# LAYER ACTIVATION SCHEDULE: NOVEL METHOD FOR EFFICIENT STRUCTURING AND PRUNING OF NEURAL NETWORKS USING MONOTONIC GATING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Deep neural networks typically exhibit highly unstructured and entangled internal representations, which obscures interpretability, necessitates external regularization, and limits the hardware efficiency of post-training pruning due to irregular sparsity patterns. We introduce the Layer Activation Schedule, an end-to-end differentiable gating framework that imposes a strict structural hierarchy on neural network representations during training. By modulating pre-activations with a monotonically decaying survival probability mask, our method forces robust core features to accumulate in high-probability (leading) dimensions, while relegating highly specific details and noise to low-probability (trailing) dimensions. This self-organizing topology provides intrinsic regularization, yields highly interpretable, importance-ordered feature maps, and transforms network sparsification into a trivial structural truncation of trailing dimensions. Because representations are structurally sorted, pruning yields dense, smaller matrices without requiring unstructured masks or subsequent fine-tuning. Furthermore, our framework naturally produces a continuous family of nested sub-networks within a single trained model, enabling dynamic adjustments to computational capacity based on real-time deployment constraints without retraining.

We validate our approach on ResNet and Transformer architectures, demonstrating massive reductions in parameters and Multiply-Accumulate operations (MACs) while maintaining competitive accuracy. For GPT-2 medium trained under constrained training budgets, our approach maintains good-level generation quality after removing 92.8% of FLOPs, extending the breaking point by 2.7× over the strongest baseline. On ResNet-50, it preserves 64.62% Top-1 accuracy at a 60.47% MAC reduction. Our method significantly outperforms traditional pruning techniques under strict computational constraints, offering a efficient paradigm for scalable AI deployment.

## 1 INTRODUCTION

The energy cost of advanced neural networks, particularly Large Language Models (LLMs) and deep vision architectures, has surged dramatically, consuming substantial global energy resources and presenting severe deployment challenges. In real-world applications, hardware often faces highly variable inference demands, experiencing severe bottlenecks during peak traffic and idling during off-peak periods. Furthermore, deploying AI across a spectrum of devices, from resource-constrained edge systems to high-end cloud servers, typically requires training and maintaining multiple distinct models of varying sizes, multiplying the

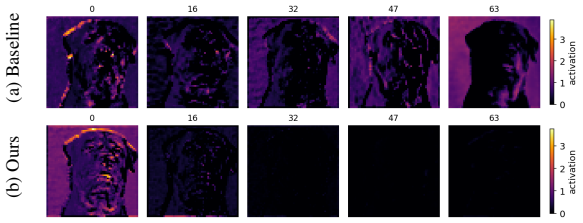



Figure 1: Post-activation feature maps (channels 0, 16, 32, 47, 63) of the first block in ResNet-50. While the Baseline (top) scatters features randomly across all channels, our model (bottom) enforces a hierarchical representation, concentrating core features in the first few channels.



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

computational and financial overhead. This inflexibility is rooted in a fundamental limitation of conventional training: networks produce highly unstructured and entangled internal representations. Unlike biological sensory neurons, which encode natural stimuli with a sparse, efficient code that concentrates structure in a small subset of informative units Barlow (1961); Olshausen & Field (1996), standard artificial neural networks distribute learned features uniformly and randomly across their hidden dimensions.

To alleviate inference costs, post-training pruning has long been employed to eliminate redundant parameters LeCun et al. (1989); Malach et al. (2020). However, standard unstructured pruning produces irregular, scattered sparsity patterns that map poorly to modern hardware accelerators, which are heavily optimized for dense matrix multiplications Wen et al. (2016); Li et al. (2017); Lin et al. (2020). While structured pruning methods attempt to remove entire structural blocks to regain hardware efficiency, they often require complex, computationally expensive iterative frameworks or extensive fine-tuning to recover lost performance Luo et al. (2019), fundamentally because the underlying representations were not organized hierarchically during training.

To address these fundamental limitations, we introduce the **Layer Activation Schedule**, a novel, end-to-end differentiable gating framework that inherently organizes the internal components of neural networks during training Rippel et al. (2014). Rather than treating all computational units (e.g., channels or attention heads) equally, our approach imposes a strict structural hierarchy. By modulating pre-activations with a deterministic, monotonically decaying survival probability mask, we force the network to accumulate the most robust, core features in the high-probability (leading) dimensions. Conversely, highly specific details and high-frequency noise are systematically relegated to the low-probability (trailing) dimensions.

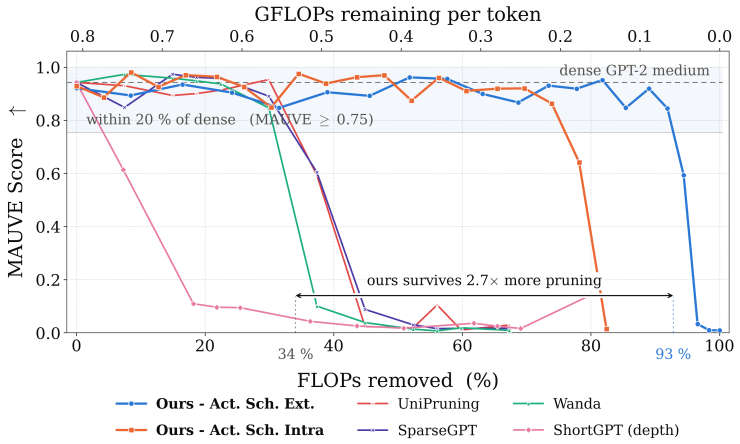



Figure 2: Generation quality against pruned compute for GPT-2 medium. Our activation-scheduled models stay within 20% of the dense model's MAUVE score until 76% and 93% of the network's FLOPs are removed, while the strongest baseline collapses at 34%.

This inherently self-organizing topology yields several profound benefits:

- **Ordered and Hierarchical Representations:** Unlike standard architectures where features are randomly entangled, our scheduled activation enforces a dimensional importance hierarchy Horváth et al. (2021). This yields highly interpretable representation maps that are strictly ordered by importance, contrasting sharply with the random feature distribution of baseline models (see Figure 1).
- **Intrinsic Regularization:** The continuous probability decay naturally regularizes the network inspired by the Information Bottleneck principle Tishby & Zaslavsky (2015), mitigating overfitting without the need for external regularization schemes.
- **Hardware-Efficient Structural Pruning:** Because representations are naturally sorted by importance, network pruning simplifies to a straightforward truncation of the least significant dimensions. This directly yields smaller, fully dense matrices, eliminating the need for complex sparse masks or costly post-training fine-tuning. On GPT-2 medium trained on the OpenWebText dataset under a constrained training budget, this approach

allows the removal of up to 92.8% of the network's FLOPs while preserving dense-level generation quality. This delays performance collapse by 2.7× compared to the strongest baseline (Figure 2).

- **Elastic Deployment (Train Once, Deploy Anywhere):** Our framework naturally produces a continuous family of nested sub-networks within a single trained parameter set. A single model can dynamically adjust its width, and consequently its computational cost (e.g., Multiply-Accumulate operations, or MACs), based on real-time deployment constraints without any retraining. On ResNet-50, our models preserve 64.62% Top-1 accuracy at a 60.47% MAC reduction, whereas standard unstructured pruning collapses to 0.90%.

We demonstrate the efficacy of our method on state-of-the-art architectures, offering a highly efficient and scalable paradigm for dynamic AI deployment.

## 2 Related Work

**Stochastic Regularization and Ordered Representations.** Standard techniques like dropout prevent overfitting via unstructured stochasticity Srivastava et al. (2014), later generalized through variational Kingma et al. (2015) and concrete dropout Gal et al. (2017), or coupled directly with pruning Salehinejad & Valaee (2022). To organize this stochasticity, nested dropout Rippel et al. (2014); Cui et al. (2023) and Matryoshka Representation Learning Kusupati et al. (2022) enforce dimensional importance hierarchies. This facilitates the extraction of nested sub-models without retraining, as demonstrated in federated learning Horváth et al. (2021). In contrast, our method imposes a deterministic structural hierarchy via a continuous, monotonically decaying probability mask, inherently priming the network for trivial structured pruning without complex iterative procedures.

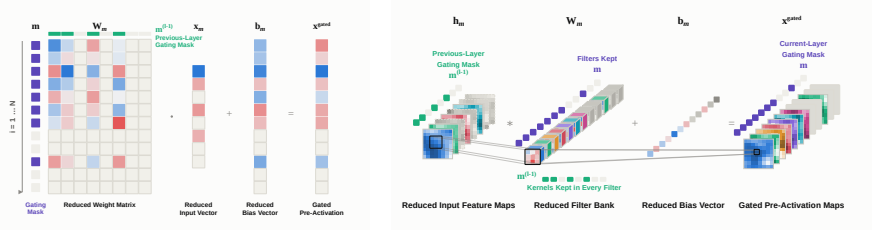
**Structural Pruning and Optimization.** Traditional pruning removes redundant weights post-training LeCun et al. (1989); Han et al. (2015), often at the filter level Li et al. (2017); Luo et al. (2017) to maintain hardware-efficient dense structures. Parallel strategies induce sparsity during training via structured regularizers Lin et al. (2020) or sensitivity analysis Tartaglione et al. (2022). Recent advancements frame pruning as a complex optimization problem, utilizing structural dependency modeling Fang et al. (2023), Information Bottleneck theory Guo et al. (2023), parameter-gradient correlations Jiang et al. (2024), and iterative frameworks Denkin et al. (2026); Asiaee (2026). Other approaches employ combinatorial optimization Yasuda et al. (2024); Mehmood et al. (2026); Shen et al. (2022), policy learning Sengupta et al. (2025), or knowledge distillation Fontana et al. (2024); Liang et al. (2025); Niu et al. (2023); Westphal et al. (2024); Kousar et al. (2024). Unlike these computationally intensive post-hoc methods, our approach inherently sorts features during the forward pass, reducing sparsification to a deterministic, zero-cost truncation of trailing units.

**Adaptive Computation.** The Lottery Ticket Hypothesis Frankle & Carbin (2019) highlighted the existence of naturally sparse sub-networks, motivating targeted dropout Gomez et al. (2019) and dynamic routing Shazeer et al. (2017); Bolukbasi et al. (2017); Ponti & Martins (2024). While slimmable networks Yu & Huang (2019) flexibly adjust capacity, our static activation schedule similarly isolates redundancy and supports scalable execution, but achieves this without requiring specialized architectural blocks or dynamic pathways.

## 3 Method

### 3.1 Stochastic Gating and Dimensional Hierarchy

Conventional neural networks lack constraints on their internal representation topology, distributing features randomly across dimensions. This unstructured entanglement hinders interpretability and complicates post-training structural pruning. To overcome this, we introduce the Layer Activation Schedule, an end-to-end differentiable gating mechanism that imposes a strict dimensional hierarchy during training. By assigning a monotonically decaying survival probability to each computational




(a) Masked Linear Layer (b) Masked Convolutional Layer

Figure 4: Dynamic parameter slicing during training. The previous layer's mask $\boldsymbol{m}^{(\ell-1)}$ (green) filters incoming inputs, while the current mask $\boldsymbol{m}$ (violet) determines the active computational units. Dropped units are physically sliced out of the computation.

unit (e.g., neuron or channel), the network is forced to accumulate core features in high-probability leading dimensions, leaving trailing dimensions to capture fine-grained details or noise.

**Training Phase (Stochastic Gating):** We formalize this hierarchy via a deterministic activation schedule $\boldsymbol{p} \in [0,1]^N$, where $p_i$ is the survival probability of the $i$-th unit. During the forward pass, we apply a stochastic mask $\boldsymbol{m} \in \{0,1\}^N$ drawn from an independent Bernoulli distribution parameterized by $\boldsymbol{p}$. Utilizing the uniform reparameterization trick with $\boldsymbol{u} \sim \mathcal{U}(0,1)^N$, the mask is defined as $m_i = \mathbb{I}[u_i \leq p_i]$ (Figure 3).

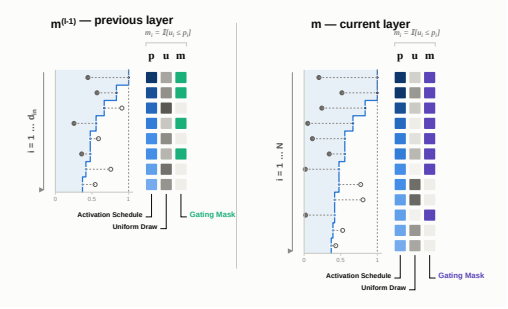



Figure 3: Mask generation, $m_i = \mathbb{I}[u_i \leq p_i]$, for two consecutive layers. The survival probability $\boldsymbol{p}$ monotonically decays over the unit index $i$. A unit survives if a uniform draw $u_i$ falls below $p_i$.

Instead of computing full dense activations and masking dropped units with zeros, we dynamically slice the layer's parameters to execute the forward pass efficiently on a structurally reduced subspace. For a parameterized layer with weights $\boldsymbol{W}$, we filter the output dimension using the current mask $\boldsymbol{m}$ and the input dimension using the preceding layer's mask $\boldsymbol{m}^{(\ell-1)}$. This explicit slicing applies natively across architectures: it reduces the weight matrix $\boldsymbol{W}$ in linear layers (Figure 4a), truncates input and output channels in `Conv2d` layers (Figure 4b), and filters affine parameters and running statistics in normalization layers (`BatchNorm2d`, `LayerNorm`).

While this independent per-element sampling establishes the expected survival probabilities, it produces non-contiguous masks. To maximize training throughput on modern GPUs, which are heavily optimized for contiguous memory accesses and dense block operations, we introduce an alternative contiguous slice sampling method. In this approach, rather than drawing an independent binary mask, we sample an integer $K$ representing the number of contiguous leading units to retain. The sampling distribution of $K$ is calibrated such that the marginal survival frequency of each unit $i$ identically matches its probability $p_i$. This ensures that parameters are sliced as a single dense block, significantly accelerating training speed while strictly preserving the required dimensional hierarchy.

**Inference Phase** During evaluation, stochasticity is removed to ensure deterministic outputs. The mask $\boldsymbol{m}$ is replaced by its expectation, $\mathbb{E}[\boldsymbol{m}] = \boldsymbol{p}$. For layers followed by a non-linear activation function, we scale the base output by the deterministic schedule: $\boldsymbol{y} = f_{\text{base}}(\boldsymbol{x}) \odot \boldsymbol{p}$. For Transformer architectures, we don't apply the scale, as the representational power of token embeddings and residual pathways are already diminished in the power.

### 3.2 Activation Schedule Generation

The efficacy of our method depends on the construction of the probability schedule $\boldsymbol{p}$. While the framework supports arbitrary scheduling functions, we focus on two continuous decay profiles that impose a structured hierarchy on the computational units.

A schedule is defined by partitioning a target dimension into multiple segments, each specifying an activation probability profile over a fractional width $\rho_i$.

**1. Piecewise Linear Ramp-Decay Schedule:** Our default configuration uses a three-segment continuous profile to transition from fully active to heavily regularized units. It consists of a leading plateau ($\rho_1$), a central linear decay ($\rho_2$), and a trailing floor ($\rho_3$), such that $\sum \rho_i = 1$. A standard setting employs a plateau of $\rho_1 = 0.1$ at a probability of 1.0, a linear decay over $\rho_2 = 0.7$ from 1.0 to 0.5, and a floor over $\rho_3 = 0.2$ at 0.5. These base probabilities are scaled by a hyperparameter $1/\alpha$ to shift the curve, establishing a steady regularization slope across the layer.

**2. Continuous Harmonic Decay Schedule:** To enforce a smooth, theoretically grounded decay, we propose a harmonic sequence divided into $K$ continuous linear segments. Within the $k$-th segment ($k \in \{1, \dots, K\}$), the probability decays linearly from $\frac{1}{k}$ to $\frac{1}{k+1}$. To ensure trailing units constitute the majority of regularized parameters, the fractional width $\rho_k$ is proportional to $k$:

$$\rho_k = \frac{k}{\sum_{i=1}^{K} i} \tag{1}$$

This yields a rapidly decaying profile, with base values similarly scaled by $1/\alpha$.

**Interpolation and Probability Clipping:** To align the raw schedule with the discrete layer dimension $N$, it is resized via continuous linear interpolation to yield $\tilde{\boldsymbol{p}}$. To ensure a valid probability distribution and prevent absolute unit death unless intended, the values are clipped:

$$p_i = \max\Big(\min(\tilde{p}_i, 1),\ p_{\min}\Big) \quad \forall i \in \{1, \dots, N\} \tag{2}$$

where $\alpha \in (0, 1]$ is a global scaling hyperparameter controlling the decay rate, and $p_{\min} \in [0, 1)$ sets the minimum survival probability. By default, $p_{\min} = 0$, allowing the network to completely drop the least important connections. The final vector $\boldsymbol{p}$ is stored as a fixed, non-trainable parameter.

### 3.3 Architectural Variants: Intra-Block vs. Extended Stream Gating

The proposed activation schedule can be integrated at different architectural scopes, dictated by its interaction with residual connections. We formalize two general gating strategies, **Intra-Block Gating** and **Extended Stream Gating**, both of which are applicable to both Transformer (e.g., GPT) and Convolutional (e.g., ResNet-50) architectures. Figure 5 illustrates the Extended Stream variant for GPT; the corresponding ResNet-50 configuration and all Intra-Block variants are provided in Appendix A.

**1. Intra-Block Gating:** This formulation restricts the schedule to the internal, hidden dimensions of a computational block, preserving the main residual stream at full capacity. For Transformers, stochastic gating targets the MLP hidden layer and attention head dimensions; for ResNet Bottlenecks, it is applied to the internal reduced-width convolutions. The final projection within each block restores the representation to its full dimensionality prior to the residual addition. Consequently, each block draws its masks independently (e.g., $\boldsymbol{m}^{(\text{att})}$ and $\boldsymbol{m}^{(\text{mlp})}$ for GPT; $\boldsymbol{m}^{(w)}$ for ResNet). Because these masks never propagate to the residual stream, they require no alignment across depth or scatter operations, enabling native components like layer normalization to function unaltered.

**2. Extended Stream Gating:** In contrast, this strategy extends the schedule to the residual pathways. In ResNet, the output of each block's final expansion convolution is masked, and the narrowed tensor is sparsely scattered into a full-width zero-buffer before the residual addition, ensuring only high-probability channels update the main stream. In Transformers, to maximize computational efficiency, we sample a single, globally shared mask per forward pass and apply it directly to the entire residual stream (Figure 5). Consequently, token embeddings, layer normalizations, attention projections, and the language modeling head all operate on a physically compressed subspace. The global sharing of this mask eliminates the need for scatter operations, forcing the entire network to execute strictly within the narrowed dimensionality.

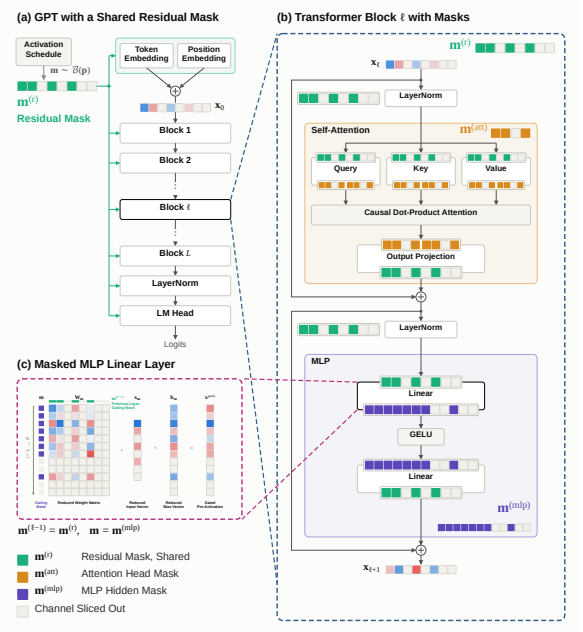



Figure 5: Extended Stream Gating in GPT. **(a)** A global residual mask $\boldsymbol{m}^{(r)}$ (green) is sampled once per forward pass and shared across all layers. **(b)** Within a transformer block, $\boldsymbol{m}^{(r)}$ defines the active width of LayerNorms and the main stream boundaries of attention and MLP projections. Concurrently, $\boldsymbol{m}^{(\text{att})}$ (amber) masks the dimension of all $n_h$ attention heads, and $\boldsymbol{m}^{(\text{mlp})}$ (violet) masks the $4d$ MLP hidden units. **(c)** The MLP expansion layer applies both masks (cf. Figure 4a): rows are sliced by $\boldsymbol{m}^{(\text{mlp})}$ and columns by $\boldsymbol{m}^{(r)}$. Dropped units are physically truncated from the computation rather than zero-masked.

### 3.4 Structured Pruning of the Scheduled Network

A central implication of the proposed activation schedule is that the network self-organizes during training, rendering post-hoc structural compression computationally trivial. Because the monotonically decaying survival probability $\boldsymbol{p}$ constrains robust features to accumulate in the leading dimensions, network pruning simplifies to a deterministic truncation of the low-probability (trailing) units.

Rather than relying on iterative fine-tuning, magnitude thresholding, or unstructured masking, we achieve structured sparsity via direct tensor slicing. To prune a layer to $N'$ active units, we physically truncate its weight matrix (e.g., $\boldsymbol{W}' = \boldsymbol{W}_{1:N',:}$) and correspondingly reduce the input dimension of the subsequent layer. This direct structural truncation extends natively to convolutional filters, normalization parameters, and token embeddings. By physically removing these trailing dimensions across all scheduled components, we immediately yield a smaller, fully dense architecture that strictly reduces memory footprint and execution cost. A detailed formalization of this operator across all layer types is provided in the supplementary material (Appendix B).

**Depth-wise Pruning Strategies.** To analyze the distribution of feature redundancy across network depth, we define two layer-wise pruning allocation strategies across the $L$ scheduled layers. In the **Uniform** strategy, a constant drop ratio is applied uniformly across all scheduled layers, such that $r_\ell = r$ for all $\ell \in \{0, \ldots, L-1\}$. Alternatively, the **Graded** strategy allocates an aggressive pruning ratio to earlier layers and monotonically decays it deeper in the network. The effective drop ratio for the $\ell$-th layer is defined as $r_\ell = \min\left(1, r\frac{L-\ell}{L}\right)$. This strategy probes the hypothesis that lower-level representational features exhibit higher compressibility than deep, task-specific features.

## 4 Experiments

We evaluate our Layer Activation Schedule on language modeling using GPT-2 medium on OpenWebText (Radford et al., 2019; Gokaslan et al., 2019), and on image classification using

ResNet-50 on ImageNet-1K (He et al., 2016; Deng et al., 2009). Detailed hardware setups, hyperparameters, data augmentations, and unpruned baseline performance metrics for both architectures are provided in Appendix C.

### 4.1 Effective Pruning at Extreme Sparsity

**GPT-2 Generation Quality and Extreme Sparsity.** We evaluate the generation quality and downstream performance of pruned GPT-2 models using metrics covering generation quality (MAUVE (Pillutla et al., 2021)), language modeling (WikiText-2 (Merity et al., 2017), LAMBADA (Paperno et al., 2016)), and zero-shot downstream tasks (Gao et al., 2023). As illustrated in Figure 2, standard pruning causes GPT-2 generation quality to fall off a cliff at moderate sparsity. Every evaluated baseline exhibits this catastrophic failure. In contrast, our Extended Stream (Act. Sch. Ext.) pruning extends this breaking point significantly, maintaining dense-level generation quality (within 20% of the dense model's MAUVE score) until 92.8% of the network's FLOPs are removed. The Intra-Block variant also survives up to 76.0% FLOPs removed, whereas the strongest baseline, UniPruning (Ding et al., 2025; Mishra et al., 2021), collapses at a mere 34.0% reduction. Consequently, our activation-scheduled pruning allows for a 2.7× larger share of the network to be removed before quality drops, requiring 9.2× less compute than the best baseline at its breaking point.

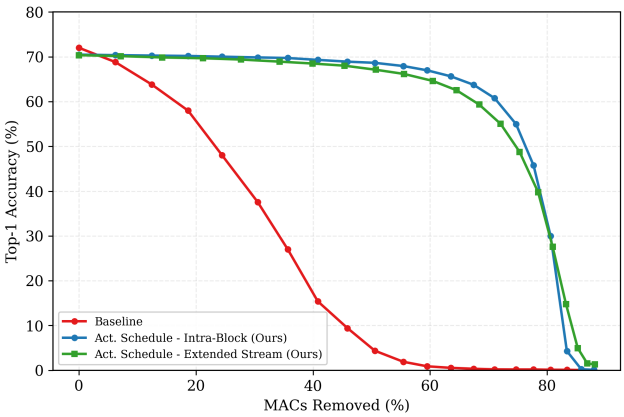



Figure 6: Top-1 validation accuracy versus percentage of MACs removed under uniform pruning. Both scheduled variants are plotted against the single unscheduled Baseline; the per-variant panels are given in the supplementary material (Appendix G).

Crucially, this stark contrast highlights a fundamental limitation of standard training: our GPT-2 checkpoints are trained for only 200,000 iterations on the OpenWebText dataset. This constrained training budget does not provide sufficient iterations or capacity for standard models to naturally structure their internal representations. Because of this unstructured entanglement, traditional pruning methods fail catastrophically when attempting to remove parameters. By explicitly forcing the network to consolidate essential features into a compact subset from the beginning, our method overcomes this limitation, proving highly effective and explaining why it succeeds where all other pruning techniques collapse.

Table 1 reports every method at three matched budgets. At 20% FLOPs removed, the field is competitive, but from 40% onwards, only the two scheduled models remain near the dense reference. At an extreme 67% FLOPs removed (0.267 GFLOPs/token remaining, the deepest budget every baseline reaches before their curves end, our scheduled models continue to generate high-quality representations, while baselines degenerate to random noise. The Extended Stream model maintains a WikiText-2 perplexity of 43.43 (compared to SparseGPT (Frantar & Alistarh, 2023) diverging to $8.9 \times 10^4$) and a zero-shot task average of 0.462 (compared to chance levels near 0.378).

**ResNet-50 Uniform Pruning.** We evaluate the structural pruning capabilities of ResNet-50 using a uniform pruning strategy, comparing the scheduled networks against the standard baseline under increasing sparsity levels. As depicted in Figure 6, the baseline model's performance deteriorates catastrophically even at moderate pruning rates. At roughly 55% MAC reduction, the Intra-Block model maintains a Top-1 accuracy of 67.94%, whereas the Baseline collapses to 1.89%. Similarly, the Extended Stream architecture preserves 64.62% accuracy at 60.47% MAC reduction, vastly outperforming the Baseline (0.90%). Our method guarantees strict hardware efficiency since sparsification is achieved through contiguous structural truncation rather than unstructured masking. For accuracy budget analysis and graded pruning profiles, see Appendix G and Appendix H.

Table 1: GPT-2 medium under pruning at three matched compute budgets. Each block reports one budget: 20%, 40% and 60% of the dense FLOPs removed (0.646, 0.485 and 0.323 GFLOPs/token against a dense 0.808). Values are interpolated between the two measured configurations that bracket the budget, in log space for the perplexities and linearly for the accuracies. **WT2** is the WikiText-2 perplexity (lower is better); **MAUVE** is the generation-quality score and **0-shot** the mean accuracy over the nine zero-shot tasks (higher is better). Best value per column in bold; the dense row is the unpruned reference and is not eligible.

| **Method** | **20% removed** | | | **40% removed** | | | **60% removed** | | |
|---|---|---|---|---|---|---|---|---|---|
| | **WT2** | **MAUVE** | **0-shot** | **WT2** | **MAUVE** | **0-shot** | **WT2** | **MAUVE** | **0-shot** |
| *Ours (structured)* | | | | | | | | | |
| Act. Sch. - Intra-Block | **27.0** | **0.966** | 0.495 | **30.4** | **0.944** | **0.480** | 41.6 | 0.918 | **0.471** |
| Act. Sch. - Extended Stream | 28.3 | 0.921 | 0.475 | 31.3 | 0.904 | 0.467 | **38.6** | **0.932** | 0.460 |
| *Structured pruning methods* | | | | | | | | | |
| ShortGPT (depth) | 115.3 | 0.103 | 0.455 | $1.5\times10^{3}$ | 0.034 | 0.406 | $2.3\times10^{4}$ | 0.033 | 0.365 |
| Magnitude (structured) | $1.6\times10^{8}$ | 0.012 | 0.384 | $3.9\times10^{9}$ | 0.015 | 0.382 | $4.2\times10^{6}$ | 0.008 | 0.382 |
| Taylor (structured) | $2.8\times10^{4}$ | 0.017 | 0.360 | $1.6\times10^{5}$ | 0.016 | 0.356 | $2.8\times10^{5}$ | 0.014 | 0.361 |
| *Semi-structured pruning method* | | | | | | | | | |
| UniPruning (2:4) | 29.1 | 0.905 | **0.514** | 111.9 | 0.394 | 0.458 | $1.7\times10^{5}$ | 0.011 | 0.366 |
| *Unstructured pruning methods* | | | | | | | | | |
| SparseGPT | 27.4 | 0.960 | 0.504 | 274.2 | 0.424 | 0.456 | $6.7\times10^{4}$ | 0.017 | 0.355 |
| RIA | 28.2 | 0.875 | 0.508 | 231.9 | 0.159 | 0.450 | $1.2\times10^{5}$ | 0.011 | 0.381 |
| Wanda | 29.1 | 0.944 | 0.513 | 478.9 | 0.079 | 0.430 | $7.0\times10^{4}$ | 0.019 | 0.376 |
| OWL | 28.7 | 0.935 | 0.511 | 287.2 | 0.171 | 0.441 | $7.5\times10^{4}$ | 0.009 | 0.372 |
| UniPruning | 28.6 | 0.830 | 0.511 | 108.6 | 0.494 | 0.460 | $3.1\times10^{5}$ | 0.008 | 0.365 |
| Magnitude | 31.6 | 0.795 | 0.504 | $1.3\times10^{4}$ | 0.087 | 0.375 | $1.8\times10^{5}$ | 0.014 | 0.366 |
| *Dense GPT-2 medium* | *25.5* | *0.942* | *0.510* | *25.5* | *0.942* | *0.510* | *25.5* | *0.942* | *0.510* |

### 4.2 Well-Structured Neural Networks: Ordered Representations

To elucidate the underlying mechanisms that enable scheduled networks to withstand aggressive truncation, we conduct an internal analysis of the post-residual ReLU activations in ResNet-50, recording each channel's mean activation $\mathbb{E}[a]$ and power $\mathbb{E}[a^2]$.

The analysis reveals a fundamental divergence in representational topology. In the Baseline architecture, essential channels are distributed uniformly and stochastically across the channel indices (Figure 7a). In stark contrast, the Extended Stream model exhibits a highly ordered self-organization: channel activation peaks monotonically at the leading indices and decays systematically (Figure 7b). Quantitatively, averaging across all 16 blocks, the initial 10% of the Baseline's channel indices account for only 9.9% of the total power, mirroring a mathematically flat profile. For our well-ordered Extended Stream model, the same initial 10% of indices capture 37.9% of the total power.

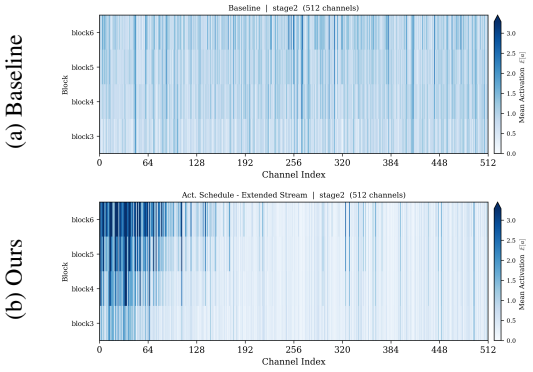



Figure 7: Per-channel mean activation $\mathbb{E}[a]$ over the four 512-channel blocks of Stage 2, for (a) the Baseline and (b) our Extended Stream model, on a shared colour scale. The Baseline spreads its activation evenly over the channel index. The scheduled model concentrates it in the leading channels, so the trailing channels can be truncated as a contiguous block.

Because the network is structurally ordered by importance, the trailing channels carry minimal representational power and can be efficiently eliminated as a contiguous block. This fundamentally transforms pruning into a computationally trivial direct structural truncation. Furthermore, this paradigm naturally yields a continuous family of nested sub-networks within a single trained parameter set, allowing dynamic adjustment of model capacity based on deployment constraints without retraining. Additional per-stage visualizations and spatial feature maps are detailed in Appendix J and Appendix K.

### 4.3 Intrinsic Regularization and Training Stability

Beyond structured pruning and elastic deployment, the Layer Activation Schedule imposes a strong intrinsic regularization that significantly stabilizes the training dynamics of deep architectures. Standard GPT-2 models frequently suffer from training instability and divergence caused by gradient explosions, necessitating strict optimization constraints. For instance, training a baseline GPT-2 medium model with a learning rate of $3 \times 10^{-4}$ and a gradient clipping norm of $1.0$ (or even $0.25$) leads to catastrophic divergence. To achieve stable training, we had to apply an aggressive gradient clipping norm of $0.1$ and lower the learning rate to $1 \times 10^{-4}$. In contrast, our monotonically decaying survival probability mask naturally regulates the flow of gradients and prevents the over-accumulation of error signals across dimensions. This intrinsic regularization allows our scheduled models to train reliably with higher learning rates ($3 \times 10^{-4}$) and relaxed gradient clipping constraints ($1.0$). By reducing the need for overly conservative hyperparameters, our method provides a robust and highly stable training environment.

## 5 Conclusion

We introduced the Layer Activation Schedule, an end-to-end differentiable gating framework that inherently organizes neural network representations by imposing a strict dimensional hierarchy during training. By modulating computational units with a monotonically decaying survival probability mask, our method consolidates core features into leading dimensions, reducing network sparsification to a trivial structural truncation. This eliminates the need for unstructured masks or post-training fine-tuning and naturally yields a continuous family of nested sub-networks for elastic deployment. Extensive evaluations demonstrate massive efficiency gains: on GPT-2 medium, our approach maintains dense-level generation quality after removing 92.8% of FLOPs, extending the breaking point by 2.7$\times$ over the strongest baseline. On ResNet-50, it preserves 64.62% Top-1 accuracy at a 60.47% MAC reduction, where standard pruning collapses to 0.90%. Ultimately, our structured framework offers a robust and highly efficient paradigm for scalable AI deployment under strict compute constraints.

### AI Assistance

During the preparation of this work, Large Language Models (LLMs) were utilized to aid in the writing process. Specifically, LLMs were used to rephrase and polish the manuscript to enhance readability and improve the flow of the text. Furthermore, we employed LLMs for retrieval and discovery, including searching for and identifying relevant related work. The authors carefully reviewed and verified all AI-generated suggestions and assume full responsibility for the final content of the paper.

## References

Amir Asiaee. Causal Mechanism Reduction: Mechanism Replacement for Neural Network Pruning and Abstraction. *arXiv preprint arXiv:2602.24266*, February 2026. doi: 10.48550/arXiv.2602.24266.

Horace B. Barlow. Possible Principles Underlying the Transformations of Sensory Messages. In Walter A. Rosenblith (ed.), *Sensory Communication*, pp. 217–234. MIT Press, 1961.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Tolga Bolukbasi, Joseph Wang, Ofer Dekel, and Venkatesh Saligrama. Adaptive Neural Networks for Efficient Inference. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 527–536. PMLR, July 2017.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Yufei Cui, Yu Mao, Ziquan Liu, Qiao Li, Antoni B. Chan, Xue Liu, Tei-Wei Kuo, and Chun Jason Xue. Variational Nested Dropout. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(8):10519–10534, August 2023. ISSN 1939-3539. doi: 10.1109/TPAMI.2023.3241945.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Roman Denkin, Ida Akerholm, Prashant Singh, and Ida-Maria Sintorn. Squeeze-Release: Iterative Pruning with Exact Structural Minimization. *arXiv preprint arXiv:2606.14346*, June 2026. doi: 10.48550/arXiv.2606.14346.

Yizhuo Ding, Wanying Qu, Jiawei Geng, Wenqi Shao, and Yanwei Fu. UniPruning: Unifying local metric and global feedback for scalable sparse LLMs. *arXiv preprint arXiv:2510.03291*, 2025.

Gongfan Fang, Xinyin Ma, Mingli Song, Michael Bi Mi, and Xinchao Wang. DepGraph: Towards Any Structural Pruning. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16091–16101, June 2023. doi: 10.1109/CVPR52729.2023.01544.

Federico Fontana, Romeo Lanzino, Marco Raoul Marini, Danilo Avola, Luigi Cinque, Francesco Scarcello, and Gian Luca Foresti. Distilled Gradual Pruning With Pruned Fine-Tuning. *IEEE Transactions on Artificial Intelligence*, 5(8):4269–4279, August 2024. ISSN 2691-4581. doi: 10.1109/TAI.2024.3366497.

Jonathan Frankle and Michael Carbin. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *International Conference on Learning Representations*, May 2019.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 2023.

Yarin Gal, Jiri Hron, and Alex Kendall. Concrete Dropout. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. OpenWebText corpus. `http://Skylion007.github.io/OpenWebTextCorpus`, 2019.

Aidan N. Gomez, Ivan Zhang, Siddhartha Rao Kamalakara, Divyam Madaan, Kevin Swersky, Yarin Gal, and Geoffrey E. Hinton. Learning Sparse Networks Using Targeted Dropout. *arXiv preprint arXiv:1905.13678*, May 2019. doi: 10.48550/arXiv.1905.13678.

Sylvain Gugger, Lysandre Debut, Thomas Wolf, Philipp Schmid, Zachary Mueller, Sourab Mangrulkar, Marc Sun, and Benjamin Bossan. Accelerate: Training and inference at scale made simple, efficient and adaptable. `https://github.com/huggingface/accelerate`, 2022.

Song Guo, Lei Zhang, Xiawu Zheng, Yan Wang, Yuchao Li, Fei Chao, Chenglin Wu, Shengchuan Zhang, and Rongrong Ji. Automatic Network Pruning via Hilbert-Schmidt Independence Criterion Lasso under Information Bottleneck Principle. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 17412–17423, October 2023. doi: 10.1109/ICCV51070.2023.01601.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both Weights and Connections for Efficient Neural Network. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Samuel Horváth, Stefanos Laskaridis, Mario Almeida, Ilias Leontiadis, Stylianos Venieris, and Nicholas Lane. FjORD: Fair and Accurate Federated Learning under heterogeneous targets with Ordered Dropout. In *Advances in Neural Information Processing Systems*, volume 34, pp. 12876–12889. Curran Associates, Inc., 2021.

Zihao Jiang, Tao Deng, Lei You, Siwei Feng, and Juncheng Jia. CoPruning: Exploring the Parameter-Gradient Nonlinear Correlation for Neural Network Pruning Using Copula Function. *OpenReview preprint*, October 2024.

Andrej Karpathy. nanoGPT: The simplest, fastest repository for training/finetuning medium-sized GPTs. `https://github.com/karpathy/nanoGPT`, 2023.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations (ICLR)*, 2015. URL `https://arxiv.org/abs/1412.6980`.

Durk P Kingma, Tim Salimans, and Max Welling. Variational Dropout and the Local Reparameterization Trick. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Humaira Kousar, Hasnain Irshad Bhatti, and Jaekyun Moon. Pruning-based Data Selection and Network Fusion for Efficient Deep Learning. In *Second NeurIPS Workshop on Attributing Model Behavior at Scale*, December 2024.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka Representation Learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30233–30249. Curran Associates, Inc., 2022.

Yann LeCun, John Denker, and Sara Solla. Optimal Brain Damage. In *Advances in Neural Information Processing Systems*, volume 2. Morgan-Kaufmann, 1989.

Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations (ICLR)*, April 2017. doi: 10.48550/arXiv.1608.08710.

Yongqing Liang, Dawei Xiang, and Xin Li. SEArch: A Self-Evolving Framework for Network Architecture Optimization. *Neurocomputing*, 651:130980, October 2025. doi: 10.1016/j.neucom.2025.130980.

Shaohui Lin, Rongrong Ji, Yuchao Li, Cheng Deng, and Xuelong Li. Toward Compact ConvNets via Structure-Sparsity Regularized Filter Pruning. *IEEE Transactions on Neural Networks and Learning Systems*, 31(2):574–588, February 2020. doi: 10.1109/TNNLS.2019.2906563.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations (ICLR)*, 2019. URL `https://openreview.net/forum?id=Bkg6RiCqY7`.

Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5058–5066, 2017.

Jian-Hao Luo, Hao Zhang, Hong-Yu Zhou, Chen-Wei Xie, Jianxin Wu, and Weiyao Lin. ThiNet: Pruning CNN Filters for a Thinner Net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2525–2538, October 2019. ISSN 1939-3539. doi: 10.1109/TPAMI.2018.2858232.

Eran Malach, Gilad Yehudai, Shai Shalev-Shwartz, and Ohad Shamir. Proving the Lottery Ticket Hypothesis: Pruning is All You Need. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 6682–6691. PMLR, November 2020.

Rayeesa Mehmood, Sergei Koltcov, Anton Surkov, and Vera Ignatenko. Modeling Pruning as a Phase Transition: A Thermodynamic Analysis of Neural Activations. *Computers, Materials & Continua*, 86(3), 2026. ISSN 1546-2218, 1546-2226. doi: 10.32604/cmc.2025.072735.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. ShortGPT: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-acl.1035.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017.

Asit Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *arXiv preprint arXiv:2104.08378*, 2021.

Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11264–11272, 2019.

Tao Niu, Yinglei Teng, and Yiding Liu. Pruning-as-Reconstruct: Masked Autoencoders are Efficient Importance Indicators. Preprint, October 2023.

Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996. doi: 10.1038/381607a0.

OpenAI. tiktoken: A fast BPE tokeniser for use with OpenAI's models. `https://github.com/openai/tiktoken`, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1525–1534, 2016. doi: 10.18653/v1/P16-1144.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 8024–8035, 2019.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 4816–4828, 2021.

Edoardo Maria Ponti and André F. T. Martins. Dynamic Sparsity in Machine Learning: Routing Information through Neural Pathways. In *Advances in Neural Information Processing Systems (NeurIPS), Tutorial Track*, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL `https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf`.

Oren Rippel, Michael Gelbart, and Ryan Adams. Learning Ordered Representations with Nested Dropout. In *Proceedings of the 31st International Conference on Machine Learning*, pp. 1746–1754. PMLR, June 2014.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pp. 90–95, 2011.

Hojjat Salehinejad and Shahrokh Valaee. EDropout: Energy-Based Dropout and Pruning of Deep Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10): 5279–5292, October 2022. doi: 10.1109/TNNLS.2021.3069970.

Ayan Sengupta, Siddhant Chaudhary, and Tanmoy Chakraborty. You Only Prune Once: Designing Calibration-Free Model Compression With Policy Learning. In *The Thirteenth International Conference on Learning Representations*, April 2025.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. In *International Conference on Learning Representations*, February 2017.

Maying Shen, Hongxu Yin, Pavlo Molchanov, Lei Mao, Jianna Liu, and Jose M. Alvarez. Structural Pruning via Latency-Saliency Knapsack. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12894–12908. Curran Associates, Inc., 2022.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. ISSN 1533-7928.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2306.11695.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–9, 2015. doi: 10.1109/CVPR.2015.7298594.

Enzo Tartaglione, Andrea Bragagnolo, Francesco Odierna, Attilio Fiandrotti, and Marco Grangetto. SeReNe: Sensitivity-Based Regularization of Neurons for Structured Sparsity in Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):7237–7250, December 2022. doi: 10.1109/TNNLS.2021.3084527.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, April 2015. doi: 10.1109/ITW.2015.7133169.

TorchVision maintainers and contributors. TorchVision image classification reference training scripts. `https://github.com/pytorch/vision/tree/main/references/classification`, 2016. Accessed 2026-09-12. Defines the 90-epoch ResNet-50 recipe: SGD with momentum $0.9$, initial learning rate $0.1$ at a total batch size of $256$, weight decay $10^{-4}$, and step decay $\gamma = 0.1$ every $30$ epochs.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5998–6008, 2017.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text (W-NUT) at EMNLP*, pp. 94–106, 2017. doi: 10.18653/v1/W17-4413.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning Structured Sparsity in Deep Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Charles Westphal, Stephen Hailes, and Mirco Musolesi. Mutual Information Preserving Neural Network Pruning. *arXiv preprint arXiv:2411.00147*, November 2024.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 10524–10533. PMLR, 2020.

Taisuke Yasuda, Kyriakos Axiotis, Gang Fu, Mohammadhossein Bateni, and Vahab Mirrokni. SequentialAttention++ for Block Sparsification: Differentiable Pruning Meets Combinatorial Optimization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024.

Lu Yin, You Wu, Zhenyu Zhang, Cheng-Yu Hsieh, Yaqing Wang, Yiling Jia, Gen Li, Ajay Jaiswal, Mykola Pechenizkiy, Yi Liang, Michael Bendersky, Zhangyang Wang, and Shiwei Liu. Outlier weighed layerwise sparsity (OWL): A missing secret sauce for pruning LLMs to high sparsity. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*. PMLR, 2024. arXiv:2310.05175.

Jiahui Yu and Thomas S. Huang. Universally Slimmable Networks and Improved Training Techniques. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1803–1811, 2019.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 4791–4800, 2019. doi: 10.18653/v1/P19-1472.

Yingtao Zhang, Haoli Bai, Haokun Lin, Jialin Zhao, Lu Hou, and Carlo Vittorio Cannistraci. Plug-and-play: An efficient post-training pruning method for large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

## A Additional Gating Architecture Diagrams

This section complements the architectural variants of Section 3.3. Figure 8 shows the Extended Stream variant for ResNet-50, the counterpart of Figure 5 in the main text. Figures 9 and 10 then show the Intra-Block counterparts of Figures 5 and 8, with the same layout and colours. In the Intra-Block variant, the tensors passed between blocks keep their full width, so the residual stream is never masked. Each block draws its own masks, and the masks of different blocks never have to agree because they never reach the residual addition. The only masked layers are therefore inside the blocks: the attention heads and the MLP hidden layer in GPT, and the reduced-width convolutions in the ResNet-50 bottleneck.

## B Detailed Formulation of Structured Pruning

This section details the pruning operator introduced in Section 3.4 and illustrates its application across different layers.

**Prunable Layers.** Pruning is applied to any parameterized layer equipped with the activation schedule. This includes linear projections (`Linear`), convolutional layers (`Conv2d`), normalization layers (`BatchNorm2d`, `LayerNorm`), and token embeddings (`Embedding`). Crucially, the final classification or language modeling head is excluded from this process. This layer maps the terminal hidden representations directly to output logits without an intermediate non-linear activation or probability schedule.

**Pruning Operator: Direct Structural Truncation.** Consider a scheduled layer parameterized by weights and optionally biases, where the output dimensions are strictly ordered according to their assigned survival probabilities. To achieve a target drop ratio, we target the least probable trailing units (e.g., neurons or channels) for removal.

Rather than merely masking these units with zeros, we perform true structural pruning by physically truncating the layer's parameters across the network. For a **Linear** layer, the retained weight matrix

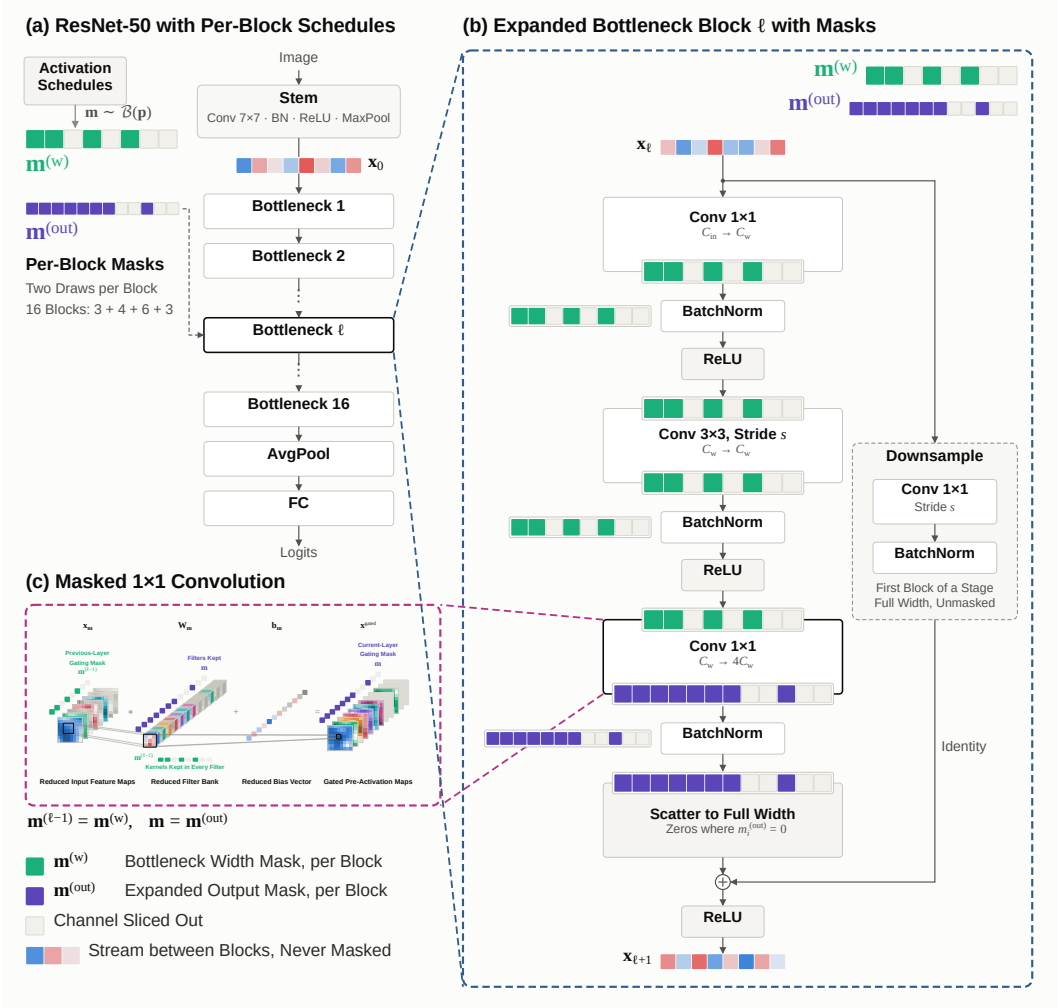



Figure 8: Activation schedule extended to the expanded output of the ResNet-50 bottleneck (Extended Stream Gating). **(a)** Each of the 16 bottleneck blocks ($3 + 4 + 6 + 3$) draws its own two masks per forward pass. The stem, the feature maps passed between blocks, the average pool and the classifier always keep their full width. **(b)** An expanded bottleneck block, with each mask drawn on the edge where a tensor crosses it. $\boldsymbol{m}^{(w)}$ (green) masks the internal width: the output of the first $1 \times 1$ convolution, both sides of the $3 \times 3$ convolution and the input of the last $1 \times 1$ convolution, together with the two BatchNorms in between. $\boldsymbol{m}^{(\text{out})}$ (violet) masks the $4C_w$ output of the last $1 \times 1$ convolution and its BatchNorm. This narrow result is scattered back into a zero-filled full-width tensor before the residual addition, so the dropped channels carry the identity alone. The downsample on the identity path of a stage's first block is never masked. **(c)** The last $1 \times 1$ convolution with both masks applied, as in Figure 4b (drawn with the $3 \times 3$ kernels of the generic masked convolution): input channels are sliced by $\boldsymbol{m}^{(\ell-1)} = \boldsymbol{m}^{(w)}$ and filters by $\boldsymbol{m} = \boldsymbol{m}^{(\text{out})}$. Empty cells are sliced out of this forward pass; they are not zeroed.

$\boldsymbol{W}'$ and bias $\boldsymbol{b}'$ are obtained by slicing the output dimension to keep only the high-probability components:

$$\boldsymbol{W}' = \boldsymbol{W}_{1:N',:}, \quad \boldsymbol{b}' = \boldsymbol{b}_{1:N'} \tag{3}$$

To preserve dimensional integrity, the input dimension of the subsequent linear layer must also be reduced by discarding the corresponding input channels: $\boldsymbol{W}'_{\text{next}} = \boldsymbol{W}_{\text{next}_{:,1:N'}}$.

This direct truncation naturally extends to other architectural components:

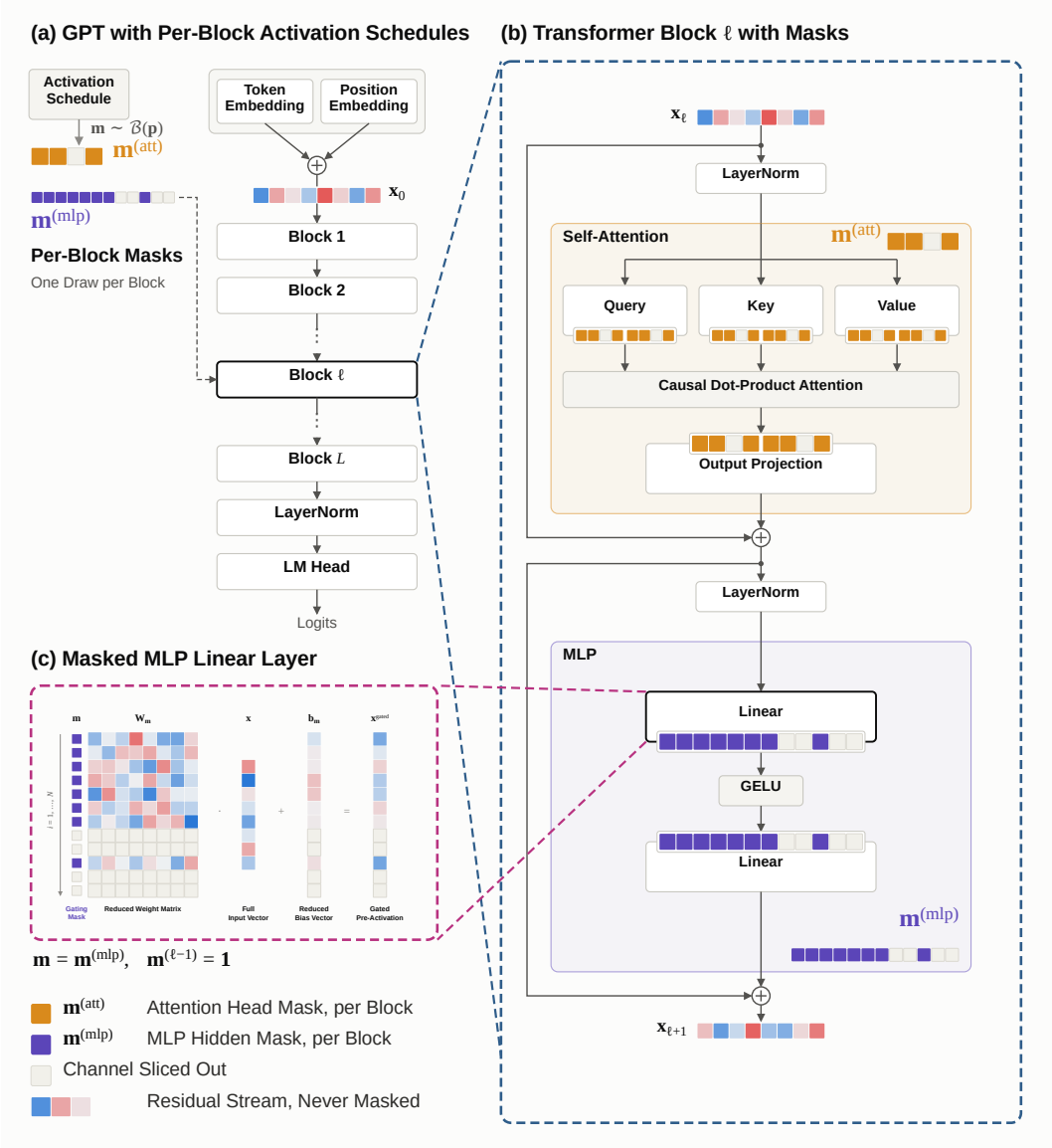



Figure 9: Activation schedule inside the transformer blocks of GPT (Intra-Block Gating). **(a)** The residual stream $\boldsymbol{x}$ keeps its full width $d$ along its whole length. The token and position embeddings, the final LayerNorm and the tied head are plain modules with no mask. Each block has its own activation schedules and draws its masks independently of the other blocks. **(b)** A pre-LN transformer block, with each mask drawn on the edge where a tensor crosses it. $\boldsymbol{m}^{(\text{att})}$ (amber) masks the head dimension and is repeated over the $n_h$ heads. It is the output mask of the query, key and value projections and the input mask of the attention output projection. $\boldsymbol{m}^{(\text{mlp})}$ (violet) masks the $4d$ hidden units. It is the output mask of the MLP expansion and the input mask of its contraction. Both LayerNorms, the query, key and value inputs, the output of the output projection, and the residual additions stay at full width. **(c)** The MLP expansion layer with its mask applied, as in Figure 4a: rows are sliced by $\boldsymbol{m} = \boldsymbol{m}^{(\text{mlp})}$, and no column is sliced because the input is unmasked ($\boldsymbol{m}^{(\ell-1)} = \mathbf{1}$). Empty cells are sliced out of this forward pass; they are not zeroed.

- **Convolutional Layers (`Conv2d`):** The 4D weight tensor $\boldsymbol{W}$ is sliced along the output channel dimension to drop trailing filters, and along the input channel dimension to match the preceding layer's narrowed output (Figure 11).

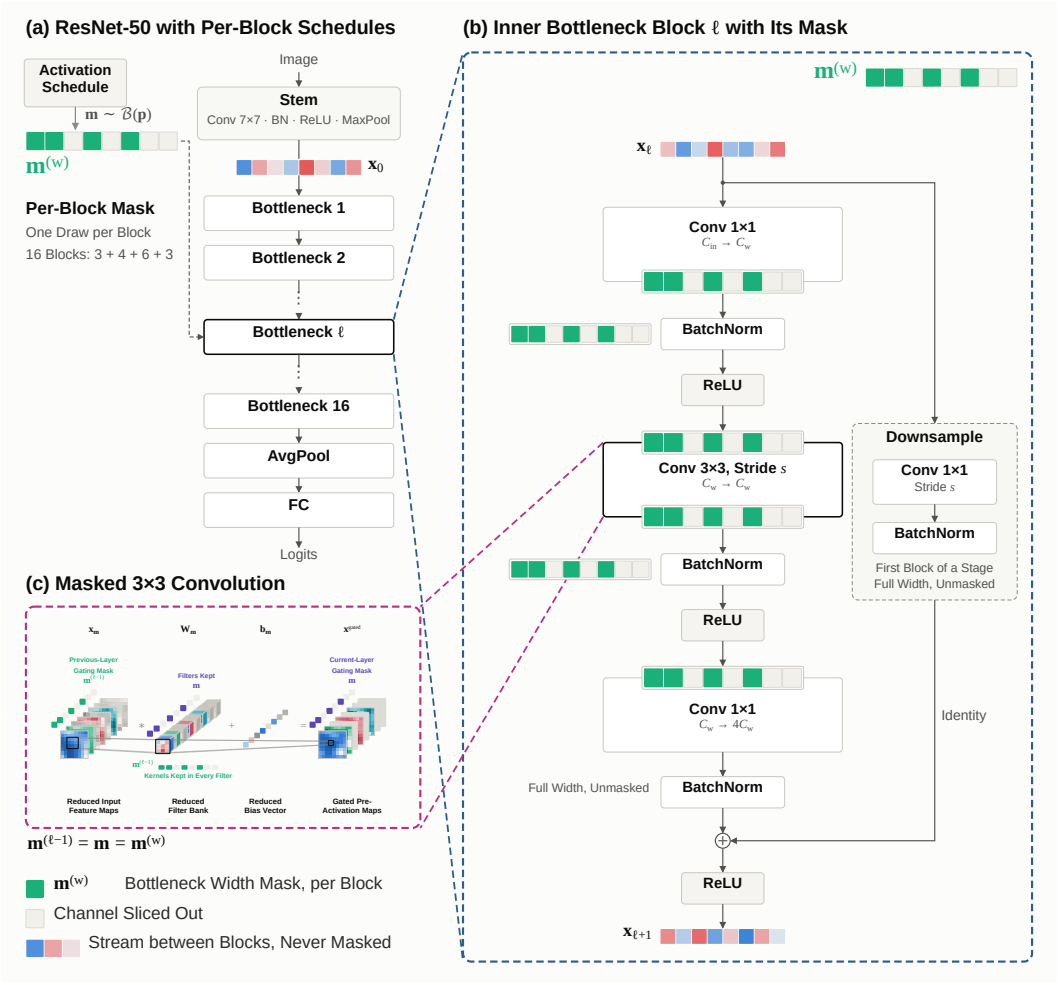



Figure 10: Activation schedule on the inner width of the ResNet-50 bottleneck (Intra-Block Gating). **(a)** Each of the 16 bottleneck blocks $(3+4+6+3)$ draws its own mask per forward pass. The stem, the feature maps passed between blocks, the average pool and the classifier always keep their full width. **(b)** A bottleneck block, with the mask drawn on each edge where a tensor crosses it. $\boldsymbol{m}^{(w)}$ (green) masks the internal width: the output of the first $1 \times 1$ convolution, both sides of the $3 \times 3$ convolution and the input of the last $1\times1$ convolution, together with the two BatchNorms in between. The last $1 \times 1$ convolution widens back to the full $4C_w$ output with no mask. Its BatchNorm, the residual addition and the downsample on the identity path of a stage's first block therefore all stay at full width, and no scatter is needed. **(c)** The $3\times3$ convolution with the mask applied, as in Figure 4b. Because this convolution maps $C_w$ to $C_w$ channels, input channels and filters are sliced by the same mask, $\boldsymbol{m}^{(\ell-1)} = \boldsymbol{m} = \boldsymbol{m}^{(w)}$. Empty cells are sliced out of this forward pass; they are not zeroed.

- **Normalization Layers (`BatchNorm2d`, `LayerNorm`):** The affine transformation parameters (scale and shift) and any tracked running statistics are sliced to retain only the high-probability channels.

- **Token Embeddings (`Embedding`):** The embedding weight matrix is sliced along the embedding dimension to reduce the representational width of the tokens.

By physically removing these dimensions across all scheduled components, we directly shrink the model size and execution cost.

**Pruning Illustrations.** Figure 11 applies the direct structural truncation to a convolutional layer, on both of its channel axes. Figure 12 contrasts it with unstructured pruning of a linear layer trained without the schedule. In both cases, the schedule has already ordered the units by importance during

training. The pruning operator is therefore a slice, with no importance scoring, magnitude threshold, or fine-tuning required.

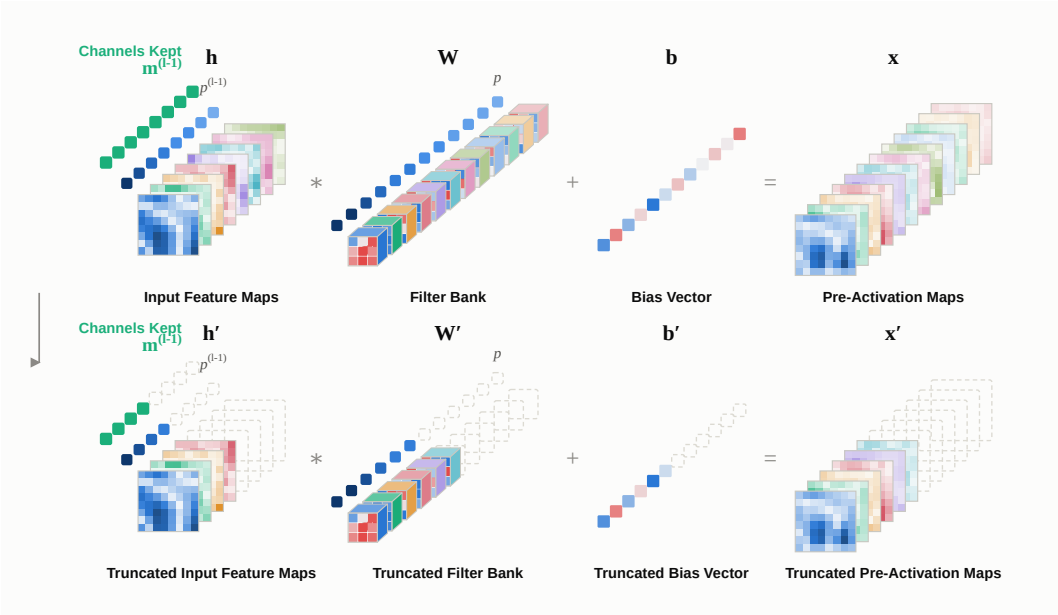



Figure 11: Structured truncation of a convolutional layer, $\boldsymbol{W}' = \boldsymbol{W}_{1:N',\,1:d',\,:,\,:}$ and $\boldsymbol{b}' = \boldsymbol{b}_{1:N'}$. **Top:** the layer as it is after training under the schedule. The blue strips are the schedules, $\boldsymbol{p}^{(\ell-1)}$ for the input channels and $\boldsymbol{p}$ for the filters: dark where a channel is almost always active and pale where it rarely is. The green strip is the previous layer's mask $\boldsymbol{m}^{(\ell-1)}$. Because the layer was trained under these schedules, a filter's magnitude decays with its survival probability, and a kernel's magnitude decays with the product of its filter's and its input channel's probabilities. The tensors are therefore already ordered by importance. The feature maps share one colour scale, which shows that the trailing output maps carry almost nothing. **Bottom:** the same layer truncated with $r = 0.5$ on both channel axes ($N = 12 \to N' = 6$ filters and $d_{\text{in}} = 8 \to d' = 4$ input channels). On the output axis, whole filters are removed together with their bias entries and output maps. On the input axis, whole input channels are removed together with the kernel slices that read them. Halving both axes leaves a quarter of the kernels. Dashed footprints mark the removed parts and only keep the two rows comparable: in the pruned model these channels no longer exist, so both the tensor and the convolution are smaller.

## C Detailed Experimental Setup

### C.0.1 GPT-2

Our language-modelling protocol follows the GPT-2 pre-training recipe (Radford et al., 2019) in the form standardised by nanoGPT (Karpathy, 2023), which is the codebase our training script is derived from.

**Dataset and Architecture.** Models are trained from scratch on OpenWebText (Gokaslan et al., 2019), tokenised with the GPT-2 byte-level BPE encoder as shipped in `tiktoken` (OpenAI, 2022) (vocabulary size 50,257, padded to 50,304). Before tokenising, we hold out a 0.05% random slice of the documents as the validation split. The architecture is GPT-2 medium: a decoder-only pre-LN Transformer (Vaswani et al., 2017; Xiong et al., 2020) with $L = 24$ blocks, $n_{\text{head}} = 16$ attention heads, model width $d_{\text{model}} = 1024$, context length $T = 1024$ and an MLP inner width of $4d_{\text{model}} = 4096$, for 353.80M non-embedding parameters. Biases are disabled in every linear and normalisation layer, and dropout is set to 0.

**Optimisation.** All runs use AdamW (Loshchilov & Hutter, 2019; Kingma & Ba, 2015) with moment coefficients $\beta_1 = 0.9$ and $\beta_2 = 0.95$, and decoupled weight decay 0.1. For the Baseline

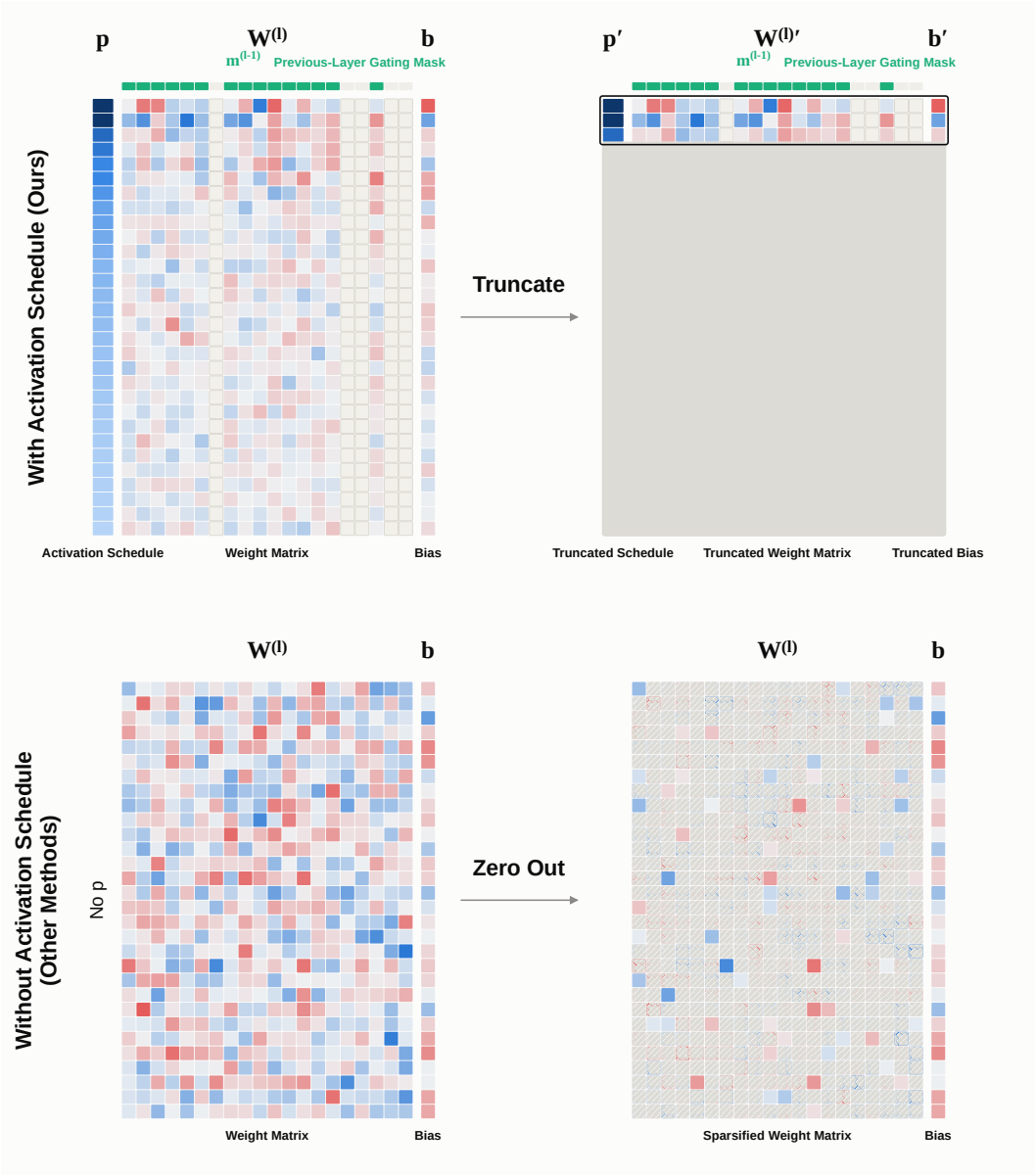



Figure 12: Structured truncation with the activation schedule versus unstructured pruning without it, at the same drop ratio $r = 0.9$ for a linear layer with $N = 30$ neurons and $d_{\text{in}} = 20$ inputs. **Top (ours):** the column $\boldsymbol{p}$ is the survival probability of each row of $\boldsymbol{W}^{(\ell)}$. The green strip is the previous layer's mask $\boldsymbol{m}^{(\ell-1)}$, and the columns it drops are left empty, as in Figure 4a. After training under the schedule, a row's magnitude decays with its probability, so the matrix is already ordered by importance. Pruning is one contiguous truncation of the trailing rows. $\boldsymbol{W}$, $\boldsymbol{b}$ and $\boldsymbol{p}$ keep their leading $N'$ entries (outlined), the grey block is removed, and the result is a smaller dense matrix. **Bottom (without the schedule):** a standard layer has no ordering, so its weights are scattered uniformly. Removing the same fraction zeroes individual entries wherever they fall (hatched cells). The matrix keeps its original shape, so neither memory nor multiplications are saved, and the learned function is disrupted indiscriminately.

model, we use a peak learning rate of $\eta_{\max} = 1 \times 10^{-4}$ and gradients are clipped to a global $\ell_2$ norm of $0.1$ before each step. For the Layer Activation Schedule models, we use a higher peak learning rate of $\eta_{\max} = 3 \times 10^{-4}$ and a relaxed gradient clipping norm of $1.0$. The learning rate is warmed up linearly over the first $2000$ steps and then annealed by a cosine schedule towards $10\%$ of $\eta_{\max}$ over $200{,}000$ total training steps. Each optimiser step consumes a micro-batch of $12$ sequences

per process, accumulated over $40$ micro-batches in total on $8$ processes, for a global batch of $480$ sequences ($\approx 0.5$M tokens).

**Evaluation and Hardware.** Training is implemented in PyTorch (Paszke et al., 2019) and distributed with HuggingFace Accelerate (Gugger et al., 2022) on a single node of $8\times$ NVIDIA A100 GPUs. Forward and backward passes run in `bfloat16` mixed precision, with master weights and the optimiser state in FP32, and TF32 enabled for FP32 matmuls.

#### C.0.2 RESNET50

Our ResNet-50 protocol closely follows the original training recipe for this architecture (He et al., 2016), as standardised by the TorchVision classification reference scripts (TorchVision maintainers and contributors, 2016).

**Dataset and Architecture.** All models are trained from scratch on ImageNet-1K (ILSVRC-2012) (Deng et al., 2009). The network is the standard ResNet-50 adopting the v1.5 convention, where the downsampling stride is carried by the $3 \times 3$ convolution rather than the $1 \times 1$ convolution.

**Optimisation.** All runs are trained for $90$ epochs at a batch size of $512$ with AdamW (Loshchilov & Hutter, 2019), an initial learning rate of $\eta_0 = 10^{-3}$, decoupled weight decay $10^{-4}$, and default Adam moment coefficients. The learning rate follows a step-decay schedule, dropping by a factor of ten at epoch $50$. The objective is cross-entropy with label smoothing $\epsilon = 0.1$. Automatic mixed precision is disabled to remove loss-scaling dynamics as a possible confounder.

**Data augmentation.** Training images are augmented with a scale-and-aspect-ratio random resized crop (Szegedy et al., 2015) to $224 \times 224$, followed by a random horizontal flip with probability $0.5$. Evaluation uses the standard resize-then-centre-crop protocol. Both pipelines conclude by scaling pixels to $[0, 1]$ and normalising each channel with ImageNet statistics.

## D ADDITIONAL GPT-2 PRUNING RESULTS

Table 1 of Section 4.1 reports three headline metrics at the three matched compute budgets. This section gives the remaining nine, over the same thirteen methods and the same budgets, so that all twelve metrics of the evaluation are available in full. Every value is read exactly as in the main table: a budget rarely lands on a measured configuration, so each reading is interpolated between the two measured points that bracket it, in log space for the perplexities and linearly for the losses and accuracies. Nothing is extrapolated. Table 2 covers the OpenWebText validation loss together with the fixed-length and LAMBADA metrics and keeps the layout of the main table, with the budget running across the columns. Table 3 breaks the zero-shot average into its five individual tasks; because five tasks by three budgets does not fit across one row of columns, the budget runs down the table there instead, as three blocks of thirteen methods.

## E GPT-2 PRUNING CURVES: STRUCTURED METHODS

Figure 13 and Figure 14 show two of the twelve evaluation metrics across all thirteen methods together, for the two scheduled models and the four strongest baselines. This section gives the full sweep for the structured family on its own, over all twelve metrics. Restricting each panel to a single family keeps the curves separable, where the combined view above puts thirteen of them in the same band. The methods here are the two activation-scheduled variants, ShortGPT (Men et al., 2025) as the depth-pruning baseline, and structured magnitude (Li et al., 2017) and Taylor (Molchanov et al., 2019) pruning, which remove whole units by weight magnitude and by a first-order saliency estimate respectively.

The axes are the same in every panel of this section and of Appendix F. The lower axis is the percentage of the dense FLOPs removed and the upper axis the GFLOPs per token that remain, against a dense GPT-2 medium of 0.808 GFLOPs per token; the dashed horizontal line is the dense reference. Three conventions are worth stating before the panels are read. First, the two scheduled

Table 2: Validation loss, fixed-length and LAMBADA metrics for every method at the three matched compute budgets, read the same way as Table 1. **Loss** is the OpenWebText validation loss, **FL ppl** the fixed-length perplexity at stride 512 and **LMB ppl / LMB acc** the LAMBADA (OpenAI split) perplexity and last-word accuracy. Lower is better for the loss and the two perplexities, higher for the accuracy. Best value per column in bold; the dense row is the unpruned reference and is not eligible.

| Method | 20% removed | | | | 40% removed | | | | 60% removed | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Loss | FL ppl | LMB ppl | LMB acc | Loss | FL ppl | LMB ppl | LMB acc | Loss | FL ppl | LMB ppl | LMB acc |
| *Ours (structured)* | | | | | | | | | | | | |
| Act. Sch. - Intra-Block | 2.75 | 16.5 | 11.3 | 0.388 | **2.82** | **17.8** | **13.8** | **0.352** | 3.08 | 23.3 | 29.6 | 0.253 |
| Act. Sch. - Extended Stream | 2.85 | 18.5 | 13.2 | 0.367 | 2.90 | 19.4 | 15.8 | 0.340 | **3.02** | **22.1** | **21.2** | **0.303** |
| *Structured pruning methods* | | | | | | | | | | | | |
| ShortGPT (depth) | 4.07 | 54.2 | 167.2 | 0.128 | 6.61 | 738.2 | $2.9\times10^{3}$ | 0.041 | 8.70 | $1.2\times10^{4}$ | $1.0\times10^{5}$ | 0.001 |
| Magnitude (structured) | 18.20 | $1.0\times10^{8}$ | $3.6\times10^{6}$ | 0.000 | 22.57 | $5.8\times10^{9}$ | $7.6\times10^{7}$ | 0.000 | 14.53 | $2.2\times10^{6}$ | $1.0\times10^{6}$ | 0.000 |
| Taylor (structured) | 9.43 | $1.3\times10^{4}$ | $1.2\times10^{5}$ | 0.000 | 12.02 | $1.6\times10^{5}$ | $3.7\times10^{5}$ | 0.000 | 13.38 | $3.9\times10^{5}$ | $7.9\times10^{5}$ | 0.000 |
| *Semi-structured pruning method* | | | | | | | | | | | | |
| UniPruning (2:4) | 2.78 | 15.0 | 11.6 | 0.393 | 3.77 | 42.2 | 105.4 | 0.182 | 10.54 | $6.9\times10^{4}$ | $1.2\times10^{5}$ | 0.000 |
| *Unstructured pruning methods* | | | | | | | | | | | | |
| SparseGPT | **2.74** | **14.4** | **10.2** | **0.413** | 4.04 | 69.1 | 69.4 | 0.254 | 9.79 | $2.5\times10^{4}$ | $1.3\times10^{5}$ | 0.000 |
| RIA | 2.75 | 14.5 | 10.9 | 0.399 | 4.37 | 82.0 | 226.5 | 0.157 | 11.38 | $8.7\times10^{4}$ | $2.5\times10^{5}$ | 0.000 |
| Wanda | 2.76 | 14.7 | 11.6 | 0.389 | 4.96 | 155.5 | 725.1 | 0.104 | 10.64 | $3.8\times10^{4}$ | $4.3\times10^{5}$ | 0.000 |
| OWL | 2.76 | 14.7 | 11.6 | 0.387 | 4.39 | 90.4 | 282.7 | 0.129 | 10.71 | $3.8\times10^{4}$ | $3.7\times10^{5}$ | 0.000 |
| UniPruning | 2.78 | 15.0 | 10.9 | 0.407 | 3.74 | 40.5 | 89.6 | 0.194 | 11.00 | $1.1\times10^{5}$ | $1.2\times10^{5}$ | 0.000 |
| Magnitude | 2.82 | 15.6 | 13.4 | 0.358 | 8.21 | $5.2\times10^{3}$ | $1.3\times10^{4}$ | 0.008 | 11.75 | $1.1\times10^{5}$ | $5.6\times10^{5}$ | 0.000 |
| Random (control) | 11.02 | $7.0\times10^{4}$ | $1.0\times10^{6}$ | 0.000 | 10.63 | $4.0\times10^{4}$ | $4.7\times10^{5}$ | 0.000 | 15.63 | $7.3\times10^{6}$ | $1.1\times10^{6}$ | 0.000 |
| *Dense GPT-2 medium* | *2.68* | *13.6* | *10.1* | *0.410* | *2.68* | *13.6* | *10.1* | *0.410* | *2.68* | *13.6* | *10.1* | *0.410* |

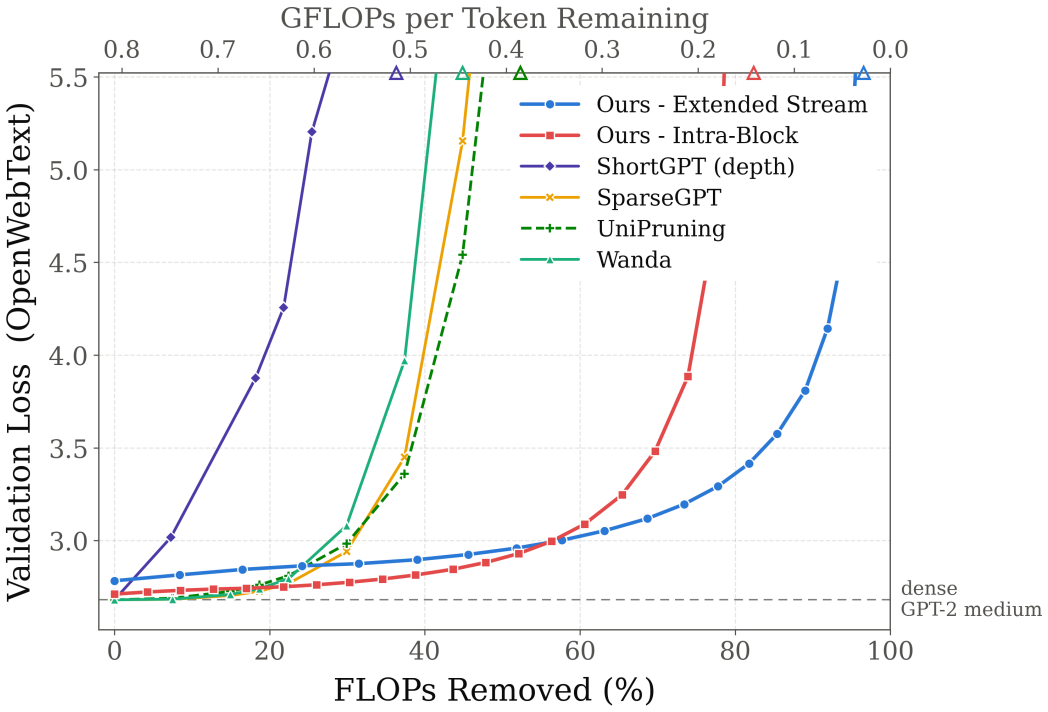



Figure 13: OpenWebText validation loss for GPT-2 medium under pruning across the full sweep of compute budgets, for the two scheduled models and the four strongest baselines, against the percentage of dense FLOPs removed on the lower axis and the GFLOPs per token that remain on the upper axis. The dashed line is the dense GPT-2 medium reference at 2.68. Lower is better, and the vertical axis is capped at a loss of 5.5, with an open triangle marking where a curve leaves it.

variants are separately trained checkpoints rather than post-hoc compressions of the dense model, so their curves do not start on the dense line at 0% removed; the gap at the left edge of every axis is a property of the checkpoints, not measurement noise. Second, collapsed configurations reach perplexities of $10^{31}$, which would flatten every curve worth reading, so the loss and perplexity axes are capped at a multiple of the dense reference and an open triangle on the top spine marks where a curve leaves the axis; nothing is capped in the underlying data. Third, the curves end at different budgets because the sweeps reach different depths, and nothing is interpolated past a

Table 3: The five individual zero-shot tasks that the 0-shot average of Table 1 is taken over, for every method at each of the three matched compute budgets. The budget runs down the table here rather than across it, because five tasks by three budgets is too wide for one row of columns. ARC-Easy (Clark et al., 2018), HellaSwag (Zellers et al., 2019) and PIQA (Bisk et al., 2020) are length-normalised accuracies, SciQ (Welbl et al., 2017) and COPA (Roemmele et al., 2011) plain accuracies; higher is better throughout. Best value per column within each budget block in bold; the dense row is the unpruned reference and is not eligible.

| **Method** | **ARC-E** | **HellaSwag** | **PIQA** | **SciQ** | **COPA** |
|---|---|---|---|---|---|
| *20% of dense FLOPs removed* | | | | | |
| Act. Sch. - Intra-Block | 0.433 | 0.363 | 0.638 | 0.776 | 0.683 |
| Act. Sch. - Extended Stream | 0.403 | 0.337 | 0.645 | 0.727 | 0.655 |
| ShortGPT (depth) | 0.352 | 0.335 | 0.587 | 0.576 | 0.610 |
| Magnitude (structured) | 0.248 | 0.262 | 0.505 | 0.206 | 0.584 |
| Taylor (structured) | 0.271 | 0.266 | 0.516 | 0.240 | 0.547 |
| UniPruning (2:4) | **0.439** | 0.364 | 0.652 | 0.782 | 0.696 |
| SparseGPT | 0.434 | **0.374** | 0.651 | 0.773 | 0.686 |
| RIA | 0.433 | 0.370 | **0.654** | 0.781 | 0.686 |
| Wanda | 0.431 | 0.372 | 0.652 | 0.780 | **0.720** |
| OWL | 0.435 | 0.372 | 0.649 | 0.782 | 0.690 |
| UniPruning | 0.438 | 0.364 | 0.651 | **0.786** | 0.687 |
| Magnitude | 0.422 | 0.369 | 0.653 | 0.774 | 0.707 |
| Random (control) | 0.261 | 0.266 | 0.515 | 0.214 | 0.540 |
| *40% of dense FLOPs removed* | | | | | |
| Act. Sch. - Intra-Block | **0.411** | **0.344** | 0.630 | **0.759** | **0.637** |
| Act. Sch. - Extended Stream | 0.394 | 0.322 | **0.635** | 0.712 | 0.634 |
| ShortGPT (depth) | 0.283 | 0.281 | 0.535 | 0.369 | 0.585 |
| Magnitude (structured) | 0.236 | 0.259 | 0.517 | 0.203 | 0.557 |
| Taylor (structured) | 0.258 | 0.264 | 0.514 | 0.209 | 0.527 |
| UniPruning (2:4) | 0.352 | 0.310 | 0.593 | 0.645 | 0.600 |
| SparseGPT | 0.351 | 0.317 | 0.589 | 0.600 | 0.610 |
| RIA | 0.352 | 0.306 | 0.583 | 0.585 | 0.598 |
| Wanda | 0.322 | 0.293 | 0.575 | 0.468 | 0.603 |
| OWL | 0.339 | 0.306 | 0.585 | 0.548 | 0.616 |
| UniPruning | 0.350 | 0.308 | 0.595 | 0.646 | 0.606 |
| Magnitude | 0.290 | 0.278 | 0.539 | 0.361 | 0.531 |
| Random (control) | 0.253 | 0.259 | 0.503 | 0.186 | 0.455 |
| *60% of dense FLOPs removed* | | | | | |
| Act. Sch. - Intra-Block | **0.395** | **0.311** | 0.611 | **0.713** | **0.639** |
| Act. Sch. - Extended Stream | 0.392 | 0.303 | **0.615** | 0.704 | 0.587 |
| ShortGPT (depth) | 0.278 | 0.261 | 0.516 | 0.252 | 0.536 |
| Magnitude (structured) | 0.239 | 0.266 | 0.528 | 0.199 | 0.540 |
| Taylor (structured) | 0.243 | 0.265 | 0.504 | 0.191 | 0.567 |
| UniPruning (2:4) | 0.269 | 0.256 | 0.500 | 0.200 | 0.550 |
| SparseGPT | 0.275 | 0.266 | 0.503 | 0.207 | 0.531 |
| RIA | 0.263 | 0.270 | 0.515 | 0.200 | 0.590 |
| Wanda | 0.269 | 0.265 | 0.498 | 0.197 | 0.600 |
| OWL | 0.275 | 0.268 | 0.512 | 0.202 | 0.551 |
| UniPruning | 0.259 | 0.260 | 0.496 | 0.205 | 0.589 |
| Magnitude | 0.253 | 0.263 | 0.504 | 0.220 | 0.608 |
| Random (control) | 0.263 | 0.256 | 0.499 | 0.196 | 0.549 |
| *Dense GPT-2 medium* | *0.440* | *0.379* | *0.661* | *0.783* | *0.690* |

method's measured range: the Extended Stream model is measured to 100% of FLOPs removed and the Intra-Block model to 87.2%, while ShortGPT stops at 80.0% and the two structured baselines at 66.7%.

Read at 66% of the dense FLOPs removed, the deepest budget every curve in this family reaches, the separation of Section 4.1 holds on all twelve axes. The Extended Stream model is at a validation loss of 3.09 and the Intra-Block model at 3.28, against 11.11 for ShortGPT, 12.02 for Taylor and 13.06 for structured magnitude pruning. MAUVE reads 0.919 for Intra-Block and 0.883 for Extended Stream, against 0.023 or below for all three baselines, and the nine-task zero-shot average reads 0.461 and 0.459 against 0.358–0.377.

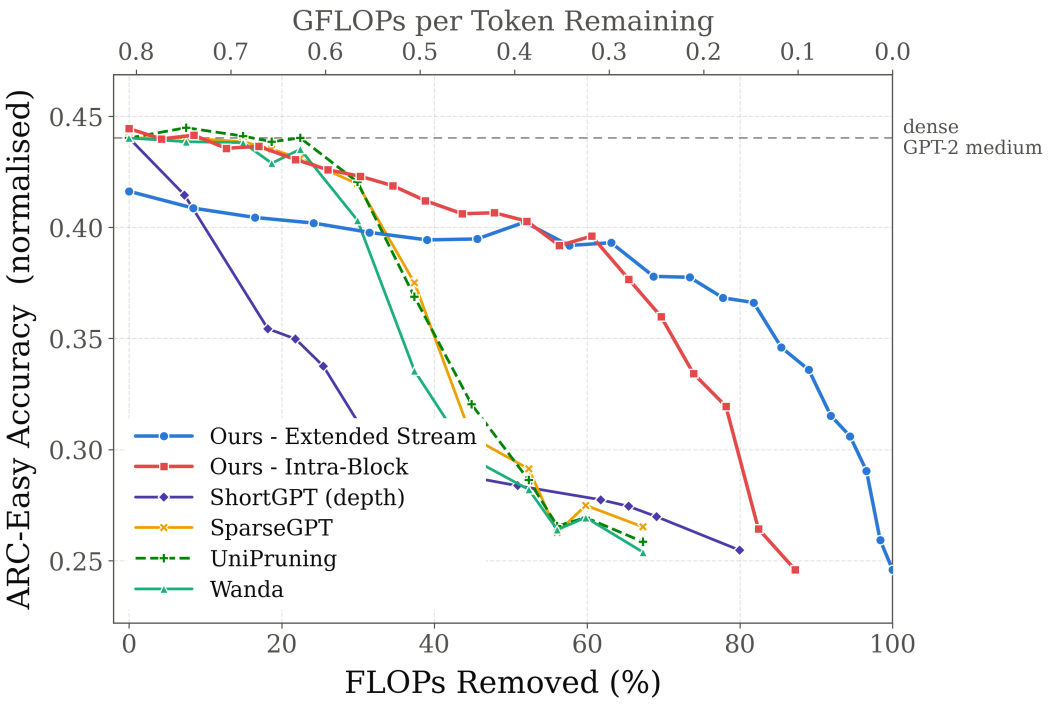



Figure 14: Length-normalised ARC-Easy accuracy for GPT-2 medium under pruning across the full sweep of compute budgets, for the two scheduled models and the four strongest baselines, against the percentage of dense FLOPs removed on the lower axis and the GFLOPs per token that remain on the upper axis. The dashed line is the dense GPT-2 medium reference at 0.440. Higher is better.

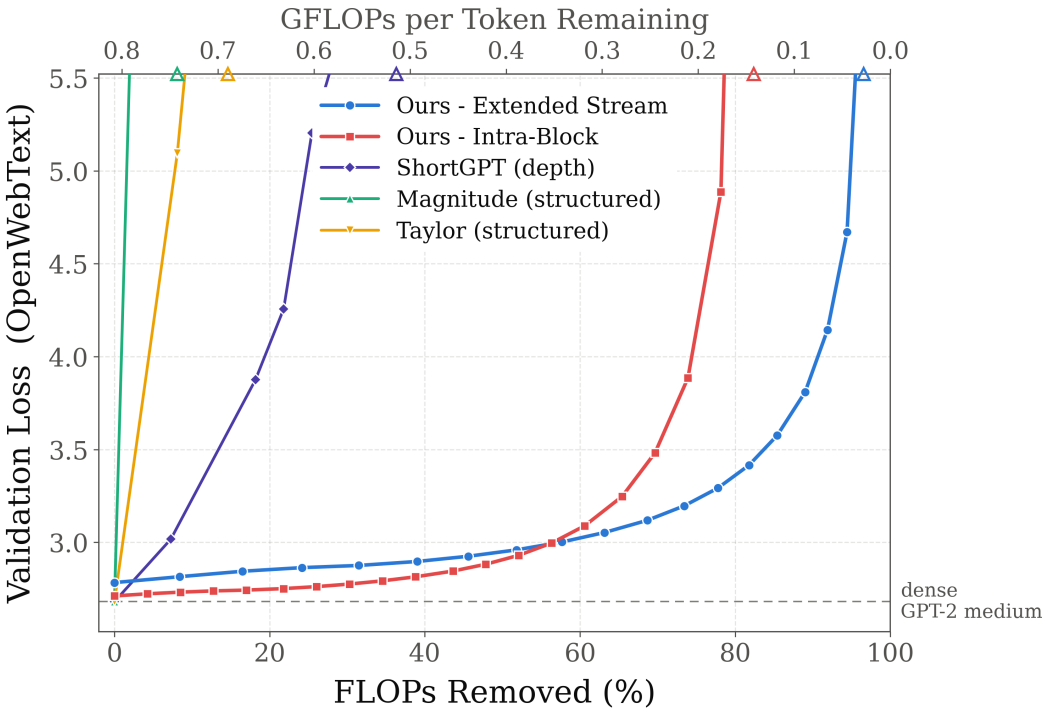



Figure 15: GPT-2 medium under structured pruning: OpenWebText validation loss across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 2.68. The vertical axis is capped at 2× the dense value; an open triangle on the top spine marks a curve leaving the axis.

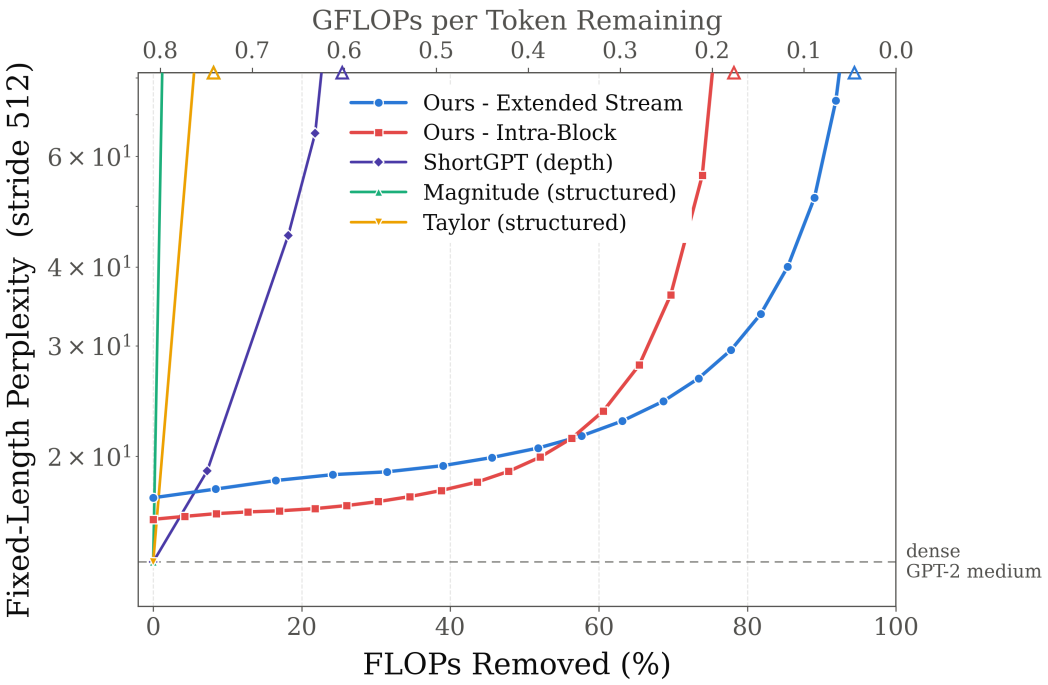



Figure 16: GPT-2 medium under structured pruning: fixed-length perplexity at stride 512 across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 13.59. The vertical axis is capped at 6× the dense value; an open triangle on the top spine marks a curve leaving the axis.

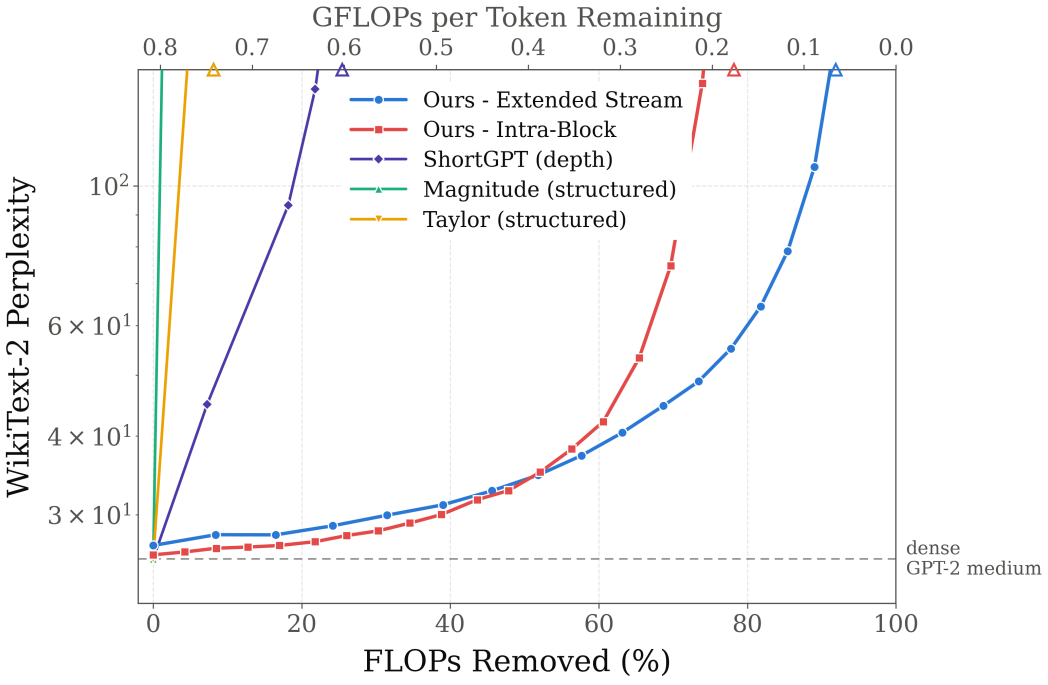



Figure 17: GPT-2 medium under structured pruning: WikiText-2 perplexity across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 25.51. The vertical axis is capped at 6× the dense value; an open triangle on the top spine marks a curve leaving the axis.

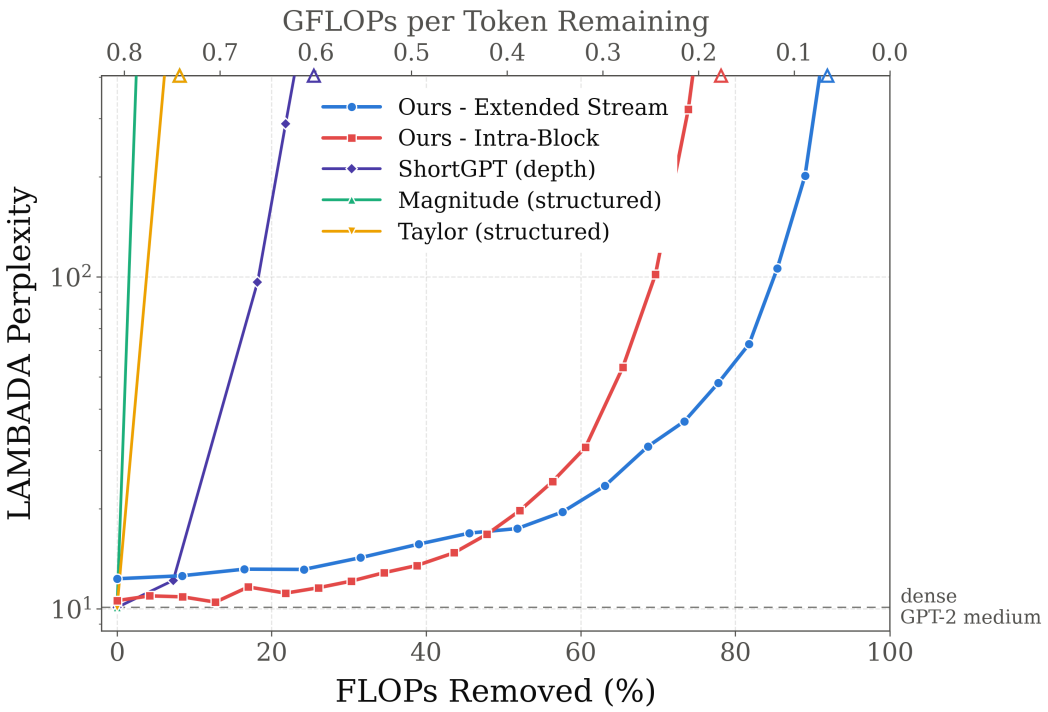



Figure 18: GPT-2 medium under structured pruning: LAMBADA (OpenAI split) perplexity across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 10.10. The vertical axis is capped at 40× the dense value; an open triangle on the top spine marks a curve leaving the axis.

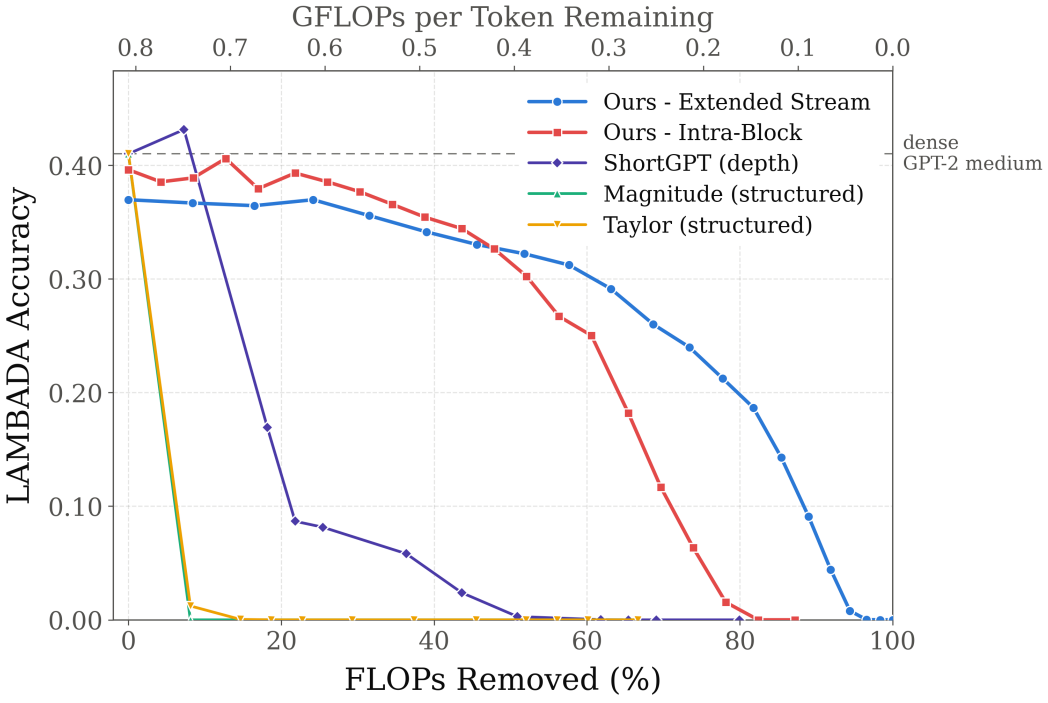



Figure 19: GPT-2 medium under structured pruning: LAMBADA last-word accuracy across the full sweep of compute budgets. Higher is better, and the dashed line is the dense reference at 0.410.

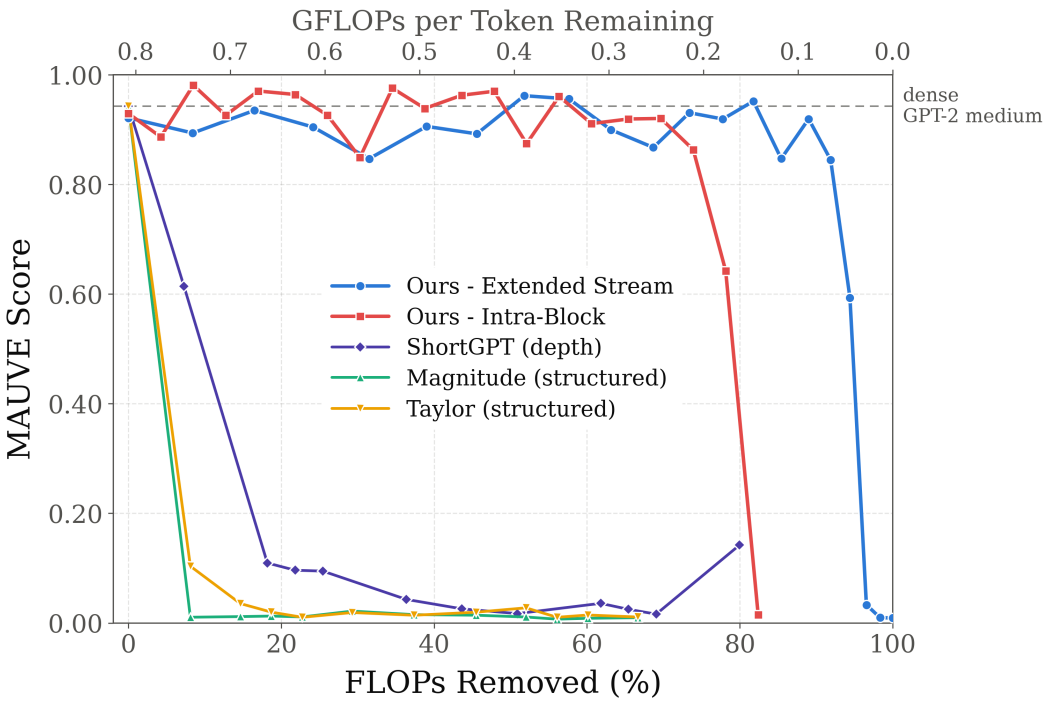



Figure 20: GPT-2 medium under structured pruning: the MAUVE generation-quality score across the full sweep of compute budgets. Higher is better, and the dashed line is the dense reference at 0.942. This is the cliff of Figure 2 restricted to the structured family: all three baselines have fallen below 0.15 by 20% of FLOPs removed, while the two scheduled models stay within 20% of the dense score until 76.0% (Intra-Block) and 92.8% (Extended Stream) of the FLOPs are gone. The deepest Intra-Block MAUVE reading, at 87.2% removed, is not plotted: it rebounds to 0.332 after the curve has already collapsed to 0.015 at 82.4%, which a connecting line would render as a recovery. The value is unchanged in the underlying data.

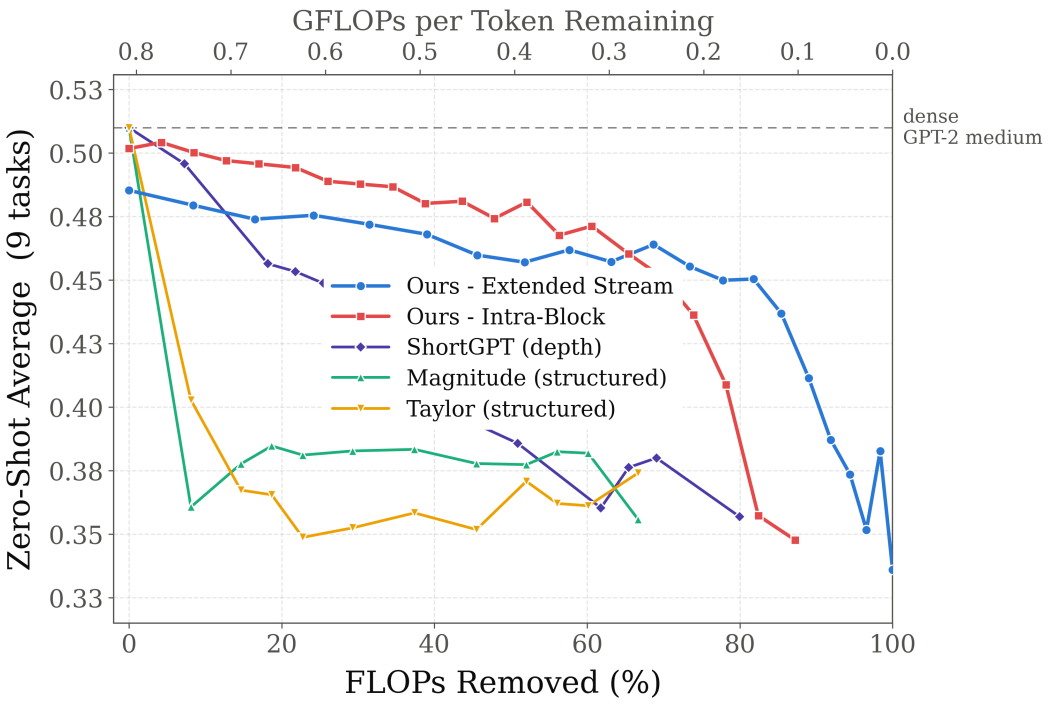



Figure 21: GPT-2 medium under structured pruning: mean accuracy over the nine zero-shot tasks across the full sweep of compute budgets. Higher is better, and the dashed line is the dense reference at 0.510.

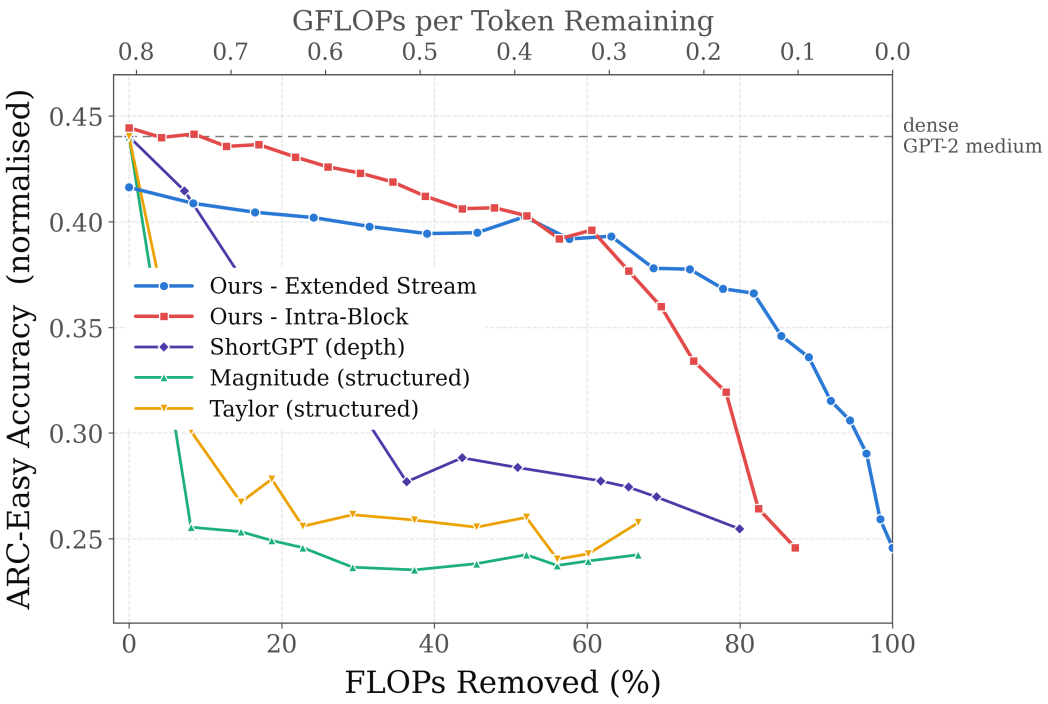



Figure 22: GPT-2 medium under structured pruning: length-normalised ARC-Easy accuracy, one of the five zero-shot tasks whose average is Figure 21. Higher is better, and the dashed line is the dense reference at 0.440. The vertical axis is fitted to the range of the data rather than starting at zero: chance is 0.25, so a zero baseline would spend most of the panel on empty space. This is a line rather than a bar, so position carries the value and no area is being compared; a drop should be read against the axis, not against the apparent size of the mark.

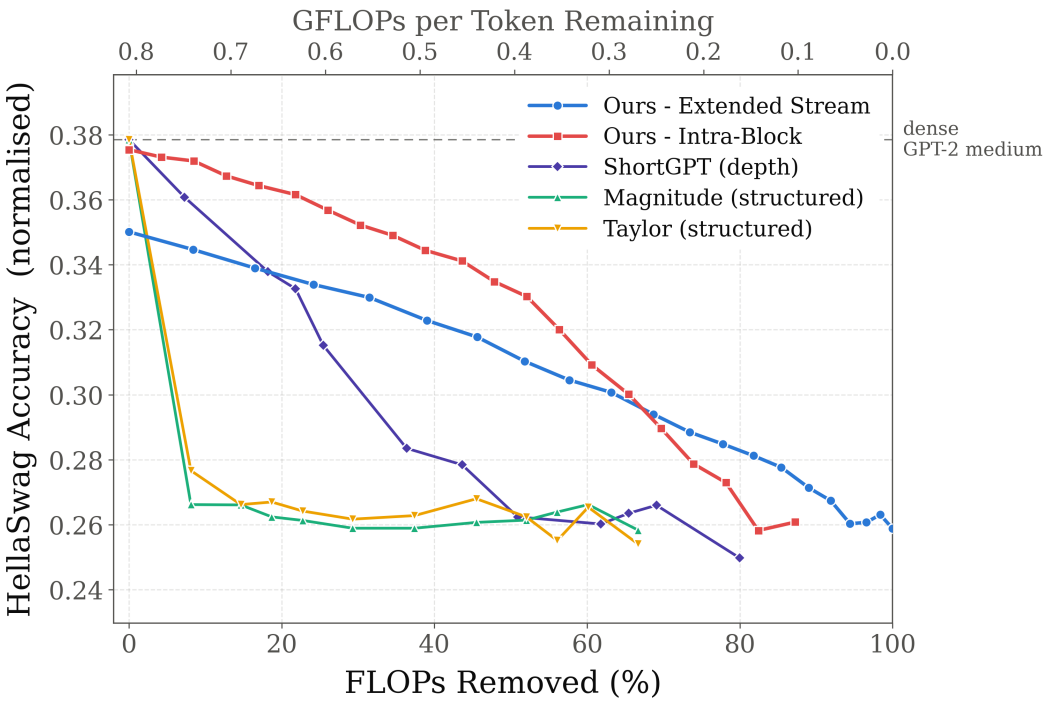



Figure 23: GPT-2 medium under structured pruning: length-normalised HellaSwag accuracy, one of the five zero-shot tasks whose average is Figure 21. Higher is better, and the dashed line is the dense reference at 0.379. The vertical axis is fitted to the range of the data rather than starting at zero: chance is 0.25, so a zero baseline would spend most of the panel on empty space.

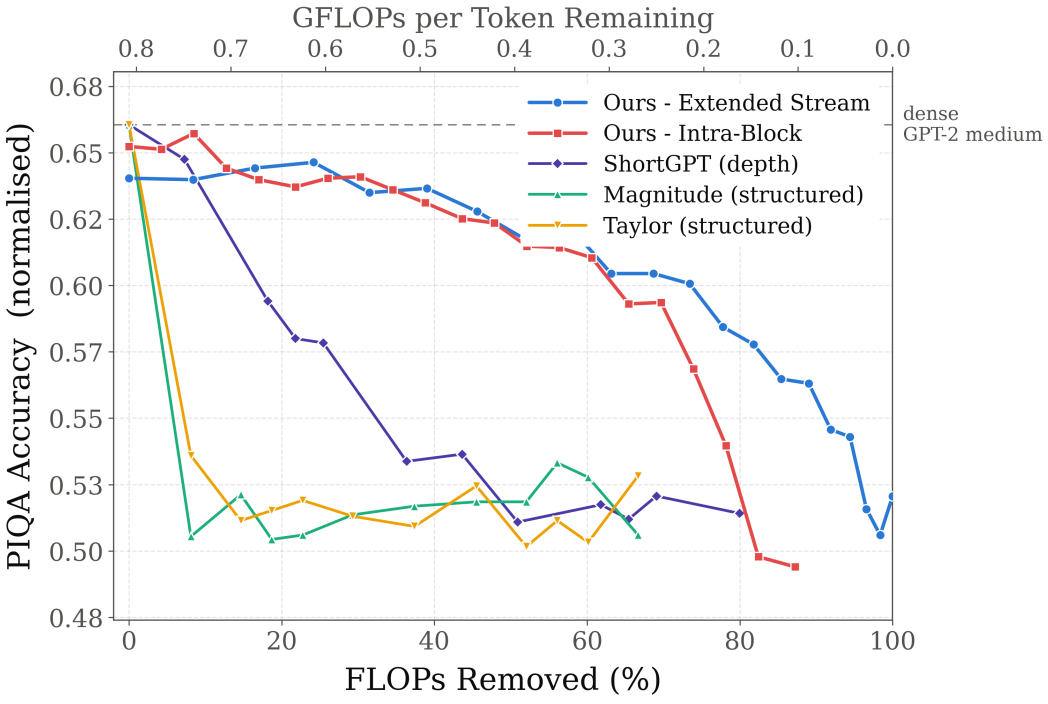



Figure 24: GPT-2 medium under structured pruning: length-normalised PIQA accuracy, one of the five zero-shot tasks whose average is Figure 21. Higher is better, and the dashed line is the dense reference at 0.661. The vertical axis is fitted to the range of the data rather than starting at zero: chance is 0.5, so a zero baseline would spend most of the panel on empty space.

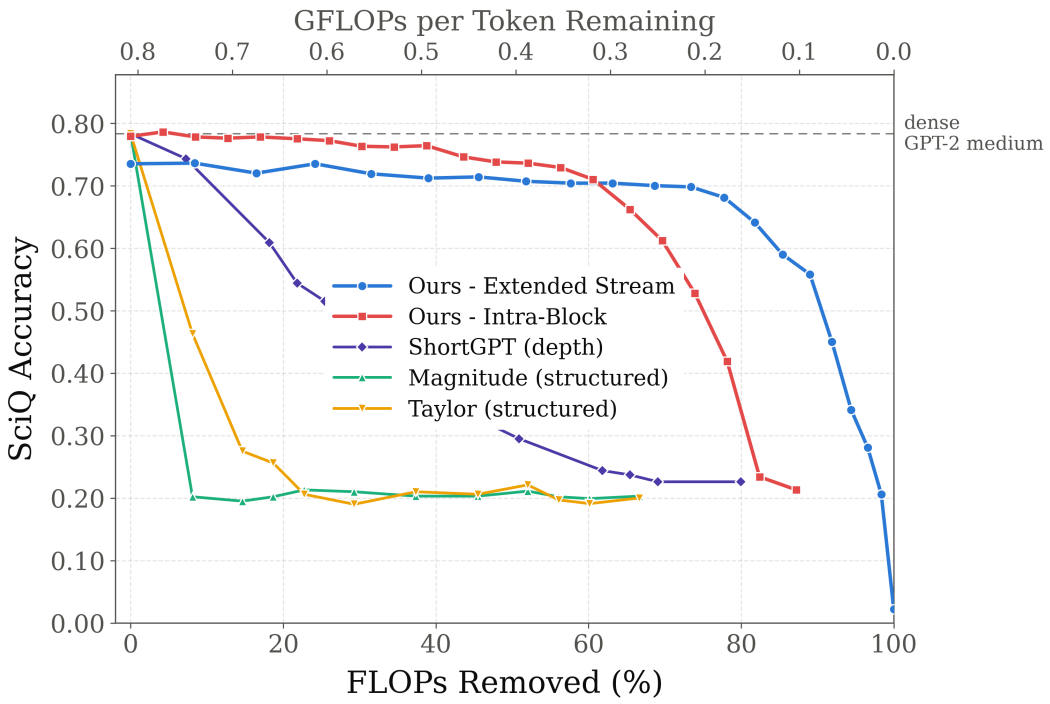



Figure 25: GPT-2 medium under structured pruning: SciQ accuracy, one of the five zero-shot tasks whose average is Figure 21. Higher is better, and the dashed line is the dense reference at 0.783. The vertical axis is fitted to the range of the data rather than starting at zero: chance is 0.25, so a zero baseline would spend most of the panel on empty space.

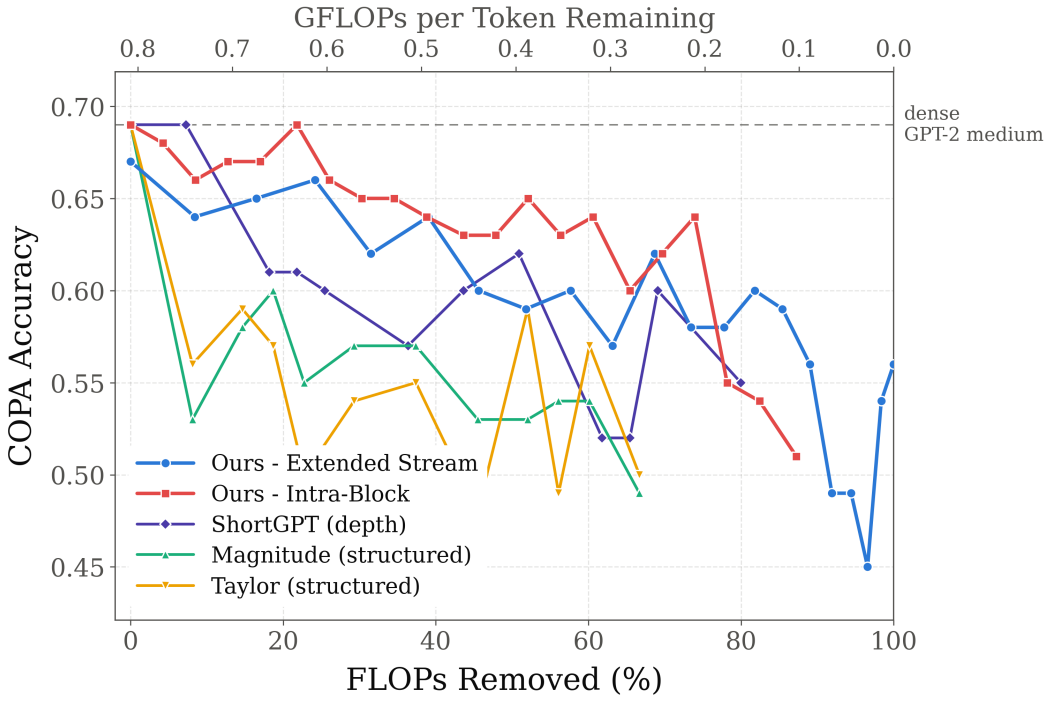



Figure 26: GPT-2 medium under structured pruning: COPA accuracy, one of the five zero-shot tasks whose average is Figure 21. Higher is better, and the dashed line is the dense reference at 0.690. The vertical axis is fitted to the range of the data rather than starting at zero: chance is 0.5, so a zero baseline would spend most of the panel on empty space.

# F GPT-2 Pruning Curves: Semi-Structured and Unstructured Methods

This section is the counterpart of Appendix E for the semi-structured and unstructured family, over the same twelve metrics, the same two scheduled models and the same axes. The baselines here are SparseGPT (Frantar & Alistarh, 2023), RIA (Zhang et al., 2024), Wanda (Sun et al., 2024), OWL (Yin et al., 2024), UniPruning (Ding et al., 2025) and magnitude pruning (Han et al., 2015) in their free-form variants, UniPruning (Ding et al., 2025; Mishra et al., 2021) as the semi-structured baseline, and a random control that sparsifies at the same rates without a saliency criterion. All eight are measured to 67.3% of the dense FLOPs removed, or 0.264 GFLOPs per token, while the Extended Stream model is measured to 100% and the Intra-Block model to 87.2%; as in the structured panels, no curve is drawn past its measured range.

The comparison these panels support is the one the main text makes in Table 1: this family is the stronger of the two at shallow budgets and still collapses well before the scheduled models do. Up to roughly 30% of FLOPs removed the baselines track the dense line closely, more closely than the scheduled checkpoints do, which start slightly below it. Between 30% and 45% removed they fall together, and by the common deepest budget of 67% the ordering is unambiguous. There the Extended Stream model reads a validation loss of 3.10 and the Intra-Block model 3.33, against 10.81 for SparseGPT as the best baseline and up to 14.42 for OWL across the seven saliency-based baselines. MAUVE reads 0.919 for Intra-Block and 0.877 for Extended Stream against 0.037 or below for those seven, and the nine-task zero-shot average reads 0.462 and 0.457 against 0.356–0.378, a band that sits close to the chance level of the constituent tasks.

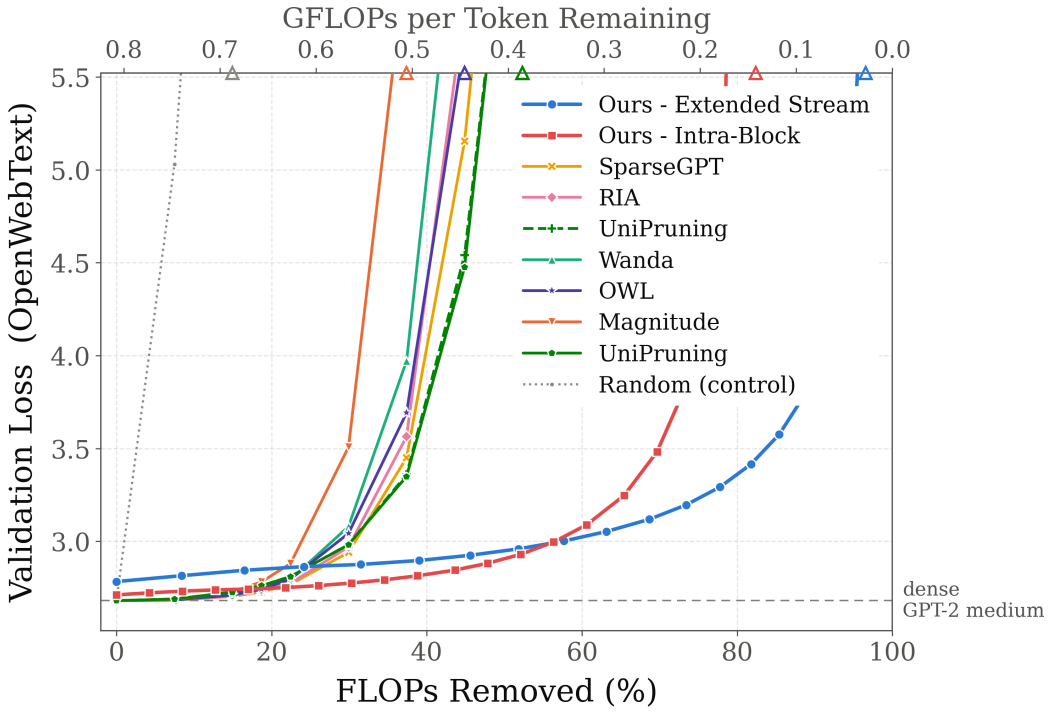



Figure 27: GPT-2 medium under semi-structured and unstructured pruning: OpenWebText validation loss across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 2.68. The axis cap (2× the dense value) and the open-triangle marker follow the convention introduced in Appendix E (see Figure 15). The seven saliency-based baselines leave the axis within a narrow band of budgets, between roughly 37% and 52% of FLOPs removed, which is the same cliff that Figure 32 shows on the MAUVE axis; the random control leaves it at about 15%.

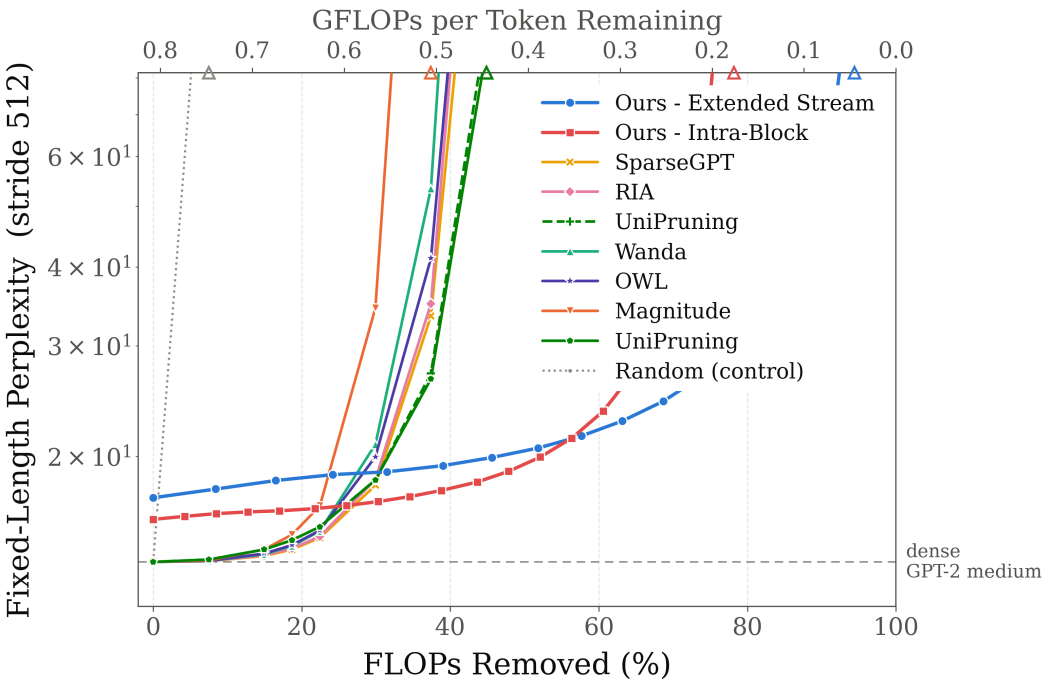



Figure 28: GPT-2 medium under semi-structured and unstructured pruning: fixed-length perplexity at stride 512 across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 13.59. The axis cap (6× the dense value) and the open-triangle marker follow the convention introduced in Appendix E.

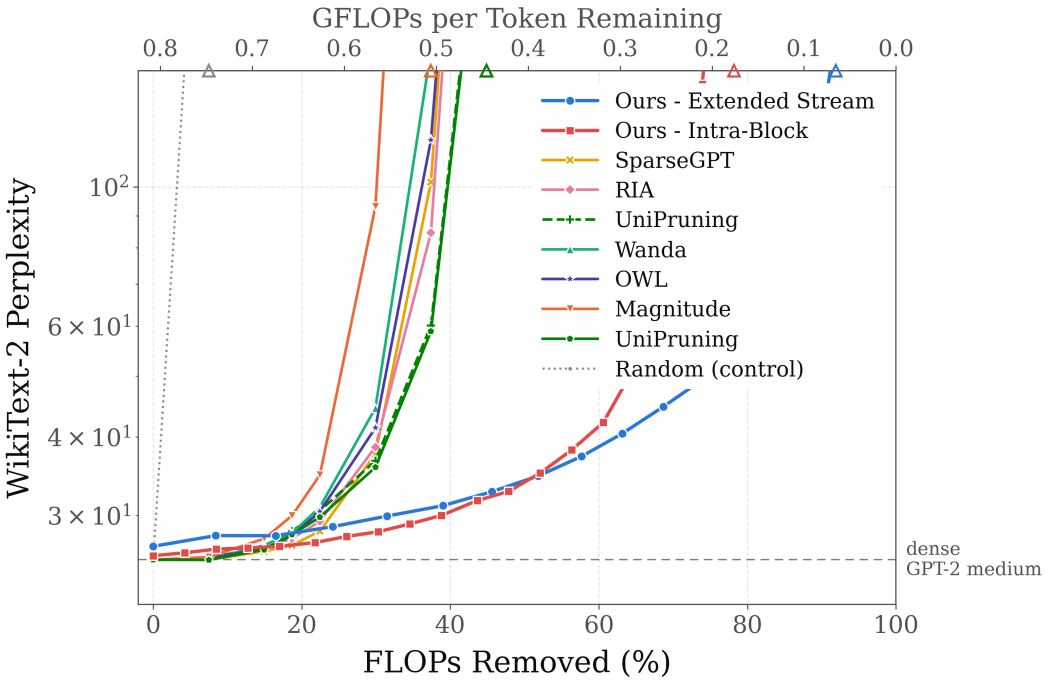



Figure 29: GPT-2 medium under semi-structured and unstructured pruning: WikiText-2 perplexity across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 25.51. The axis cap (6× the dense value) and the open-triangle marker follow the convention introduced in Appendix E.

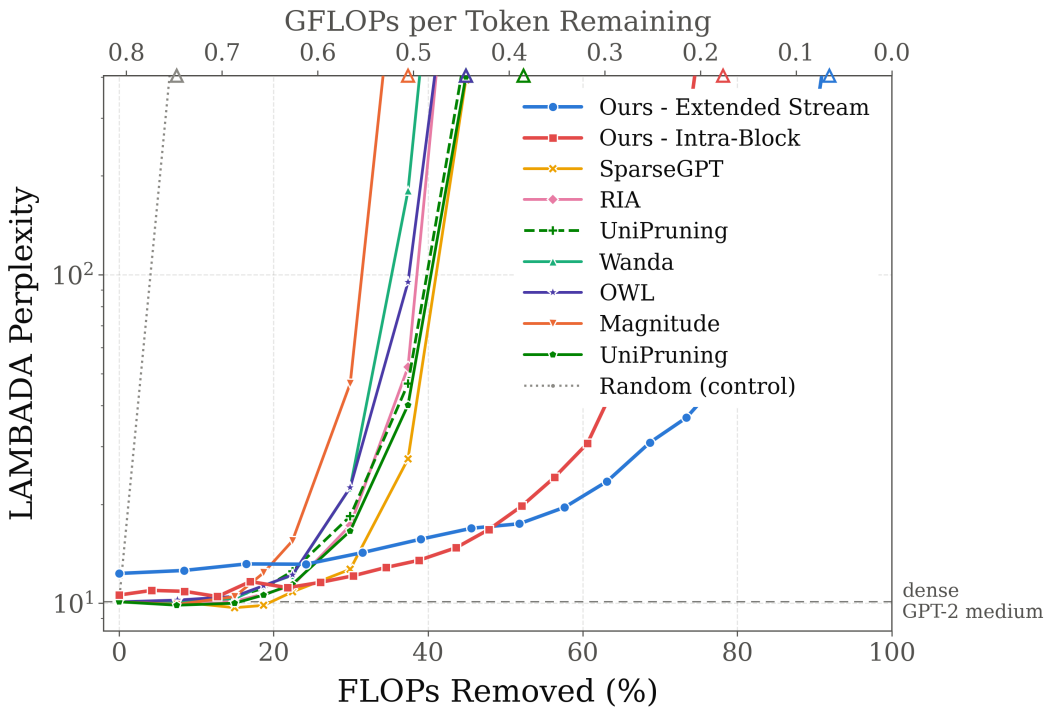



Figure 30: GPT-2 medium under semi-structured and unstructured pruning: LAMBADA (OpenAI split) perplexity across the full sweep of compute budgets. Lower is better, and the dashed line is the dense GPT-2 medium reference at 10.10. The axis cap (40× the dense value) and the open-triangle marker follow the convention introduced in Appendix E.

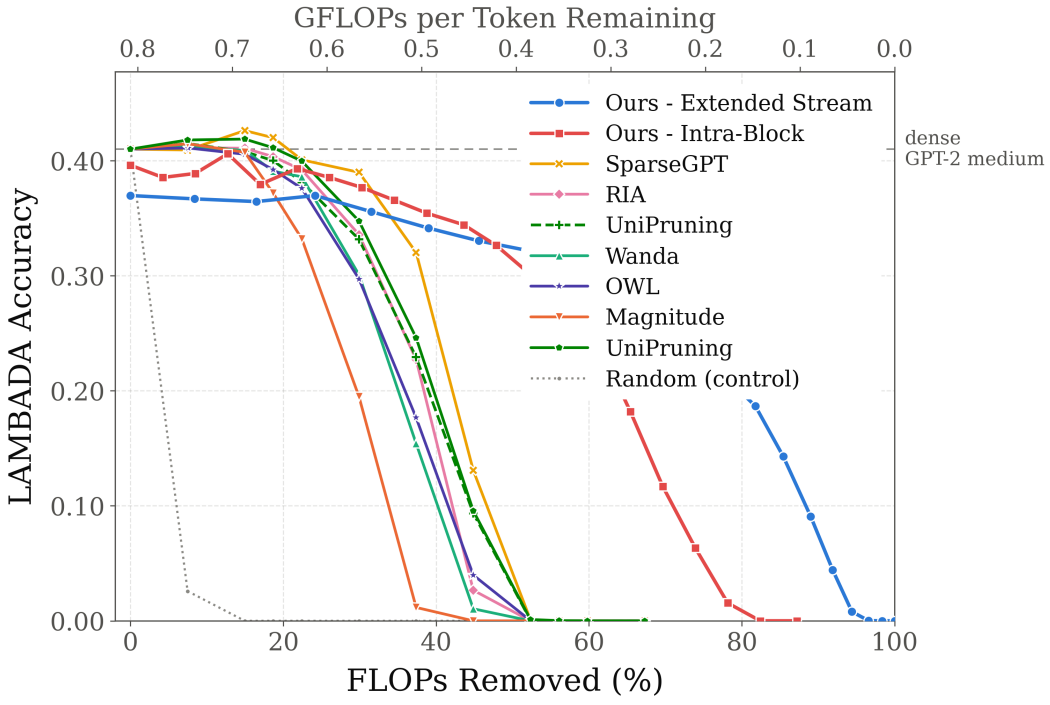



Figure 31: GPT-2 medium under semi-structured and unstructured pruning: LAMBADA last-word accuracy across the full sweep of compute budgets. Higher is better, and the dashed line is the dense reference at 0.410.

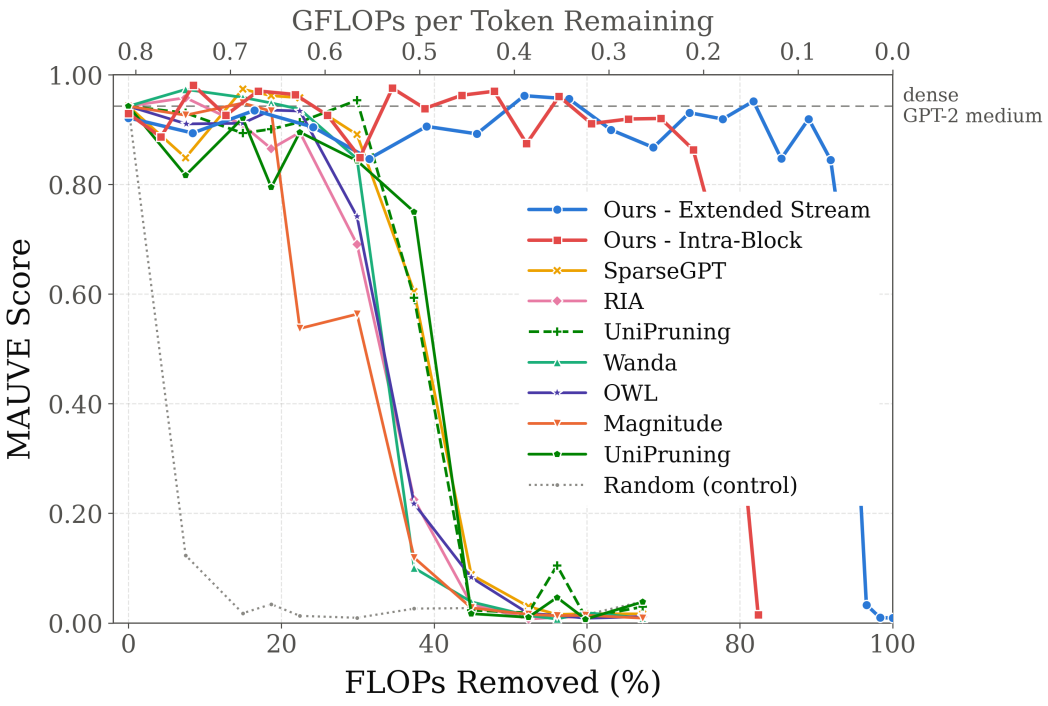



Figure 32: GPT-2 medium under semi-structured and unstructured pruning: the MAUVE generation-quality score across the full sweep of compute budgets. Higher is better, and the dashed line is the dense reference at 0.942. This panel makes the point of Section 4.1 on its own: the seven saliency-based baselines hold dense-level MAUVE and then fall from above 0.8 to below 0.05 within about fifteen percentage points of budget, between roughly 30% and 45% removed, whereas the two scheduled models cross the same band only beyond 78% (Intra-Block) and 94% (Extended Stream). The random control is already collapsed at the shallowest budgets, which fixes the floor these curves fall to. As in Figure 20, the deepest Intra-Block MAUVE reading at 87.2% removed is not plotted.

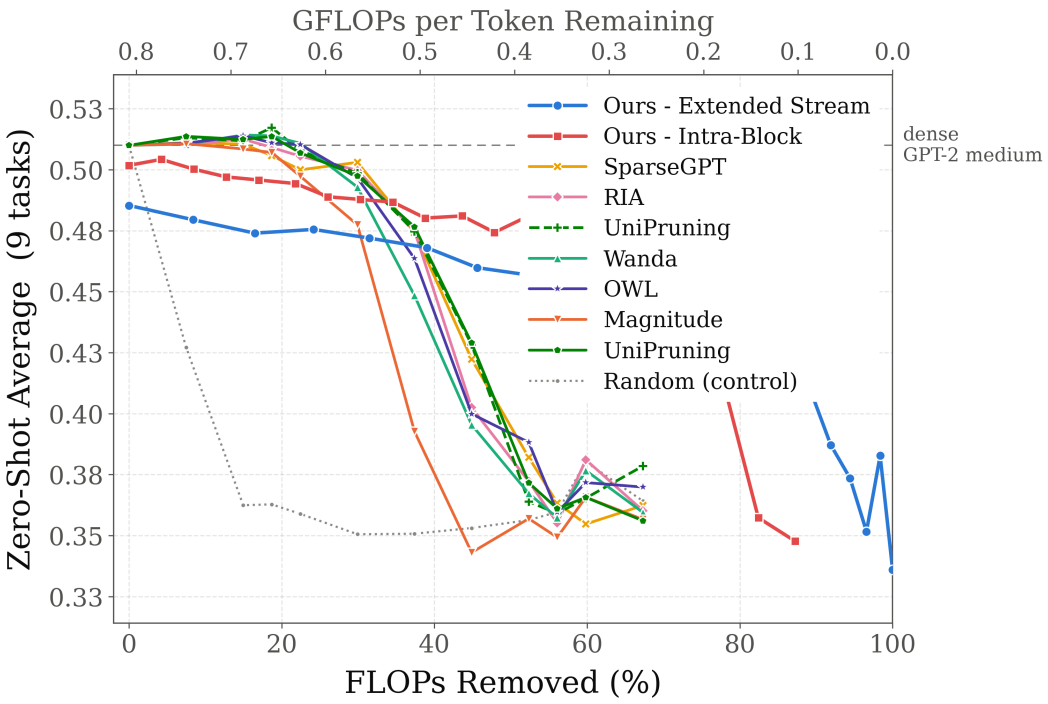



Figure 33: GPT-2 medium under semi-structured and unstructured pruning: mean accuracy over the nine zero-shot tasks across the full sweep of compute budgets. Higher is better, and the dashed line is the dense reference at 0.510.

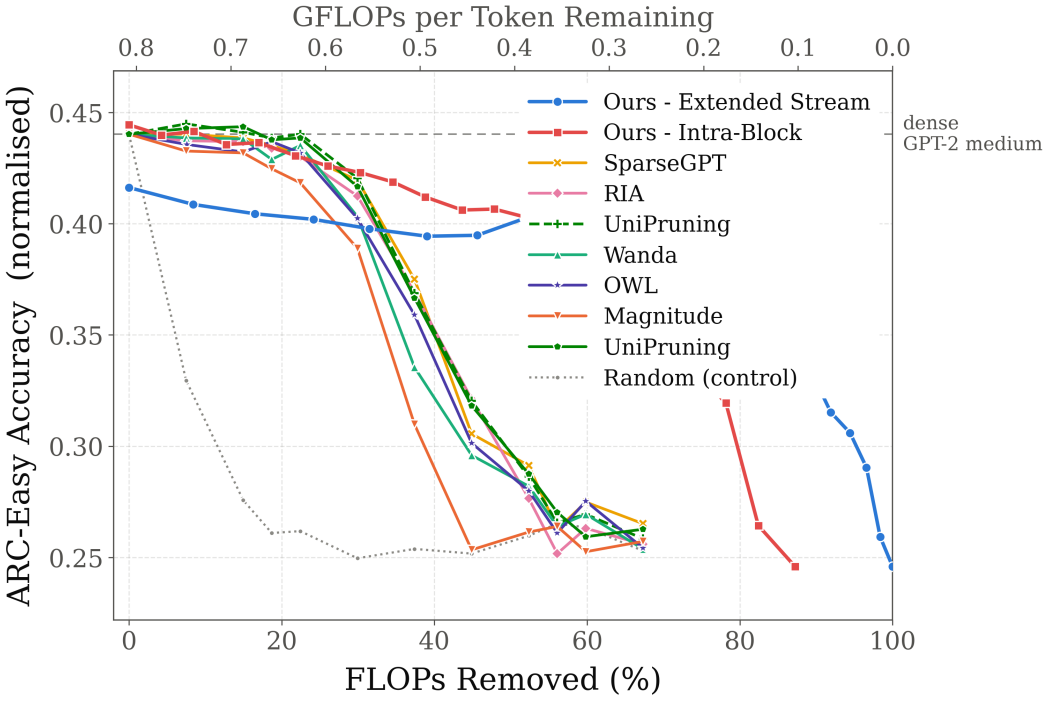



Figure 34: GPT-2 medium under semi-structured and unstructured pruning: length-normalised ARC-Easy accuracy, one of the five zero-shot tasks whose average is Figure 33. Higher is better, and the dashed line is the dense reference at 0.440. The vertical axis is fitted to the range of the data rather than starting at zero, as in Appendix E.

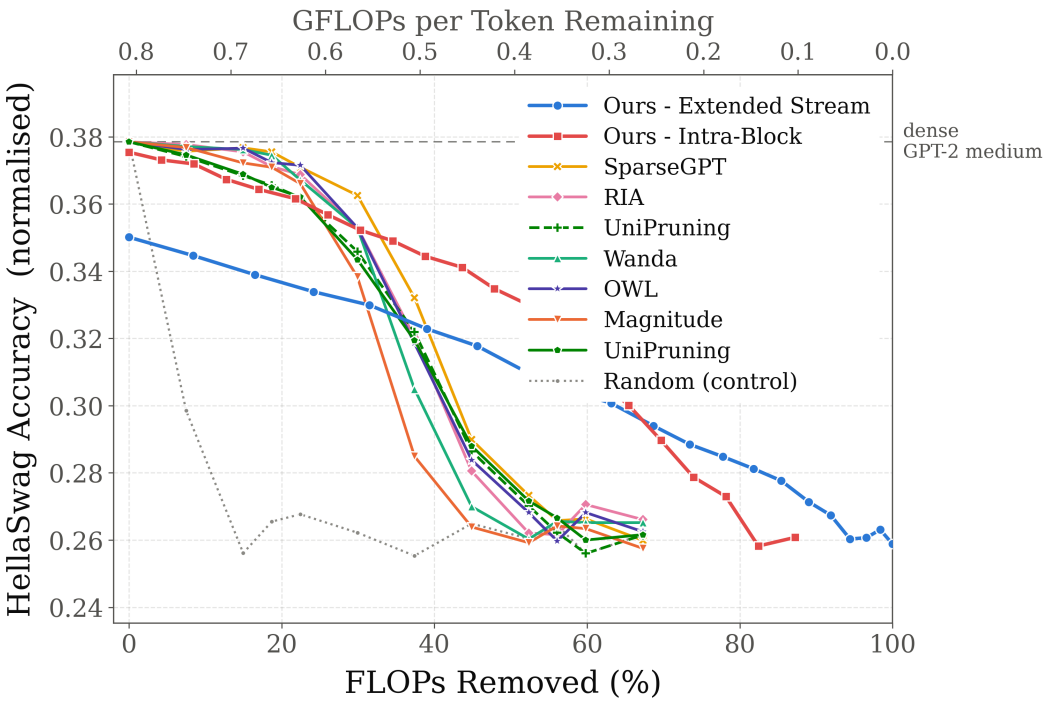



Figure 35: GPT-2 medium under semi-structured and unstructured pruning: length-normalised HellaSwag accuracy, one of the five zero-shot tasks whose average is Figure 33. Higher is better, and the dashed line is the dense reference at 0.379.

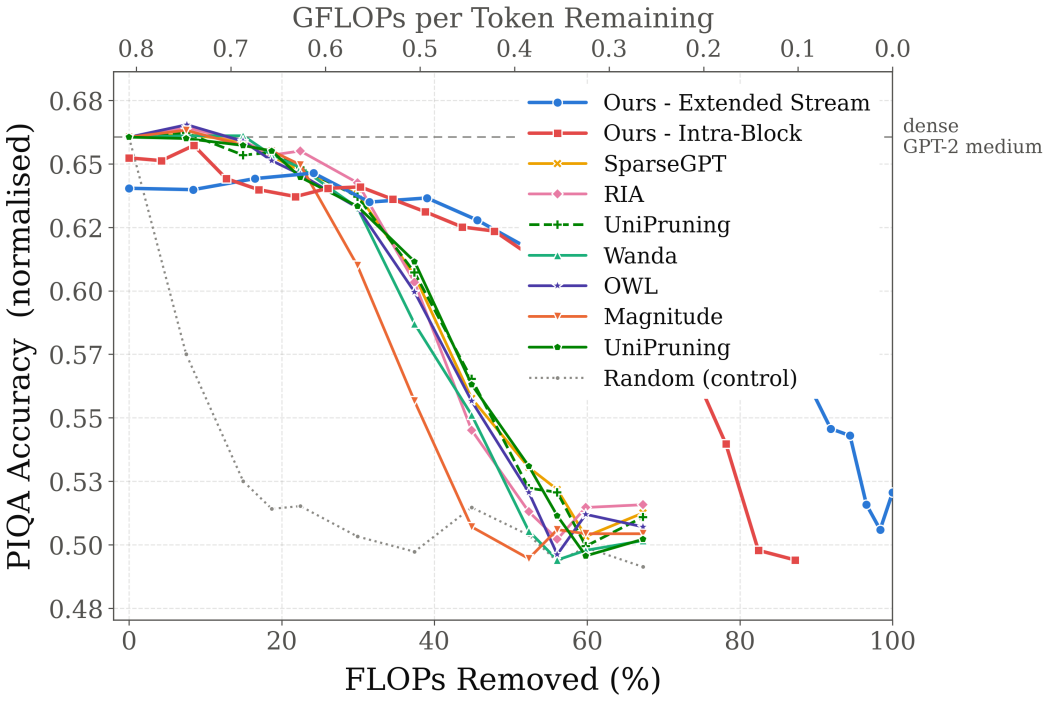



Figure 36: GPT-2 medium under semi-structured and unstructured pruning: length-normalised PIQA accuracy, one of the five zero-shot tasks whose average is Figure 33. Higher is better, and the dashed line is the dense reference at 0.661.

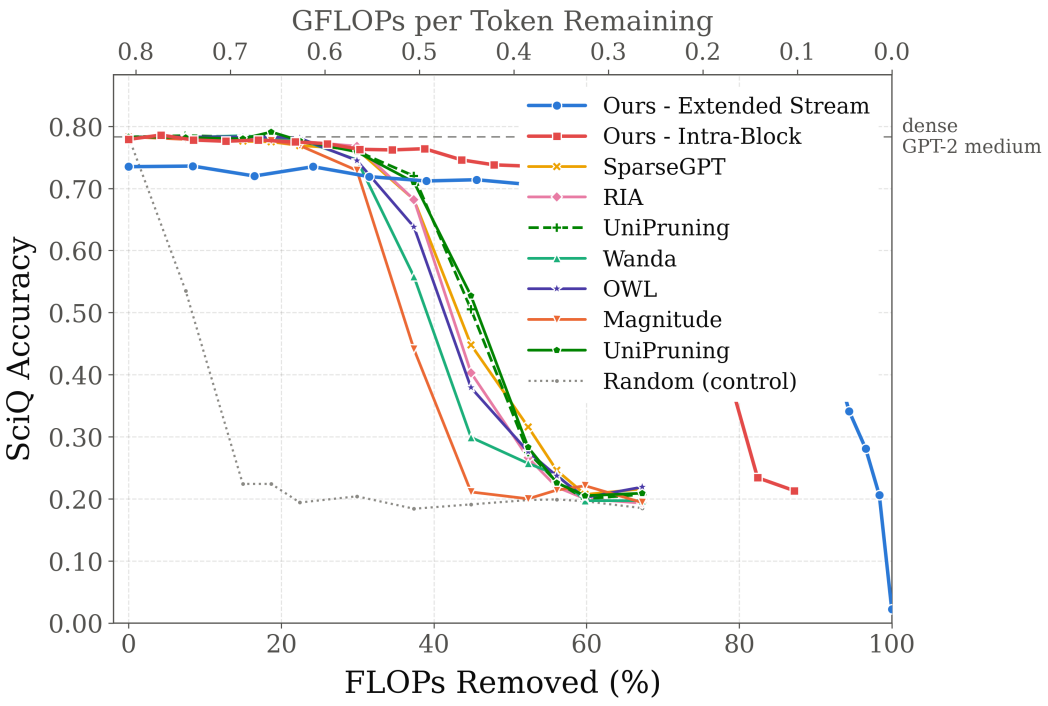



Figure 37: GPT-2 medium under semi-structured and unstructured pruning: SciQ accuracy, one of the five zero-shot tasks whose average is Figure 33. Higher is better, and the dashed line is the dense reference at 0.783. SciQ separates the methods most sharply of the five tasks, because it is the only one whose dense accuracy sits far enough above chance for a collapse to be visible over a wide range.

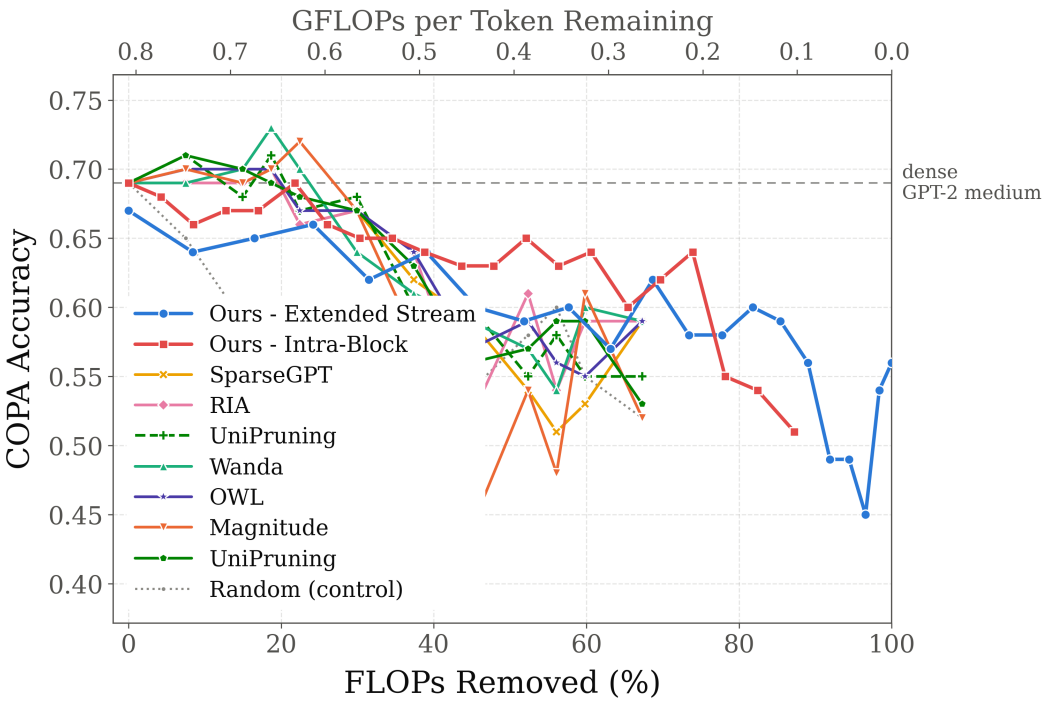



Figure 38: GPT-2 medium under semi-structured and unstructured pruning: COPA accuracy, one of the five zero-shot tasks whose average is Figure 33. Higher is better, and the dashed line is the dense reference at 0.690. COPA separates the methods least of the five tasks, since its dense accuracy is already close to the 0.5 chance level of a binary choice.

## G RESNET-50 DENSE EVALUATION, UNIFORM PRUNING TABLE AND PER-VARIANT BUDGET BARS

For completeness we report the dense validation accuracy of the three ResNet-50 checkpoints, the full uniform pruning table, and the per-variant accuracy budget bars under uniform pruning, i.e. the maximum MAC reduction each model tolerates within a 1%, 2%, or 5% Top-1 accuracy drop. Table 4 lists the dense accuracy of the three checkpoints, and Table 5 reports the Top-1 and Top-5 accuracy at the MAC reduction levels discussed in Section 4.1. Figures 40–42 are the full-resolution, panel-wise counterparts of the aggregated comparison in Figure 39 and of Table 6. Figures 43 and 44 likewise give the per-variant accuracy versus MACs curves that are combined into a single plot in Figure 6.

Table 4: Evaluation of the ResNet-50 configurations on the validation set.

| **Model** | **Loss** | **Top-1 (%)** | **Top-5 (%)** | **ΔTop-1** | **ΔTop-5** |
|---|---|---|---|---|---|
| Baseline | 2.17 | 72.02 | 89.95 | 0.00 | 0.00 |
| Act. Schedule - Intra-Block | 2.28 | 70.49 | 89.23 | -1.53 | -0.73 |
| Act. Schedule - Extended Stream | 2.28 | 70.37 | 89.17 | -1.65 | -0.78 |

Table 5: Top-1 and Top-5 accuracy (%) under Uniform Pruning at various MAC reduction levels.

| **Model** | **MACs Removed (%)** | **Top-1 (%)** | **Top-5 (%)** |
|---|---|---|---|
| *Intra-Block Gating vs. Baseline* | | | |
| Baseline | 0.00 | 72.03 | 89.95 |
| Act. Schedule - Intra-Block | 0.00 | 70.49 | 89.22 |
| Baseline | 24.45 | 48.05 | 71.32 |
| Act. Schedule - Intra-Block | 24.45 | 70.03 | 88.91 |
| Baseline | 55.44 | 1.89 | 5.86 |
| Act. Schedule - Intra-Block | 55.44 | 67.94 | 87.53 |
| Baseline | 77.68 | 0.18 | 0.81 |
| Act. Schedule - Intra-Block | 77.68 | 45.74 | 70.23 |
| *Extended Stream Gating vs. Baseline* | | | |
| Baseline | 0.00 | 72.02 | 89.95 |
| Act. Schedule - Extended Stream | 0.00 | 70.37 | 89.17 |
| Baseline | 27.67 | 28.61 | 50.21 |
| Act. Schedule - Extended Stream | 27.67 | 69.39 | 88.42 |
| Baseline | 60.47 | 0.90 | 3.15 |
| Act. Schedule - Extended Stream | 60.47 | 64.62 | 85.28 |
| Baseline | 80.94 | 0.38 | 1.49 |
| Act. Schedule - Extended Stream | 80.94 | 27.55 | 50.91 |

Table 6: Maximum MAC reduction (%) achievable within a 1%, 2%, and 5% Top-1 accuracy drop budget under uniform pruning.

| **Model** | **$\leq$ 1% Drop** | **$\leq$ 2% Drop** | **$\leq$ 5% Drop** |
|---|---|---|---|
| *Intra-Block Gating vs. Baseline* | | | |
| Baseline | 0.00 | 0.00 | 6.21 |
| Act. Schedule - Intra-Block | 35.69 | 50.56 | 63.53 |
| *Extended Stream Gating vs. Baseline* | | | |
| Baseline | 0.00 | 0.00 | 0.00 |
| Act. Schedule - Extended Stream | 27.67 | 39.91 | 55.57 |

## H RESNET-50 GRADED PRUNING RESULTS

In addition to the uniform pruning strategy, we investigate a graded pruning approach where the effective drop ratio is allocated aggressively to earlier layers and monotonically decayed deeper in the network. This evaluates the hypothesis that lower-level features are more compressible than deep, task-specific features.

Figures 45 and 46 and Table 7 summarize the Top-1 accuracy under this strategy. The results demonstrate that the scheduled models continue to significantly outperform the Baseline. However,

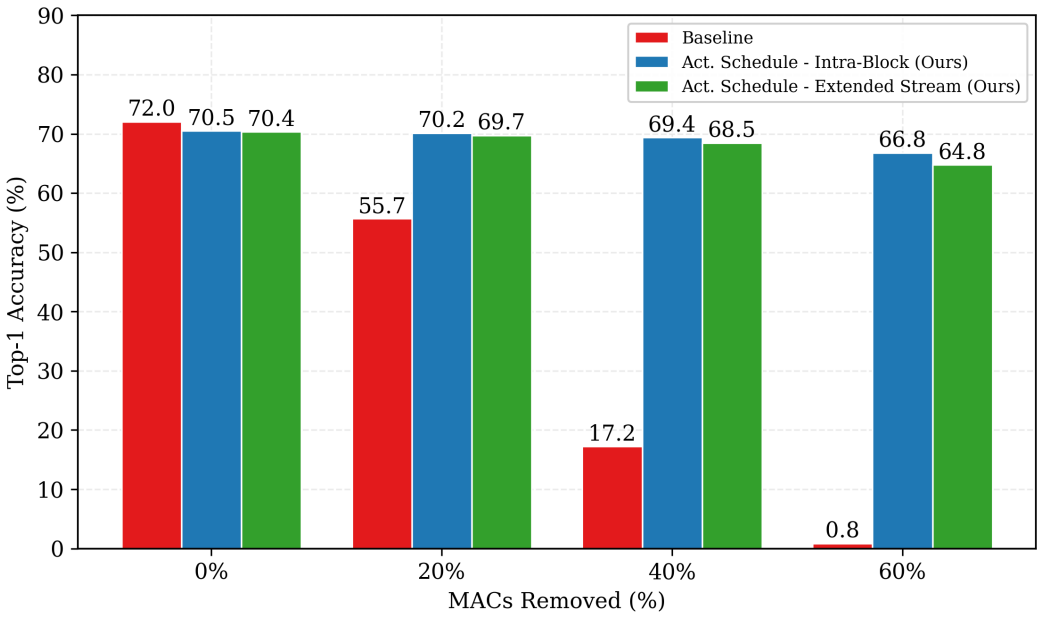



Figure 39: Top-1 validation accuracy of the three ResNet-50 checkpoints at fixed MAC reduction levels under uniform pruning. The 0% group is the dense evaluation; the 20%, 40%, and 60% groups are interpolated on each model's own pruning sweep. Both scheduled variants stay within a few points of their dense accuracy, whereas the Baseline degrades sharply.

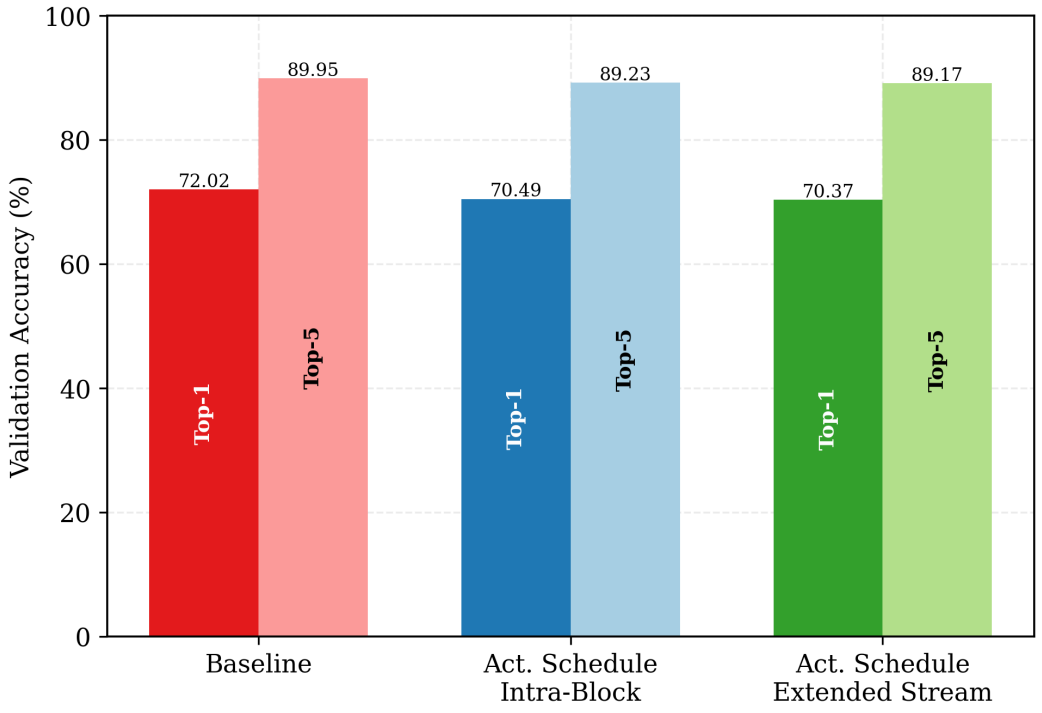



Figure 40: Top-1 and top-5 validation accuracy comparison of the three ResNet-50 checkpoints.

the graded strategy removes fewer total MACs for the same maximal drop ratio compared to uniform pruning, leading to a different accuracy-efficiency trade-off curve.

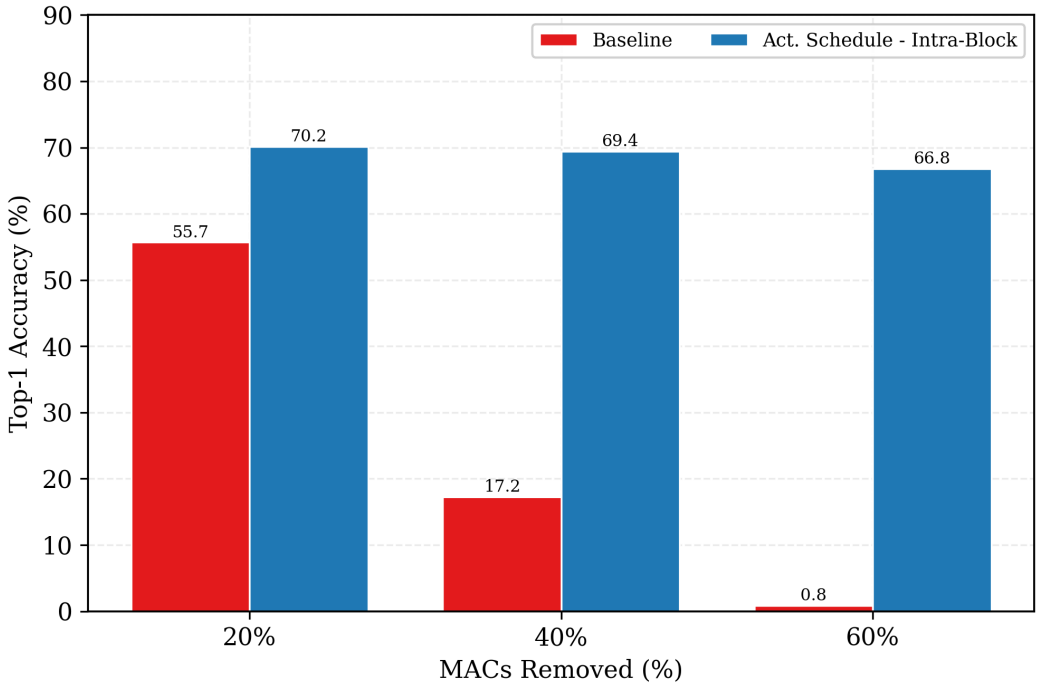



Figure 41: MAC reduction achievable within 1%, 2%, and 5% Top-1 accuracy degradation budgets under uniform pruning for Intra-Block gating against the baseline.

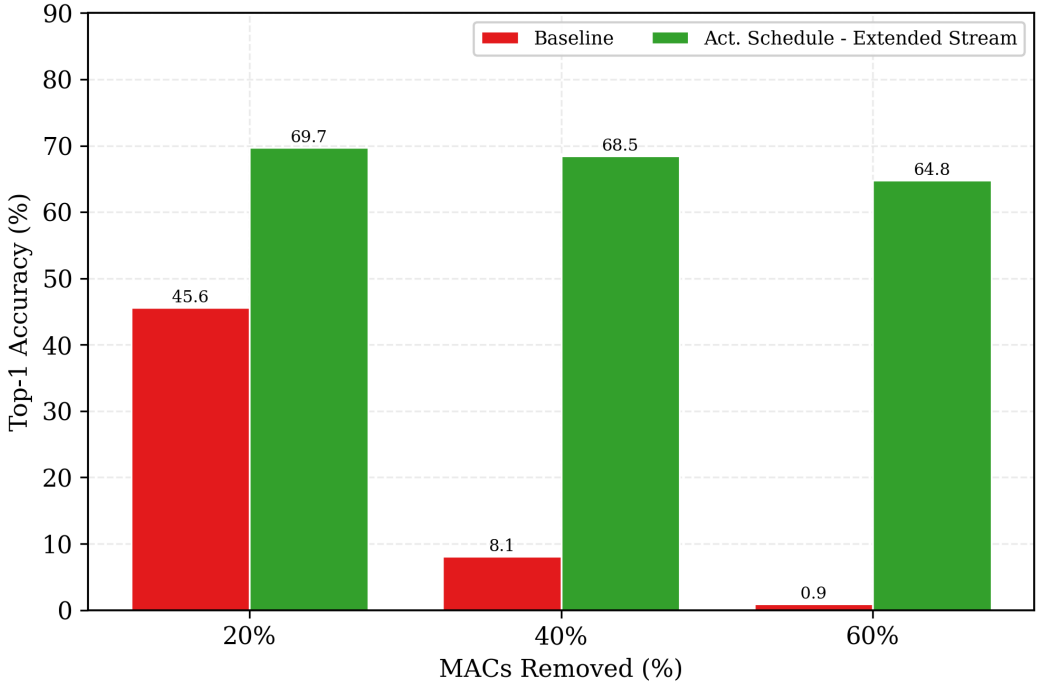



Figure 42: MAC reduction achievable within 1%, 2%, and 5% Top-1 accuracy degradation budgets under uniform pruning for Extended Stream gating against the baseline.

### H.1 Accuracy Budget under Graded Pruning

Similar to the uniform pruning analysis, we quantify the MAC reduction achievable within fixed accuracy degradation budgets under the graded pruning strategy. Table 8 and Figures 47 and 48

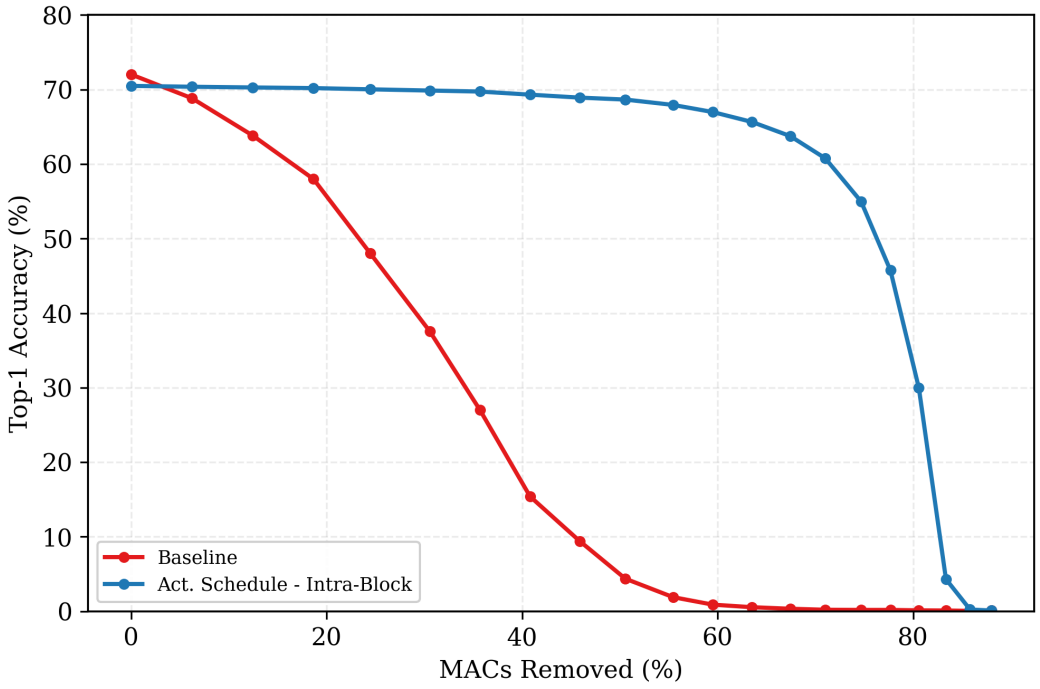



Figure 43: Top-1 validation accuracy versus percentage of MACs removed under uniform pruning for Intra-Block gating against the baseline.

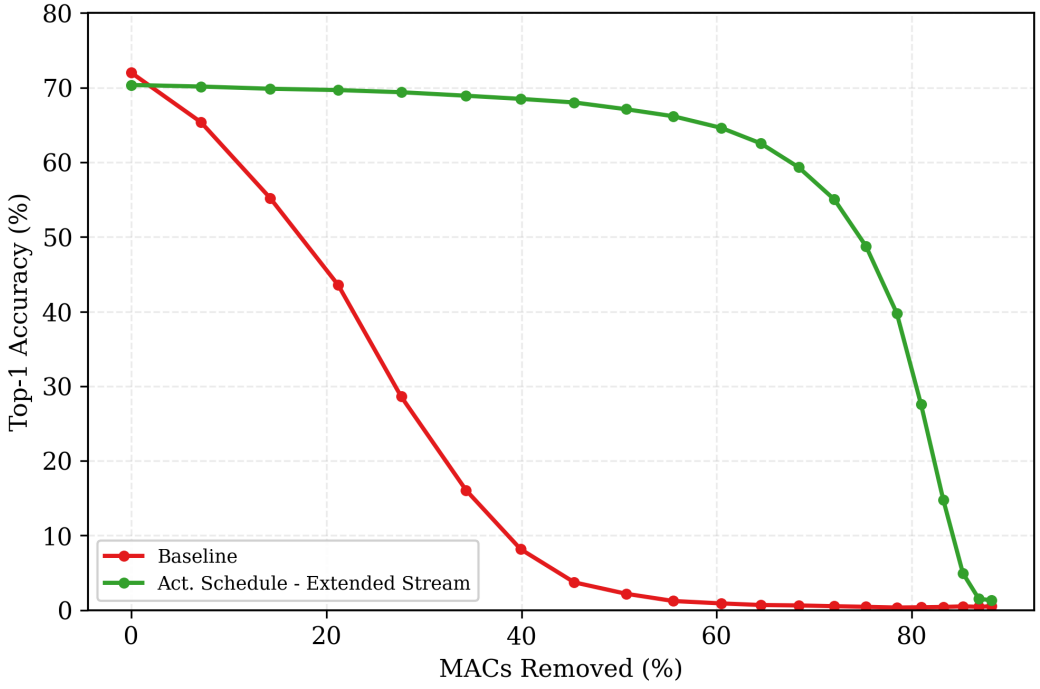



Figure 44: Top-1 validation accuracy versus percentage of MACs removed under uniform pruning for Extended Stream gating against the baseline.

present the maximum MAC reduction for 1%, 2%, and 5% Top-1 accuracy drops. The scheduled models consistently allow for significant MAC reduction compared to the Baseline. For example, the Intra-Block model maintains within a 1% drop up to a 38.59% MAC reduction, whereas the Baseline tolerates no reduction at this budget and only 9.90% within a 5% drop budget.

Table 7: Top-1 Accuracy (%) under Graded Pruning at various MAC reduction levels.

| Model | MACs Removed (%) | Top-1 (%) |
|---|---|---|
| *Intra-Block Gating vs. Baseline* | | |
| Baseline | 0.00 | 72.03 |
| Act. Schedule - Intra-Block | 0.00 | 70.49 |
| Baseline | 31.09 | 41.59 |
| Act. Schedule - Intra-Block | 31.09 | 69.97 |
| Baseline | 45.58 | 4.73 |
| Act. Schedule - Intra-Block | 45.58 | 68.81 |
| Baseline | 53.90 | 1.13 |
| Act. Schedule - Intra-Block | 53.90 | 44.93 |
| *Extended Stream Gating vs. Baseline* | | |
| Baseline | 0.00 | 72.02 |
| Act. Schedule - Extended Stream | 0.00 | 70.37 |
| Baseline | 34.60 | 25.84 |
| Act. Schedule - Extended Stream | 34.60 | 69.56 |
| Baseline | 49.45 | 4.92 |
| Act. Schedule - Extended Stream | 49.45 | 67.51 |
| Baseline | 57.25 | 2.10 |
| Act. Schedule - Extended Stream | 57.25 | 58.67 |

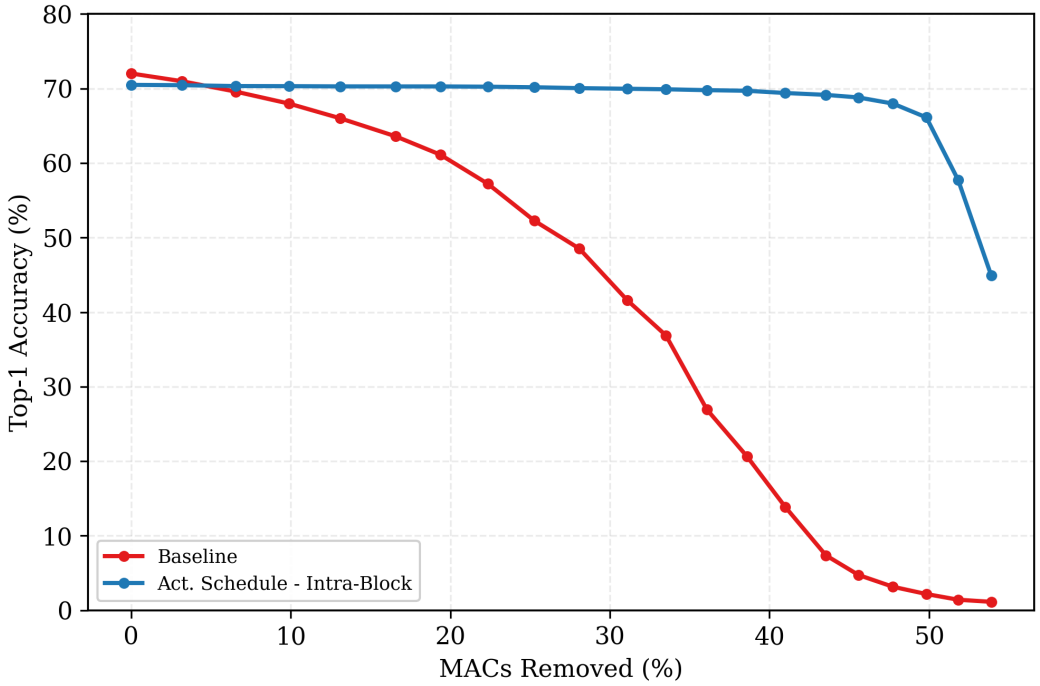



Figure 45: Top-1 validation accuracy versus percentage of MACs removed under graded pruning for Intra-Block gating against the baseline.

# I ResNet-50 Top-5 Accuracy Results

In this section, we present the Top-5 accuracy results for both uniform and graded pruning strategies. The Top-5 metrics provide further insight into the models' robustness under extreme sparsity, complementing the Top-1 accuracy findings.

## I.1 Uniform Pruning (Top-5)

Figures 49 and 50 and Table 9 summarize the Top-5 accuracy of the scheduled models compared to the baseline under uniform pruning. The Layer Activation Schedule models exhibit superior

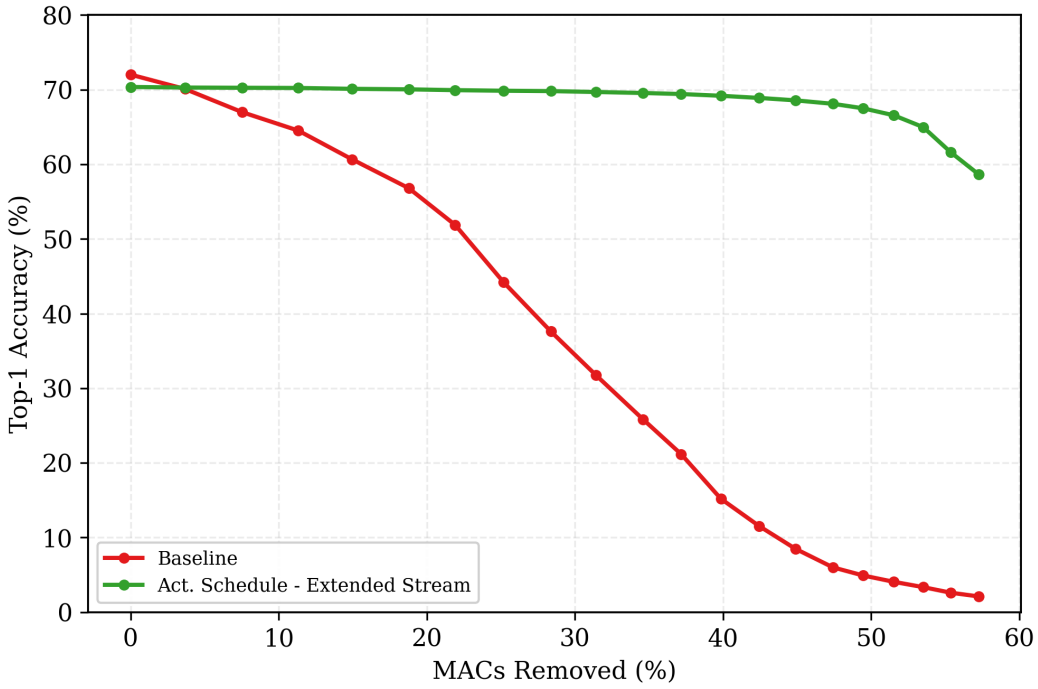



Figure 46: Top-1 validation accuracy versus percentage of MACs removed under graded pruning for Extended Stream gating against the baseline.

Table 8: Maximum MAC reduction (%) achievable within a 1%, 2%, and 5% Top-1 accuracy drop budget under graded pruning.

| **Model** | **$\leq$ 1% Drop** | **$\leq$ 2% Drop** | **$\leq$ 5% Drop** |
|---|---|---|---|
| *Intra-Block Gating vs. Baseline* | | | |
| Baseline | 0.00 | 3.17 | 9.90 |
| Act. Schedule - Intra-Block | 38.59 | 45.58 | 49.83 |
| *Extended Stream Gating vs. Baseline* | | | |
| Baseline | 0.00 | 3.68 | 3.68 |
| Act. Schedule - Extended Stream | 37.17 | 44.89 | 51.54 |

preservation of Top-5 accuracy at high MAC reduction levels. For instance, at approximately 55% MAC reduction, the Intra-Block model maintains a Top-5 accuracy of 87.53%, whereas the Baseline drops to 5.86%. Similarly, the Extended Stream architecture preserves 85.28% accuracy at a 60.47% MAC reduction compared to the Baseline's 3.15%.

Table 10 outlines the maximum MAC reduction achievable without losing more than 1%, 2%, or 5% of Top-5 accuracy relative to each model's unpruned performance. The scheduled models can be pruned heavily with minimal Top-5 accuracy degradation.

### I.2 Graded Pruning (Top-5)

Similarly, Figures 53 and 54 and Table 11 present the Top-5 accuracy for the graded pruning strategy. The graded strategy further demonstrates the resilience of the activation schedule.

The maximum MAC reduction thresholds for the graded pruning under Top-5 accuracy are detailed in Table 12. The models tolerate extreme structured reduction, showcasing that early layers can be sparsified aggressively without compromising final Top-5 classification.

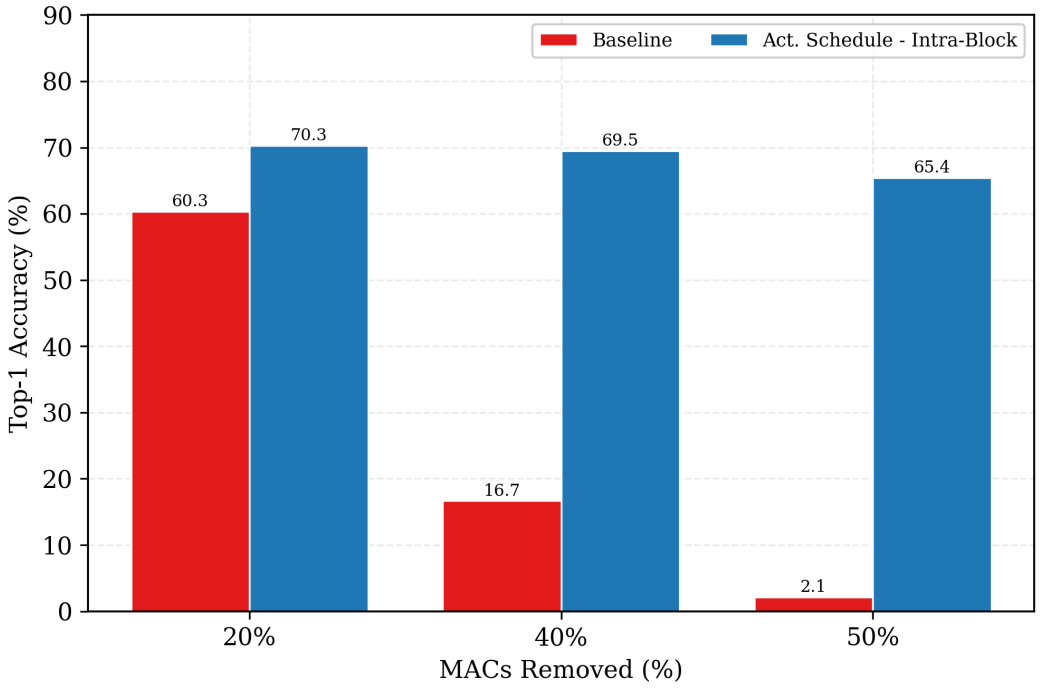



Figure 47: MAC reduction achievable within 1%, 2%, and 5% accuracy degradation budgets under graded pruning for Intra-Block gating against the baseline.

Table 9: Top-5 Accuracy (%) under Uniform Pruning at various MAC reduction levels.

| **Model** | **MACs Removed (%)** | **Top-5 (%)** |
|---|---|---|
| *Intra-Block Gating vs. Baseline* | | |
| Baseline | 0.00 | 89.95 |
| Act. Schedule - Intra-Block | 0.00 | 89.22 |
| Baseline | 24.45 | 71.32 |
| Act. Schedule - Intra-Block | 24.45 | 88.91 |
| Baseline | 55.44 | 5.86 |
| Act. Schedule - Intra-Block | 55.44 | 87.53 |
| Baseline | 77.68 | 0.81 |
| Act. Schedule - Intra-Block | 77.68 | 70.23 |
| *Extended Stream Gating vs. Baseline* | | |
| Baseline | 0.00 | 89.95 |
| Act. Schedule - Extended Stream | 0.00 | 89.17 |
| Baseline | 27.67 | 50.21 |
| Act. Schedule - Extended Stream | 27.67 | 88.42 |
| Baseline | 60.47 | 3.15 |
| Act. Schedule - Extended Stream | 60.47 | 85.28 |
| Baseline | 80.94 | 1.49 |
| Act. Schedule - Extended Stream | 80.94 | 50.91 |

## J ResNet-50 Channel Activation Structure

This section completes the channel analysis of Section 4.2. It compares the Baseline (left column of each figure) with the Extended Stream model (right column). All statistics are measured at the post-residual ReLU of every bottleneck block and averaged over all spatial positions of the ImageNet validation split. Figures 58, 59 and 60, and Figures 61–64, show one map per stage, with the colour scale fitted to that stage alone. These colours can therefore be compared between the two models, but not between stages. Table 13 lists the per-block numbers.

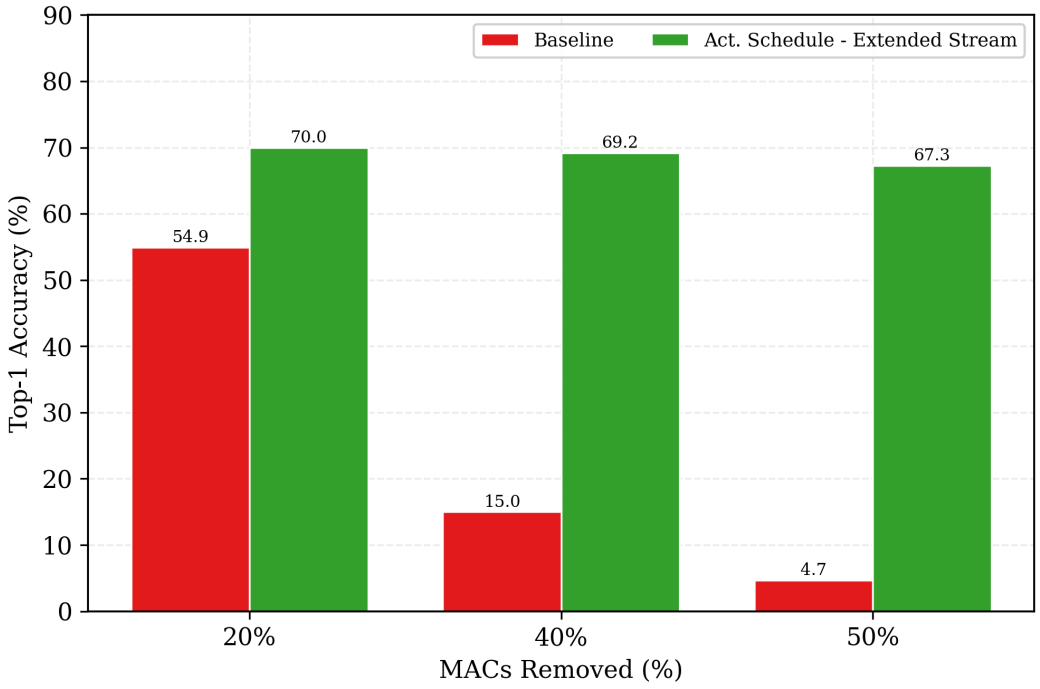



Figure 48: MAC reduction achievable within 1%, 2%, and 5% accuracy degradation budgets under graded pruning for Extended Stream gating against the baseline.

Table 10: Maximum MAC reduction (%) achievable within a 1%, 2%, and 5% Top-5 accuracy drop budget under uniform pruning.

| **Model** | $\leq$ **1% Drop** | $\leq$ **2% Drop** | $\leq$ **5% Drop** |
|---|---|---|---|
| *Intra-Block Gating vs. Baseline* | | | |
| Baseline | 0.00 | 0.00 | 6.21 |
| Act. Schedule - Intra-Block | 45.88 | 55.44 | 67.45 |
| *Extended Stream Gating vs. Baseline* | | | |
| Baseline | 0.00 | 0.00 | 7.15 |
| Act. Schedule - Extended Stream | 27.67 | 45.40 | 60.47 |

The same pattern holds in every stage. The Extended Stream model concentrates its activation and power in the leading channel indices, while the Baseline spreads them evenly. The ordering is weakest in Stage 4 (blocks 13–15), where the first 10% of indices hold 26.7% of the power, against 37.7%–45.8% in the earlier stages. These blocks also carry the least power overall: the mean channel power is 0.115 for the Extended Stream model and 0.151 for the Baseline.

## K Spatial Feature Maps

This section complements the feature maps of Section 4.2. Figures 65–69 repeat the comparison of Figure 1 for five further ImageNet validation images, with the same layer, the same five channels and the same shared colour scale within each pair. The images are chosen to span the kinds of content the network sees: a second animal silhouette, a man-made object against flat sky, a studio shot on a white background, a close-up face and a cluttered natural texture. The reading is the same in every case. The Baseline responds in all five channels, and its most complete rendering of the object is often in a trailing channel rather than a leading one. The Extended Stream model concentrates the content in channel 0, keeps sparse edges in channel 16, and is close to zero in channels 32, 47 and 63. Note that the two rows of each figure share one colour bar, so the darkness of the trailing panels in the bottom row is a real drop in activation and not a rescaling.

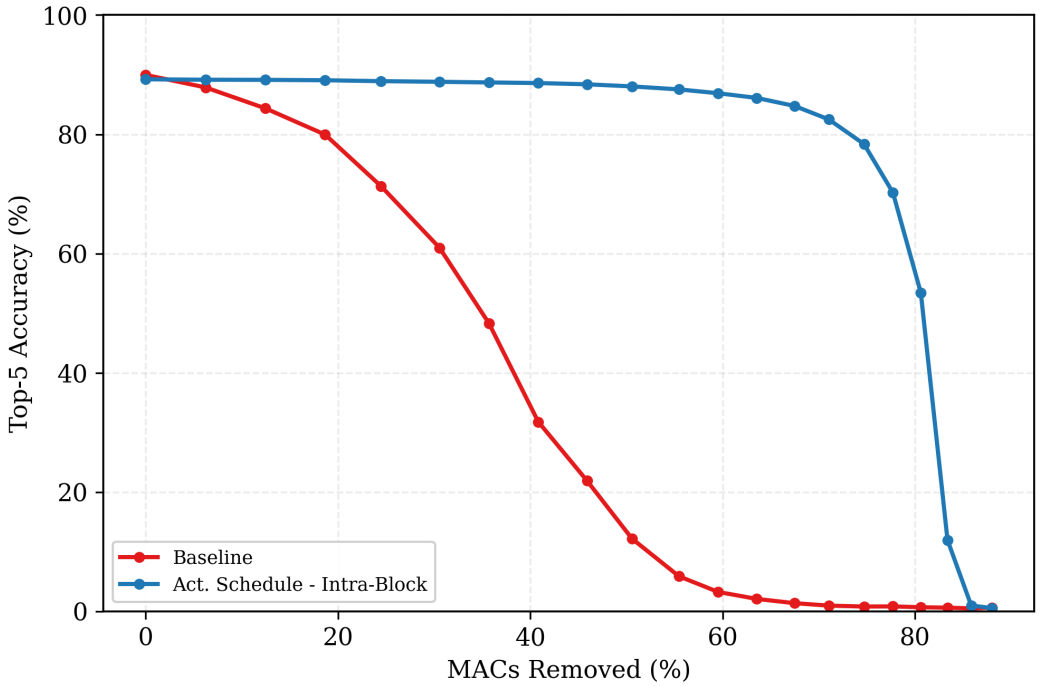



Figure 49: Top-5 validation accuracy versus percentage of MACs removed under uniform pruning for Intra-Block gating against the baseline.

Table 11: Top-5 Accuracy (%) under Graded Pruning at various MAC reduction levels.

| **Model** | **MACs Removed (%)** | **Top-5 (%)** |
|---|---|---|
| *Intra-Block Gating vs. Baseline* | | |
| Baseline | 0.00 | 89.95 |
| Act. Schedule - Intra-Block | 0.00 | 89.22 |
| Baseline | 31.09 | 64.74 |
| Act. Schedule - Intra-Block | 31.09 | 88.90 |
| Baseline | 45.58 | 12.42 |
| Act. Schedule - Intra-Block | 45.58 | 87.95 |
| Baseline | 53.90 | 3.64 |
| Act. Schedule - Intra-Block | 53.90 | 68.07 |
| *Extended Stream Gating vs. Baseline* | | |
| Baseline | 0.00 | 89.95 |
| Act. Schedule - Extended Stream | 0.00 | 89.17 |
| Baseline | 34.60 | 46.52 |
| Act. Schedule - Extended Stream | 34.60 | 88.59 |
| Baseline | 49.45 | 12.51 |
| Act. Schedule - Extended Stream | 49.45 | 87.25 |
| Baseline | 57.25 | 6.44 |
| Act. Schedule - Extended Stream | 57.25 | 81.04 |

Table 12: Maximum MAC reduction (%) achievable within a 1%, 2%, and 5% Top-5 accuracy drop budget under graded pruning.

| **Model** | **$\leq$ 1% Drop** | **$\leq$ 2% Drop** | **$\leq$ 5% Drop** |
|---|---|---|---|
| *Intra-Block Gating vs. Baseline* | | | |
| Baseline | 3.17 | 6.56 | 13.12 |
| Act. Schedule - Intra-Block | 40.99 | 47.74 | 49.83 |
| *Extended Stream Gating vs. Baseline* | | | |
| Baseline | 0.00 | 3.68 | 7.54 |
| Act. Schedule - Extended Stream | 42.44 | 49.45 | 53.52 |

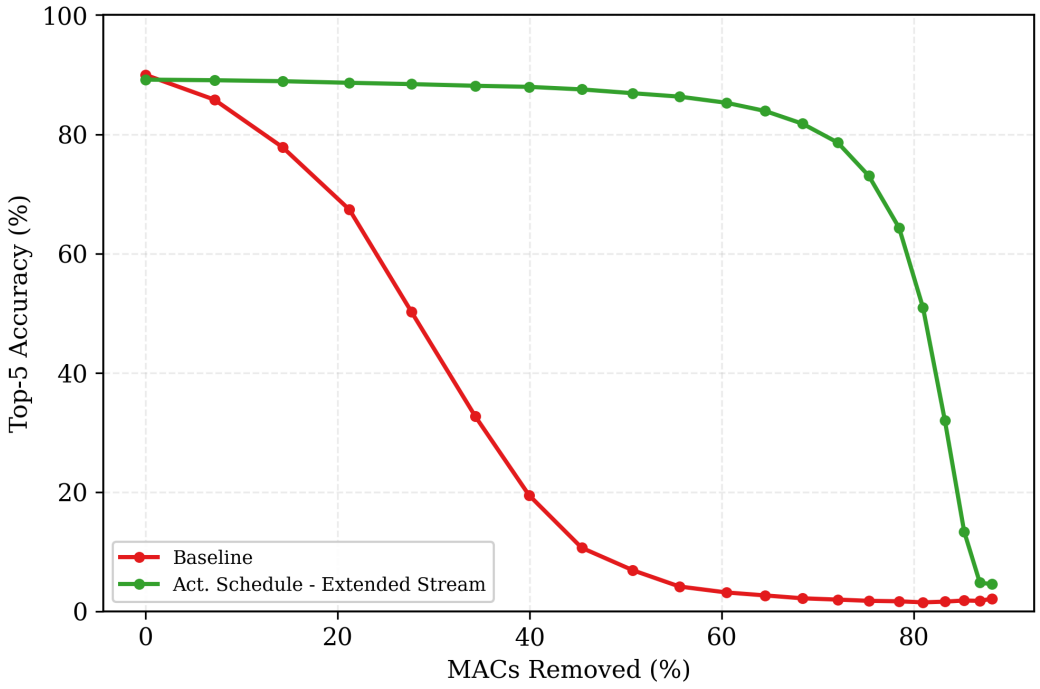



Figure 50: Top-5 validation accuracy versus percentage of MACs removed under uniform pruning for Extended Stream gating against the baseline.

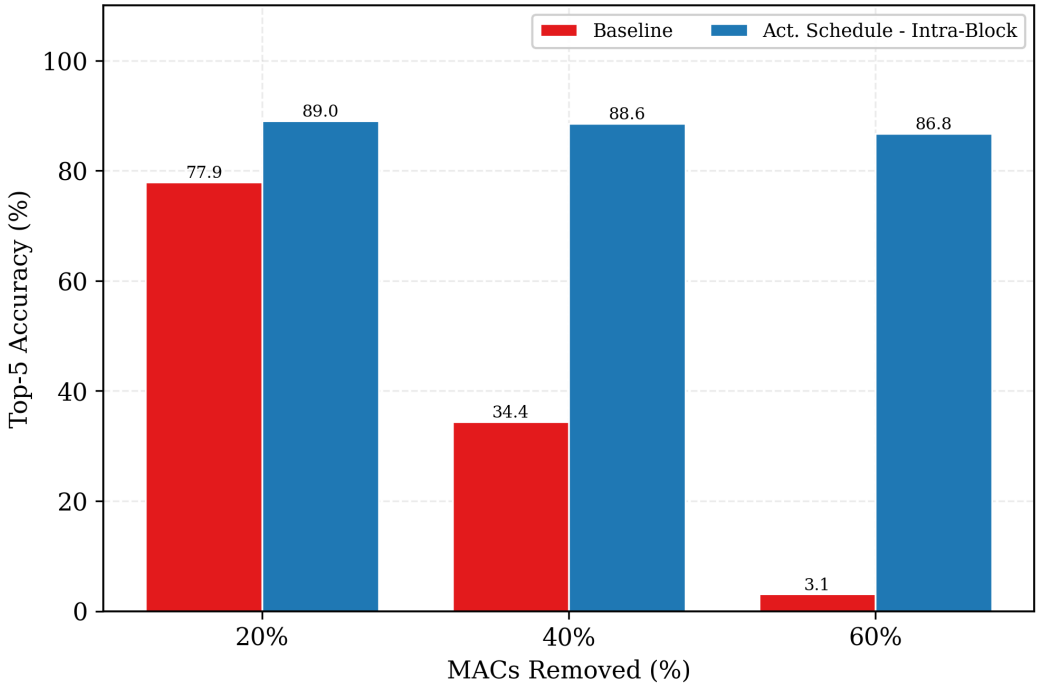



Figure 51: MAC reduction achievable within 1%, 2%, and 5% Top-5 accuracy degradation budgets under uniform pruning for Intra-Block gating against the baseline.

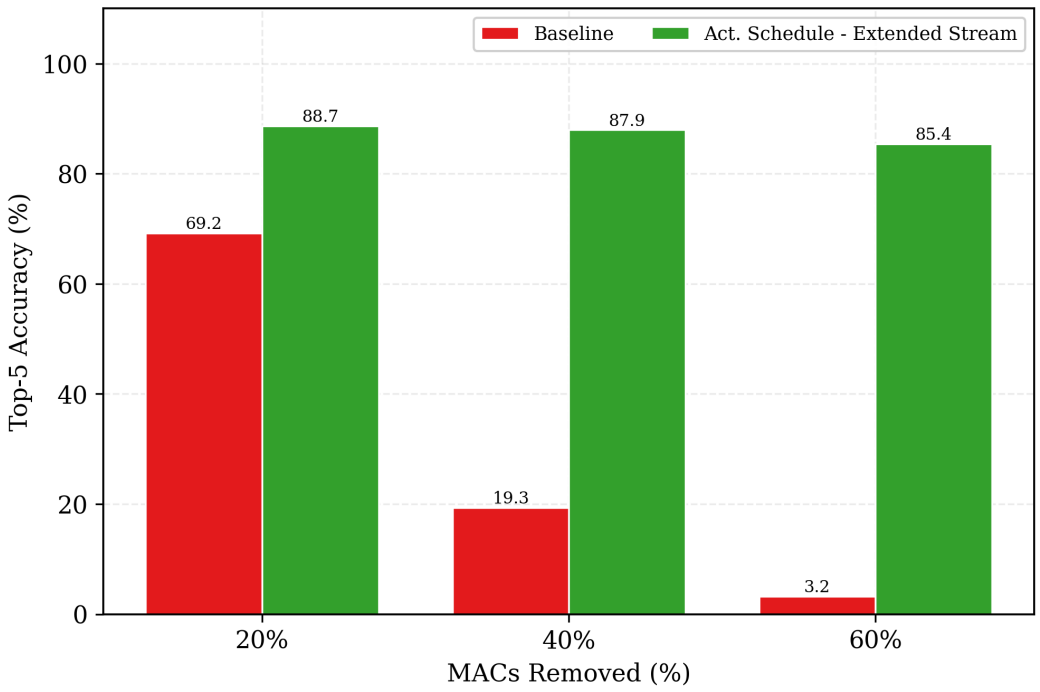



Figure 52: MAC reduction achievable within 1%, 2%, and 5% Top-5 accuracy degradation budgets under uniform pruning for Extended Stream gating against the baseline.

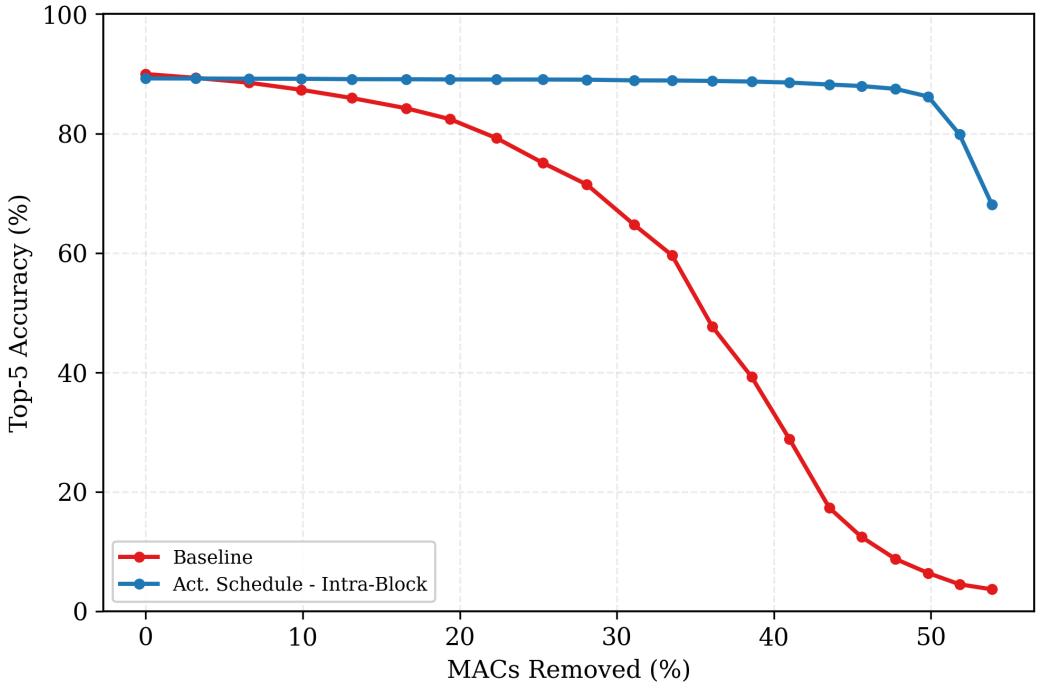



Figure 53: Top-5 validation accuracy versus percentage of MACs removed under graded pruning for Intra-Block gating against the baseline.

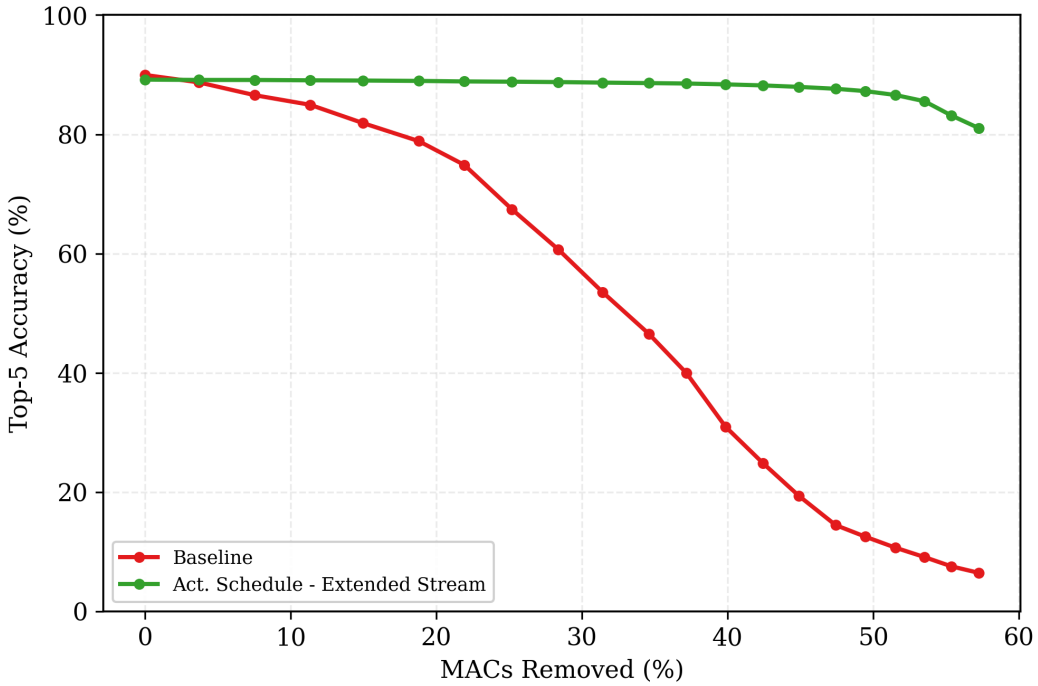



Figure 54: Top-5 validation accuracy versus percentage of MACs removed under graded pruning for Extended Stream gating against the baseline.

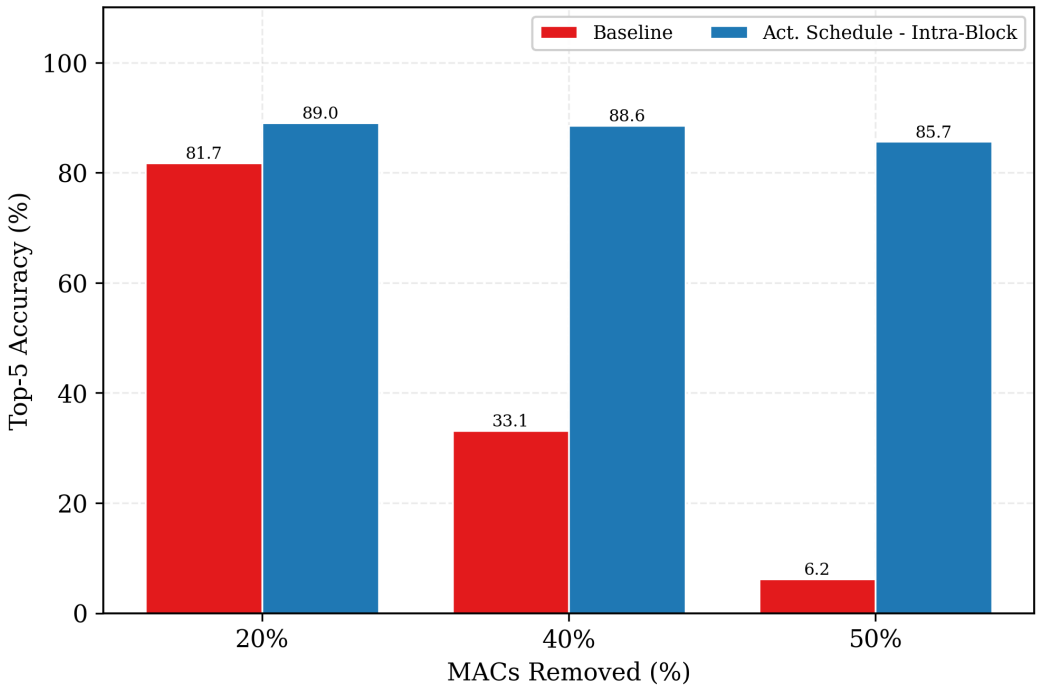



Figure 55: MAC reduction achievable within 1%, 2%, and 5% Top-5 accuracy degradation budgets under graded pruning for Intra-Block gating against the baseline.

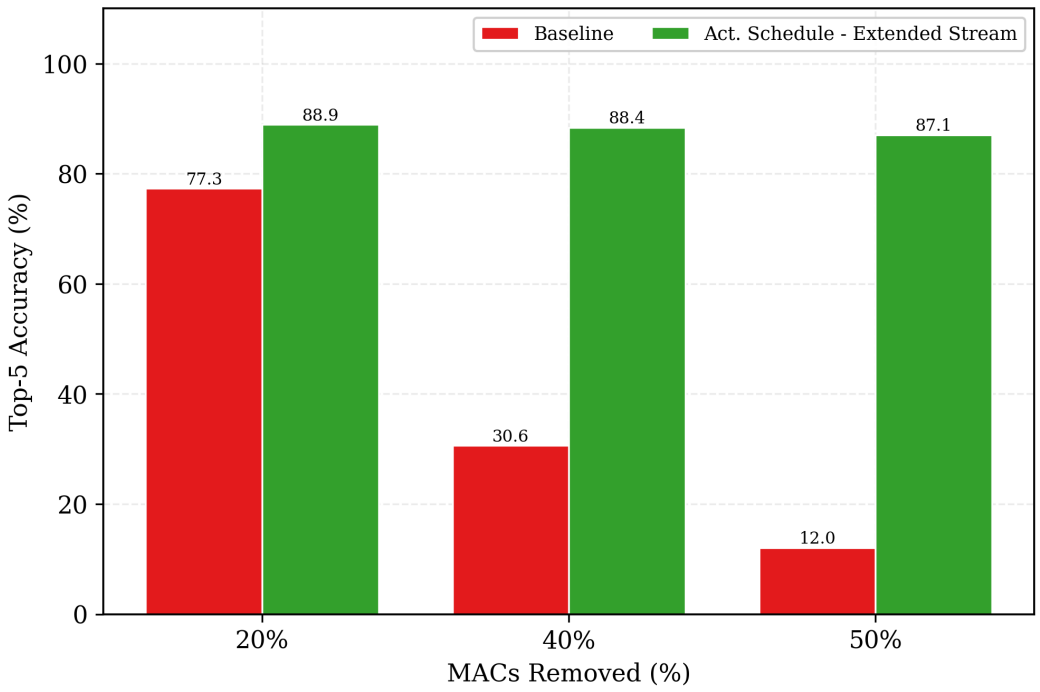
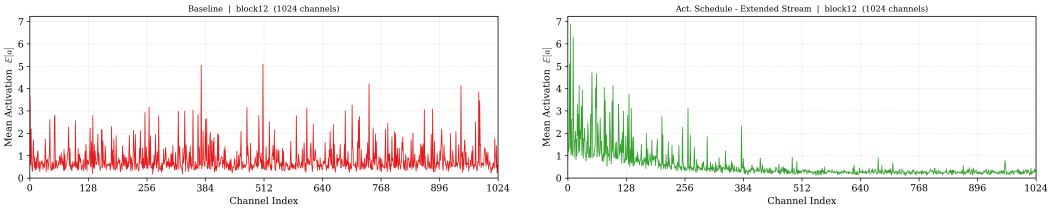
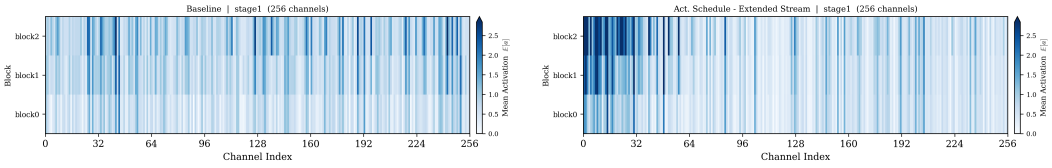



Figure 56: MAC reduction achievable within 1%, 2%, and 5% Top-5 accuracy degradation budgets under graded pruning for Extended Stream gating against the baseline.



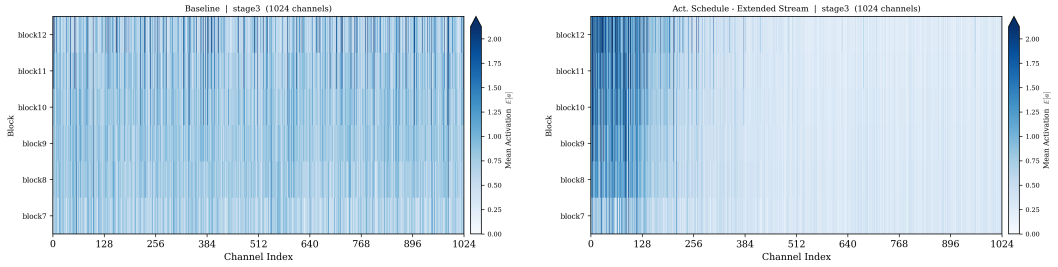

(a) Baseline, block 12 (b) Extended Stream, block 12

Figure 57: Per-channel mean activation $\mathbb{E}[a]$ of block 12 (1024 channels) for (a) the Baseline and (b) the Extended Stream model, on a shared y-axis. The Baseline's largest response sits at channel 510, whereas the scheduled model peaks at channel 6 and decays with the channel index.



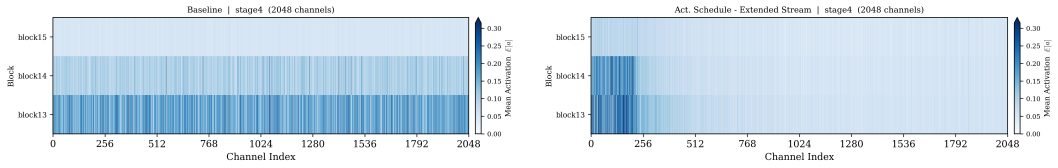

(a) Baseline, stage 1 (b) Extended Stream, stage 1

Figure 58: Per-channel mean activation $\mathbb{E}[a]$ over Stage 1, for (a) the Baseline and (b) the Extended Stream model. Stage 2 is shown in Figure 7. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.



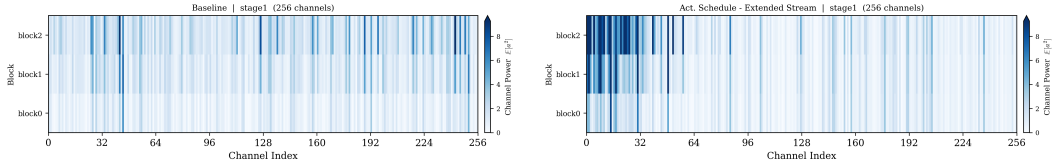

(a) Baseline, stage 3 (b) Extended Stream, stage 3

Figure 59: Per-channel mean activation $\mathbb{E}[a]$ over Stage 3, for (a) the Baseline and (b) the Extended Stream model. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.



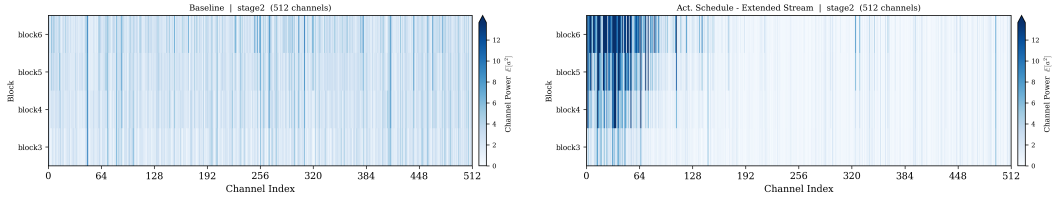

(a) Baseline, stage 4 (b) Extended Stream, stage 4

Figure 60: Per-channel mean activation $\mathbb{E}[a]$ over Stage 4, for (a) the Baseline and (b) the Extended Stream model. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.



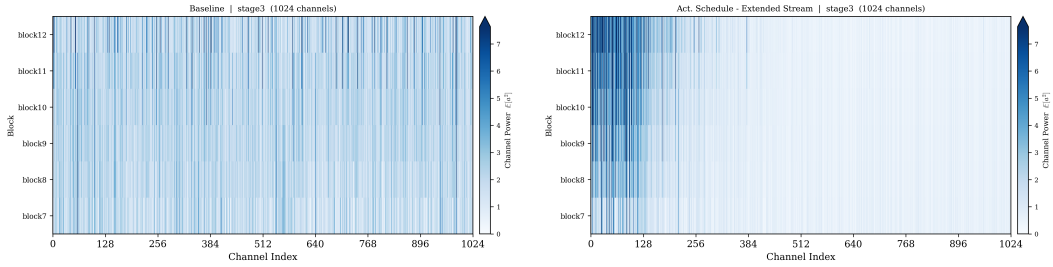

(a) Baseline, stage 1 (b) Extended Stream, stage 1

Figure 61: Per-channel power $\mathbb{E}[a^2]$ over Stage 1, for (a) the Baseline and (b) the Extended Stream model. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.



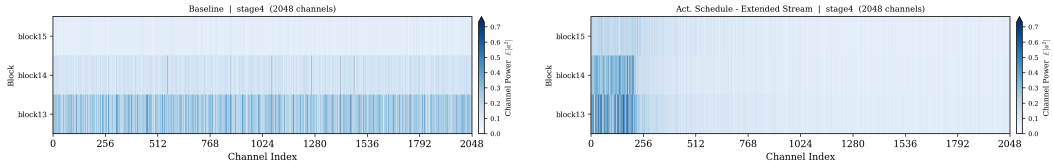

(a) Baseline, stage 2 (b) Extended Stream, stage 2

Figure 62: Per-channel power $\mathbb{E}[a^2]$ over Stage 2, for (a) the Baseline and (b) the Extended Stream model. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.




(a) Baseline, stage 3 (b) Extended Stream, stage 3

Figure 63: Per-channel power $\mathbb{E}[a^2]$ over Stage 3, for (a) the Baseline and (b) the Extended Stream model. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.




(a) Baseline, stage 4 (b) Extended Stream, stage 4

Figure 64: Per-channel power $\mathbb{E}[a^2]$ over Stage 4, for (a) the Baseline and (b) the Extended Stream model. The colour scale is fitted to this stage (0 to its 99th percentile) and shared by both models.

Table 13: Per-block channel statistics of the Baseline (B) and the Extended Stream model (E): mean channel power $\mathbb{E}[a^2]$, the share (%) of the block's power held by the first 10% and 25% of channel indices (10% and 25% for a flat profile), and the index of the strongest channel.

| **Block** | **Width** | **Mean power** | | **First 10% idx** | | **First 25% idx** | | **Strongest ch.** | |
|---|---|---|---|---|---|---|---|---|---|
| | | **B** | **E** | **B** | **E** | **B** | **E** | **B** | **E** |
| 0 | 256 | 0.851 | 1.014 | 8.1 | 23.8 | 28.6 | 39.6 | 44 | 14 |
| 1 | 256 | 1.458 | 1.583 | 8.2 | 39.5 | 27.8 | 58.2 | 44 | 14 |
| 2 | 256 | 2.029 | 2.570 | 8.3 | 49.8 | 26.2 | 70.1 | 242 | 14 |
| 3 | 512 | 2.124 | 0.972 | 9.3 | 32.4 | 24.1 | 49.3 | 47 | 31 |
| 4 | 512 | 2.462 | 1.433 | 9.4 | 43.2 | 23.9 | 63.6 | 47 | 13 |
| 5 | 512 | 2.636 | 2.245 | 10.0 | 51.3 | 24.2 | 74.4 | 309 | 13 |
| 6 | 512 | 2.832 | 3.485 | 9.5 | 56.4 | 24.0 | 81.4 | 309 | 1 |
| 7 | 1024 | 1.774 | 0.878 | 11.7 | 26.1 | 26.6 | 47.3 | 59 | 93 |
| 8 | 1024 | 1.939 | 1.045 | 11.5 | 32.9 | 26.2 | 55.9 | 984 | 93 |
| 9 | 1024 | 2.055 | 1.152 | 11.2 | 36.9 | 25.6 | 60.3 | 984 | 3 |
| 10 | 1024 | 2.136 | 1.318 | 10.9 | 40.6 | 25.4 | 65.1 | 984 | 6 |
| 11 | 1024 | 2.243 | 1.510 | 10.6 | 44.5 | 25.1 | 69.5 | 375 | 6 |
| 12 | 1024 | 2.332 | 1.778 | 10.0 | 48.1 | 24.6 | 73.7 | 510 | 6 |
| 13 | 2048 | 0.250 | 0.133 | 9.8 | 31.8 | 25.5 | 50.6 | 317 | 84 |
| 14 | 2048 | 0.130 | 0.113 | 9.6 | 29.2 | 23.8 | 47.0 | 559 | 86 |
| 15 | 2048 | 0.074 | 0.098 | 9.7 | 19.2 | 24.5 | 40.7 | 1805 | 229 |

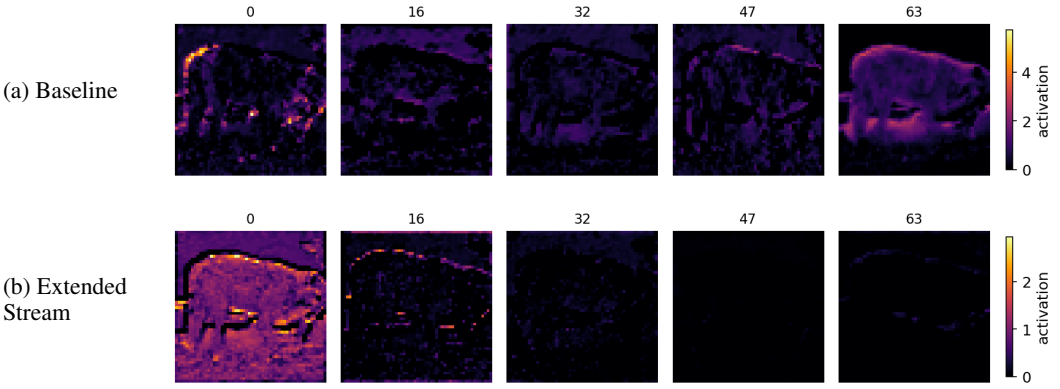



Figure 65: Post-ReLU feature maps of the first $1 \times 1$ convolution in block 0 for a second ImageNet validation image (white wolf). Each row shows channels 0, 16, 32, 47 and 63 of the 64 channels, on one shared colour scale. The Baseline (top) responds strongly in all of these channels, and gives its most complete silhouette in channel 63. The Extended Stream model (bottom) puts the image content in the leading channel, and its trailing channels are close to zero.

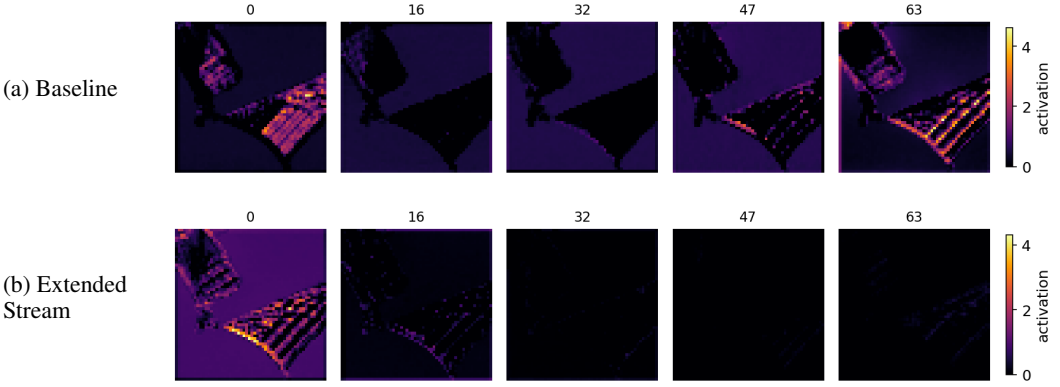



Figure 66: The same comparison for a man-made object on a flat background (parachute). This image separates the two models most sharply. In the Baseline (top) the trailing channel 63 is the brightest and most detailed of the five: it resolves the individual rigging lines and the stripes of the flag, while the leading channel 0 is comparatively muted. The importance ordering therefore runs against the channel index. In the Extended Stream model (bottom) that ordering is reversed and enforced: channel 0 carries the canopy, the lines and the flag at full contrast, channel 16 retains only their edges, and channels 32 and 47 are empty. Channel 63 keeps a faint trace of the strongest edges alone.

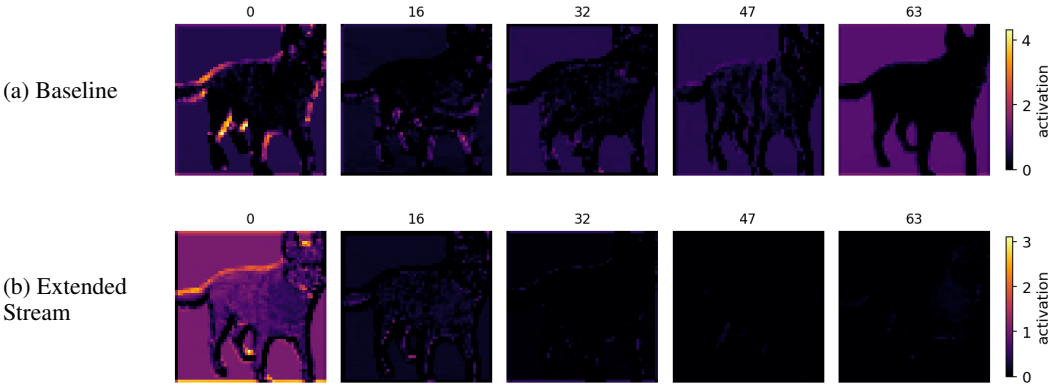



Figure 67: The same comparison for a studio shot on a white background (kelpie). The flat background makes the figure-ground assignment easy to read. In the Baseline (top) every channel carries a version of the animal, and channel 63 holds the cleanest one, as a uniform bright background with the body cut out of it. In the Extended Stream model (bottom) that whole description sits in channel 0, which holds the background, the body and the outline together; channel 16 keeps the outline and a little fur texture, and channels 32, 47 and 63 are blank.

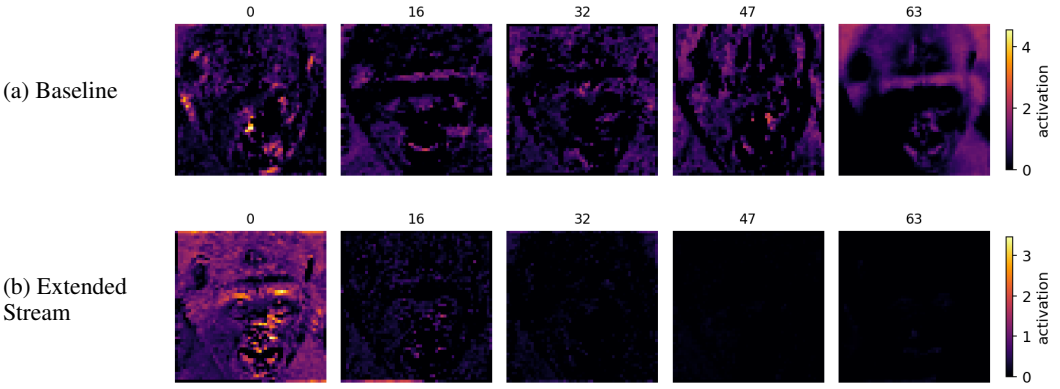
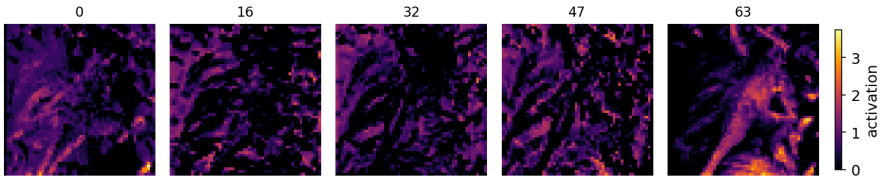



Figure 68: The same comparison for a close-up face (gorilla). The Baseline (top) spreads the face across all five channels: channel 0 gives scattered highlights, channels 16 and 47 give edges around the brow and muzzle, and the smooth, complete rendering of the face again appears last, in channel 63. The Extended Stream model (bottom) places the complete face in channel 0, with the brow, eyes and muzzle all legible in that single map; channel 16 reduces to their contours, and channels 32, 47 and 63 are near-silent.

(a) Baseline




(b) Extended Stream

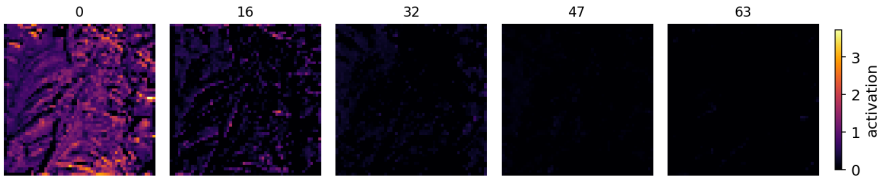



Figure 69: The same comparison for a cluttered natural scene with no flat background (common iguana on a tree trunk). Here the Baseline (top) is close to uniform across the five channels: each one carries the scales, the bark and the surrounding leaves at a similar strength, with channel 63 the strongest, so no channel can be dropped without losing part of the description. In the Extended Stream model (bottom) the same texture is compressed into channel 0, channel 16 keeps only the high-contrast scale and bark edges, and channels 32, 47 and 63 fade to black. The contrast between a flat profile and a decaying one is clearest on this image.