DIGA: Definição e Implementação de Modelos com Aprendizado Generativo para Aplicações Inteligentes

Lucas Airam C. de Souza[1,2], Guilherme Araújo Thomaz[1], Fernando Dias de M. Silva[1], Mateus da Silva Gilbert[1], Nadjib Achir[2], Miguel Elias M. Campista[1] e Luís Henique M. K. Costa[1]

[1]Universidade Federal do Rio de Janeiro, GTA/DEL-Poli/PEE-COPPE, Brasil
[2]École Polytechnique, INRIA Saclay, França

{airam,guiaraujo,fernandodias,gilbert,miguel,luish}@gta.ufrj.br

nadjib.achir@inria.fr

Resumo. *O aprendizado de máquina já está presente em diversas aplicações cotidianas. Entretanto, determinar qual modelo utilizar é uma tarefa árdua devido ao grande número de opções disponíveis. Esse cenário torna-se ainda mais complexo em redes veiculares, onde as aplicações exigem baixa latência e os dispositivos são heterogêneos. Assim, torna-se necessário um mecanismo que estime o tempo de inferência e selecione o modelo mais apropriado para um dispositivo no qual será executado. Este trabalho propõe o DIGA, um sistema de seleção de modelos de aprendizado de máquina projetado para o setor automotivo. Considerando características do dispositivo e do modelo candidato, o sistema estima o tempo de inferência necessário para processar uma amostra e verifica se o modelo atende aos critérios de latência estabelecidos pela aplicação em um cenário com múltiplos dispositivos. Para estimar o tempo de inferência de cada modelo em um dispositivo, este trabalho compara o uso de um modelo matemático determinístico com quatro modelos diferentes de inteligência artificial generativa, usando como referência o tempo real de execução nos dispositivos. Os resultados mostram que modelos matemáticos determinísticos usados na literatura subestimam o tempo de inferência em 80% dos casos. Portanto, demonstra-se a necessidade de considerar a estrutura do modelo, pois o número de operações de ponto flutuante encontrado matematicamente é insuficiente para determinar o tempo de inferência.*

Abstract. *Machine learning is already integrated into many everyday applications. However, selecting the appropriate model is challenging due to the numerous available options. This challenge becomes even more complex in vehicular networks, where applications require low latency and operate on heterogeneous devices. Thus, a mechanism that estimates inference time and selects the most suitable model for the target device becomes essential. This work proposes DIGA, a machine-learning model selection system designed for the automotive sector. Considering both device and candidate model characteristics, the system estimates the inference time required to process a sample and verifies whether the model meets the application's latency criteria in a multi-device scenario. To estimate the inference time of each model on a device, this work compares a deterministic mathematical model with four different generative artificial intelligence models, using real execution times as a reference.*

Este trabalho foi realizado com recursos do CNPq, CAPES, FAPERJ, FAPESP (2023/00811-0 e 2023/00673-7) e Fundação de Desenvolvimento da Pesquisa - Fundep - Rota 2030, e nossos parceiros Stellantis e Mobway.

The results show that deterministic mathematical models used in the literature underestimate inference time in 80% of cases. Therefore, this study highlights the need to consider model structure, as the number of floating-point operations derived mathematically is insufficient to accurately determine inference time.

1. Introdução

A alta variedade de modelos de aprendizado de máquina e dispositivos utilizados para inferência cria um desafio para determinar qual modelo a ser utilizado para uma aplicação, dadas suas restrições de tempo de resposta. Apesar da existência de testes padronizados, o desempenho do modelo varia de acordo com os recursos computacionais, formato de dados utilizado, entre outros aspectos do ambiente de execução. Assim, é importante caracterizar o ambiente no qual o modelo será executado para garantir que o mesmo atende aos critérios de tempo de resposta da aplicação.

Um trabalho anterior da equipe demonstrou que a inteligência artificial generativa pode auxiliar na definição de modelos e hiperparâmetros, reduzindo o tempo de definição quando comparado às técnicas de otimização, busca exaustiva ou AutoML [de Souza et al. 2024]. Entretanto, o treinamento dos modelos de aprendizado de máquina frequentemente desconsidera as condições do dispositivo no qual serão executados e os requisitos da aplicação que os utiliza. Outros trabalhos focam a determinação do desempenho de modelos em diferentes ambientes [Boutros et al. 2020, Reddi et al. 2020, Mattson et al. 2020]. No entanto, estes trabalhos visam ambientes de execução em nuvem.

Este trabalho propõe o DIGA, um sistema para caracterização do tempo de inferência de modelos de aprendizado de máquina cuja finalidade é definir qual modelo será executado em um dispositivo embarcado, baseado em suas restrições computacionais e da aplicação inteligente. A métrica utilizada para determinar se a execução de um modelo é viável é o tempo de inferência estimado, dado o tempo máximo tolerado de atraso da aplicação inteligente. Para a escolha do modelo consideram-se informações da aplicação e dos dados utilizados para realizar a inferência. A caracterização do tempo de inferência é realizada de três formas distintas: (i) execução no dispositivo, (ii) modelagem matemática do tempo de inferência [Asperti et al. 2021] e (iii) utilização de modelos extensos de linguagem (*Large Language Models* - LLMs). O tempo de inferência estimado é utilizado para filtrar modelos que não atendam às restrições de tempo da aplicação inteligente.

Diferentemente de trabalhos anteriores, o DIGA caracteriza as tarefas envolvidas na execução de modelos de aprendizado de máquina para aplicações inteligentes. O objetivo é melhorar o desempenho da aplicação inteligente pelo mapeamento modelo-cliente, considerando a heterogeneidade dos dispositivos embarcados. As informações obtidas sobre as características de modelos, dispositivos e dados são armazenadas em um banco de dados e auxiliam a decisão de escolha do modelo. Os resultados mostram que a organização dos neurônios influencia o tempo de inferência. Além disso, o número menor de operações de ponto flutuante (*FLOating Point Operations* - FLOPs)[1] de um modelo não acarreta necessariamente economia de memória. Dessa forma, é necessário considerar também o tamanho do modelo a ser transferido em casos nos quais os dispositivos possuem conexões limitadas ou em que o tempo de transferência do modelo seja relevante. Os experimentos mostram que o DIGA, ao utilizar LLMs, identifica corretamente 25% mais modelos inviáveis para os dispositivos em comparação com a abordagem exclusivamente baseada em FLOPs utilizando o modelo Mistral.

[1]Neste artigo, utiliza-se FLOPs para indicar o plural da quantidade de operações de ponto flutuante necessária para realizar uma inferência, enquanto FLOPS é a taxa de operações por segundo realizadas por um dispositivo.

2. Trabalhos Relacionados

Aplicações que utilizam inteligência artificial possuem diferentes requisitos em relação ao chamado tempo de resposta do modelo, ou tempo de inferência. O tempo de inferência de um modelo de aprendizado de máquina depende de parâmetros do modelo, dados de entrada e dos recursos disponíveis no dispositivo que o executa. Esta seção discute as principais técnicas utilizadas para estimar o tempo de inferência de modelos de aprendizado de máquina e o potencial da inteligência artificial (IA) generativa em tarefas similares.

2.1. Técnicas para Estimar o Tempo de Inferência de Modelos de IA

Comparar diferentes modelos de aprendizado de máquina de maneira justa é fundamental para classificar e selecionar a opção mais adequada para cada aplicação. O desempenho desses modelos é sensível a diferentes fatores que podem favorecer um modelo em relação ao outro se não forem ajustados adequadamente. Chitty *et al.* discutem quatro formas de estimar o tempo de inferência de um modelo de aprendizado de máquina: i) execução no dispositivo, ii) modelos preditivos, iii) aproximação a partir de um modelo matemático, também denominado *proxy*, ou iv) simulação [Chitty-Venkata et al. 2023]. Entretanto, nota-se que simulações envolvem o conhecimento de parâmetros específicos, como arquitetura do dispositivo, para refletir bons resultados, tornando esta alternativa complexa para cenários com múltiplos dispositivos.

2.1.1. Execução em Dispositivo

pCAMP (*performance CompArison of Machine learning Packages*) [Zhang et al. 2018] é um estudo para caracterizar o tempo de inferência, uso de memória e consumo de energia de dispositivos de borda ao executar dois modelos de aprendizado profundo. Os autores caracterizam os dispositivos utilizando implementações dos modelos empregando quatro bibliotecas e dispositivos de borda. Entretanto, a alternativa de determinação do tempo de inferência a partir da execução no dispositivo é complexa em um cenário distribuído, pois os clientes possuem recursos computacionais limitados e são numerosos. Portanto, realizar esta estimativa utilizando o dispositivo do cliente aumenta a sua utilização de energia e o tempo necessário para obter o modelo. Assim, uma alternativa para essa abordagem é a utilização de modelos preditivos.

Mattson *et al.* propõem uma estrutura padronizada para avaliar o desempenho de diferentes modelos de aprendizado de máquina para treinamento [Mattson et al. 2020] e inferência [Reddi et al. 2020]. A padronização consiste em parâmetros fixos como conjuntos de dados, métrica e formato de dados de entrada para garantir justiça na comparação. A avaliação combina diferentes dispositivos e bibliotecas para auxiliar na arquitetura de aplicações inteligentes. Wang *et al.* desenvolvem a ferramenta FLINT (*Federated Learning Integration*), adotada no contexto de transição do aprendizado centralizado para o aprendizado federado *cross-device* no LinkedIn [Wang et al. 2023]. O trabalho revela que os usuários do aplicativo possuem *smartphones* com poder de processamento distinto. Assim, uma seleção de clientes que não leva esse fator em consideração impacta negativamente a reputação da empresa, pois dispositivos mais restritos computacionalmente podem levar a um desempenho insatisfatório na experiência do usuário. O trabalho propõe um conjunto de dispositivos de teste em nuvem e um ajuste de complexidade de modelo para evitar esse problema, e realiza experimentos assumindo tarefas típicas utilizadas no LinkedIn. Porém, a avaliação de desempenho ocorre apenas com dispositivos reais, e sem comparação com outras alternativas.

Contudo, a disponibilidade dos dispositivos para estimar o tempo de inferência, como nos trabalhos anteriores, nem sempre é possível. Assim, uma alternativa é armazenar dados de execuções prévias para ajustar modelos preditivos. Desta forma, os modelos podem acumular o conhecimento sem requerer o acesso ao dispositivo para futuras previsões do tempo de inferência.

2.1.2. Modelos Preditivos

Os modelos preditivos consistem em modelos de aprendizado de máquina utilizados em um problema de regressão para prever o tempo de inferência de outros modelos. Para isso, os modelos preditivos necessitam de dados reais obtidos a partir da execução de inferências em dispositivos para ajustar os parâmetros do modelo.

Idelbayev e Carreira-Perpiñán propõem um modelo para calcular o tempo de inferência de redes neurais em dispositivos para determinar o grau de compressão a utilizar no modelo [Idelbayev e Carreira-Perpiñán 2021]. Para isso, os autores propõem um modelo que consiste na soma do tempo de processamento em cada camada. Os autores revelam, porém, que o tempo de inferência é difícil de ser estimado, devido a otimizações da execução e aquecimento dos núcleos de processamento que podem alterar o resultado obtido. Por fim, para determinar o tempo de inferência considerando diferentes níveis de compressão, os autores realizam execuções e ajustam um modelo linear sobre os pontos obtidos.

Weng *et al.* focam na redução do desperdício de recursos computacionais ao realizar tarefas de inferência em nuvem [Weng et al. 2022]. Para isso, estimam o tempo de execução de uma tarefa de inferência e demonstram que há repetições de conjuntos de instruções nas tarefas de aprendizado de máquina. Assim, os autores utilizam o modelo de árvore de classificação e regressão (*Classification And Regression Tree* - CART) com informações sobre o usuário, grupo e quantidade de recursos solicitados para determinar o tempo de execução da tarefa de aprendizado. Cui *et al.* propõem um serviço para determinação da latência de inferência em um ambiente de nuvem para compartilhamento de recursos [Cui et al. 2021]. Dessa forma, é possível garantir qualidade de serviço para clientes diferentes executando inferência sobre o mesmo dispositivo de unidade de processamento gráfico (*Graphics Processing Unit* - GPU). Para estimar o tempo de execução de uma predição, os autores utilizam um modelo Perceptron Multicamadas (*Multi-Layer Perceptron* - MLP) treinado com dados coletados a partir da execução de diferentes modelos. Entretanto, a abordagem necessita executar diversas vezes o processo de inferência para coleta de dados para o treinamento da MLP e é aplicada para um dispositivo específico.

Assim, a utilização de modelos preditivos implica na obtenção de dados para ajuste do modelo, o que é complexo em um ambiente distribuído com diferentes recursos computacionais por cada cliente. Portanto, é necessário utilizar um modelo que seja capaz de generalizar a estimativa do tempo de inferência em diferentes cenários sem assumir que o estimador possua acesso ao dispositivo físico. Assim, os LLMs surgem como uma alternativa para processar informações textuais e retornar estimativas sobre os modelos.

2.1.3. Modelos Matemáticos

A aproximação a partir de *proxies* ou modelos matemáticos consiste em descrever por meio de uma equação o comportamento esperado para o tempo de inferência de acordo com os parâmetros que mais o influênciam. Apesar de depender da qualidade da modelagem para gerar resultados próximos do real, a principal vantagem desse método é mitigar a necessidade de acesso ao dispositivo no qual o modelo será executado e a facilidade de adaptar ao cenário distribuído.

Desislavov *et al.* utilizam um modelo matemático para determinar o consumo energético de diferentes modelos de aprendizado de máquina [Desislavov et al. 2023]. O modelo, simples, aproxima o consumo estimado do real, e pode ser facilmente adaptado para estimar o tempo de inferência.

Asperti *et al.* propõem uma fórmula mais complexa para modelar a correlação entre o número de FLOPs, a energia utilizada e o tempo de inferência de forma mais realista [Asperti et al. 2021]. O modelo possui resultados melhores, entretanto, aumenta o conhecimento necessário de parâmetros para ajustar o modelo para diferentes dispositivos e modelos.

Como os modelos matemáticos acurados necessitam de informações específicas de dispositivos e modelos, isso pode dificultar a sua rápida implementação, apesar do baixo custo computacional envolvido para gerar as estimativas. Por outro lado, modelos mais simples podem gerar respostas distantes dos valores reais. Assim, uma alternativa de baixo custo de implementação são os modelos de aprendizado generativo, que possuem conhecimentos adquiridos por meio de imensas bases de dados, reduzindo o alto nível de conhecimento técnico para ajustar os modelos matemáticos e ao mesmo tempo produzindo uma resposta acurada.

2.2. Aprendizado Generativo para Definição de Modelos

Os LLMs possuem alta eficiência em tarefas de processamento de linguagem natural. Entretanto, o desempenho em tarefas de computação de alto desempenho (*High-Performance Computing* - HPC) é limitado [Ding et al. 2023]. Assim, Ding *et al.* propõem um ajuste fino do modelo LLaMA (*Large Language Model Meta AI*) para aumentar o seu desempenho em duas tarefas de HPC em aprendizado de máquina: tratamento de dados e detecção de condição de corrida. Entretanto, o foco é voltado para a produção e análise de códigos, o que está fora do escopo deste trabalho. Em um trabalho anterior [de Souza et al. 2024] demonstrou-se que a inteligência artificial generativa pode auxiliar na definição de modelos e hiperparâmetros, reduzindo o tempo de definição quando comparada a técnicas de otimização, busca exaustiva ou AutoML. Porém, considerar a heterogeneidade de dispositivos de borda durante a escolha do modelo ainda é uma questão em aberto, tratada no presente trabalho.

Diferentemente dos trabalhos anteriores, este trabalho propõe DIGA para caracterizar e definir modelos de aprendizado profundo considerando as restrições dos dispositivos de borda. Enquanto trabalhos anteriores focam a avaliação de desempenho em nuvem, o trabalho atual foca em dispositivos distribuídos e embarcados em veículos por questões de latência da aplicação. A seguir, são apresentados os detalhes da proposta, descrevendo os serviços e componentes implementados.

3. DIGA: Um Sistema de Caracterização de Modelos

Este trabalho propõe e implementa o DIGA, um sistema concebido para a definição de modelos para aplicações inteligentes que executam em dispositivos de borda heterogêneos. Assume-se que as aplicações impõem restrições relativas ao tempo máximo para processamento de uma amostra, também denominado tempo de inferência [Reddi et al. 2020, Clancy et al. 2024]. Dessa forma, neste trabalho preocupa-se com a avaliação da requisição dos modelos quando os dispositivos possuem restrições computacionais. Assim, o DIGA implementa serviços que permitem caracterizar os principais componentes envolvidos na inferência, como visto a seguir.

3.1. Arquitetura do DIGA

A Figura 1 exibe a arquitetura do sistema DIGA. Em trabalhos anteriores [de Souza et al. 2024], uma primeira versão da arquitetura foi proposta, na qual o serviço de segurança, o serviço de inteligência distribuída e o treinamento centralizado de modelos são implementados. A Figura 1 apresenta em azul os serviços implementados no trabalho anterior e que são reutilizados neste trabalho, enquanto os serviços em verde-claro são implementados ou

adaptados e avaliados neste trabalho. Assim, a proposta implementa o serviço de caracterização de modelos, serviço de caracterização veicular e adapta o serviço de inteligência distribuída.

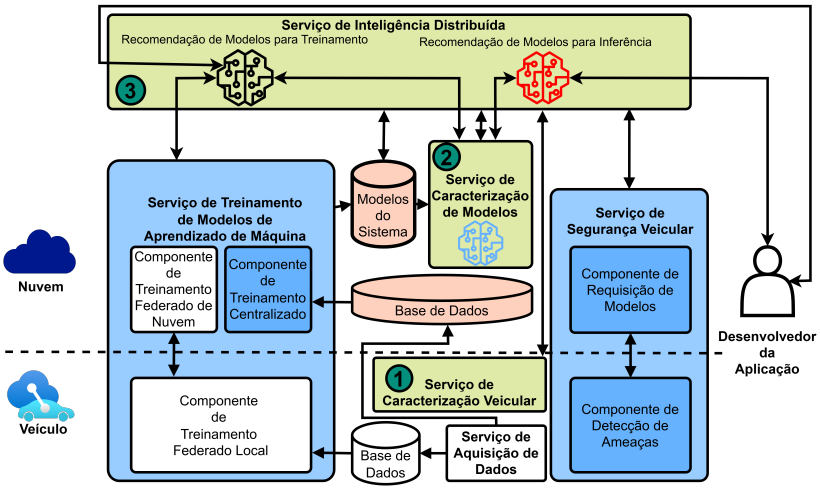


Figura 1: Arquitetura da proposta. Este trabalho implementa o (1) serviço de caracterização veicular, o (2) serviço de caracterização de modelos e adapta o funcionamento do (3) serviço de inteligência distribuída.

O serviço de caracterização veicular estima a capacidade computacional do dispositivo cliente, tipos de sensores disponíveis, canal de comunicação e o formato dos dados produzidos pelo veículo para a aplicação inteligente. Essas informações são utilizadas pelo serviço de inteligência distribuída para definir quais modelos o dispositivo é capaz de executar conforme as restrições da aplicação. O foco deste trabalho é avaliar a capacidade computacional do dispositivo, taxa de transmissão e o formato dos dados, tendo como restrição o tempo de inferência da aplicação e a redução do tempo de transferência do modelo.

O serviço de caracterização de modelos recebe informações relativas aos modelos, como nome, caso seja um modelo conhecido, hiperparâmetros e tamanho. Além disso, o serviço consome os dados gerados pelo serviço de caracterização veicular. Com essas informações, o serviço gera uma estimativa do tempo de inferência, utilizado como um critério de seleção de modelos. Entre as estratégias avaliadas neste trabalho, a inteligência artificial generativa é proposta para a caracterização dos modelos do sistema, como exibido na Figura 1.

Por fim, o serviço de inteligência distribuída, anteriormente definido como apenas um LLM [de Souza et al. 2024], é implementado por dois LLMs. O primeiro LLM é especialista em definir modelos e hiperparâmetros para o treinamento, fora do escopo do trabalho atual, pois foi analisado em trabalhos anteriores. O segundo identifica as restrições da aplicação inteligente ao recomendar modelos para um dispositivo. Ambos interagem com o serviço de caracterização de modelos a fim de determinar qual modelo treinar ou qual modelo retornar ao cliente com base nas restrições da aplicação. Dessa forma, o sistema atual utiliza um conjunto de agentes, especialistas em determinadas tarefas [Zhao et al. 2024]. A interação entre esses componentes é exibida na Figura 1.

3.2. Estimativa do Tempo de Inferência

O objetivo deste trabalho é determinar o tempo de inferência de um modelo de aprendizado de máquina para utilizar essa informação como uma restrição do problema de definição de modelos para um dispositivo. Esse tempo é influenciado por diversos fatores [Reddi et al. 2020]. Cada modelo de aprendizado de máquina impõe requisitos mínimos

de recursos computacionais para procedimentos de inferência. Os recursos computacionais também variam conforme os dados utilizados, dependendo do tamanho e formato, e da necessidade de pré-processamento antes da inferência.

Este trabalho propõe o uso da inteligência artificial generativa como uma nova maneira de estimar o tempo de inferência, e compara os resultados gerados com os valores obtidos pela execução no dispositivo real e um *proxy* que utiliza a modelagem matemática do tempo de inferência. Para inteligência artificial generativa, investigam-se quatro modelos: Gemma, Mistral, LLaMA3.1 e LLaMA3.2. Esses modelos foram escolhidos por serem de código aberto e possuírem uma interface de programação de aplicações (*Application Programming Interface* - API) fácil de utilizar e integrar ao sistema.

Proxy com Modelagem Matemática: O *proxy* utilizado a partir da modelagem matemática é determinístico e descrito pela Equação

$$\hat{T_i} = \frac{F_m}{Fs_d}. \tag{1}$$

$\hat{T_i}$ é o tempo de inferência estimado, que é igual à quantidade de FLOPs do modelo F_m e a capacidade de FLOPs por segundo do dispositivo Fs_d.

Inteligência Artificial Generativa: Para estimar o tempo de inferência utilizando LLMs, aplica-se o *prompt* "*You are an expert in Machine Learning and Computer Architecture. You have expertise with the most popular machine learning models. Which is the expected inference time for M to execute one inference in a device with a memory of $M_{e_{RAM}}$ GB and a CPU frequency of F_{CPU} GHz, with C cores. Give me the answer in the format : inference time = mean +/- deviation with a specific mean value and a range deviation.*". As duas primeiras frases foram utilizadas para preparação do *prompt*, que tende a aumentar a precisão da resposta gerada [Memon et al. 2024]. A seguir, M representa o nome do modelo avaliado, que é substituído a cada avaliação. O mesmo ocorre para os valores $M_{e_{RAM}}$, F_{CPU} e C_{CPU}. Além disso, caso o dispositivo possua GPU, é adicionada a frase "The device has also a GPU with $M_{e_{GPU}}$ GB of memory and C_{GPU} cores with F_{GPU} GHz of frequency." ao *prompt*, após a descrição da CPU. O mesmo *prompt* foi utilizado para os diferentes modelos de inteligência artificial generativa avaliados para igualdade na comparação. As informações coletadas e geradas são armazenadas no banco de dados do sistema.

4. Experimentos e Análise dos Resultados

A primeira tarefa realizada neste trabalho é a caracterização de modelos disponíveis no sistema. Neste artigo, assume-se que o banco de dados de modelos foi previamente preenchido a partir do serviço de treinamento de modelos de aprendizado de máquina, seja de forma centralizada ou federada. Portanto, o DIGA é capaz de gerar informações do modelo como número de FLOPs, tamanho em memória, número de camadas e número de parâmetros. Outra tarefa é caracterizar as condições de execução do modelo. Assume-se que o cliente comunica-se com o servidor, especificando características dos seus dispositivos e dados por meio do serviço de caracterização veicular. Entre as informações compartilhadas estão: a quantidade de memória RAM nominal do dispositivo, taxa de FLOPS, frequência da CPU, quantidade de núcleos de processamento, se há disponibilidade de GPU, e características da GPU caso esteja disponível. Em relação aos dados, o veículo informa o formato de entrada e saída dos dados e o tipo de tarefa de aprendizado. Por fim, o objetivo é, a partir das informações coletadas acima, determinar o tempo de inferência do modelo, executando no dispositivo do cliente sobre amostras do

Tabela 1: Dispositivos utilizados na avaliação.

Dispositivo	N. Núcleos	N. Núcleos GPU	F. CPU (GHz)	F. GPU (MHz)	RAM (GB)	GFLOPS
PC i9	20		3,7		32	1664
Raspberry Pi 4	4		1,8		8	4.4
Raspberry Pi 5	4		2,4		8	10
Jetson Nano	20	128	1,43	0,921	4	235.8 (472)

mesmo formato que o conjunto de dados. A seguir, são discutidos os dispositivos e modelos utilizados nos experimentos desenvolvidos no artigo.

4.1. Dispositivos e Modelos Avaliados

Os dispositivos selecionados, descritos na Tabela 1 incluem desde um servidor de borda até placas SoC (*System on a Chip*) utilizadas em sistemas embarcados mais leves. Além disso, foram coletadas informações sobre a capacidade de processamento de cada um dos dispositivos em relação ao número de FLOPS para utilização no *proxy* matemático [TechPowerUp 2025]. Os dispositivos são utilizados para executar os diferentes modelos avaliados.

O caso de uso dos experimentos é o tempo de inferência de redes neurais convolucionais (*Convolutional Neural Network* - CNN) utilizadas para classificação de imagens, uma vez que esta é uma das tarefas mais cruciais e desafiadoras no cenário de assistência de segurança veicular e de pedestres em veículos autônomos. Assume-se que a entrada da rede é uma imagem urbana de 640 x 480 pixels redimensionada para 224 x 224 pixels, para torná-la compatível com a entrada das redes. A imagem aleatória de 640 x 480 foi obtida em um conjunto de dados de domínio público[2]. Outro pré-processamento comum é a normalização dos *sub-pixels* dos canais de cor RGB (*Red/Green/Blue*). Entretanto, esses tempos não foram considerados, uma vez que o escopo do artigo limita-se ao tempo de inferência. Evidentemente, em aplicações veiculares inteligentes, a codificação do vídeo e das imagens, a resolução da câmera e os outros módulos da aplicação também influenciam a sobrecarga final percebida para o usuário. Deve-se notar também que a acurácia da tarefa não é foco deste trabalho, uma vez que a imagem oferecida como entrada e a confiança gerada na saída não têm relação com o tempo de inferência. Ademais, diferentes tarefas, como detecção de objetos ou segmentação de imagens, utilizam as mesmas estruturas convolucionais, mas possuem métricas de desempenho diferentes, como p. ex. a IOU (*Intersection Over Union*).

A escolha das redes listadas na Figura 2 permite avaliar a acurácia das estimativas propostas para o tempo de inferência de modelos com diferentes características em termos de profundidade, número de operações em ponto flutuante (FLOPs) e disposição de parâmetros. Além disso, comparar modelos com diferentes arquiteturas permite identificar a influência de cada característica das redes neurais no tempo de inferência esperado. Por exemplo, a profundidade da MobileNetV2 e da InceptionV3 difere em menos de 3%, mas o número de FLOPs é nove vezes maior para a InceptionV3. Assim, ao comparar o tempo de inferência dos dois modelos, pode-se focar na influência que o número de FLOPs tem nos resultados encontrados. Diferentemente, a DenseNet121 e a InceptionV3 possuem FLOPs similares, apresentando uma diferença menor que 2%, porém a profundidade da DenseNet é mais que o dobro da InceptionV3. Assim, pode-se avaliar a influência da profundidade no tempo de inferência. Por fim, diferentes pares permitem analisar como cada característica de rede influencia o tempo de inferência encontrado, e quais estimativas capturam melhor essas relações. Se o tempo de inferência entre os dois mo-

[2]https://www.kaggle.com/datasets/pavansanagapati/images-dataset.

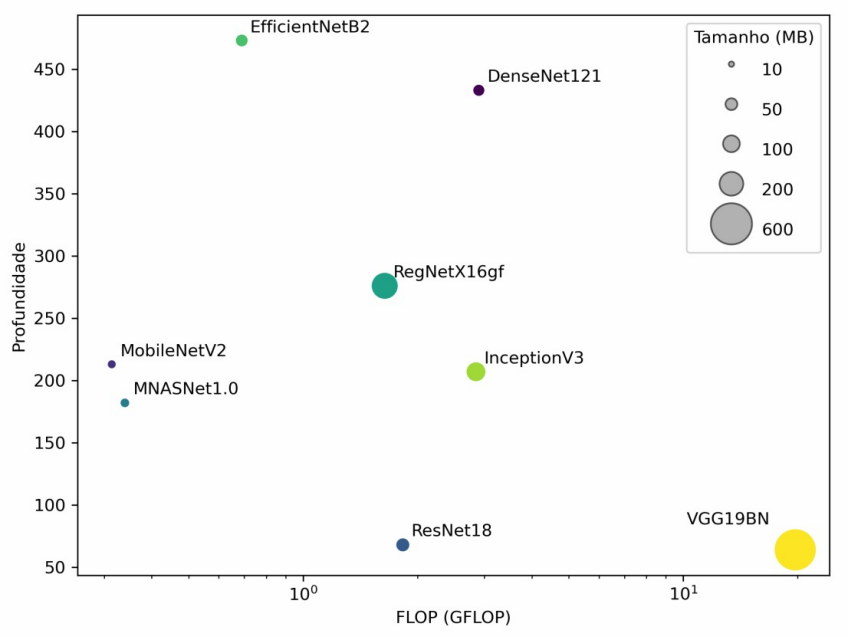


Figura 2: Comparação das características principais dos modelos de redes neurais avaliados.

delos for similar, então pode-se esperar que a profundidade da rede seja a característica mais relevante, e espera-se que os estimadores sejam capazes de identificar essa tendência. Assim, utilizando os diferentes dispositivos e modelos, são conduzidos dois experimentos: o primeiro visa a análise comparativa de diferentes modelos de IA generativa, neste caso, diferentes LLMs, e a avaliação de diferentes estimadores do tempo de inferência dos modelos. Por fim, o segundo foca a avaliação do tempo de transferência do modelo do servidor para o cliente, ou tempo de comunicação.

4.2. Avaliação de Estimadores do Tempo de Inferência de Modelos

A forma mais exata de obter o tempo de inferência, assumido como valor de referência neste trabalho, é executando a inferência com um modelo embarcado no dispositivo. Todos os modelos utilizados possuem pesos pré-treinados com a base de dados ImageNet, uma vez que o experimento não foca no treinamento. Assim, o primeiro experimento consiste na comparação da resposta para o tempo de inferência entre diferentes modelos de inteligência artificial generativa, utilizando como entrada o mesmo *prompt* definido na Seção 3. Além disso, os resultados obtidos são comparados aos valores obtidos por meio do modelo matemático e por meio da execução no dispositivo.

Os resultados apresentam os tempos para cada rede classificar uma imagem, obtidas com um código utilizando o *framework* PyTorch, na linguagem Python. As redes, previamente treinadas com o popular *dataset* de imagens ImageNet v2, foram obtidas no repositório TorchHub, que possui os modelos mais adotados em trabalhos de aprendizado profundo. O número de FLOPs de cada modelo, bem como o número de camadas, foi medido utilizando ferramentas do PyTorch, como Pthflops. A imagem foi redimensionada e normalizada utilizando o pacote *TorchVision*, sendo que este tempo é desprezível e portanto foi desconsiderado.

Os valores de intervalo de confiança para o modelo matemático foram estimados como 50% do valor nominal obtido pelo modelo, pois o modelo utilizado é determinístico em relação aos dados de entrada para a estimativa. Além disso, o melhor modelo de LLM foi utilizado para

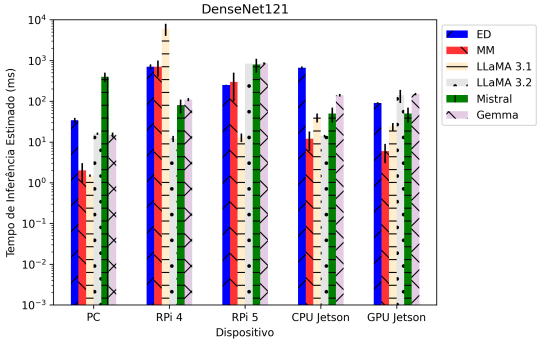
comparar os tempos retornados em relação ao valor real e o modelo matemático. Em todas as figuras de resultados, ED significa o tempo de inferência medido na Execução no Dispositivo e MM o tempo de inferência obtido pelo Modelo Matemático.

Pode-se observar que há uma grande variedade entre os valores retornados pelos diferentes modelos para um par dispositivo-modelo nas Figuras de 3(a) a 3(h). O texto retornado possui grande variedade, porém observa-se três métodos principais para estimar o tempo de inferência. Uma das técnicas foi utilizar informações do modelo e do dispositivo, como o *proxy* matemático, para retornar o tempo de inferência. A segunda alternativa comum no texto de saída foi a utilização de informações presentes em artigos de avaliação de modelos para sugerir o tempo de inferência. Por fim, a terceira alternativa foi uma combinação das duas alternativas anteriores, criando um modelo matemático e combinando informações geradas por artigos de avaliação de modelos para configurar os parâmetros da resposta.

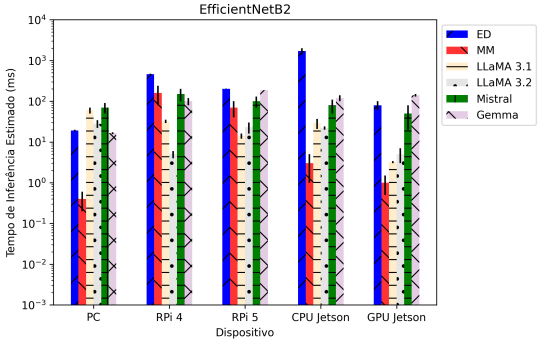
Comparando-os ao tempo real (barras ED), o desempenho está relacionado com a capacidade do modelo em apresentar estimativas próximas ao valor real e se a sua estimativa subestima o tempo de inferência. Os LLMs apresentaram estimativas bem distintas para os dispositivos e apresentando alta dificuldade para estimar o tempo de inferência nos Raspberry Pi. O Mistral ofereceu as estimativas mais próximas do real em mais de um dispositivo, apresentando os menores erros para estimativas em PC (excluindo os modelos DenseNet121 e EfficientNetB2) e em GPU Jetson (excluindo ResNet18 e RegNext). Outro LLM que se destaca é o Gemma, que obteve os menores erros dentre as LLMs para estimativas em Raspberry Pi, e teve desempenho comparável ao Mistral para estimativas de latências em GPU Jetson. Focando as situações onde a latência é subestimada, novamente o Mistral e o Gemma se destacam como os LLMs que apresentam a menor ocorrência de estimativas menores que a real, com o Gemma superando o Mistral. Outro objetivo dos estimadores é que esses sejam capazes de filtrar os modelos que conseguem operar em um intervalo de tempo. Para latências entre 20 ms e 100 ms, o Mistral é o estimador mais adequado. Exceto para os dispositivos RPi 4 e CPU Jetson, o Mistral recomenda apenas um modelo a mais que o ideal em todos os cenários (RegNetX16gf para o RPi 5).

Os valores estimados para o atraso máximo de uma aplicação veicular variam conforme a sua categoria [Clancy et al. 2024]. Para aplicações críticas de segurança é estipulado o tempo máximo entre 10 e 100 milissegundos. Assim, estes tempos de referência são utilizados para avaliar o tempo de inferência das aplicações inteligentes.

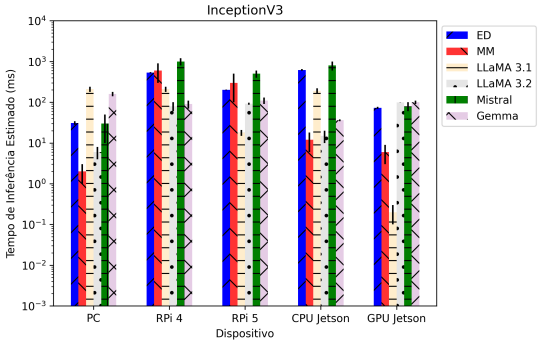
Dentre os modelos analisados, MNASNet1.0 apresenta os menores tempos de inferência nos modelos que utilizam CPU, sendo superado apenas pelo ResNet18 executado no Raspberry Pi 4. Retornando à Figura 2, percebe-se que a MNASNet1.0 é um dos menores modelos analisados, contendo o menor número de parâmetros e FLOPs, além de apresentar a terceira menor profundidade. Logo, o tempo de execução encontrado se alinha ao esperado quando analisamos as características da rede. Comparando modelos com profundidade similar, pode-se notar que os modelos com maior número de parâmetros (indicado pelo tamanho do modelo) e FLOPs executam sua inferência em um tempo maior. Por exemplo, InceptionV3 leva ao menos o dobro do tempo que MobileNetV2 para executar a inferência, e VGG19BN supera em mais de 7 vezes o tempo de execução de ResNet18. Esses resultados confirmam a hipótese esperada de que modelos com mais parâmetros e mais FLOPs apresentam uma maior latência de processamento. Isso também tende a ocorrer com modelos com tamanho e profundidade próximos, com DenseNet121 apresentando um tempo de inferência superior a EfficientNetB2, modelo que possui 2,21 GFLOPs a menos que o primeiro. Logo, o número de FLOPs tende a ser um fator preponderante para o tempo de inferência para modelos executados em CPU.



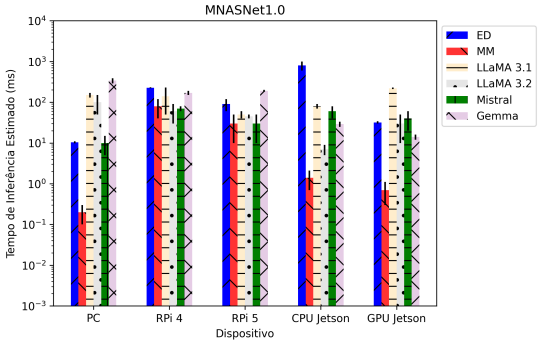
(a) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para DenseNet121.



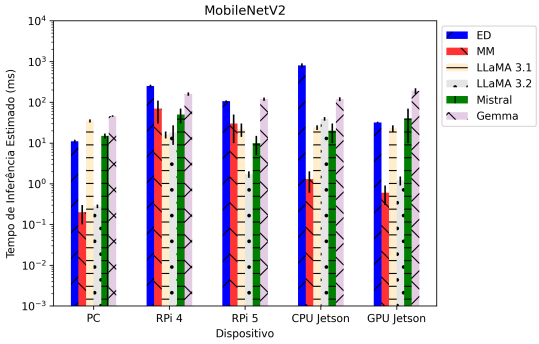
(b) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para EfficientNetB2.



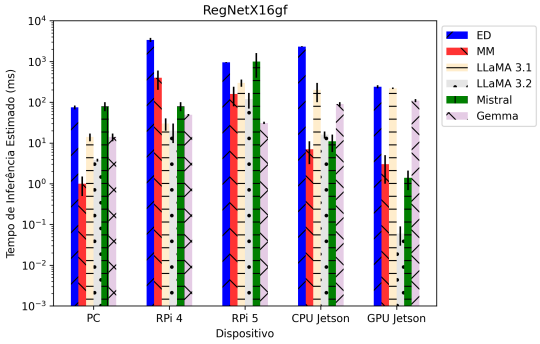
(c) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para InceptionV3.



(d) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para MNASNet1.0.



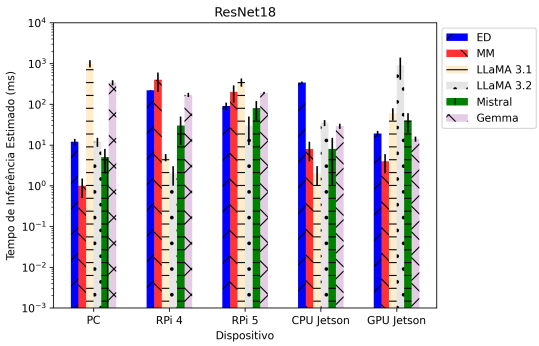
(e) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para MobileNetV2.



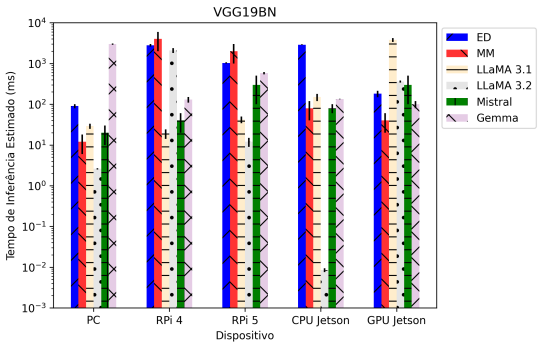
(f) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para RegNetX16gf.

Anasalisando os resultados é possível concluir que o modelo matemático subestima em 80% dos casos analisados o tempo de inferência, 32 dos 40 cenários. Isso implica que a utilização de um modelo simples para realizar a estimativa do tempo é insuficiente, uma vez que o tempo real pode ser acima do exigido pela aplicação. Dessa forma, o DIGA, que utiliza um LLM para prever o tempo de inferência, possui um resultado melhor quando usando o LLM Mistral, subestimando o tempo de inferência em 55% dos casos. Além disso, os demais LLMs subestimam menos o tempo de inferência em comparação com o modelo matemático, sendo subestimado em 60% para o Gemma, 65% para o LLaMMa 3.1 e 70% LLaMMa 3.2. Dessa forma, a utilização do modelo Mistral, reduz o erro de subestimação em 25% comparado ao modelo matemático.

Por fim, apesar do número de FLOPs ser importante, os resultados mostram que outros



(g) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para ResNet18.



(h) Estimativa do tempo médio de inferência de cada modelo em cada dispositivo, acompanhado do desvio padrão para VGG19BN.

parâmetros podem exercer influência sobre o tempo de inferência, como, a estrutura da rede, a disposição de camadas e parâmetros. Comparando MNASNet1.0 e MobileNetV2, modelos com número de FLOPs parecidos, é notável que o modelo menos profundo (MNASNet1.0) apresenta tempos de inferência ligeiramente menores. Adicionalmente, ResNet18, o segundo modelo menos profundo analisado, obteve as menores latências junto a MNASNet1.0. Esse resultado torna-se mais relevante ao notar que RegNetX16gf, modelo consideravelmente mais profundo e com mais parâmetros, possui tempo de inferência superior ao ResNet18, embora possua número de FLOPs inferior. Esses resultados sugerem que somente o número de FLOPs é insuficiente para estimar o tempo de latência. Logo, um bom estimador deve ser capaz de considerar todos esses fatores simultaneamente para oferecer uma estimativa mais próxima do real.

4.3. Avaliação do Tempo de Comunicação do Modelo entre Servidor e Cliente

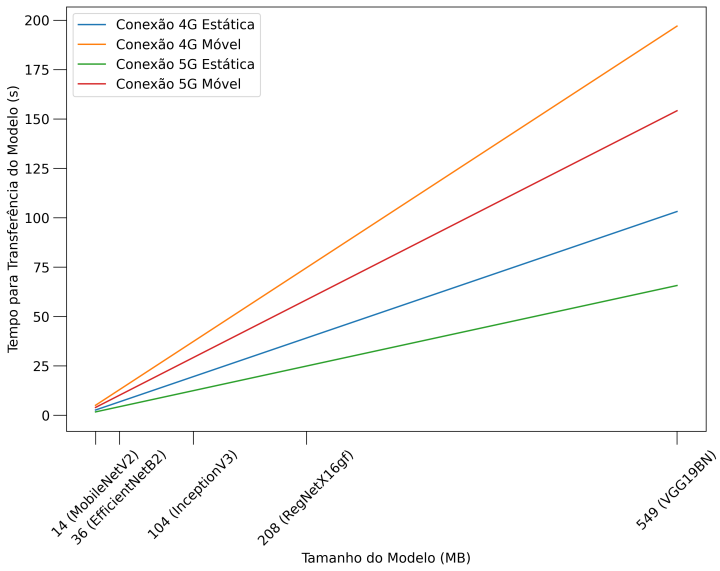


Figura 3: Validação do tempo necessário para transferir o modelo ao cliente considerando conexões sem fio 4G e 5G, com e sem mobilidade em função do tamanho do modelo.

Por fim, outra métrica importante a ser analisada é o tempo de transferência do modelo do servidor para o cliente. Apesar deste tempo influenciar majoritariamente o treinamento dos modelos a partir do aprendizado federado, a sua influência impacta também a aplicação, pois o

modelo pode estar relacionado com alguma função crítica do veículo [Salay et al. 2017]. Desta forma, é recomendável que o condutor espere o tempo necessário para atualização do modelo.

Assim, a fim de oferecer uma melhor experiência ao usuário, é importante avaliar o tempo de transferência do modelo em nuvem para o veículo. A Tabela 3 indica os valores simulados do tempo de transferência dos modelos utilizados em redes 4G e 5G, com e sem mobilidade. Os tempos de transferência foram calculados utilizando medidas de taxa de transferência média em um cenário de transferência de arquivos [Raca et al. 2020], sendo 42,6 e 66,9 Mbps para os cenários 4G e 5G sem mobilidade e 22,3 e 28,5 Mbps para os cenários com mobilidade, respectivamente.

Os resultados mostram que o tempo esperado para transferência do modelo entre a nuvem e o veículo é da ordem de alguns segundos para modelos menores que 50 MB, como a MobileNetV2 e a EfficientNetB2, em todas as condições de comunicação avaliadas. Entretanto, o tempo para modelos maiores que 200 MB é da ordem de minutos em enlaces 4G. Portanto, para modelos maiores, a informação sobre as condições de conexão também se torna um critério importante para a seleção dos modelos no sistema.

5. Conclusão e Trabalhos Futuros

Este trabalho apresenta DIGA, um sistema para definição de modelos para aplicações inteligentes que considera restrições computacionais dos dispositivos. Utilizando informações do dispositivo, dados e aplicação, é possível determinar quais modelos atendem os requisitos mínimos de tempo de inferência. Os modelos LLMs demonstraram uma grande diversidade em suas respostas retornadas, utilizando diferentes tipos de informações para estimar o tempo de inferência. Porém, apesar da grande diversidade, os LLMs subestimam menos o tempo de inferência do que o modelo matemático, com o Mistral apresentando uma redução de 25% do erro de subestimação. Entretanto, o modelo ainda subestima o tempo de inferência em 55% dos casos. Dessa forma, para aumentar a precisão das respostas é necessário realizar ajustes-finos nos modelos e garantir a qualidade dos dados utilizados no treinamento, uma vez que esses modelos tendem a memorizar os dados utilizados. Os resultados mostram que a estrutura da rede, ou seja, a disposição de camadas e parâmetros, influencia no tempo de inferência de maneira significativa. Comparando modelos com número de FLOPs parecidos, é possível verificar que os modelos menos profundos apresentam tempos de inferência menores. Esses resultados mostram que o número de FLOPs é insuficiente para estimar o tempo de inferência de um modelo. Logo, um bom estimador deve ser capaz de considerar todos esses fatores simultaneamente para oferecer uma estimativa mais próxima do real. Por outro lado, modelos maiores possuem um tempo maior de transferência entre a nuvem e os clientes. Dessa forma, a informação sobre as condições de conexão se torna um critério importante para a seleção dos modelos no sistema para proporcionar uma melhor experiência de usuário, uma vez que as aplicações parem sua execução ou podem depender que o veículo esteja estático durante a sua atualização. Por fim, o ajuste fino dos LLMs e dos estimadores deve ser um ponto de pesquisa para aumentar a eficiência do sistema em trabalhos futuros.

Referências

Asperti, A., Evangelista, D. e Marzolla, M. (2021). Dissecting FLOPs along Input Dimensions for GreenAI Cost Estimations. Em *International Conference on Machine Learning, Optimization, and Data Science*, páginas 86–100. Springer.

Boutros, A. et al. (2020). Beyond Peak Performance: Comparing the Real Performance of AI-Otimized FPGAs and GPUs. Em *IEEE International Conference on Field-Programmable Technology (ICFPT)*, páginas 10–19.

Chitty-Venkata, K. T., Emani, M., Vishwanath, V. e Somani, A. K. (2023). Neural Architecture Search Benchmarks: Insights and Survey. *IEEE Access*, 11:25217–25236.

Clancy, J. et al. (2024). Wireless Access for V2X Communications: Research, Challenges and Opportunities. *Communications Surveys & Tutorials*.

Cui, W. et al. (2021). Enable Simultaneous DNN Services Based on Deterministic Operator Overlap and Precise Latency Prediction. Em *International Conference for High Performance Computing, Networking, Storage and Analysis*, páginas 1–15.

de Souza, L. A. C. et al. (2024). AutoMHS-GPT: Automated Model and Hyperparameter Selection with Generative Pre-Trained Model. Em *IEEE International Conference on Cloud Networking*.

Desislavov, R., Martínez-Plumed, F. e Hernández-Orallo, J. (2023). Trends in ai inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustainable Computing: Informatics and Systems*, 38:100857.

Ding, X. et al. (2023). HPC-GPT: Integrating Large Language Model for High-Performance Computing. Em *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, páginas 951–960.

Idelbayev, Y. e Carreira-Perpiñán, M. Á. (2021). Beyond FLOPs in Low-Rank Compression of Neural Networks: Optimizing Device-Specific Inference Runtime. Em *IEEE International Conference on Image Processing (ICIP)*, páginas 2843–2847.

Mattson, P., et al. (2020). MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance. *IEEE Micro*, 40(2):8–16.

Memon, Z., Arham, M., Ul-Hasan, A. e Shafait, F. (2024). LLM-Informed Discrete Prompt Optimization. Em *Workshop on LLMs and Cognition (ICML)*.

Raca, D., Leahy, D., Sreenan, C. J. e Quinlan, J. J. (2020). Beyond Throughput, the Next Generation: A 5G Dataset with Channel and Context Metrics. Em *ACM Multimedia Systems Conference*, páginas 303–308.

Reddi, V. J. et al. (2020). MLPerf Inference Benchmark. Em *ACM/IEEE Annual International Symposium on Computer Architecture (ISCA)*, páginas 446–459. IEEE.

Salay, R., Queiroz, R. e Czarnecki, K. (2017). An Analysis of ISO 26262: Using Machine Learning Safely in Automotive Software. *arXiv preprint arXiv:1709.02435*.

TechPowerUp (2025). Hardware specification.

Wang, E., Chen, B., Chowdhury, M., Kannan, A. e Liang, F. (2023). FLINT: A Platform for Federated Learning Integration. *Proceedings of Machine Learning and Systems*, 5:21–34.

Weng, Q. et al. (2022). MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters. Em *USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, páginas 945–960.

Zhang, X., Wang, Y. e Shi, W. (2018). pCAMP: Performance Comparison of Machine Learning Packages on the Edges. Em *USENIX Workshop on Hot Topics in Edge Computing (HotEdge)*.

Zhao, A. et al. (2024). ExpeL: LLM Agents are Experiential Learners. Em *Conference on Artificial Intelligence (AAAI)*, páginas 19632–19642.