: *Latência e emissões de carbono para particionamentos diferentes da GPU utilizando o mesmo modelo (B. Li, Samsi et al., 2023)*

Primeiro, seja x^p um vetor onde x_i^p denota a configuração de particionamento escolhida para a GPU de índice i, entre as GPUs alocadas para a resolução do problema, e seja x^v um vetor onde x_{ij}^v o modelo escolhido para a partição j da GPU i. Assim, temos $A(x^p, x^v)$ e $C(x^p, x^v)$, que respectivamente expressam a acurácia e a emissão de CO_2 de um treinamento/inferência com uma configuração e modelos determinados.

Com A_{base} e C_{base} representando a acurácia e emissão de gás carbônico do modelo mais acurado conhecido treinado sem particionamento, podemos calcular o diferencial de acurácia e energia de (x^p, x^v) da seguinte forma:

$$\Delta C = \frac{C_{\text{base}} - E(x^p, x^v) * IC}{C_{\text{base}}}$$

onde IC é a intensidade de carbono da energia utilizada na implementação, e

$$\Delta A = \frac{A(x^p, x^v) - A_{\text{base}}}{A_{\text{base}}}$$

A função de otimização, a ser maximizada, é expressa por

$$f(x^p, x^v) = \lambda * \Delta C + (1 - \lambda) * \Delta A$$

onde λ é um parâmetro entre 0 e 1 que define o balanço entre maior acurácia e menor gasto energético buscado no processo de otimização. A única restrição sugerida pelos autores é uma restrição de latência máxima L_{max} na inferência das configurações. Assim, a otimização pode ser expressa por

$$\max_{x^p, x^v} f(x^p, x^v)$$
$$\text{t.q. } L(x^p, x^v) \leq L_{max}$$

Como o número de configurações possíveis pode ser grande demais para que seja viável testar todas elas, os autores sugerem representá-las por meio de um grafo bipartido onde um conjunto de vértices representando modelos diferentes se ligam a outro conjunto representando os tipos de partições, com arestas cujo peso indica a quantidade de vezes

em que um modelo específico ocupa um certo tipo de partição dentro da configuração. A partir dessas representações, aplica-se o algoritmo de arrefecimento simulado (*simulated annealing*), descrito no programa 2.2, para encontrar a configuração ótima. A função de distância usada para definir as vizinhanças no algoritmo é a distância de edição de grafos (*graph edit distance*), e a probabilidade de escolher um vizinho $x^{g'}$ da configuração x^g é definida por

$$P(x^g, x^{g'}) = exp(-\frac{h(x^{g'}) - h(x^g)}{T})$$

onde T é o parâmetro de temperatura e

$$h(x^g) = -f(x^g) * min(1, \frac{L_{max}}{L(x^g)})$$

Nos testes realizados pelos autores, o arrefecimento simulado obtém resultados muito próximos de uma busca exaustiva, em uma fração do tempo de execução, e ainda permite certa flexibilidade na otimização a partir do parâmetro λ. Em comparação com a implementação dos maiores modelos sem particionamento, o método Clover conseguiu, para tarefas de linguagem, utilizando os modelos ALBERT v2 que variam entre 12 e 223 milhões de parâmetros, reduzir em mais de 80% a emissão de carbono, com um custo de 4% na precisão. Para modelos de classificação de imagens — os EfficientNet, também da Google, a redução é um pouco maior, diminuindo a precisão em aproximadamente 2%. Em todos os casos, como é possível observar no gráfico mais à direita na figura 2.12, a latência é curiosamente reduzida, ou seja, os gastos com a disputa de recursos pelas partições é compensado pelo menor número médio de operações realizados por elas.

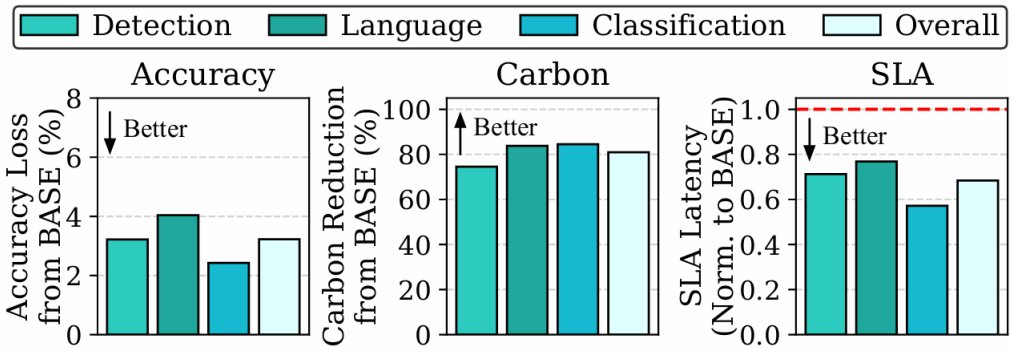


Figura 2.12: *Resultados da técnica CLOVER para tarefas de detecção de objetos, classificação de imagens e processamento de linguagem natural (B. Li, Samsi et al., 2023)*

Programa 2.2 Arrefecimento simulado, como implementado por Li et al. (B. Li, Samsi et al., 2023)

```
1  FUNCAO arrefecimento_simulado(X, d, R, K_max, I_max, λ, T, T_min)
2      k ← 0
3      i ← 0
```

cont ⟶

$\longrightarrow$ *cont*

4 $x^g \leftarrow$ `escolha_aleatoria`(X)

5 **enquanto** $k < K_{max}$ **e** $i < I_{max}$

6 $x^{g'} \leftarrow$ `vizinho_aleatorio`(x^g, X, d)

7 **se** $h(x^{g'}, \lambda) < h(x^g, \lambda)$ **ou** $P(x^g, x^{g'}, T) \geq$ `aleatorio`$(0, 1)$

8 $x^g \leftarrow x^{g'}$

9 $k \leftarrow 0$

10 *senão*

11 k += 1

12 $T \leftarrow$ `maximo`$(T_{min}, T - R)$

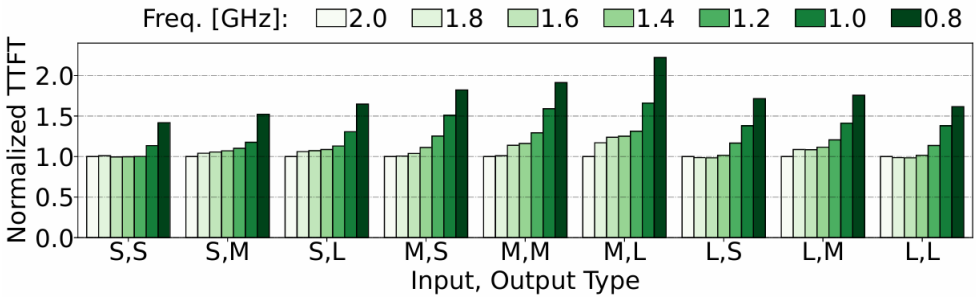
13 i += 1

14 *retorne* x^g

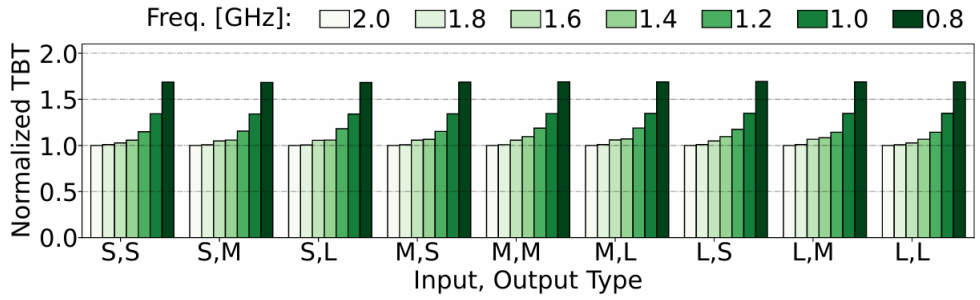
Por fim, uma simples estratégia de redução de energia é limitar a frequência máxima das GPUs, como propõe Stojkovic et al. (Stojkovic *et al.*, 2024). Para os testes, os autores utilizam uma implementação da LLM LLama2 em 8 GPUs NVIDIA H100 com entradas e saídas de três tamanhos diferentes, analisando as nove combinações possíveis para gasto energético, potência máxima e três métricas de latência:

- *time to first token* ou TTFT, que denota o tempo até a geração do primeiro *token* de saída.
- *time between tokens* ou TBT, expressando o tempo médio entre *tokens* gerados
- *throughput* ou T-put, ou seja, o número de trabalhos de inferência realizados por segundo

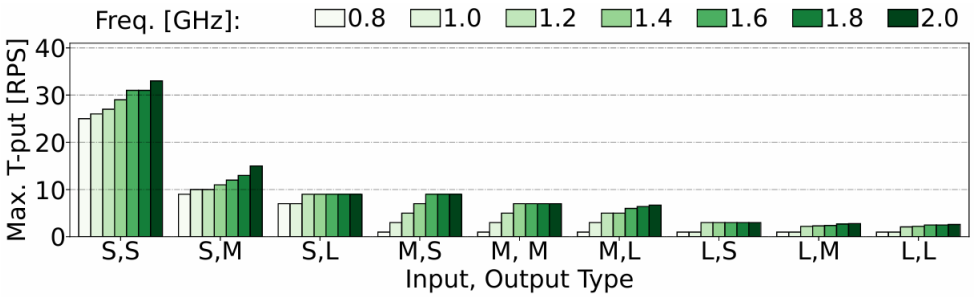
Nos testes, os autores encontraram que existem situações onde reduzir a frequência diminui o gasto energético sem grande impacto na latência, como é o caso das entradas e saídas grandes (1024 e 256 *tokens*, respectivamente), no qual a redução de frequência de 2.0 GHz para 1.2 GHz degrada em menos de 20% as métricas de latência, mas diminui o gasto energético em aproximadamente 30%. No entanto, a relação entre frequência e gasto energético nem sempre é proporcional: nas figuras 2.13, observa-se que para entradas médias diminuir a frequência para 1.8 ou 1.6 GHz tende a aumentar o consumo de energia. A técnica também potencializa o uso mais intenso de paralelismo de tensores entre as placas, pois vê-se nas imagens 2.14 que entre o paralelismo de grau 2 (TP2) com alta frequência e o de grau 8 (TP8) com menor frequência, o gasto energético é similar, porém o TBT e o TTFT são quase 50% menores para o TP8.



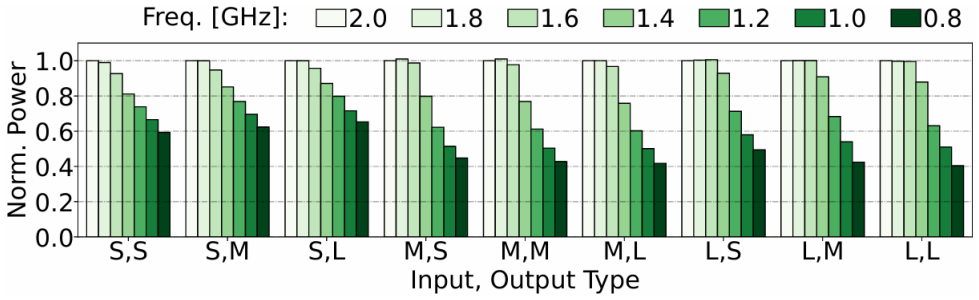
(a) *TTFT (tempo até o primeiro* token*)*



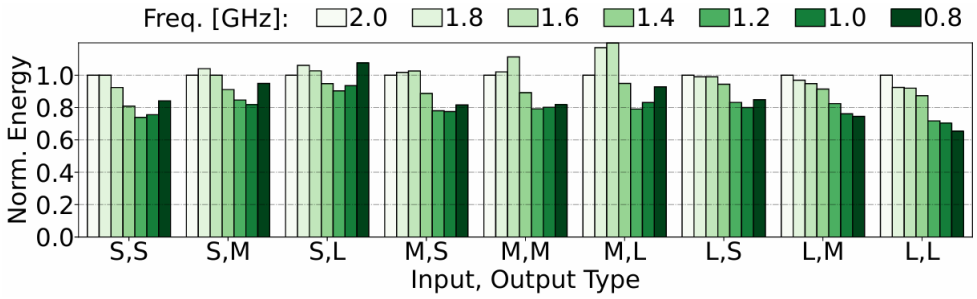
(b) *TBT (tempo entre* tokens *consecutivos)*



(c) throughput*, ou número de inferências por segundo*

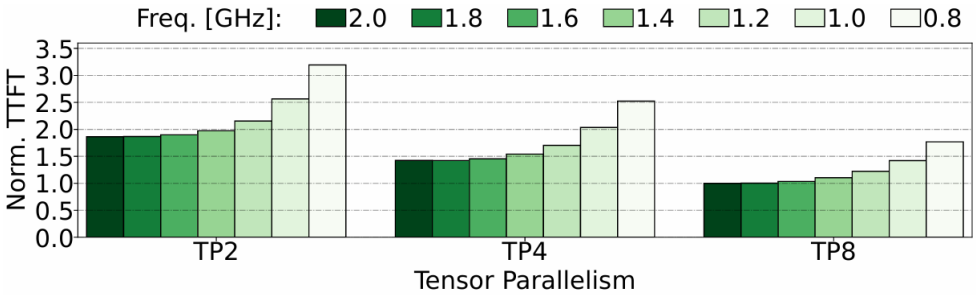


(d) *potência média das GPUs*

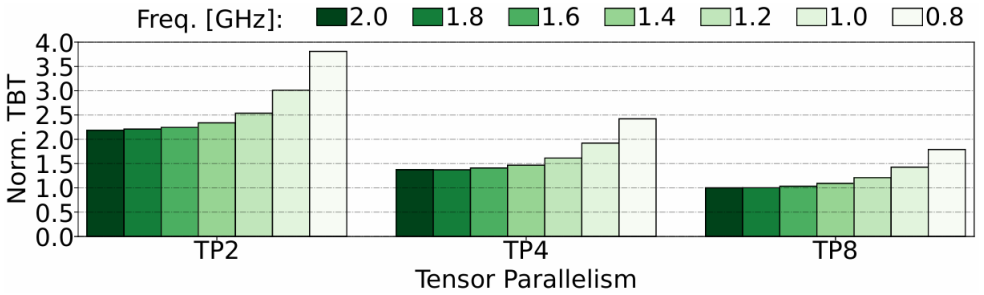


(e) *energia média consumida pelas GPUs*

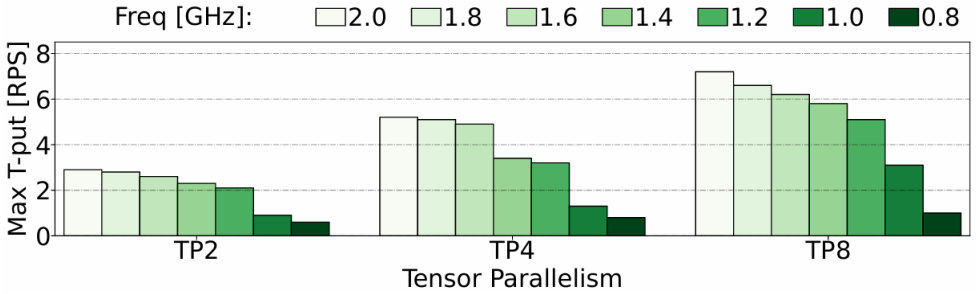
Figura 2.13: *Resultados da redução de frequência das GPUs para inferências de LLMs em três tamanhos de entrada e saída diferentes (S, M, L) (*Stojkovic *et al., 2024)*



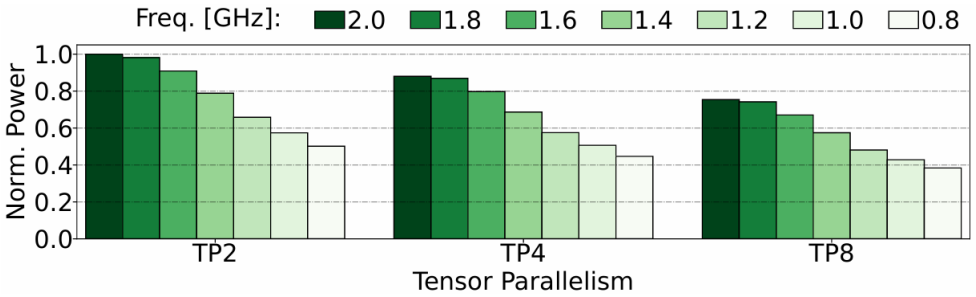
(a) *TTFT (tempo até o primeiro* token*)*



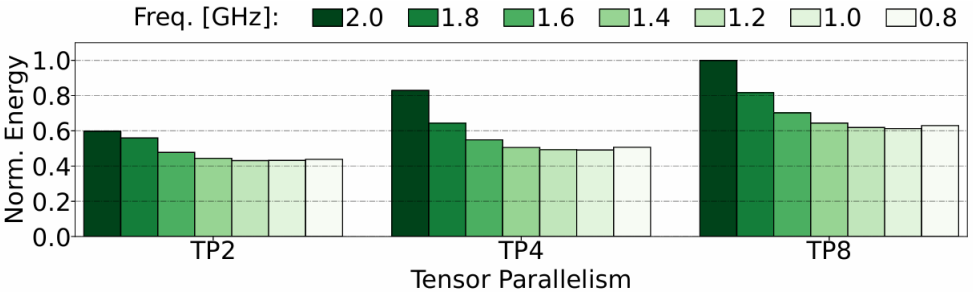
(b) *TBT (tempo entre* tokens *consecutivos)*



(c) throughput*, ou número de inferências por segundo*



(d) *potência média das GPUs*



(e) *energia média consumida pelas GPUs*

Figura 2.14: *Resultados da redução de frequência das GPUs para inferências de LLMs em três configurações de paralelismo de tensores distintas (*Stojkovic *et al., 2024)*

Conclusão

Neste trabalho, foi feito um estudo sobre maneiras de calcular e reduzir a emissão de CO_2 e demais gases poluentes, principalmente a partir da redução do gasto energético, de aplicações de aprendizado de máquina, com um foco circunstancial em grandes modelos de linguagem baseados em *Transformers*.

Percebe-se que a necessidade de se desenvolver formas de estimar as emissões surge da falta de dados oficiais confiáveis sobre essas emissões. Sobre o carbono incorporado, a falta de dados ou dados de baixa qualidade tornam o cálculo inviável, mesmo com o nível de granularidade maior fornecido pelo modelo STEC em relação ao ACT, enquanto o cálculo do carbono operacional se torna trivial pelo uso de *profilers*, mas o resultado do *profiling* é bastante dependente do *hardware* utilizado. Logo, mesmo com modelos públicos como os modelos DeepSeek, é difícil replicar perfeitamente o processo de treinamento e inferência.

Dos métodos de redução, aqueles baseados em otimizar o uso do *hardware* são provavelmente mais promissores no âmbito dos modelos comerciais, tornam o treinamento e a inferência mais rápidos sem piorar as métricas de qualidade da inferência. No entanto, é necessário explorar mais as relações de custo-benefício entre as métricas dos modelos e o custo operacional, para que seja possível aplicar os demais métodos, tornando as aplicações mais sustentáveis.

É interessante também que trabalhos futuros sejam dedicados a alinhar perspectivas ecológicas sobre IA/ML com outras teorias de ecologia e sustentabilidade, como o princípio dos 3 Rs (reduzir, reutilizar e reciclar) e os Objetivos de Desenvolvimento Sustentável definidos, definidos pelo programa da ONU *Transformando o nosso mundo: a Agenda 2030 para o Desenvolvimento Sustentável* (Nações Unidas, 2016). Os 17 objetivos não tratam apenas do cuidado com o meio-ambiente e redução das diversas formas de poluição ambiental; tratam também de questões humanitárias como o fim da pobreza e da fome (objetivos 1 e 2), promoção da igualdade de gênero (objetivo 5) e manutenção da paz (objetivo 16).

Referências

[Altman 2025] Sam Altman. "The gentle singularity" (2025). URL: https://blog.samaltman.com/the-gentle-singularity (citado na pg. 13).

[Bradbury *et al.* 2018] James Bradbury *et al. JAX: composable transformations of Python+NumPy programs*. Versão 0.3.13. 2018. URL: http://github.com/jax-ml/jax (citado na pg. 14).

[Casson 2023] Adam Casson. "Transformer flops" (2023). URL: https://adamcasson.com/posts/transformer-flops (citado na pg. 15).

[Chatterji *et al.* 2025] Aaron Chatterji *et al. How People Use ChatGPT*. Working Paper 34255. National Bureau of Economic Research, set. de 2025. DOI: 10.3386/w34255. URL: http://www.nber.org/papers/w34255 (citado na pg. 1).

[Chawla 2025] Avi Chawla. "Transformer vs. mixture of experts in llms" (2025). URL: https://www.dailydoseofds.com/p/transformer-vs-mixture-of-experts-in-llms/ (citado nas pgs. vii, 3).

[Chowdhery *et al.* 2023] Aakanksha Chowdhery *et al.* "Palm: scaling language modeling with pathways". *J. Mach. Learn. Res.* 24.1 (jan. de 2023). ISSN: 1532-4435 (citado na pg. 15).

[DeepSeek-AI *et al.* 2025] DeepSeek-AI *et al. DeepSeek-V3 Technical Report*. 2025. arXiv: 2412.19437 [cs.CL]. URL: https://arxiv.org/abs/2412.19437 (citado na pg. 2).

[Diaz 2025] Jaclyn Diaz. "Why google's search engine trial is about ai". *NPR.org* (2025). URL: https://www.npr.org/2025/04/29/nx-s1-5377353/google-antitrust-remedies-trial-ai (citado na pg. 1).

[Elsworth *et al.* 2025] Cooper Elsworth *et al. Measuring the environmental impact of delivering AI at Google Scale*. 2025. arXiv: 2508.15734 [cs.AI]. URL: https://arxiv.org/abs/2508.15734 (citado na pg. 13).

[Gupta *et al.* 2022] Udit Gupta *et al.* "Act: designing sustainable com puter systems with an architectural carbon modeling tool". In: *Proceedings of The 49th Annual International Symposium on Computer Architecture (ISCA '22)*. ACM, 2022 (citado nas pgs. vii, 6, 7, 9–11).

[Hammond e Jones 2009] Geofrrey Hammond e Craig Jones. *Embodied Carbon: The Inventory of Carbon and Energy(ICE)*. 2ª ed. BSRIA, 2009 (citado na pg. 5).

[He 2022] Horace He. *The "Ideal" PyTorch FLOP Counter (with __torch_dispatch__)*. 2022. url: https://dev-discuss.pytorch.org/t/the-ideal-pytorch-flop-counter-with-torch-dispatch/505 (citado na pg. 13).

[Hoffmann *et al.* 2022] Jordan Hoffmann *et al. Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL]. url: https://arxiv.org/abs/2203.15556 (citado nas pgs. vii, 15).

[Ivanov *et al.* 2021] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li e Torsten Hoefler. *Data Movement Is All You Need: A Case Study on Optimizing Transformers*. 2021. arXiv: 2007.00072 [cs.LG]. url: https://arxiv.org/abs/2007.00072 (citado nas pgs. 15, 25).

[B. Li, Basu Roy *et al.* 2023] Baolin Li, Rohan Basu Roy *et al.* "Toward sustainable hpc: carbon footprint estimation and environmental implications of hpc systems". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '23. Denver, CO, USA: Association for Computing Machinery, 2023. isbn: 9798400701092. doi: 10.1145/3581784.3607035. url: https://doi.org/10.1145/3581784.3607035 (citado nas pgs. 8, 9).

[B. Li, Samsi *et al.* 2023] Baolin Li, Siddharth Samsi, Vijay Gadepally e Devesh Tiwari. "Clover: toward sustainable ai with carbon-aware machine learning inference service". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '23. ACM, nov. de 2023, pp. 1–15. doi: 10.1145/3581784.3607034. url: http://dx.doi.org/10.1145/3581784.3607034 (citado nas pgs. viii, 25–27).

[C. Li *et al.* 2016] Chao Li, Yi Yang, Min Feng, Srimat Chakradhar e Huiyang Zhou. "Optimizing memory efficiency for deep convolutional neural networks on gpus". In: *SC '16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. 2016, pp. 633–644. doi: 10.1109/SC.2016.53 (citado nas pgs. viii, 21, 23, 24).

[Nações Unidas 2016] Assembleia Geral das Nações Unidas. *Transformando nosso Mundo: A Agenda 2030 para o Desenvolvimento Sustentável*. 2016. url: https://www.mds.gov.br/webarquivos/publicacao/brasil_amigo_pesso_idosa/agenda2030.pdf (citado na pg. 31).

[O'Brien 2024] Isabel O'Brien. "Data center emissions probably 662% higher than big tech claims. can it keep up the ruse?" *theguardian.com* (2024). url: https://www.theguardian.com/technology/2024/sep/15/data-center-gas-emissions-tech (citado na pg. 3).

[Paszke *et al.* 2019] Adam Paszke *et al.* "Pytorch: an imperative style, high-performance deep learning library". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019 (citado na pg. 13).

[Puvis de Chavannes *et al.* 2021] Lucas Høyberg Puvis de Chavannes, Mads Guldborg Kjeldgaard Kongsbak, Timmie Rantzau e Leon Derczynski. "Hyperparameter power impact in transformer language model training". In: *Proceedings of the Second Workshop on Simple and Efficient Natural Language Processing*. Ed. por Nafise Sadat Moosavi *et al.* Virtual: Association for Computational Linguistics, nov. de 2021, pp. 96–118. doi: 10.18653/v1/2021.sustainlp-1.12. url: https://aclanthology.org/2021.sustainlp-1.12/ (citado nas pgs. vii, viii, 19–21).

[Schwartz *et al.* 2020] Roy Schwartz, Jesse Dodge, Noah A. Smith e Oren Etzioni. "Green ai". *Commun. ACM* 63.12 (nov. de 2020), pp. 54–63. issn: 0001-0782. doi: 10.1145/3381831. url: https://doi.org/10.1145/3381831 (citado na pg. 13).

[Shehabi *et al.* 2024] A. Shehabi *et al.* "2024 united states data center energy usage report" (2024). url: http://dx.doi.org/10.71468/P1WC7Q (citado na pg. 2).

[Singla *et al.* 2025] Alex Singla *et al. The state of AI in 2025: Agents, innovation, and transformation*. 2025. url: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai (acesso em 11/12/2025) (citado nas pgs. vii, 1, 2).

[Sovrasov 2024] Vladislav Sovrasov. *ptflops: a flops counting tool for neural networks in pytorch framework*. 2024. url: https://github.com/sovrasov/flops-counter.pytorch (citado na pg. 13).

[Srivathsan *et al.* 2024] Bhargs Srivathsan *et al. AI power: Expanding data center capacity to meet growing demand*. 2024. url: https://www.mckinsey.com/industries/technology-media-and-telecommunications/our-insights/ai-power-expanding-data-center-capacity-to-meet-growing-demand/ (acesso em 09/10/2025) (citado nas pgs. 2, 5).

[Stojkovic *et al.* 2024] Jovan Stojkovic, Esha Choukse, Chaojie Zhang, Inigo Goiri e Josep Torrellas. *Towards Greener LLMs: Bringing Energy-Efficiency to the Forefront of LLM Inference*. 2024. arXiv: 2403.20306 [cs.AI]. url: https://arxiv.org/abs/2403.20306 (citado nas pgs. viii, 28–30).

[*Synergy Identifies the World's Top 20 Locations for Hyperscale Data Centers* 2024] *Synergy Identifies the World's Top 20 Locations for Hyperscale Data Centers*. Rel. técn. Reno, Nevada, USA: Synergy Research Group, 2024 (citado na pg. 3).

[Tannu e Nair 2023] Swamit Tannu e Prashant J. Nair. "The dirty secret of ssds: embodied carbon". *SIGENERGY Energy Inform. Rev.* 3.3 (out. de 2023), pp. 4–9. doi: 10.1145/3630614.3630616. url: https://doi.org/10.1145/3630614.3630616 (citado nas pgs. vii, 11).

[Team 2025] Qwen Team. *Qwen3-Max: Just Scale it*. Set. de 2025 (citado na pg. 2).

[TOP500 2025] TOP500. *TOP500 List - June 2025*. 2025. URL: https://top500.org/lists/top500/list/2025/06/ (acesso em 09/10/2025) (citado na pg. 9).

[Trask *et al.* 2015] Andrew Trask, David Gilmore e Matthew Russell. *Modeling Order in Neural Word Embeddings at Scale*. 2015. arXiv: 1506.02338 [cs.CL]. URL: https://arxiv.org/abs/1506.02338 (citado na pg. 1).

[Vaswani *et al.* 2023] Ashish Vaswani *et al. Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762 (citado na pg. 1).

[Wadenstein e Vanderbauwhede 2025] Mattias Wadenstein e Wim Vanderbauwhede. "Life cycle analysis for emissions of scientific computing centres". *The European Physical Journal C* 85.8 (ago. de 2025). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-025-14650-8. URL: http://dx.doi.org/10.1140/epjc/s10052-025-14650-8 (citado na pg. 10).

[Wang *et al.* 2019] Yue Wang *et al. E2-Train: Training State-of-the-art CNNs with Over 80% Energy Savings*. 2019. arXiv: 1910.13349 [cs.LG]. URL: https://arxiv.org/abs/1910.13349 (citado nas pgs. vii, 16–18).

[Zhang *et al.* 2024] Xiaoyang Zhang, Yijie Yang e Dan Wang. "Spatial-temporal embodied carbon models for the embodied carbon accounting of computer systems". In: *Proceedings of the 15th ACM International Conference on Future and Sustainable Energy Systems*. e-Energy '24. Singapore, Singapore: Association for Computing Machinery, 2024, pp. 464–471. ISBN: 9798400704802. DOI: 10.1145/3632775.3661939. URL: https://doi.org/10.1145/3632775.3661939 (citado nas pgs. vii, 7–9).