CATransformers: Carbon Aware Transformers Through Joint Model-Hardware Optimization

Irene Wang[1,2,*], Mostafa Elhoushi[2], H. Ekin Sumbul[3], Samuel Hsia[2], Daniel Jiang[4], Newsha Ardalani[2], Divya Mahajan[1], Carole-Jean Wu[2], Bilge Acun[2]

[1]Georgia Institute of Technology, [2]FAIR at Meta, [3]Reality Labs at Meta, [4]Meta, [*]Work done at Meta
irene.wang@gatech.edu acun@meta.com

Abstract

Machine learning solutions are rapidly adopted to enable a variety of key use cases, from conversational AI assistants to scientific discovery. This growing adoption is expected to increase the associated lifecycle carbon footprint, including both *operational carbon* from training and inference and *embodied carbon* from AI hardware manufacturing. We introduce CATransformers—the first carbon-aware co-optimization framework for Transformer-based models and hardware accelerators. By integrating both operational and embodied carbon into early-stage design space exploration, CATransformers enables sustainability-driven model architecture and hardware accelerator co-design that reveals fundamentally different trade-offs than latency- or energy-centric approaches. Evaluated across a range of Transformer models, CATransformers consistently demonstrates the potential to reduce total carbon emissions—by up to 30%—while maintaining accuracy and latency. We further highlight its extensibility through a focused case study on multi-modal models. Our results emphasize the need for holistic optimization methods that prioritize carbon efficiency without compromising model capability and execution time performance. The source code of CATransformers is available at https://github.com/facebookresearch/CATransformers.

1 Introduction

As machine learning (ML) systems become more widespread across various industries, it is crucial to take a closer examination of their carbon footprint and find strategies to mitigate it across the system stack. Sustainable ML system design requires a holistic approach that considers both operational carbon (energy used during training and inference) and embodied carbon (emissions from hardware manufacturing and lifecycle) [GEH+22], which together contribute to total emissions. This work tackles the question: *How does incorporating carbon footprint metrics into optimization workflows influence the design of ML models and hardware architectures as a co-optimization?*

A fundamental challenge in designing sustainable AI systems stem from the tight coupling between model architecture and hardware design, which together shape both operational and embodied carbon. Model execution on a particular hardware determines runtime characteristics, such as compute intensity and memory access patterns, which affect operational carbon, while hardware parameters like chip area, memory hierarchy, and fabrication technology drive embodied carbon. Carbon optimization requires joint model-hardware exploration, particularly during early accelerator design, to uncover opportunities that align model demands with hardware capabilities in a carbon-efficient way. Co-design is even more important for multi-modal workloads like vision-language models. Each modality introduces distinct bottlenecks—e.g., vision transformers usually have compute-bound workloads [DGC24], while text transformers have more memory-bound workloads [Fu24]. Co-

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

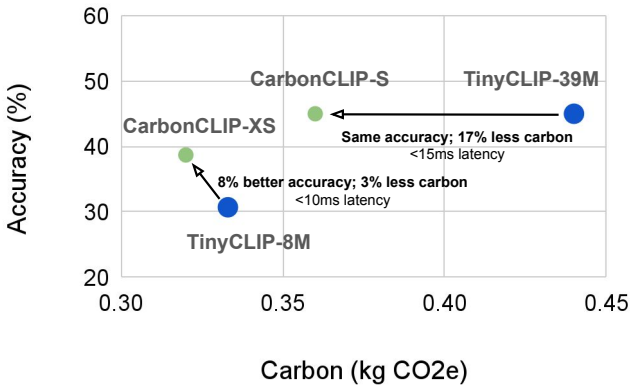


Figure 1: CarbonCLIP models achieve lower carbon footprint and higher accuracy compared to baseline CLIP models.

optimization across modalities and system layers is essential to minimize the total carbon footprint without compromising performance.

While this sustainability challenge applies across the entire AI lifecycle, the rapid adoption of AI on edge devices presents a particularly acute and fast-growing problem [QPFM+23, GKL+22]. With billions of products like smartphones and AR/VR headsets requiring specialized, custom accelerators [WLS+24, SWL+22, HAI25, Goo20, Nvi25], the embodied carbon from manufacturing constitutes a massive, upfront environmental cost. Furthermore, the cumulative operational carbon from continuous inference over a device's multi-year lifespan represents a substantial and often overlooked energy demand [App21]. Addressing the tightly coupled model-hardware system at the edge is therefore a critical first step, offering a high-impact domain to develop and validate the carbon-aware co-design principles needed for sustainable AI.

Prior hardware-aware neural architecture search (NAS) methods have focused primarily on latency or energy and are not suitable for total carbon optimization. This is because carbon is a complex metric that cannot be captured by optimizing latency or energy in isolation. These approaches either neglect embodied carbon by optimizing only the model on fixed hardware [WWL+20] or treat latency, energy, and chip area as orthogonal constraints [ZDM+22], overlooking the tightly coupled nature of operational (latency- and energy-driven) and embodied (chip area-driven) carbon. As a result, these methods are fundamentally limited in their ability to minimize total carbon emissions. Furthermore, prior work often assumes fixed hardware or fixed model backbones, limiting their applicability to a broader range of emerging hardware platforms and modern, multi-modal models.

We introduce CATransformers—*the first framework to jointly optimize model and hardware architecture with the specific goal of minimizing total carbon emissions for edge inference-only devices.* Distinct from prior latency- or energy-centric methods, our approach incorporates both operational and embodied carbon into a **unified optimization pipeline**, revealing carbon-efficient configurations previously overlooked. We observe that optimizing traditional metrics in isolation frequently results in suboptimal trade-offs. For example, accelerating inference by increasing compute density may inadvertently increase power consumption and embodied carbon due to larger chip area or more resource-intensive fabrication. By prioritizing total carbon as a first-class objective, CATransformers yields fundamentally different and more sustainable design choices.

CATransformers systematically explores the joint design space of model architectures and hardware accelerators using multi-objective optimization. It comprises three main components: (1) a *Multi-Objective Bayesian Optimizer* that balances accuracy, latency, energy, and carbon emissions; (2) an *ML Model Evaluator* that efficiently navigates model variants using importance-based pruning and fine-tuning; and (3) a *Hardware Estimator* that profiles latency, energy, and carbon footprint.

We evaluate CATransformers on language, vision, and multi-modal Transformer-based models and show that its co-optimized model–hardware configurations reduce total carbon by 30% over latency-optimized and 8% over energy-optimized baselines. To further highlight its utility, we present CarbonCLIP, a family of CATransformers-optimized CLIP models.

Figure 1 shows CarbonCLIP achieves up to 17% lower total carbon emissions compared to state-of-the-art CLIP variants on edge devices, while maintaining similar accuracy and latency. Joint

optimization enables low-carbon, high-accuracy designs that outperform approaches that optimize only hardware configurations.

Our key contributions include:

1. **Insights and Analysis:** Empirical insights showing how carbon-aware optimization shifts model–hardware trade-offs relative to traditional metrics, enabling early-stage design exploration for next-generation ML accelerators.
2. **Quantification Framework**: We develop the first open-source toolchain to estimate both operational and embodied carbon for custom accelerators during design-space exploration.
3. **Carbon-Aware Co-optimization:** CATransformers uses multi-objective Bayesian optimization to jointly explore model and hardware design spaces, balancing accuracy, latency, energy, and carbon. It leverages fine-tuning–based proxy signals to reduce evaluation cost.
4. **Sustainable Multi-Modal Models**: Using CATransformers, we co-optimize CLIP variants (CarbonCLIP) that reduce total carbon by up to 17% compared to edge-deployed CLIP baselines, without sacrificing accuracy or latency.

CATransformers is designed as a **modular, extensible framework** that supports the seamless integration of diverse model architectures, optimization strategies, and carbon estimation techniques. The goal is not to outperform other NAS frameworks on traditional metrics, but to highlight the fundamentally different design trade-offs that emerge under carbon-centric optimization objectives. Through systematic comparisons across multiple optimization modes and model families, we demonstrate the practical impact of carbon-aware design, which is the core contribution of this work.

2 Background and Related Works

Hardware Accelerator Search: Specialized hardware accelerators have been developed to efficiently run deep learning workloads, with tensor cores for matrix operations [JKL+23, MPA+16, PSM+17, CKES16] and vector cores for element-wise operations [JYP+17, GKX+24]. Prior accelerator search frameworks [APK+24, PMK+22, WTPM24, PMT25, ZHS+22, SSL23] optimize for throughput and energy, but do not jointly optimize hardware and model architectures or consider carbon footprint in the design process.

Hardware and Neural Architecture Co-optimization: Co-optimization methods [ZDM+22, CHY+21, JYS+20, LYH21] typically focus on latency or energy, ignoring carbon as a primary design objective. Approaches that do consider carbon [ECA+23, ECA+25, ZG23] often optimize either the model or hardware in isolation. Furthermore, prior work often assumes fixed hardware or fixed model backbones, limiting applicability to a broader range of emerging hardware platforms and modern, heterogeneous models—particularly multi-modal architectures. Our work addresses this gap by jointly optimizing both hardware and model architectures with total carbon footprint—including embodied and operational emissions—as a first-class objective.

The Carbon Footprint of AI Systems: Much of the research on the carbon footprint of AI focuses on operational emissions, with three main directions: (1) quantifying the emissions from training [BBL23, SGM20, LLSD19, WRG+22], (2) analyzing energy use during deployment and inference [LJS24, LGH+24], and (3) developing techniques to reduce emissions while exploring energy–performance trade-offs [GHL+25, ZLJG24, LJGT24, ALK+23].

Embodied carbon, however, remains underexplored. Tools like ACT [GEH+22], IMEC.netzero [IME25], and LLMCarbon [FKW+24] have begun to estimate embodied emissions, but a holistic framework that quantifies and minimizes both operational and embodied carbon remains lacking. CORDOBA [ECA+25] introduces a carbon-aware accelerator design tool, but it is not open-sourced, limiting reproducibility and extensibility. Unlike CE-NAS [ZLJG24], which reduces carbon emissions during neural architecture search, CATransformers is the first to optimize total carbon emissions from both model and hardware, during early-stage design.

AI Systems on the Edge: AI models are no longer confined to large-scale datacenters and are now widely deployed on edge devices like smartphones, AR/VR headsets, robotics, and vehicles [QPFM+23, SWL+22, HAI25]. These edge settings present unique challenges, including strict resource constraints, hardware heterogeneity, and communication bottlenecks that impact

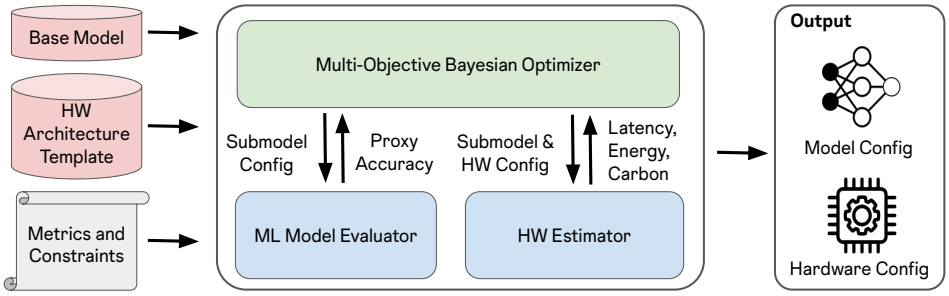


Figure 2: Overview of the CATransformers framework. The Bayesian optimizer iteratively explores model and hardware configurations using accuracy, carbon, and latency estimates from evaluation modules, outputting optimized co-designs.

performance [WNM23, HYW+23, bJRE+23]. This wide adoption, however, also raises significant environmental concerns. The sheer scale of device production means embodied carbon from chip fabrication is a major contributor to total emissions [GKL+22]. Simultaneously, the cumulative operational carbon from frequent inference over a device's multi-year lifespan presents a distinct and growing sustainability challenge. CATransformers aims to address this gap by enabling holistic, carbon-aware co-design of AI models and hardware accelerators specifically for edge systems.

CLIP Models and Edge Variants: Multi-modal models like CLIP [RKH+21] combine Transformer-based text encoders with image encoders (ResNet [HZRS16] or ViT [DBK+21]) and are trained on large datasets [SVB+21, XXT+24, SBV+22, GIF+23] to learn cross-modal associations. These models support tasks like zero-shot classification and retrieval. Several variants [WPZ+23, LBL+24, STJ+23, VPF+24] adapt CLIP for edge devices through pruning and efficient training, but focus solely on accuracy and latency. Our work is the first to optimize CLIP models for edge deployment with total carbon emissions in mind, without sacrificing accuracy or latency.

3 Framework Overview

In this section, we introduce CATransformers, a carbon-aware architecture search framework for sustainability-driven co-optimization of ML models and hardware architectures. As illustrated in Figure 2, CATransformers framework takes three inputs: (1) a base ML model, (2) a hardware architecture template, and (3) a set of optimization objectives and constraints that define the joint model–hardware search space. The framework consists of three key components: a multi-objective optimizer, an ML model evaluator, and a hardware estimator.

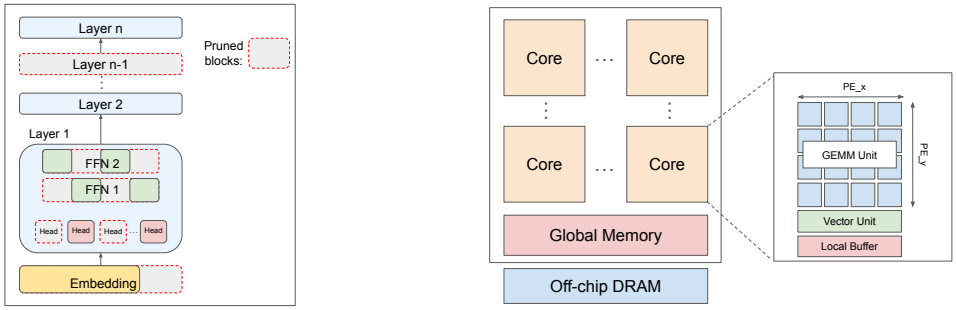
3.1 CATransformers Inputs

Base Model: The base model is a large, pre-trained Transformer that serves as the foundation for generating pruned variants. It defines the architecture's shape, functionality, and search space. As shown in Figure 3a, pruning is applied along several dimensions, including the number of layers, hidden size, feedforward network width, attention heads, and embedding dimensions.

Hardware Architecture Template: The accelerator template (Figure 3b) captures key components and tunable parameters inspired by academic and industry designs [JYP+17, APK+24, ZHS+22, WTPM24]. It features tensor cores with Processing Elements (PEs) arranged in X and Y dimensions for GEMM operations, vector units for element-wise operations, local buffers for data reuse, and a shared global SRAM and off-chip memory—consistent with existing edge inference accelerators [SWL+22, WLS+24]. Increasing PEs or cores boosts performance but raises area and energy costs; more on-chip memory lowers latency but adds energy overhead. The optimal hardware configuration depends on the model architecture, size, and performance constraints. Table 1 summarizes the hardware design space parameters.

3.2 ML Model Evaluator

The ML model evaluator estimates the accuracy of candidate model architectures during the search.



(a) Overview of the pruning dimensions

(b) Architecture template

Figure 3: Overview of model pruning dimensions and hardware template for CATransformers

Table 1: Architecture design space parameters.

Parameter Description	Notation	Potential Values
Design Space (S)		
Number of Cores	TC	1 to 4 powers of 2
PE Array X dim	PE_x	1 to 256 powers of 2
PE Array Y dim	PE_y	1 to 256 powers of 2
Global Buffer Size	GLB	1 to 8 MB powers of 2
Local Buffer Size	L2	256 KB to 4 MB powers of 2
Local Bandwidth	$\mathrm{L2}_{\mathrm{bw}}$	1 to 256 words/cycle
Vector Unit width	V_{pe}	$= \mathrm{PE}_x$
Fixed Parameters		
Global Bandwidth	GLB-BW	256 words/cycle
Off-chip DRAM Size	HBM	1 GB
Technology	Tech	22 nm
BitWidth	B	8
Maximum TOPS	$T_{\max}$	20 TOPS
Frequency	f	500 MHz

Pruning: The evaluator prunes the pre-trained base model along key dimensions, including the number of layers, feedforward network size, attention heads, and embedding dimension. For multi-modal models like CLIP, each Transformer (text and vision) is pruned independently using strategies from prior work [LBL+24, SDDN20, WWL+20]. Each layer is pruned uniformly. See Appendix A for pruning details and empirical observations.

Fine-tuning for Accuracy Proxy: Direct evaluation of untrained pruned models results in poor accuracy, while fully retraining every model is computationally infeasible. Instead, we use lightweight fine-tuning to approximate accuracy. Our ablation studies show that this method yields a high Spearman correlation (0.98) with fully trained accuracy, making it a reliable proxy for ranking candidate models (Appendix C). This approach allows accurate and efficient evaluation within the optimization loop, enabling scalable exploration of Transformer-based models without full retraining. For language models we use the MRPC task from the GLUE dataset [WSM+18] for semantic similarity detection, for vision models, the CIFAR-10 dataset [Kri09] for image classification; and for CLIP models, the MSCOCO dataset [LMB+15] for retrieval.

3.3 Hardware Estimator

The hardware estimator provides unified, end-to-end analysis of inference latency and total carbon footprint—including both embodied and operational emissions—for each model-hardware configuration. It leverages performance and energy models to estimate operator-level latency, energy, and area [WES19, ONFL23], and incorporates carbon modeling for manufacturing-related emissions and location-specific operational carbon, scaled over the system's deployment lifetime [GEH+22, Map25]. To the best of our knowledge, this represents the first open-source toolchain built for early-stage design space exploration that holistically quantifies latency, energy, and total carbon emissions of custom ML accelerators. A detailed workflow is provided in Appendix B.

3.4 Multi-Objective Optimization

CATransformers uses multi-objective Bayesian optimization to efficiently explore the joint design space of model architectures and hardware configurations building on Ax [MP24] and

BoTorch [BKJ+20] with the qNEHVI algorithm [DBB21]. Compared to reinforcement learning or evolutionary search, Bayesian optimization offers better sample efficiency and uncertainty modeling, making it ideal for our static, high-dimensional search space where each configuration takes 5-15 minutes to evaluate. While prior work [WWL+20] explored just 125 models over 30 iterations, CATransformers scales to a search space of 100 million configurations in only 100 iterations.

The optimization maximizes accuracy while minimizing latency, energy, and total carbon (embodied + operational), exploring model (Figure 3a) and hardware parameters (Table 1) under a TOPS-based compute budget based on publicly available edge accelerators. Our framework supports four compute-constrained modes: (1) Accuracy & Total Carbon (with latency constraint), (2) Accuracy & Latency, (3) Accuracy & Energy, and (4) Accuracy, Latency, & Total Carbon.

3.5 Outputs

CATransformers outputs optimized combinations of model and hardware configurations that, when deployed together, improve overall efficiency. Once the optimizer identifies carbon-efficient, pruned model architectures, these models can optionally be fine-tuned to recover any accuracy loss. Preliminary results show these pruned models require far fewer training steps than full pre-training. Specifically, CarbonCLIP models are fine-tuned for just 2 epochs on MetaCLIP [XXT+24], using only 40% of the training steps compared to prior works [RKH+21, XXT+24, WPZ+23].

4 Evaluation

4.1 Experimental Settings

Model: To demonstrate the versatility of CATransformers, we apply it to diverse transformer-based architectures: encoder-only Bert-Base [DCLT19], decoder-only Llama3-8B [GDJ+24], vision transformer ViT-B/16 [DBK+21], and multi-modal CLIP models (CLIP-ViT-B/16 and CLIP-ViT-B/32)[RKH+21]. We use pre-trained weights from HuggingFace[WDS+19] for Bert, Llama, and ViT, and adopt OpenCLIP's [IWW+21] DataComp-1B [GIF+23] models for CLIP. Building on CLIP optimization results, we generate **CarbonCLIP** models with corresponding hardware designs.

Baselines: Prior work primarily targets performance or energy efficiency, rarely considering carbon-aware hardware-model co-design. These approaches are typically evaluated on fixed hardware and model architectures, making direct comparison difficult. To ensure a fair and meaningful evaluation, we benchmark CarbonCLIP against two widely adopted baselines: (1) standard CLIP models (ViT-B/16, ViT-B/32), and (2) TinyCLIP, a state-of-the-art variant optimized for latency. For each, we apply CATransformers's hardware search with fixed model architectures to derive comparable accelerator designs. This allows us to isolate the benefits of carbon-aware co-optimization. Our results show that carbon-optimized models can match or exceed the accuracy and latency of traditional baselines—demonstrating the practical value of our approach.

Hardware Estimation and Validation: We estimate accelerator area, latency, and energy using open-source tools (Appendix B), assuming 22nm process technology and 8-bit integer operations. To validate latency estimates, we use SCALE-Sim [SZW+18, SJZ+20] for CLIP's QKV projections. Our estimates have an average latency error of 13%, consistent with prior work [PRS+19, WES19], and are validated on standard compute array sizes (16×16 to 64×64). We further validate energy and latency predictions against real GPU hardware (V100, A100, H100) by modeling GPU-like architectures. Our estimates closely match measured results, with average errors of 8% (energy) and 9% (latency), and Spearman's rank-order correlation r ranging from 0.5 to 1.0 and $p < 0.05$, confirming toolchain accuracy. Details of the validation is in Appendix M.

Carbon Emission Estimation: Operational carbon is estimated using California grid intensity over a 3-year lifespan (1 inference/sec, 6 hrs/day), reflecting typical mobile usage [App21, NSY24, Har25]. Embodied carbon is calculated assuming fabrication in Taiwan. Appendix K explores the impact of different deployment regions, energy sources, and optimization metrics.

Execution Setup: Bayesian optimization is performed on a single node (8×V100 GPUs, 80 CPUs) over 100 trials, taking 5-20 hours depending on the model. Post-pruning training of CarbonCLIP uses 224 GPUs on the MetaCLIP-2.5B dataset [XXT+24], with a batch size of 128, learning rate of

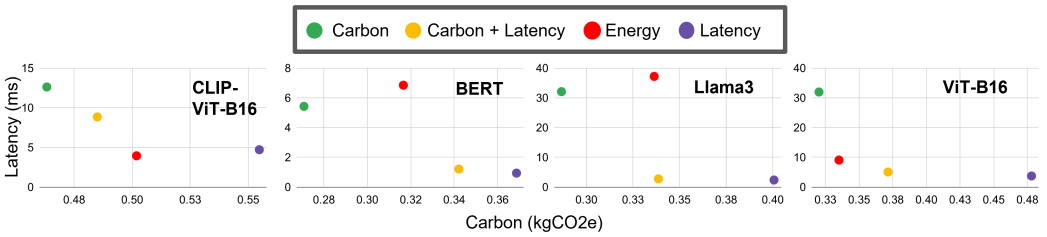


Figure 4: ISO-Accuracy plot showing the latency–carbon trade-off across optimization strategies, with accuracy matched within ±1%.

5×10^{-4}, and 2 distillation epochs. We quantify the Carbon footprint of running CATransformers in Appendix N.

4.2 Joint Model and Hardware Architecture Search Using Different Metrics

In this section, we use CATransformers to perform joint model–hardware architecture search, with carbon footprint as a central design metric alongside traditional objectives like accuracy, latency, and energy. We evaluate different optimization modes under a 20 TOPS compute budget, representative of modern edge accelerators [HAI25, Nvi25]. When latency is not an optimization target, a maximum latency constraint of 50ms is enforced [vMVJ22] to ensure realistic specifications. Each experiment is repeated three times, and we compute the Hypervolume (HV) indicator to assess consistency. HV measures the portion of the objective space dominated by the Pareto front relative to a reference point. Across runs, its standard deviation is below 0.03, with an average coefficient of variation under 3.5%, indicating statistically consistent results. Appendix F provides a detailed breakdown of the statistics.

Takeaway 1: *Carbon optimization yields the lowest footprint but at the cost of latency; energy optimization strikes a better balance.*

Figure 4 shows the latency-carbon trade-off at iso-accuracy points (based on proxy accuracy before fine-tuning). Full configurations across additional accuracy levels for each model are detailed in Appendix D. Comparing optimization modes, we find that carbon optimization significantly reduces carbon footprint by an average of 30%, but with a 7.7× increase in latency compared to latency-optimized baselines. Energy-optimized configurations reduce carbon by 24% while limiting latency overhead to 4×. Joint optimization for Carbon and Latency achieves a 18% carbon reduction with minimal latency increase, effectively balancing sustainability and performance.

Takeaway 2: *Energy optimization indirectly reduces latency, but not as effective as direct latency optimization.*

Minimizing energy tends to reduce both power and computation time, often lowering latency. In contrast, carbon-focused optimization emphasizes minimizing hardware area to reduce embodied carbon, often resulting in slower, smaller designs. Because total energy consumption is proportional to latency (i.e., energy = power × delay), latency cannot be extended arbitrarily when using smaller-area hardware. Excessively long latencies not only fail to meet practical performance targets but also begin to significantly increase operational carbon costs. Still, energy optimization does not always guarantee low latency. For example, Bert and Llama3 configurations in Figure 4 show higher latency despite energy-focused design. Direct latency optimization, seen in Latency-only and Carbon+Latency modes, consistently delivers the lowest-latency results across all models. These trade-offs highlight the need to carefully balance carbon constraints and real-world performance demands.

Takeaway 3: *Latency-optimized designs favor large, high-throughput hardware; carbon-optimized designs use compact, low-power accelerators.*

Latency-focused designs often use up to 3× larger area, enabled by more compute units and larger memory hierarchies. These systems frequently select larger models that fully utilize hardware to meet latency targets. In contrast, energy-optimized designs typically pair smaller models with larger accelerators to minimize delay and energy per inference. Carbon-optimized designs prioritize smaller hardware to reduce embodied carbon, at the cost of performance. For example, on BERT-Base (Figure 4), the carbon-optimized system uses a single-core accelerator with 512 Processing Elements (PEs), 64KB local memory, and 2MB global memory. The latency-optimized design uses two cores,

Table 2: Hardware architecture search for fixed baselines CLIP model architectures.

Model Architecture	Total Params (M)	Minimum Carbon					Minimum Latency				
		Carbon (kgCO2e)	Latency (ms)	Hardware Architecture			Carbon (kgCO2e)	Latency (ms)	Hardware Architecture		
				# Cores	Core Dimension	Memory Config {Local, Global}			# Cores	Core Dimension	Memory Config {Local, Global}
CLIP-B/16	149	0.54	18.5	1	(256,8)	64 KB, 2MB	0.69	5.4	2	(256,16)	256KB, 4MB
CLIP-L/14	427	1.43	68.7	2	(128,16)	128 KB, 2MB	1.76	66.4	4	(64,64)	256KB, 4MB
CLIP-H/14	986	1.92	71.0	1	(128,32)	128KB, 4MB	2.60	70.2	4	(256,4)	512KB, 4MB
TinyCLIP-8M/16	41	0.34	3.0	2	(256,4)	64KB, 2MB	0.56	1.3	1	(256,64)	256KB, 8MB
TinyCLIP-39M/16	83	0.44	9.4	1	(256,8)	64KB, 2MB	0.59	2.2	4	(256,16)	128KB, 4MB
TinyCLIP-40M/32	84	0.37	8.6	1	(32,32)	64KB, 2MB	0.46	1.1	4	(128,32)	64KB, 2MB
TinyCLIP-61M/32	115	0.39	9.7	1	(128,8)	64KB, 2MB	0.49	1.4	4	(128,32)	64KB, 2MB

each with 4K PEs, 128KB local memory, and 4MB global memory—cutting latency by over 4× but increasing carbon footprint by 26%.

Takeaway 4: *Model sensitivity to pruning varies by model architecture: CLIP and ViT are especially sensitive to hidden dimension pruning.*

Each model architecture responds differently to pruning. CLIP and ViT models are particularly sensitive to reductions in the hidden dimension, leading CATransformers to typically follow this pruning order for these models: FFN dimension → attention heads → layers → embedding dimension. This sensitivity arises because ViTs split images into relatively few patch tokens, requiring each token to carry rich semantic and spatial information. The hidden dimension governs the expressiveness of these token embeddings, and pruning it severely limits ViT's ability to model visual content—especially since ViTs lack the hierarchical features of CNNs and rely on global attention.

CLIP models are especially sensitive due to their multi-modal nature, where precise alignment between visual and textual embeddings is crucial. Pruning hidden or embedding dimensions can significantly degrade performance (Appendix A), as excessive pruning of the vision encoder disrupts alignment with the text encoder, harming model accuracy. Additionally, the text encoder is often pruned more aggressively than the vision encoder, which is essential for processing complex visual inputs. In contrast, language models such as Bert and Llama3 tend to be more robust to pruning in the hidden dimension. These observations underscore the challenges of pruning multi-modal models like CLIP and motivate our focus on their optimization throughout the paper.

4.3 Hardware Optimization for Fixed CLIP Models

While the previous section highlighted CATransformers' general utility, we now focus on CLIP—a widely used multi-modal model combining text and vision encoders. CLIP poses additional challenges due to its heterogeneous computation and the complex interplay between modality-specific pruning and system performance. We first evaluate the carbon footprint of state-of-the-art CLIP variants. To ensure fair comparison, we use CATransformers' hardware search to optimize inference latency and carbon footprint under a fixed model and a 20 TOPS compute budget. This yields a Pareto frontier illustrating optimal latency–carbon trade-offs. Table 2 summarizes configurations with minimum carbon and minimum latency for each model. As with joint optimization, carbon-focused hardware design results in smaller accelerators with fewer compute and memory resources. While this reduces area and emissions, it typically increases inference latency.

Takeaway 5: *Hardware must be tailored to the model-parameter count, patch size, and modality-specific traits affect optimal configurations.*

Results from the fixed-model hardware search underscore the importance of hardware–model co-design: optimal performance and carbon efficiency require aligning hardware with model architecture. ViT patch size notably influences accelerator dimensions. For instance, TinyCLIP-61M/32—despite having more parameters—outperforms TinyCLIP-39M/16 in both latency and carbon footprint when paired with hardware better suited to its larger patch size. ViT-B/32 produces a sequence length of 50, compared to 197 for ViT-B/16, enabling more efficient execution on smaller accelerators. Embedding dimensions and patch sizes shape execution patterns, making some models inherently better matched to specific hardware. Memory configuration also scales with model size: smaller models perform well with 64KB local and 2MB global buffers, while larger models like CLIP-L/14 and CLIP-H/14 require at least 128KB local buffers and larger global memory.

Table 3: Hardware and model architecture properties of each variant of the CarbonCLIP family. Hardware configurations are specified as: $\{\text{TC}, \text{PE}_x, \text{PE}_y, \text{L2}, \text{L2}_{bw}, \text{GLB}\}$. PE denotes Processing Element. Text and Vision encoders specified as: {Num Layers, FFN Dim, Hidden Dim, Num Heads}.

Name	Carbon (kgCO2e)	Latency (ms)	Hardware Configuration	Model Configuration			Avg. Accuracy over 41 datasets
				Text Encoder Configuration	Vision Encoder Configuration	Params (M)	
CLIP-B/16 - DataComp	0.54	18.5	{1, 256, 8, 64, 128, 2}	{12, 2048, 512, 8}	{12, 3072, 768, 12}	149	53.2
TinyCLIP-8M/16	0.34	3.0	{2, 256, 4, 64, 32, 2 }	{3, 1024, 256, 4}	{10, 1024, 256, 4}	41	30.7
TinyCLIP-39M/16	0.44	9.4	{1, 256, 8, 64, 128, 2}	{6, 2048, 512, 8}	{12, 2048, 512, 8}	83	45.0
CarbonCLIP-XS	0.32	7.1	{1, 256, 4, 64, 32, 2}	{6, 1024, 284, 4}	{6, 1536, 576, 6}	41	38.7
CarbonCLIP-S	0.36	12.0	{1, 256, 4, 64, 64, 2}	{6, 1024, 512, 6}	{8, 1920, 672, 6}	63	45.0
CarbonCLIP-M	0.39	19.7	{1, 256, 4, 64, 128,2 }	{8, 1536, 512, 6}	{9, 2304, 672, 6}	79	47.9
CarbonCLIP-L	0.42	13.7	{1, 256, 8, 64, 128, 2}	{6, 1280, 384, 5}	{10, 2688, 768, 7}	83	48.7
CarbonCLIP-XL	0.49	19.1	{1, 256, 8, 64, 128, 2}	{12, 2048, 512, 4}	{12, 3072, 768, 5}	123	52.0

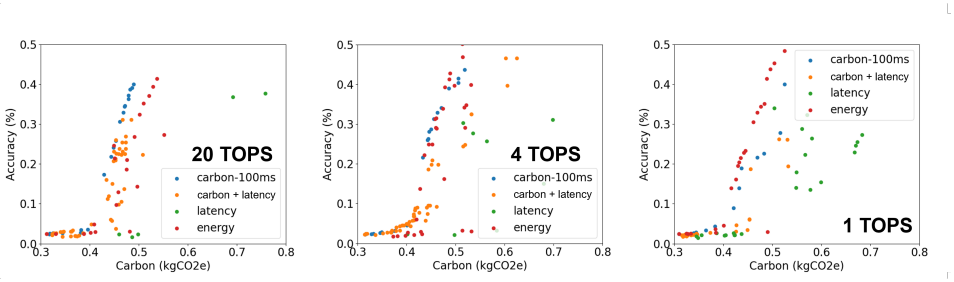


Figure 5: Pareto frontiers under varying compute constraints.

4.4 Accuracy Evaluation

We compare CarbonCLIP models to TinyCLIP and the CLIP-ViT-B/16 baseline (pretrained on DataComp-1B), using each baseline's most carbon-efficient configuration. Table 3 shows the CarbonCLIP family, selected from the Pareto frontiers in Appendix E. For each CarbonCLIP model, We perform post-pruning training, and evaluate their performance on 41 zero-shot tasks from the CLIPBenchmark [LA22]. The table summarizes model and hardware configurations for CarbonCLIP-XL to CarbonCLIP-XS (largest to smallest models), along with their carbon footprint, latency, and average accuracy. Full per-dataset results are in Appendix H. We also extend the CarbonCLIP family to the CLIP-B/32 architecture for comparison with TinyCLIP-B/32 baselines (Appendix G).

CarbonCLIP models outperform baselines across a range of carbon budgets. Notably, CarbonCLIP-XL achieves baseline-level accuracy with an 10% reduction in carbon footprint. CarbonCLIP-XS achieves an 8% increase in accuracy with a 3% reduction in carbon footprint compared to TinyCLIP-8M/16. CarbonCLIP-L, CarbonCLIP-M, and CarbonCLIP-S all achieve significant reductions in carbon footprint compared to TinyCLIP-39M/16, with CarbonCLIP-L achieving a 4% increase in accuracy and a 4.5% reduction in carbon footprint, CarbonCLIP-M achieving an 11% reduction in carbon footprint with a 3% decrease in accuracy, and CarbonCLIP-S achieving a 17% reduction in carbon footprint without any regression in accuracy.

In terms of hardware configuration, CarbonCLIP models select accelerators with cores of PE_x dimension 256 to align with the underlying operator dimensions of the CLIP ViT-B/16 architecture, with sequence length of 197 for the vision encoder. Smaller models select a total of 1024 PE units per core, whereas larger models select twice as many PEs to keep the latency of the task low. Due to the reduced size of the CarbonCLIP models, a 64KB local memory and 2MB global memory are sufficient to keep the core utilized.

4.5 Evaluating for Different Compute Constraints

We evaluate CarbonCLIP models under three peak compute constraints inspired by real-world edge devices: 20 TOPS [HAI25, Nvi25], 4 TOPS [Goo20], and 1 TOPS [WLS+24, Int25]. Figure 5 shows the resulting Pareto frontiers.

Takeaway 6: *As hardware shrinks due to tighter compute constraints, optimizing for energy (operational carbon) increasingly minimizes both total carbon and latency.*

Smaller hardware designs reduce embodied carbon, making operational energy the dominant factor in the total carbon footprint. As a result, energy-optimized designs often achieve outcomes similar to those optimized for total carbon under tight compute budgets. Appendix I explores the embodied vs. operational carbon trade-offs in more detail. Note that different grid intensities and expected hardware lifetime can also affect the ratio of embodied and operational carbon, affecting the optimization results, and the quality of each metric immensely, we show this in Appendix K. However, smaller accelerators have limited compute and memory, which increases inference delay—and thus total energy use—despite lower power consumption. This leads to higher total carbon emissions in some cases, underscoring the importance of hardware–model co-design to balance carbon efficiency and performance. We further extended the study to evaluate the impact of latency constraints on model and hardware configurations, as well as their carbon footprints in Appendix J.

5 Impact Discussion and Limitations

This work demonstrates a practical path toward sustainable AI by jointly optimizing model and hardware design for carbon reduction. Our framework is particularly effective for low-latency edge inference (e.g., chatbots, AR/VR) and highlights the potential for cross-domain collaboration across ML, hardware, and sustainability.

CATransformers currently targets Transformer models and domain-specific edge accelerators, with limited GPU support and no evaluation on CNNs. Extending to other model families (e.g., CNNs, Mamba) requires profiling the latency and energy of their unique operators and adapting pruning strategies beyond attention heads. Expanding to GPUs and data center scenarios is feasible, but the larger design space would demand significantly more search. Nonetheless, our GPU-based results (Appendices L and M) suggest scalability to training workloads and datacenter-level accelerators, where support for parallelization and distributed training could further improve carbon efficiency.

Beyond inference and training emissions, broader sustainability impacts, such as electronic waste, water usage, and rare mineral consumption, remain underexplored. While our optimizations can implicitly reduce chip area and resource use, standardized metrics and fine-grained data are currently lacking, limiting multi-objective optimization over broader environmental impacts. Moreover, custom accelerators raise concerns about hardware heterogeneity and e-waste, which may be mitigated by reusing existing hardware or co-optimizing with commercial accelerators.

Proxy-based accuracy estimation, such as the methods employed in CATransformers, enables scalable exploration but may deviate from real outcomes due to overfitting or underestimating pruning effects. Developing more robust accuracy predictors is an important direction for improving reliability.

6 Conclusion

We introduced CATransformers, a framework for co-optimizing model architectures and domain-specific accelerators to minimize carbon footprint. By jointly considering operational and embodied carbon, CATransformers supports environmentally conscious design, particularly in edge computing. We demonstrated its effectiveness across various Transformer-based models—including multi-modal models—showcasing substantial potential in carbon reductions without sacrificing performance. This work fills a key gap in sustainable AI deployment and provides a foundation for future research in carbon-aware machine learning.

7 Acknowledgements

We would like to thank Bernie Beckerman, David Eriksson, and Max Balandat for their expertise and assistance in building the optimization platform with Ax, Igor Fedorov for sharing insights on pruning models, Daniel Li and Hu Xu for providing valuable guidance on training CLIP models and working with the MetaCLIP dataset. We also would like thank Kim Hazelwood, Kristen Lauter, and Edith Beigne for supporting this work.

References

[ALK+23] Bilge Acun, Benjamin Lee, Fiodar Kazhamiaka, Aditya Sundarrajan, Kiwan Maeng, Manoj Chakkaravarthy, David Brooks, and Carole-Jean Wu. Carbon dependencies in datacenter design and management. *SIGENERGY Energy Inform. Rev.*, 3(3):21–26, October 2023.

[APK+24] Muhammad Adnan, Amar Phanishayee, Janardhan Kulkarni, Prashant J. Nair, and Divya Mahajan. Workload-aware hardware accelerator mining for distributed deep learning training, 2024.

[App21] Apple Inc. Apple environmental progress report 2021. `https://www.apple.com/environment/pdf/Apple_Environmental_Progress_Report_2021.pdf`, 2021.

[BBL23] Lucía Bouza, Aurélie Bugeau, and Loïc Lannelongue. How to estimate carbon footprint when training deep learning models? a guide and review. *Environmental Research Communications*, 5(11):115014, nov 2023.

[bJRE+23] colby banbury, Vijay Janapa Reddi, Alexander Elium, Shawn Hymel, David Tischler, Daniel Situnayake, Carl Ward, Louis Moreau, Jenny Plunkett, Matthew Kelcey, Mathijs Baaijens, Alessandro Grande, Dmitry Maslov, Arthur Beavis, Jan Jongboom, and Jessica Quaye. Edge impulse: An mlops platform for tiny machine learning. In D. Song, M. Carbin, and T. Chen, editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 254–268. Curan, 2023.

[BKJ+20] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[CHY+21] Kanghyun Choi, Deokki Hong, Hojae Yoon, Joonsang Yu, Youngsok Kim, and Jinho Lee. Dance: Differentiable accelerator/network co-exploration. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 337–342, 2021.

[CKES16] Yu-Hsin Chen, Tushar Krishna, Joel S Emer, and Vivienne Sze. Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE journal of solid-state circuits*, 52(1):127–138, 2016.

[CSL+24] Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémy Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Léval, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and MinervaBooks. mlco2/codecarbon: v2.4.1, May 2024.

[DBB21] Samuel Daulton, Maximilian Balandat, and Eytan Bakshy. Parallel bayesian optimization of multiple noisy objectives with expected hypervolume improvement. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc.

[DBK+21] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019.

[DGC24] Dayou Du, Gu Gong, and Xiaowen Chu. Model quantization and hardware acceleration for vision transformers: A comprehensive survey, 2024.

[ECA+23] Mariam Elgamal, Doug Carmean, Elnaz Ansari, Okay Zed, Ramesh Peri, Srilatha Manne, Udit Gupta, Gu-Yeon Wei, David Brooks, Gage Hills, et al. Carbon-efficient design optimization for computing systems. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, pages 1–7, 2023.

[ECA+25] Mariam Elgamal, Doug Carmean, Elnaz Ansari, Okay Zed, Ramesh Peri, Srilatha Manne, Udit Gupta, Gu-Yeon Wei, David Brooks, Gage Hills, and Carole-Jean Wu. CORDOBA: Carbon-Efficient Optimization Framework for Computing Systems . In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 1289–1303, Los Alamitos, CA, USA, March 2025. IEEE Computer Society.

[FKW+24] Ahmad Faiz, Sotaro Kaneda, Ruhan Wang, Rita Osi, Prateek Sharma, Fan Chen, and Lei Jiang. Llmcarbon: Modeling the end-to-end carbon footprint of large language models, 2024.

[Fu24] Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis, 2024.

[GDJ+24] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and et al. The llama 3 herd of models, 2024.

[GEH+22] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. Act: designing sustainable computer systems with an architectural carbon modeling tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, page 784–799, New York, NY, USA, 2022. Association for Computing Machinery.

[GHL+25] Shreyank N. Gowda, Xinyue Hao, Gen Li, Shashank Narayana Gowda, Xiaobo Jin, and Laura Sevilla-Lara. Watt for what: Rethinking deep learning's energy-performance relationship. In *Computer Vision – ECCV 2024 Workshops: Milan, Italy, September 29–October 4, 2024, Proceedings, Part XXII*, page 388–405, Berlin, Heidelberg, 2025. Springer-Verlag.

[GIF+23] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. Datacomp: In search of the next generation of multimodal datasets, 2023.

[GKL+22] Udit Gupta, Young Geun Kim, Sylvia Lee, Jordan Tse, Hsien-Hsin S. Lee, Gu-Yeon Wei, David Brooks, and Carole-Jean Wu. Chasing carbon: The elusive environmental footprint of computing. *IEEE Micro*, 42(4):37–47, July 2022.

[GKX+24] Soroush Ghodrati, Sean Kinzer, Hanyang Xu, Rohan Mahapatra, Byung Hoon Ahn, Dong Kai Wang, Lavanya Karthikeyan, Amir Yazdanbakhsh, Jongse Park, Nam Sung Kim, and Hadi Esmaeilzadeh. Tandem processor: Grappling with emerging operators in neural networks. In *ASPLOS*, 2024.

[Goo20] Google. Helping you bring local ai to applications from prototype to production, 2020.

[HAI25] HAILO. Hailo-8 ai accelerator, 2025.

[Har25] Harmony Healthcare IT. Are you addicted to your phone? american phone usage & screen time statistics. `https://www.harmonyhit.com/phone-screen-time-statistics/`, 2025.

[HYW+23] Shiqi He, Qifan Yan, Feijie Wu, Lanjun Wang, Mathias Lécuyer, and Ivan Beschastnikh. Gluefl: Reconciling client sampling and model masking for bandwidth efficient federated learning. In D. Song, M. Carbin, and T. Chen, editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 695–707. Curan, 2023.

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[IME25] IMEC. imec.netzero. "`https://netzero.imec-int.com/`", 2025.

[Int25] Intel. Intel movidius myriad x vision processing unit, 2025.

[IWW+21] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. If you use this software, please cite it as below.

[JKL+23] Norman P. Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Cliff Young, Xiang Zhou, Zongwei Zhou, and David Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings, 2023.

[JYP+17] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17, page 1–12, New York, NY, USA, 2017. Association for Computing Machinery.

[JYS+20] Weiwen Jiang, Lei Yang, Edwin Hsing-Mean Sha, Qingfeng Zhuge, Shouzhen Gu, Sakyasingha Dasgupta, Yiyu Shi, and Jingtong Hu. Hardware/software co-exploration of neural architectures. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4805–4815, 2020.

[KMH+20] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

[Kri09] Alex Krizhevsky. The cifar-10 dataset, 2009.

[LA22] LAION-AI. Clip benchmark, 2022.

[LBL+24] Haokun Lin, Haoli Bai, Zhili Liu, Lu Hou, Muyi Sun, Linqi Song, Ying Wei, and Zhenan Sun. Mope-clip: Structured pruning for efficient vision-language models with module-wise pruning error metric. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 27360–27370, 2024.

[LGH+24] Sasha Luccioni, Boris Gamazaychikov, Sara Hooker, Régis Pierrard, Emma Strubell, Yacine Jernite, and Carole-Jean Wu. Light bulbs have energy ratings - so why can't ai chatbots? *Nature*, 632:736–738, 08 2024.

[LJGT24] Baolin Li, Yankai Jiang, Vijay Gadepally, and Devesh Tiwari. Sprout: Green generative AI with carbon-efficient LLM inference. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21799–21813, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[LJS24] Sasha Luccioni, Yacine Jernite, and Emma Strubell. Power hungry processing: Watts driving the cost of ai deployment? In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '24, page 85–99, New York, NY, USA, 2024. Association for Computing Machinery.

[LLSD19] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

[LMB+15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[LYH21] Yujun Lin, Mengtian Yang, and Song Han. Naas: Neural accelerator architecture search. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pages 1051–1056, 2021.

[Map25] Electricity Maps. The leading electricity grid api, 2025.

[MP24] Inc. Meta Platforms. Adaptive experimentation platform, 2024.

[MPA+16] Divya Mahajan, Jongse Park, Emmanuel Amaro, Hardik Sharma, Amir Yazdanbakhsh, Joon Kyung Kim, and Hadi Esmaeilzadeh. Tabla: A unified template-based framework for accelerating statistical machine learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 14–26. IEEE, 2016.

[NSY24] NSYS Group. Average device lifespan: How long does a cell phone last? `https://nsysgroup.com/blog/average-device-lifespan-how-long-does-a-cell-phone-last`, 2024.

[Nvi25] Nvidia. Jetson orin nano, 2025.

[ONFL23] MohammadHossein Olyaiy, Christopher Ng, Alexandra Fedorova, and Mieszko Lis. Sunstone: A Scalable and Versatile Scheduler for Mapping Tensor Algebra on Spatial Accelerators. In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2023.

[PMK+22] Amar Phanishayee, Divya Mahajan, Janardhan Kulkarni, Miguel Castro, and Muhammad Adnan. Workload-aware hardware architecture recommendations, 2022. US Patent App. 17/965,681.

[PMT25] Amar Phanishayee, Divya Mahajan, and Jakub Michal Tarnawski. Integrated hardware architecture and distribution strategy optimization for deep learning models, 2025. US Patent App. 18/452,162.

[PRS+19] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A. Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W. Keckler, and Joel Emer. Timeloop: A Systematic Approach to DNN Accelerator Evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 304–315, 2019.

[PSM+17] Jongse Park, Hardik Sharma, Divya Mahajan, Joon Kyung Kim, Preston Olds, and Hadi Esmaeilzadeh. Scale-out acceleration for machine learning. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 367–381, 2017.

[QPFM+23] Xinchi Qiu, Titouan Parcollet, Javier Fernandez-Marques, Pedro P. B. Gusmao, Yan Gao, Daniel J. Beutel, Taner Topal, Akhil Mathur, and Nicholas D. Lane. A first look into the carbon footprint of federated learning. *J. Mach. Learn. Res.*, 24(1), January 2023.

[RKH+21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[SBV+22] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. Laion-5b: An open large-scale dataset for training next generation image-text models, 2022.

[SDDN20] Hassan Sajjad, Fahim Dalvi, Nadir Durrani, and Preslav Nakov. Poor man's BERT: smaller and faster transformer models. *CoRR*, abs/2004.03844, 2020.

[SGM20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(09):13693–13696, Apr. 2020.

[SJZ+20] Ananda Samajdar, Jan Moritz Joseph, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. A systematic methodology for characterizing scalability of dnn accelerators using scale-sim. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 58–68. IEEE, 2020.

[SSL23] Chirag Sakhuja, Zhan Shi, and Calvin Lin. Leveraging domain information for the efficient automated design of deep learning accelerators. In *2023 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 287–301, 2023.

[STJ+23] Dachuan Shi, Chaofan Tao, Ying Jin, Zhendong Yang, Chun Yuan, and Jiaqi Wang. UPop: Unified and progressive pruning for compressing vision-language transformers. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pages 31292–31311. PMLR, 2023.

[SVB+21] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *ArXiv*, abs/2111.02114, 2021.

[SWL+22] H. Ekin Sumbul, Tony F. Wu, Yuecheng Li, Syed Shakib Sarwar, William Koven, Eli Murphy-Trotzky, Xingxing Cai, Elnaz Ansari, Daniel H. Morris, Huichu Liu, Doyun Kim, Edith Beigne, Reality Labs, and Meta. System-level design and integration of a prototype ar/vr hardware featuring a custom low-power dnn accelerator chip in 7nm technology for codec avatars. In *2022 IEEE Custom Integrated Circuits Conference (CICC)*, pages 01–08, 2022.

[SZW+18] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. Scale-sim: Systolic cnn accelerator simulator. *arXiv preprint arXiv:1811.02883*, 2018.

[vMVJ22] Jakub Žádník, Markku Mäkitalo, Jarno Vanne, and Pekka Jääskeläinen. Image and video coding techniques for ultra-low latency. *ACM Comput. Surv.*, 54(11s), September 2022.

[VPF+24] Pavan Kumar Anasosalu Vasu, Hadi Pouransari, Fartash Faghri, Raviteja Vemulapalli, and Oncel Tuzel. Mobileclip: Fast image-text models through multi-modal reinforced training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15963–15974, 2024.

[WDS+19] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.

[WES19] Yannan Nellie Wu, Joel S. Emer, and Vivienne Sze. Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[WLS+24] Tony F. Wu, Huichu Liu, H. Ekin Sumbul, Lita Yang, Dipti Baheti, Jeremy Coriell, William Koven, Anu Krishnan, Mohit Mittal, Matheus Trevisan Moreira, Max Waugaman, Laurent Ye, and Edith Beigné. 11.2 a 3d integrated prototype system-on-chip for augmented reality applications using face-to-face wafer bonded 7nm logic at< 2μm pitch with up to 40% energy reduction at iso-area footprint. In *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 67, pages 210–212, 2024.

[WNM23] Irene Wang, Prashant J. Nair, and Divya Mahajan. FLuID: Mitigating stragglers in federated learning using invariant dropout. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[WPZ+23] Kan Wu, Houwen Peng, Zhenghong Zhou, Bin Xiao, Mengchen Liu, Lu Yuan, Hong Xuan, Michael Valenzuela, Xi (Stephen) Chen, Xinggang Wang, Hongyang Chao, and Han Hu. Tinyclip: Clip distillation via affinity mimicking and weight inheritance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21970–21980, October 2023.

[WRG+22] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, Michael Gschwind, Anurag Gupta, Myle Ott, Anastasia Melnikov, Salvatore Candido, David Brooks, Geeta Chauhan, Benjamin Lee, Hsien-Hsin Lee, Bugra Akyildiz, Maximilian Balandat, Joe Spisak, Ravi Jain, Mike Rabbat, and Kim Hazelwood. Sustainable ai: Environmental implications, challenges and opportunities. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 795–813, 2022.

[WSM+18] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018.

[WTPM24] Irene Wang, Jakub Tarnawski, Amar Phanishayee, and Divya Mahajan. Integrated hardware architecture and device placement search. In *International Conference on Machine Learning*, 2024.

[WWL+20] Hanrui Wang, Zhanghao Wu, Zhijian Liu, Han Cai, Ligeng Zhu, Chuang Gan, and Song Han. Hat: Hardware-aware transformers for efficient natural language processing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[XXT+24] Hu Xu, Saining Xie, Xiaoqing Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying CLIP data. In *The Twelfth International Conference on Learning Representations*, 2024.

[ZDM+22] Yanqi Zhou, Xuanyi Dong, Tianjian Meng, Mingxing Tan, Berkin Akin, Daiyi Peng, Amir Yazdanbakhsh, Da Huang, Ravi Narayanaswami, and James Laudon. Towards the co-design of neural networks and accelerators. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 141–152, 2022.

[ZG23] Yiyang Zhao and Tian Guo. Carbon-efficient neural architecture search. In *Proceedings of the 2nd Workshop on Sustainable Computer Systems*, HotCarbon '23, page 1–7. ACM, July 2023.

[ZHS+22] Dan Zhang, Safeen Huda, Ebrahim Songhori, Kartik Prabhu, Quoc Le, Anna Goldie, and Azalia Mirhoseini. A full-stack search technique for domain optimized deep learning accelerators. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, page 27–42, New York, NY, USA, 2022. Association for Computing Machinery.

[ZLJG24] Yiyang Zhao, Yunzhuo Liu, Bo Jiang, and Tian Guo. CE-NAS: An end-to-end carbon-efficient neural architecture search framework. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Appendix

A Ablation Study: Pruning and Finetuning CLIP models

In this ablation study, we studied the effect of pruning each dimension and the effect of fine-tuning on various datasets to find a good proxy for approximating the accuracy after training the pruned models on larger datasets fir CLIP-based models. All models were evaluated against MS COCO. The results are shown in Figure 6 and our key observations are as follows:

We make some key observations in terms of the importance of each dimension to the overall accuracy of the model. First, accuracy drops significantly after pruning to 50% of any dimension, even after fine-tuning. Therefore, we confine the search space to a minimum of half of each dimension. Second, the vision model has a more significant impact on accuracy compared to the text model. Finally, pruning the embedding dimension has the most significant impact on accuracy among all pruned parameters, followed by number of layers, then FFN dimensions and number of attention heads.

Fine-tuning on MS COCO: Even with just a single epoch, fine-tuning on MS COCO significantly improves the accuracy of pruned models, making it a good and fast proxy for evaluating their overall potential. **Fine-tuning on Datacomp-Tiny:** Training on a small subset of a pre-training dataset (Datacomp-Tiny, a 400k subset of Datacomp-Medium unfiltered) also improves accuracy, albeit with lower overall accuracy compared to MS COCO. **Fine-tuning on Datacomp-Medium:** Using a general pre-training dataset (Datacomp-Medium unfiltered) reduces the variance in accuracy between models of different sizes, showing that smaller models can achieve comparable accuracy when trained with enough data. This incentivizes our post-pruning training with a large and high quality MetaCLIP 2.5B Dataset.

In general, larger models will attain higher accuracy compared to models with fewer parameters when trained for the same number of steps. However, models with fewer parameters may still achieve comparable accuracy as larger models given more training steps [KMH+20]. Therefore, we fine-tune each model with the same computation FLOPS, allowing smaller models to train for more FLOPS and recover their accuracy from more extensive pruning.

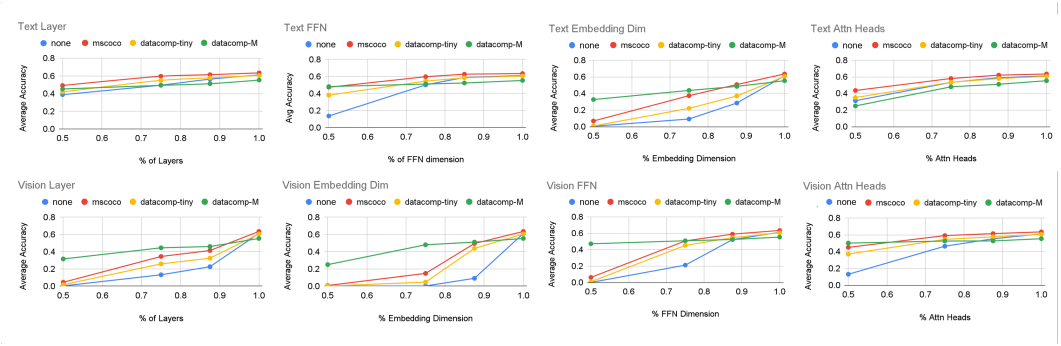


Figure 6: Pruning and fine-tuning results for each dimension of the Text and Vision encoders.

B Estimation Tool-chain Integration

In this section we present in detail the integration of the tools used in the hardware estimator as illustrated in Figure 7. We use Accelergy [WES19] to estimate the area and access energy of each component, and Sunstone [ONFL23] to estimate the per-operator latency and energy. For carbon estimations, we use ACT [GEH+22] to estimate the embodied carbon of the hardware architecture based on the area of the accelerator. Given the energy estimates from Sunstone, Electricity Maps [Map25] is used to retrieve the carbon intensity of the electricity of a given grid location and calculate the operational carbon of executing a single inference. We then scale the operational carbon to the total lifetime of the hardware architecture.

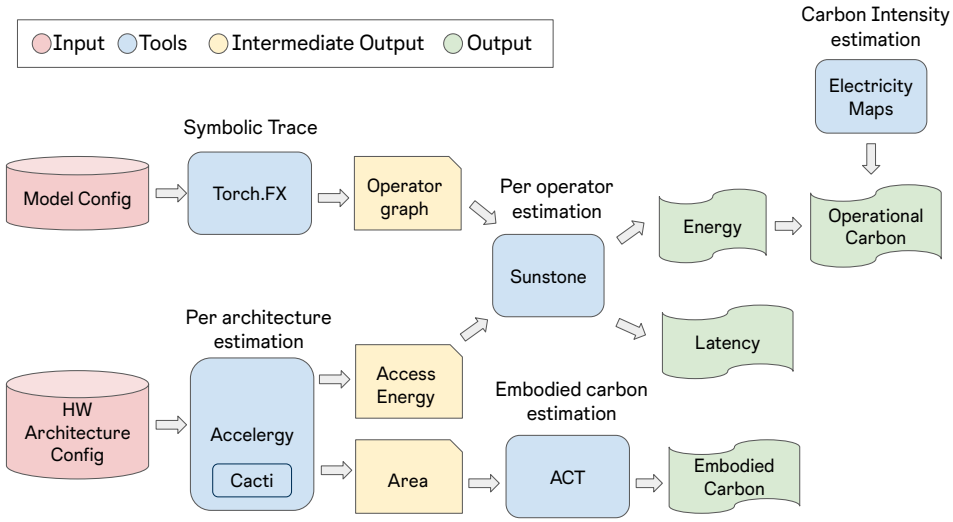


Figure 7: Integration of tools to estimate the total carbon footprint of the given hardware and model configuration.

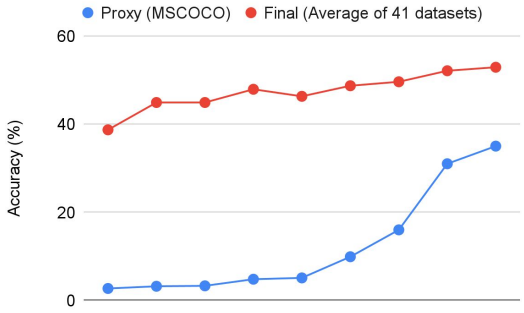


Figure 8: Correlation between the proxy accuracy, fine-tuned and evaluated on MS COCO and the final model accuracy evaluated across 41 datasets for pruned CLIP models.

C Correlation Between Proxy and Final Model Accuracy

To efficiently approximate accuracy while preserving ranking consistency, we fine-tune each pruned model to a common dataset used to evaluate each model type. As shown in Figure 8, fine-tuning maintains a high Spearman's rank correlation coefficient (0.98) with the final post-pruning training accuracy, ensuring reliable accuracy ranking. We report the mean top-1 recall accuracy on MSCOCO as the accuracy proxy for each pruned CLIP model. We incorporate this into the optimization loop to dramatically reduce the cost of evaluating candidate model architectures.

D ISO Accuracy

In this sections we present full details on different design points found by each optimization metric for a given accuracy point in Tables 4, 5, 6, and 7, for CLIP-ViT-B16, BertBase, Llama3-8b, ViT-B16, respectively. The key observations are discussed in Section 4.2. For each iso-accuracy comparison, accuracy values are reported with a tolerance of +/- 1% across optimization strategies.

E Pareto Frontier of each Optimization

In this section, we present an example of the Pareto frontier for each optimization metric using the CLIP ViT-B/16 model. Figure 9 shows the Pareto frontiers for latency-only, carbon-only, energy-only, and latency+carbon optimization objectives. Each data point in the figure represents a specific model and hardware architecture configuration, with accuracy shown on the y-axis, carbon footprint on the

Table 4: The hardware and model architecture configuration found by each optimization metric at each accuracy point for CLIP-ViT-B16. Hardware configurations are specified in the format of: $\{TC, pe_x, pe_y, L2, L2_{bw}, glb\}$. Text and Vision encoders are specified in the format of {Num Layers, FFN Dim, Hidden Dim, Num Heads}

Accuracy (+/- 1%)	Optimization Metric	Carbon (kgCO2e)	Latency (ms)	Hardware Configuration	Model Configuration		
					Text Encoder Configuration	Vision Encoder Configuration	Params (M)
31%	Carbon	0.46	12.6	{1, 256, 8, 64, 64, 2}	{9, 1536, 512, 6}	{12, 576, 768, 8}	104
	Energy	0.50	3.9	{2, 256, 16, 128, 32, 2}	{7, 1536, 384, 8}	{12, 576, 768, 8}	101
	Latency	0.55	4.7	{2, 256, 16, 128, 32, 2}	{10, 1792, 384, 7}	{12, 672, 768, 11}	111
	Carbon + Latency	0.48	8.8	{2, 256, 8, 64, 64, 2}	{9, 1280, 512, 6}	{11, 672, 768, 9}	105
19.5%	Carbon	0.44	10.9	{1,256, 8, 64, 64 2}	{9, 1536, 512, 6}	{11, 672, 768,6}	95
	Energy	0.48	3.5	{2, 256, 16, 128, 32, 2}	{6, 1536, 512, 8}	{12, 480, 768, 7}	90
	Latency	0.55	8.2	{4, 256, 4, 128, 64, 2}	{11, 1536, 384, 5}	{11, 576, 768, 12}	99
	Carbon + Latency	0.45	7.3	{2, 256, 4, 64, 128, 2}	{9, 1024, 512, 5}	{11, 576, 768, 8}	94
13%	Carbon	0.43	22.1	{1, 256, 4, 64, 64, 2}	{7, 1536, 6384 5}	{11, 576, 768, 8}	84
	Energy	0.49	7.3	{2, 256, 8, 64, 64, 2}	{8, 1792, 448, 5}	{10, 576, 768, 7}	92
	Latency	0.54	15.9	{4, 256, 2, 128, 64, 2}	{12, 1792, 320, 5}	{11, 576, 768, 12}	96
	Carbon + Latency	0.47	7.3	{1, 256, 16, 128, 64, 2}	{9, 2048, 384, 4}	{11, 3072, 768, 6}	98
2.5%	Carbon	0.32	4.6	{1, 256, 8, 64, 64, 2}	{6, 1024, 256, 4}	{6, 1536, 384, 6}	27
	Energy	0.33	1.8	{1, 256, 16, 128, 64, 2}	{6, 1024, 256, 6}	{6, 1536, 384, 6}	28
	Latency	0.46	1.3	{4, 256, 16, 128, 128, 2}	{6, 1024, 384, 8}	{9, 1536, 384, 4}	43
	Carbon + Latency	0.31	5.1	{1, 256, 4, 64, 64, 2}	{6, 1024, 256, 4}	{6, 1536, 384, 6}	27

Table 5: Iso-Accuracy points for Bert-Base model. Original model configuration: number of layers: 12, embedding dimension: 768, intermediate dimension:3072, number of Attention Heads:12

Accuracy (+/- 1%)	Optimization Metric	Carbon (kgCO2e)	Latency (ms)	Hardware Configuration	Model Configuration	Params (M)
63%	Carbon	0.26	12.7	{1, 32, 8, 64, 32, 2}	{10, 192, 48, 6}	1.7
	Energy	26	2.9	{4, 1, 62, 8, 64, 64, 2}	{6, 384, 48, 4}	1.7
	Latency	0.41	0.9	{4, 256, 8, 256, 128, 4}	{9,192,144,6}	5.5
	Carbon + Latency	0.31	0.9	{4, 4, 128, 8, 128, 64, 2}	{8, 576, 48, 4}	1.9
68%	Carbon	0.44	10.9	{1, 128, 4, 64, 64, 2}	{7, 576, 48, 12}	1.9
	Energy	0.48	3.5	{1, 128, 2, 128, 128}	{8, 1344, 48, 6}	2.5
	Latency	0.55	8.2	{2, 256, 16, 128, 128, 4}	{7, 1920, 48, 6}	2.7
	Carbon + Latency	0.45	7.3	{2, 256, 16, 256, 128, 2}	{6, 3072, 96, 8}	6.4

Table 6: Iso-Accuracy points for Llama2-7b model. Original model configuration: number of layers: 32, intermediate dimension:11008, embedding dimension: 4096, number of Attention Heads:32

Accuracy (+/- 1%)	Optimization Metric	Carbon (kgCO2e)	Latency (ms)	Hardware Configuration	Model Configuration	Params (M)
64%	Carbon	0.26	12.7	{1, 64, 8, 64, 32, 2}	{2, 1376, 256, 32}	38.2
	Energy	26	2.9	{1, 64, 8, 64, 128, 4}	{2, 3440, 256, 32}	41.2
	Latency	0.41	0.9	{4, 256, 8, 128, 128, 4}	{2, 1376, 256, 32}	38.2
	Carbon + Latency	0.31	0.9	{4, 128, 16, 64, 128, 2}	{2, 1376, 256, 32}	38.2
66%	Carbon	0.44	10.9	{1, 64, 8, 64, 64, 2}	{3, 1376, 256, 32}	41.8
	Energy	0.48	3.5	{4, 128, 16, 128, 64, 2}	{3, 1376, 256, 32}	41.8
	Latency	0.55	8.2	{2, 256, 16, 512, 64, 8}	{2, 4128, 256, 32}	42.4
	Carbon + Latency	0.45	7.3	{1, 256, 64, 512, 128, 4}	{3, 1376, 256, 32}	41.8

Table 7: Iso-Accuracy points for ViT-B-16 model. Original model configuration: number of layers: 12, intermediate dimension:3072, embedding dimension: 768, number of Attention Heads:12

Accuracy (+/- 1%)	Optimization Metric	Carbon (kgCO2e)	Latency (ms)	Hardware Configuration	Model Configuration	Params (M)
39%	Carbon	0.26	12.7	{1, 32, 32, 64, 64, 2}	{12, 1920, 2496, 6}	45.8
	Energy	26	2.9	{1, 128, 32, 64, 128, 2}	{11, 1344, 2496, 12}	31.7
	Latency	0.41	0.9	{2, 256, 32, 256, 128, 4}	{7, 2112, 2688, 6}	31.7
	Carbon + Latency	0.31	0.9	{2, 256, 16, 128, 128, 2}	{8, 1728, 2496, 6}	29.0
40%	Carbon	0.44	10.9	{2, 128, 8, 128, 64, 2}	{11, 2112, 2688, 8}	49.2
	Energy	0.48	3.5	{1, 256, 16, 64, 64, 2}	{12, 1728, 2496, 12}	39.5
	Latency	0.55	8.2	{2, 256, 16, 64, 64, 8}	{11, 1728, 2496, 8}	43.2
	Carbon + Latency	0.45	7.3	{4, 64, 32, 64, 128, 2}	{12, 1728, 2688, 8}	48.0

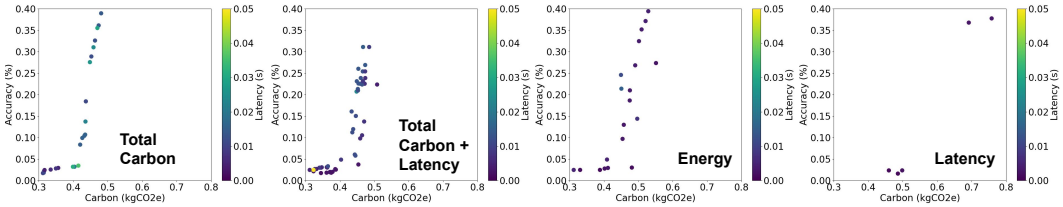


Figure 9: Pareto frontiers for different optimization modes under a 20 TOPS compute constraint.

Table 8: Summary of Hypervolume indicator difference across runs of the same configuration

Model	Metric	Hypervolume Mean	Standard Deviation (σ)	Coefficient of Variation (%)
ViT-B/16	carbon	0.27	0.0006	0.2
	latency	0.04	0.0001	0.3
	energy	0.07	0.007	9.7
	carbon + latency	0.26	0.009	3.5
BertBase	carbon	0.50	0.001	0.2
	latency	0.70	0.004	0.6
	energy	0.65	0.003	0.6
	carbon + latency	0.48	0.013	2.8
Llama3	carbon	0.45	0.033	7.4
	latency	0.66	0.004	0.6
	energy	0.35	0.003	1.1
	carbon + latency	0.43	0.003	0.7
CLIP ViT-B/16	carbon	0.19	0.021	11.1
	latency	0.18	0.016	9.2
	energy	0.39	0.015	3.8
	carbon + latency	0.13	0.004	3.2

x-axis, and latency encoded via a color map. To ensure consistency, each experiment is repeated three times, and accuracy is estimated using the MS COCO dataset. When latency is not an explicit optimization objective, a maximum latency constraint of 50 ms is enforced [vMVJ22] to ensure realistic deployment scenarios.

Consistent with the discussion in Section 4.2, examining the Pareto frontiers reveals that optimizing solely for total carbon yields configurations with the lowest carbon footprint across all accuracy levels. However, this comes at the cost of higher latency compared to the other optimization objectives.

F Consistency of Experimental Results

Each optimization is repeated three times for fair evaluation. To assess consistency, we compute the Hypervolume (HV) indicator across runs with the same configuration. HV measures the portion of the objective space dominated by the Pareto front, relative to a reference point. Across all model-hardware optimizations, the standard deviation of HV remains below 0.03, with an average coefficient of variation (std/mean) under 3.5%, indicating stable and reproducible results. Table 8 summarizes these statistics for each model architecture and optimization objective.

G CLIP ViT B/32 Accuracy Evaluations

In this section, we present results that demonstrate the applicability of CATransformers to the CLIP ViT-B/32 architecture and compare its performance against the TinyCLIP baselines. (Table 9) Our findings show that CATransformers's optimization can be effectively generalized to other model architectures, yielding up to 5% and 8% reductions in carbon footprint while achieving higher accuracy and comparable latency compared to the TinyCLIP baselines, respectively.

Table 9: The hardware and model architecture properties of each variant of the CarbonCLIP-B/32 family. Hardware configurations are specified as: $\{TC, PE_x, PE_y, L2, L2_{bw}, GLB\}$. Text and Vision encoders are specified as: {Num Layers, FFN Dim, Hidden Dim, Num Heads}

Name	Carbon (kgCO2e)	Latency (ms)	Hardware Configuration	Model Configuration		Params (M)	Avg. Accuracy over 41 datasets
				Text Encoder Configuration	Vision Encoder Configuration		
CLIP-B/32 - DataComp	0.42	15.1	{1, 32, 32 ,64 , 64,2}	{12, 2048, 512, 8}	{12, 3072, 768, 12}	144	51.1
TinyCLIP-39M/32	0.37	3.0	{1, 32, 32, 64, 32, 2 }	{6, 2048, 512, 8}	{12, 2048, 512, 8}	84	45.2
TinyCLIP-61M/32	0.39	9.4	{1, 128, 8, 64, 64, 2}	{9, 2048, 512, 8}	{12, 2560, 640, 10}	115	47.2
CarbonCLIP-32-S	0.35	7.3	{1, 64, 16, 64, 64, 2}	{6, 1536, 384, 5}	{10, 3072, 672, 12}	89	46.4
CarbonCLIP-32-M	0.36	15.3	{1, 64, 8, 64, 64, 2 }	{8, 1280, 448, 7}	{11, 2688, 768, 8}	99	47.6
CarbonCLIP-32-L	0.38	9.1	{1, 128, 8, 64, 128, 2}	{7,2048,512,6}	{11,3072,768,10}	113	49.1

Table 10: Results across all 41 evaluation benchmarks from CLIP Benchmark

Dataset	CarbonCLIP B/16 (ours)					TinyCLIP B/16		DataComp B/16	CarbonCLIP B/32 (ours)			TinyCLIP B/32		DataComp B/32
	XS	S	M	L	XL	39M/16	8M/16	ViT-B-16	S	M	L	40M/32	61M/32	ViT-B-32
cars	0.74	0.82	0.84	0.85	0.87	0.52	0.08	0.89	0.82	0.83	0.84	0.77	0.8	0.87
country211	0.11	0.14	0.16	0.17	0.2	0.18	0.12	0.22	0.15	0.16	0.17	0.13	0.15	0.18
fer2013	0.17	0.21	0.3	0.36	0.34	0.52	0.33	0.39	0.25	0.23	0.38	0.47	0.49	0.33
fgvc_aircraft	0.12	0.2	0.23	0.23	0.29	0.15	0.07	0.3	0.2	0.21	0.23	0.14	0.18	0.25
flickr30k	0.55	0.66	0.7	0.7	0.76	0.76	0.52	0.76	0.65	0.67	0.68	0.68	0.71	0.7
flickr8k	0.52	0.62	0.65	0.64	0.7	0.71	0.5	0.7	0.62	0.64	0.66	0.63	0.66	0.65
gtsrb	0.25	0.4	0.46	0.5	0.55	0.32	0.11	0.55	0.47	0.49	0.52	0.38	0.3	0.52
imagenet-a	0.11	0.19	0.26	0.32	0.39	0.33	0.15	0.48	0.22	0.22	0.24	0.17	0.21	0.3
imagenet-o	0.55	0.55	0.51	0.44	0.44	0.49	0.4	0.43	0.49	0.49	0.5	0.52	0.51	0.5
imagenet-r	0.55	0.66	0.73	0.77	0.82	0.7	0.3	0.84	0.72	0.74	0.75	0.7	0.73	0.78
imagenet1k	0.51	0.6	0.63	0.65	0.7	0.63	0.41	0.74	0.62	0.64	0.65	0.6	0.62	0.69
imagenet_sketch	0.35	0.45	0.5	0.52	0.57	0.4	0.1	0.6	0.49	0.51	0.52	0.47	0.5	0.57
imagenetv2	0.43	0.53	0.55	0.58	0.62	0.56	0.35	0.66	0.54	0.55	0.58	0.51	0.54	0.61
mnist	0.28	0.58	0.7	0.66	0.7	0.37	0.1	0.76	0.69	0.75	0.71	0.51	0.6	0.81
mscoco_captions	0.33	0.41	0.44	0.44	0.49	0.47	0.29	0.49	0.41	0.43	0.44	0.41	0.45	0.45
objectnet	0.37	0.46	0.51	0.54	0.6	0.43	0.19	0.64	0.47	0.5	0.51	0.41	0.44	0.55
renderedsst2	0.51	0.5	0.56	0.54	0.56	0.5	0.5	0.52	0.5	0.51	0.52	0.52	0.54	0.48
stl10	0.92	0.94	0.96	0.97	0.98	0.97	0.92	0.98	0.96	0.96	0.96	0.95	0.96	0.97
sun397	0.59	0.66	0.68	0.69	0.71	0.69	0.56	0.71	0.67	0.67	0.68	0.65	0.67	0.68
voc2007	0.7	0.73	0.75	0.77	0.79	0.77	0.62	0.82	0.77	0.77	0.77	0.77	0.78	0.81
voc2007_multilabel	0.75	0.79	0.81	0.81	0.83	0.82	0.74	0.81	0.79	0.8	0.8	0.76	0.79	0.79
vtab/caltech101	0.8	0.83	0.84	0.84	0.85	0.82	0.72	0.85	0.83	0.83	0.85	0.82	0.82	0.84
vtab/cifar10	0.73	0.83	0.89	0.9	0.93	0.91	0.73	0.96	0.91	0.92	0.92	0.91	0.92	0.96
vtab/cifar100	0.47	0.55	0.65	0.67	0.75	0.68	0.42	0.82	0.69	0.7	0.73	0.69	0.72	0.8
vtab/clevr_closest_object_distance	0.16	0.15	0.17	0.16	0.19	0.2	0.16	0.24	0.16	0.16	0.16	0.17	0.21	0.21
vtab/clevr_count_all	0.18	0.19	0.21	0.34	0.25	0.2	0.13	0.33	0.16	0.2	0.33	0.19	0.24	0.13
vtab/diabetic_retinopathy	0.04	0.09	0.16	0.04	0.2	0.03	0.02	0.11	0.05	0.07	0.05	0.1	0.24	0.42
vtab/dmlab	0.14	0.15	0.15	0.15	0.14	0.13	0.18	0.19	0.21	0.2	0.14	0.21	0.15	0.16
vtab/dsprites_label_orientation	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.04	0.03	0.02	0.02	0.03
vtab/dsprites_label_x_position	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.02	0.03	0.03	0.03	0.03
vtab/dsprites_label_y_position	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.03	0.03	0.03	0.03	0.03	0.03	0.03
vtab/dtd	0.41	0.51	0.54	0.51	0.58	0.47	0.29	0.58	0.48	0.51	0.53	0.51	0.52	0.57
vtab/eurosat	0.39	0.48	0.54	0.59	0.58	0.53	0.23	0.59	0.5	0.5	0.56	0.48	0.45	0.57
vtab/flowers	0.55	0.66	0.64	0.66	0.71	0.7	0.58	0.76	0.63	0.66	0.68	0.62	0.64	0.73
vtab/kitti_closest_vehicle_distance	0.37	0.32	0.3	0.26	0.35	0.11	0.15	0.29	0.28	0.19	0.32	0.15	0.17	0.16
vtab/pcam	0.59	0.57	0.59	0.63	0.6	0.61	0.53	0.6	0.61	0.58	0.54	0.52	0.57	0.53
vtab/pets	0.76	0.85	0.87	0.89	0.91	0.81	0.46	0.93	0.87	0.88	0.89	0.85	0.88	0.9
vtab/resisc45	0.45	0.57	0.62	0.64	0.66	0.55	0.21	0.65	0.58	0.64	0.63	0.54	0.58	0.63
vtab/smallnorb_label_azimuth	0.05	0.06	0.06	0.05	0.05	0.06	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.05
vtab/smallnorb_label_elevation	0.11	0.11	0.12	0.11	0.11	0.1	0.12	0.11	0.11	0.1	0.11	0.11	0.11	0.1
vtab/svhn	0.15	0.3	0.29	0.29	0.38	0.16	0.14	0.61	0.33	0.45	0.42	0.35	0.39	0.61

H CLIP Benchmark Full Result

In this section we provide a detailed breakdown on the accuracy of each dataset for CarbonCLIP and the evaluated baselines in Table 10.

I Carbon Footprint Breakdown

We provide a breakdown of the carbon footprint for each variant of the CarbonCLIP family model. As shown in Figure 10, as the model increases in size, the proportion of operational carbon in the overall carbon footprint of the model increases from 20% to over 40%. The CarbonCLIP-XL model has 3 $\times$ the number of parameters and almost 3 $\times$ the latency of CarbonCLIP-XS, but the selected hardware architecture only has double the number of compute PEs. Therefore, the operational carbon increases proportionally more than the increase in embodied carbon. This highlights need of co-optimizing the model and hardware architecture to maintain an intricate balance between operational and embodied carbon, keeping the overall carbon footprint of the system low. As such, the expected lifetime and source of power are also important factors that need to be taken into account during the optimization process.

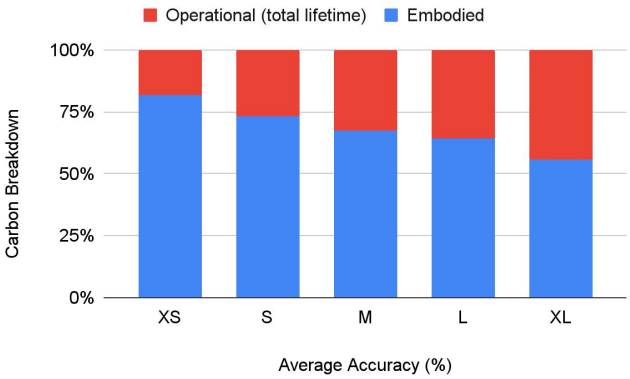


Figure 10: Operational and embodied carbon footprint breakdown for the CLIP-ViT-B/16 architecture.

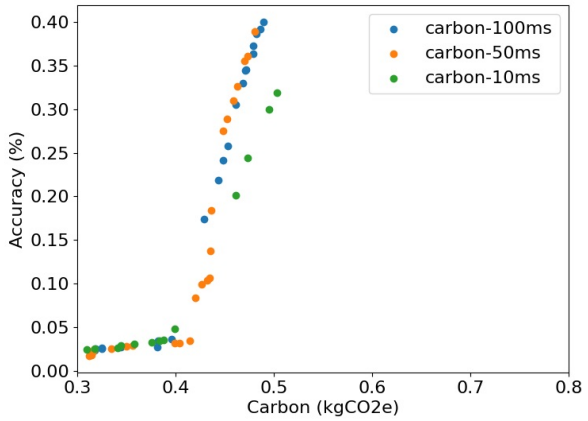


Figure 11: Latency and carbon footprint trade-offs for the CLIP-ViT-B/16 architecture across the Pareto frontier.

J Case Study: Varying Latency Constraints

We evaluate the impact of latency constraints on model and hardware configurations, as well as their carbon footprints. We categorize use cases into three categories based on latency requirements: critical real-time (<10ms), interactive (<50ms), and non-critical (<100ms) [vMVJ22]. Figure 11 shows the Pareto frontiers for each category.

Lower latency constraints typically result in less carbon-efficient designs. For example, a 10ms latency constraint achieves only an 17% carbon reduction compared to latency-optimized models, although with comparable latency values. However, increasing the constraint to 100ms does not consistently improve efficiency, as many optimal designs already meet the 50ms threshold.

K Case Study: Varying Operational Regions

The region of the operation and manufacturing of the model affects the search results, and the quality of the metric immensely. Figure 12 shows that varying the operational carbon region, in high, mid, low carbon-intensity regions, yields dramatically different results in terms of the model searched and resulting configurations. Additionally, while our case studies (Section 4.5) show that energy optimization is effective when operational carbon dominates (e.g., smaller architectures, high-carbon-intensity regions), in cleaner energy regions, total carbon optimization still yields lower emissions. Optimizing against total carbon instead of energy can reduce emissions by 8% in low-carbon regions (Canada) vs. 2% in high-carbon regions (Taiwan). The pareto frontier of each region is shown in Figure 13a and Figure 13b.

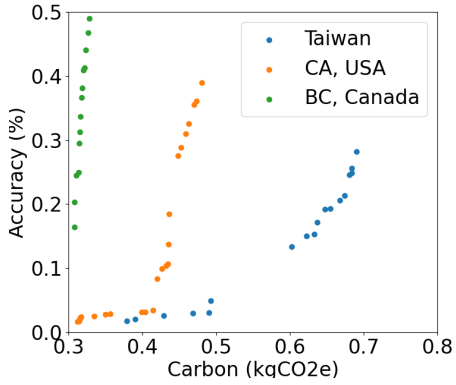
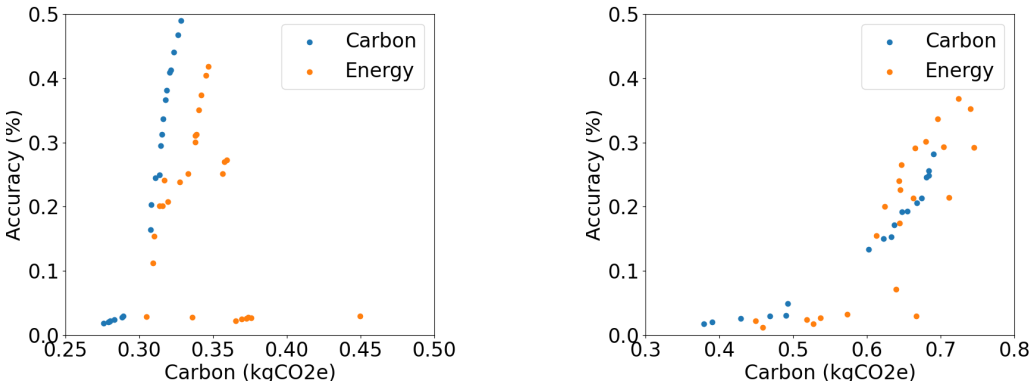


Figure 12: Pareto Frontier of Carbon-only Optimization searches across High (Taiwan), Mid (California, USA), and Low (British Columbia, Canada) carbon intensity regions.



(a) British Columbia (Low carbon intensity)

(b) Taiwan (High carbon intensity)

Figure 13: Pareto Frontier of the Carbon vs Energy Optimization result in regions with different carbon intensity.

L Case Study: Model Architecture Search with GPUs

Once a hardware architecture is selected, CATransformers supports model architecture search on fixed hardware to refine configurations and re-evaluate carbon costs before manufacturing.

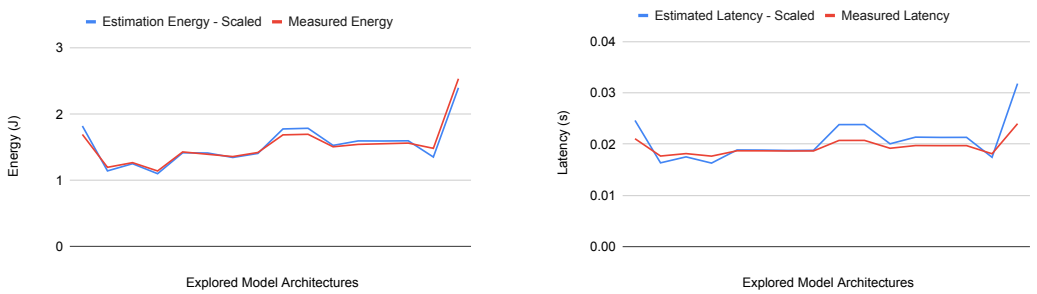
Table 11 shows an example of model architecture search on a fixed V100-like hardware architecture, optimizing total carbon and latency.

M Energy and Latency Validation with GPUs

To maintain high confidence in our estimation toolchains, we validated our energy and latency estimates using existing GPU hardware (V100, A100, and H100). Although GPUs differ from

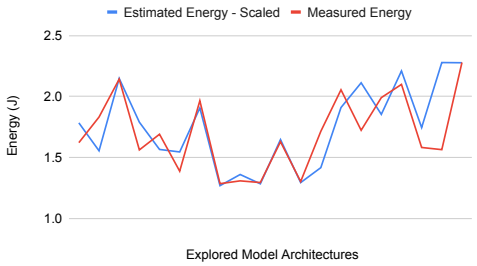
Table 11: Model architecture search with fixed V100-like architecture. Text and Vision encoders are specified in the format of {Num Layers, FFN Dim, Hidden Dim, Num Heads}

Proxy Accuracy (%)	Carbon (kgCO2e)	Latency (ms)	Text Model Config	Vision Model Config
2.8	11.13	37	6,1024,512,6	6,1536,672,6
9.8	11.41	58	6,2048,512,8	9,3072,768,6
14.4	11.45	64	6,2048,512,6	10,2688,768,6
25.3	11.57	76	6,1024,512,7	12,2688,768,6
28.7	11.58	76	6,1792,512,7	12,2688,768,6

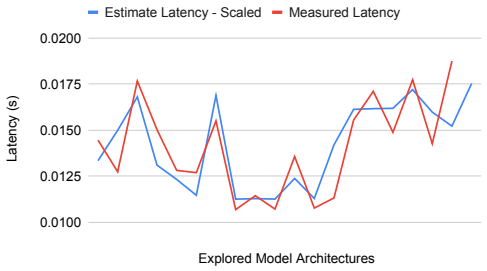


(a) V100 - Energy

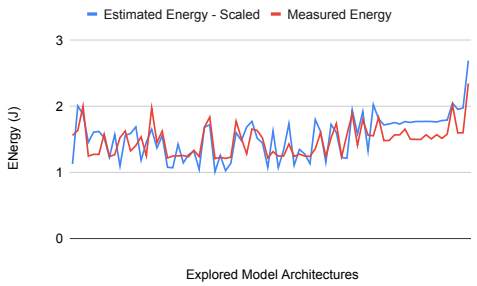
(b) V100 - Latency



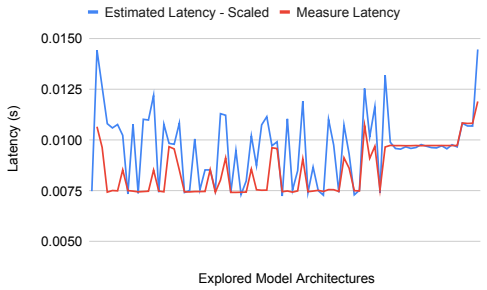
(c) A100 - Energy



(d) A100 - Latency



(e) H100 - Energy



(f) H100 - Latency

Figure 14: GPU estimated vs measure performance.

Table 12: Latency and Energy estimation error against Real GPU hardware

Hardware	Energy Estimation Error	Spearman's r (Energy)	Latency Estimation Error	Spearman's r (Latency)
V100	3.5%	0.97	7.4%	1.0
A100	8.4%	0.74	8.5%	0.85
H100	12.3%	0.7	11.1%	0.5

domain-specific accelerators, we modeled a GPU-like architecture with comparable tensor and vector units, conducted a model search on the fixed architecture, and profiled actual hardware performance for each searched data point.

The results demonstrate a strong correlation between our estimates and measured performance, with Spearman's rank-order correlation ranging from 0.5 to 1.0. When energy and latency values are scaled to a common measurement range using constants, the estimates yield average errors of 8% for energy and 9% for latency across GPU architectures. This validation confirms that our estimation tools provide accurate and reliable results, even with simulated hardware. Table 12 summarizes the results, while Figure 14 presents the estimated and measured performance for each evaluation GPU architecture.

N Carbon Footprint of the Framework

We quantify the carbon footprint of running CATransformers using CodeCarbon [CSL+24]. On average, the optimization process takes 5 hours for $\text{BERT}_{\text{base}}$ and up to 20 hours for CLIP models. For CLIP, the most resource-intensive case, 100 optimization trials emit approximately 57 kgCO_2e, while final model training emits 454 kgCO_2e per model. This means the optimization process costs roughly 1/13th the carbon budget of training the final model. Despite the one-time cost of optimization, CATransformer achieves overall efficiency gains through reduced training steps post-pruning, along with inference gains that scale with the number of devices.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: All claims made in the abstract and introduction are substantiated within the paper. An overview of the framework and its design details is provided in Section 3, while key insights, analysis, and empirical results related to carbon optimization are presented in the evaluation sections, particularly in Section 4.2. Training results for the CarbonCLIP model and comparisons to prior work are discussed in Sections 4.3 and 4.4. Additional implementation details and ablation studies are included in the Appendix.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We included a Discussion and Limitations (See Section § 5).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Paper does not include theoretical proofs.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper provides a detailed overview of all framework components in Section 3, including search parameters and integration methodology. Additional information on the estimation toolchain is provided in Appendix B to ensure full reproducibility. The experimental setup, covering evaluation baselines, dataset and model sources, as well as constants and assumptions used, is outlined in Section 4.1. Furthermore, the source code is included as supplementary material for reference.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The source code is included as part of the supplementary material and will be open-sourced upon acceptance.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental setup details, including all evaluated models, datasets used, and hyperparameters, are provided in Section 4.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Spearman's rank correlation coefficients are reported to establish the statistical significance of fine-tuning proxies (Section 3.2, Appendix C) and GPU validation

results (Appendix M). For the Bayesian Optimization results in Section 4.2, each experiment is repeated three times to ensure consistency. We demonstrate the reproducibility of the optimization outcomes by evaluating the Hypervolume (HV) indicator across runs of the same configuration, and report the standard deviation and coefficient of variation for each model and optimization objective in Appendix F. Estimation error rates for latency and energy are discussed in Section 4.1, with additional details provided in Appendix M. Accuracy deviations relevant to ISO-accuracy comparisons are clearly reported in Figure 4 in Section 4.2.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Section 4.1 provides details for the compute platform, memory, and time required for experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Authors have reviewed the NeurIPS Code of Ethic, and all submitted material preserves anonymity.

   Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss both the limitation and impact of the work in Section 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper does not pose such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites all relevant prior work, including frameworks, codebases, datasets, and pretrained models used in both the design and evaluation. Detailed references are provided in Section 3, Section 4.1, and the Appendix.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: Source code is included as part of the supplementary material (anonymized), with detailed instructions on how to run and reproduce the experiments.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Core method development in this research does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.