

1. Introduction

Hyperflux asks two questions: why do weights get pruned or regrown, and how does the network behave as it is pruned?

Main contributions:

- A novel L_0 method that evaluates a weight's importance after it is pruned, through its flux.
- A flux–pressure model that explains pruning and regrowth, yielding a scheduler for precise sparsity targeting.
- Competitive results vs. CAP, GraNet and AC/DC across ResNet-50, VGG-19 and DeiT on CIFAR-10/100 and ImageNet.

2. Method: flux & pressure

Each weight ω_i has a learnable presence t_i that masks it when $t_i \leq 0$. Two forces decide its fate:

- Flux ($G_i^- = -\partial L / \partial t_i$): the loss increase a weight's removal causes, its importance.
- Pressure ($L_\infty = (\gamma/d) \cdot \Sigma t_i$): a constant gradient γ/d pushing every t_i toward pruning.
- A straight-through estimator keeps gradients on pruned weights, so any can revive.

Weights and presences train jointly under one objective:

$$J(\omega, t) = L(\omega, T) + L_\infty(t)$$

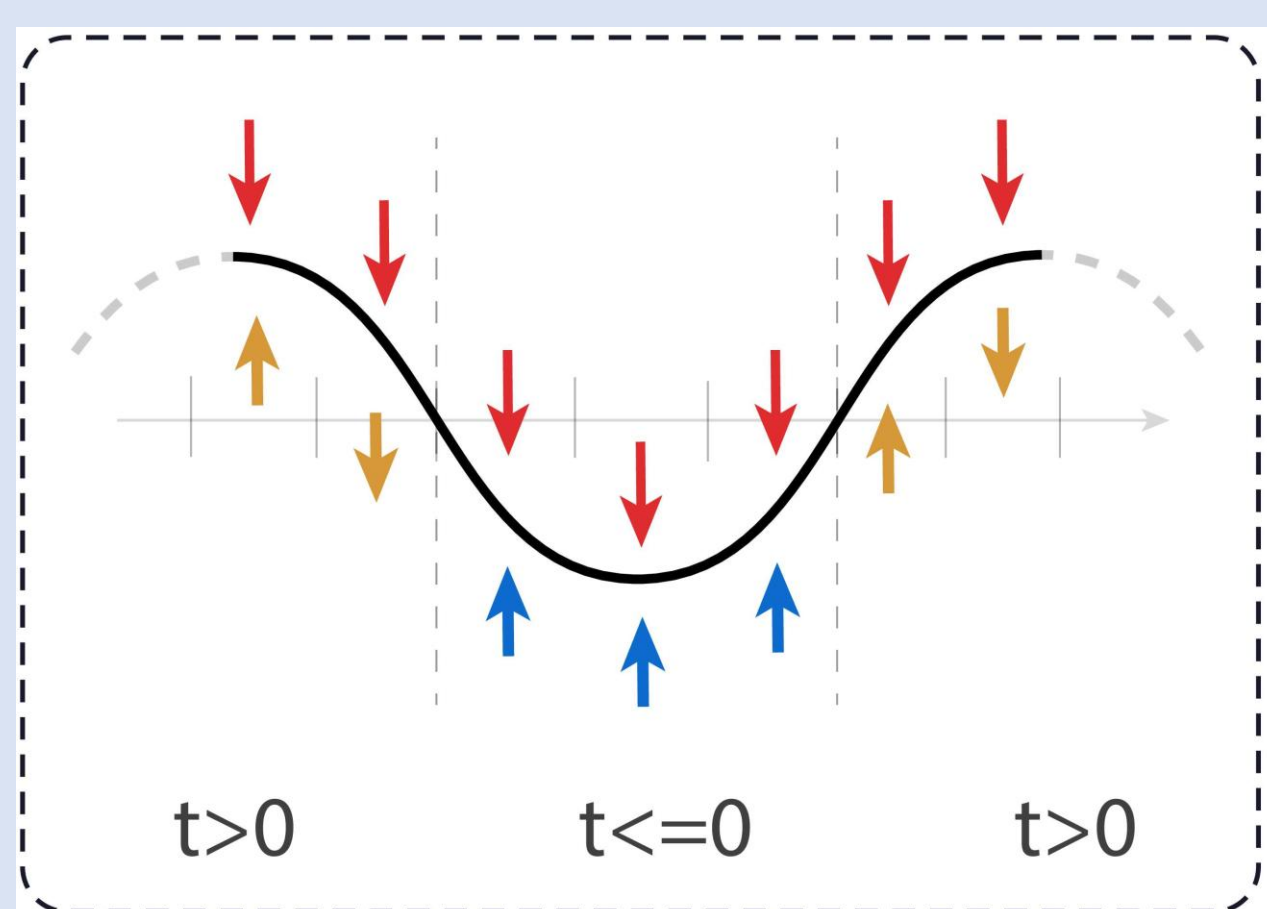


Fig. 5. Aggregated flux over the topologies visited in training. In pruned phases ($t \leq 0$) flux accumulates; when it exceeds the pressure the weight regrows ($t > 0$).

3. Properties

- Property 1: Convergence. For a fixed pressure γ , density settles at a stable equilibrium where the minimum surviving flux equals γ , rather than diverging or oscillating.
- Property 2: Predictability. That equilibrium density follows a clean power-law in γ , $\ln(s) = \ln(c) - \alpha_0 \cdot \ln(\gamma)$, which makes the target sparsity predictable and enables the scheduler.

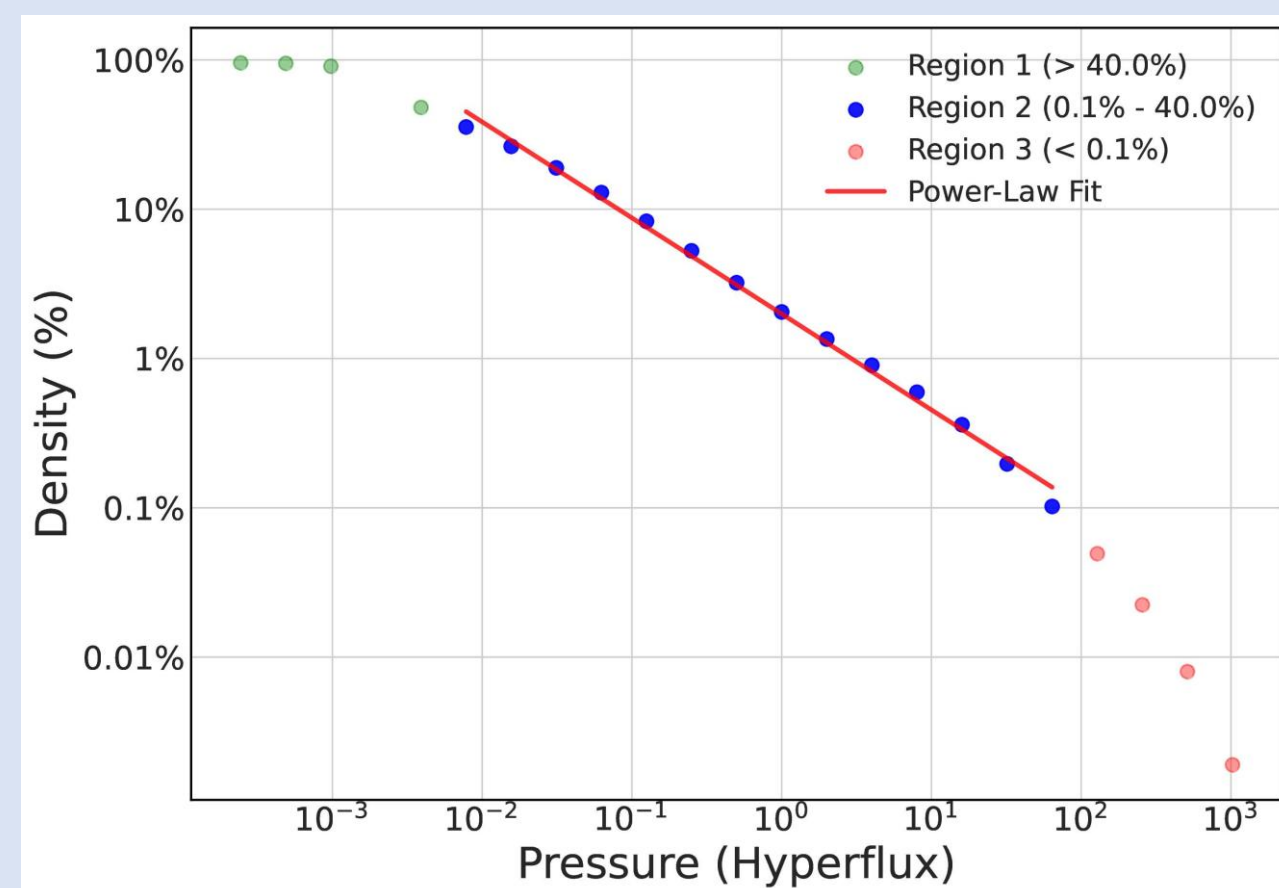


Fig. 1. Density converges to an equilibrium for fixed γ .

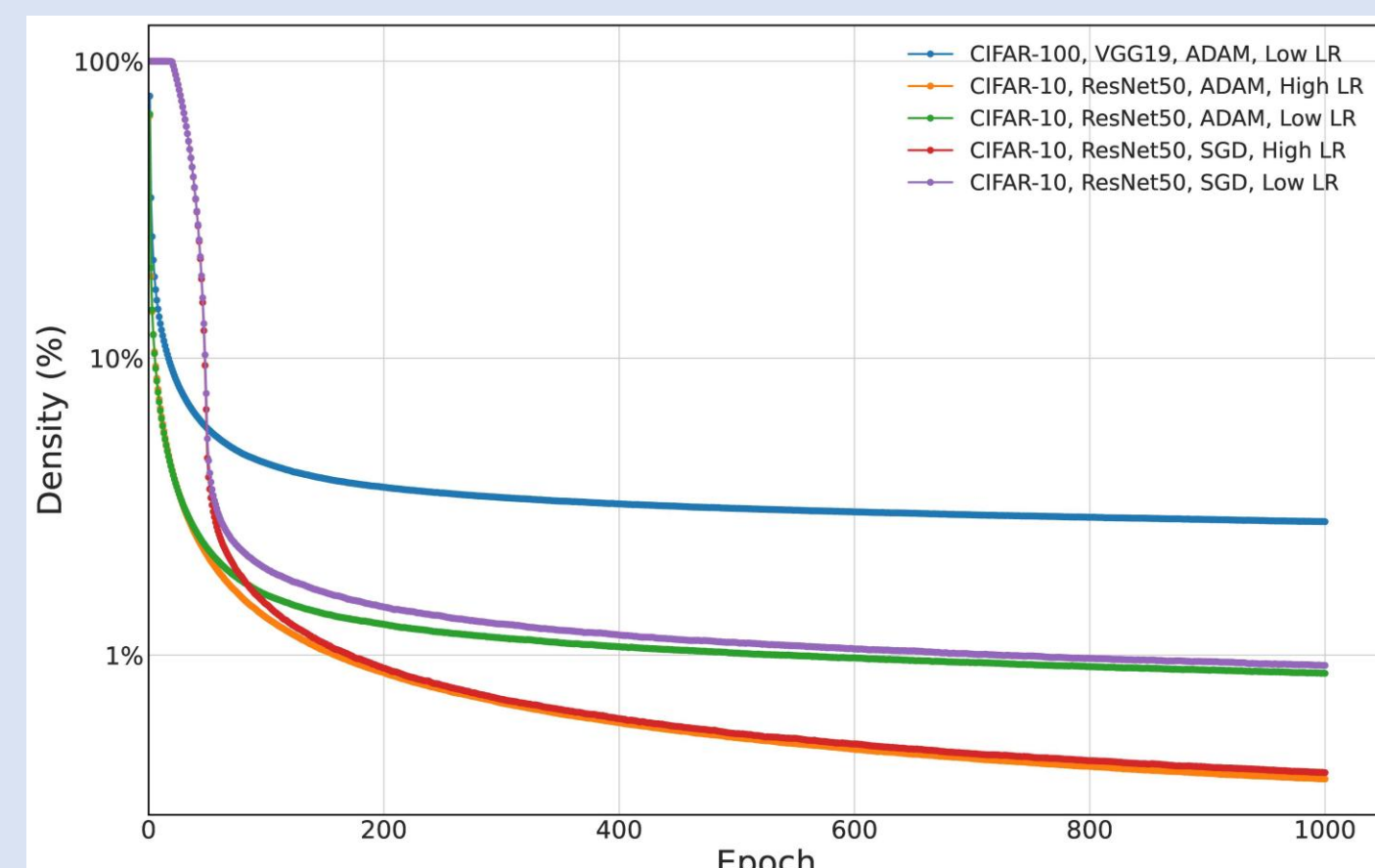


Fig. 2. γ vs. final density: a clean power-law.

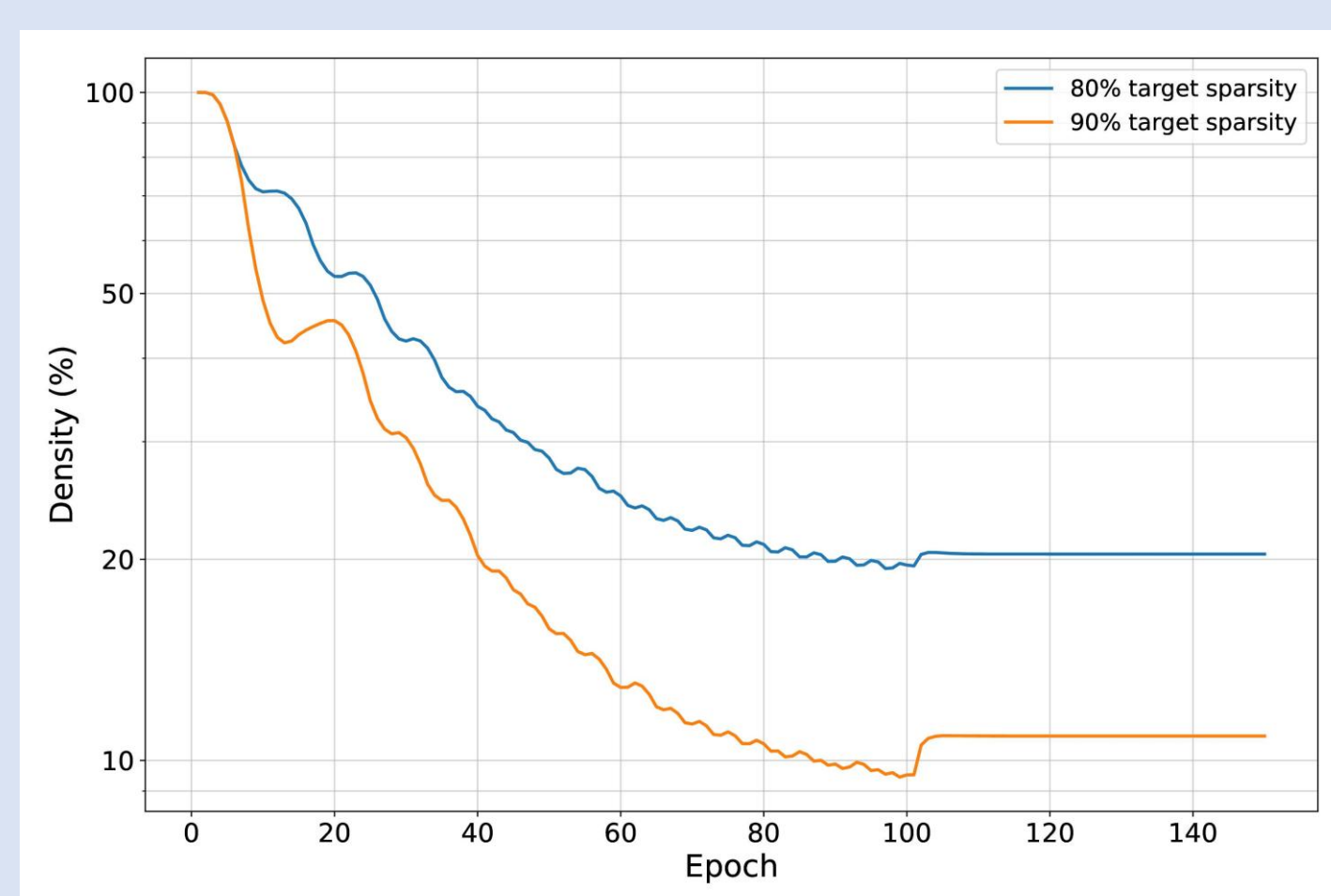


Fig. 3. Density vs. epoch under the scheduler.

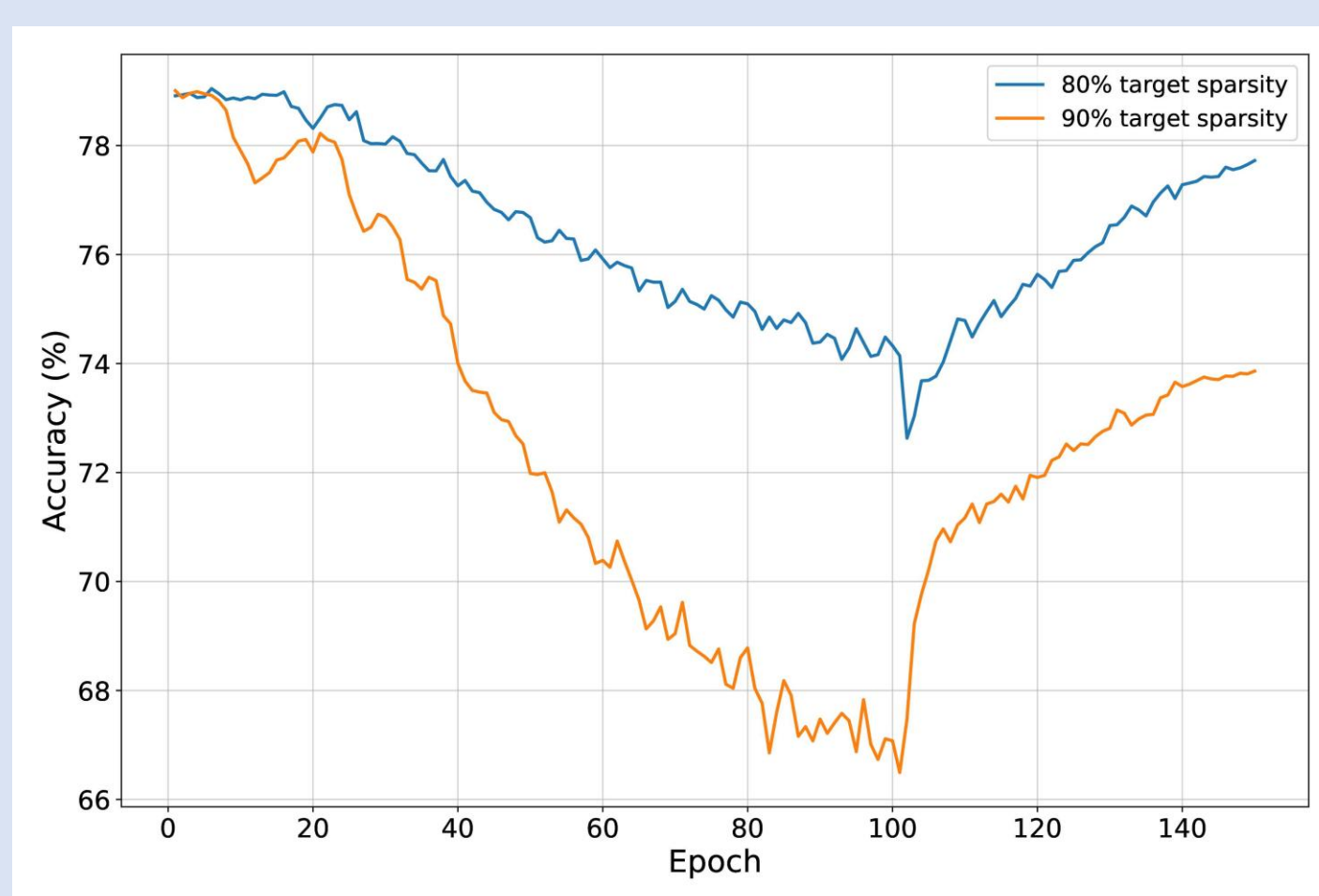


Fig. 4. Accuracy vs. epoch under the scheduler.

4. Pressure scheduler

The scheduler sets the global pressure, and thus controls the sparsity, at every epoch:

- Because density follows the power-law of Property 2, we set $\gamma = (p_e)^\alpha$.
- Like a PI controller, but with power-law increments and inertia terms for stability.
- Two policies: Π_1 (trajectory control, within 20%) vs Π_2 (final-density precision, within 5%).

5. Why weights regrow

A pruned weight keeps receiving gradient, so its presence drifts back up whenever its flux is large (implicit regrowth). Its t-value gradients over training show:

- Present ($t > 0$, blue): the gradient stays small.
- Pruned but important ($t \leq 0$, red): large flux pulls it back above the threshold.
- Pruned and unimportant: little flux, so it stays pruned.

Importance is thus assessed across the topologies visited during training, not at a single step.

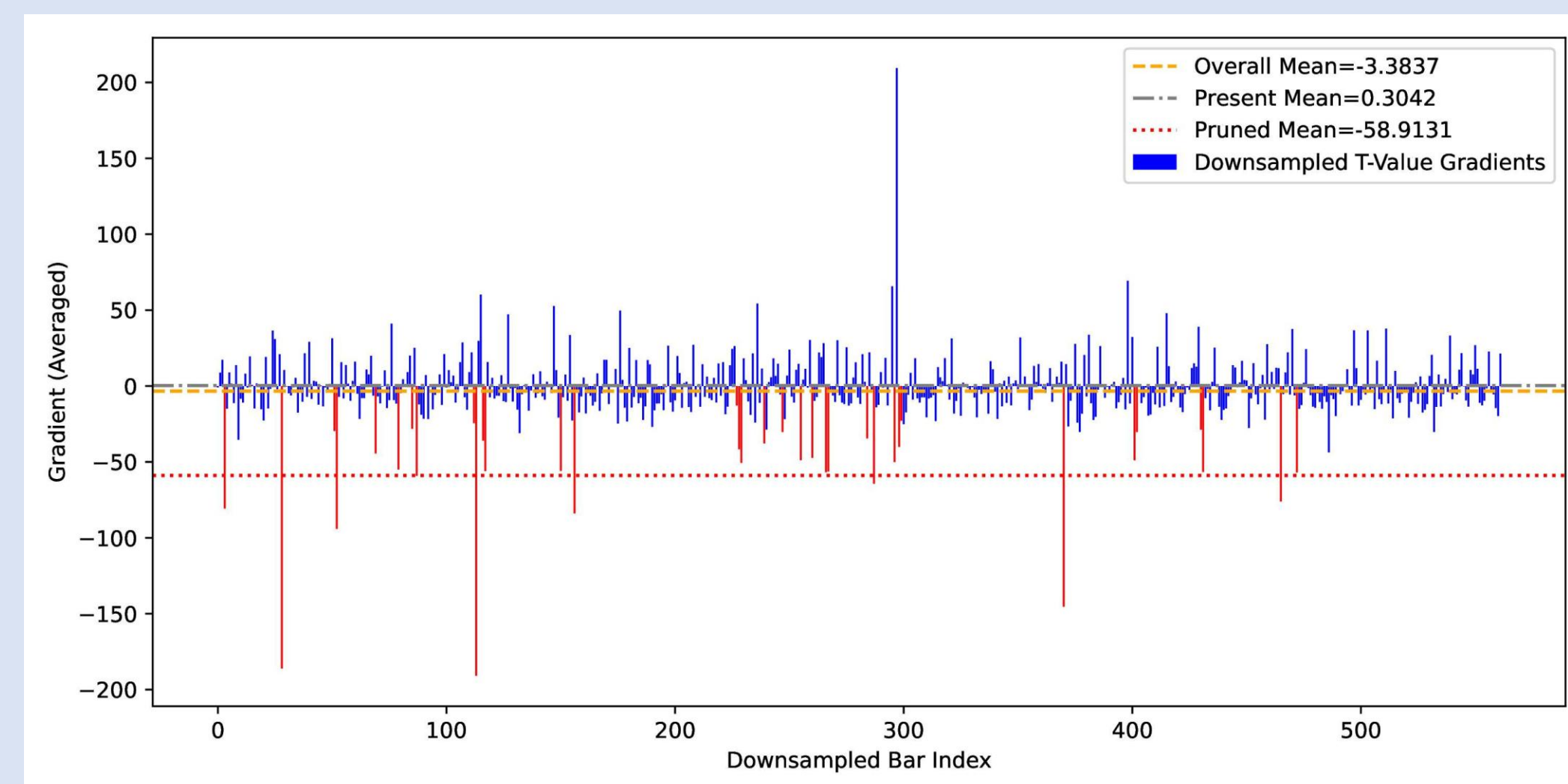


Fig. 6. t-value gradients of pruned-network weights over time (pruned mean ≈ -59 vs present mean ≈ 0).

6. Results

Against strong baselines run under an identical budget from the same pretrained weights, Hyperflux beats the dense model on CIFAR-100 at 90% and 95%, tracks AC/DC within 0.6% on ImageNet, and scales to Vision Transformers within 0.7% of CAP using half the training budget.

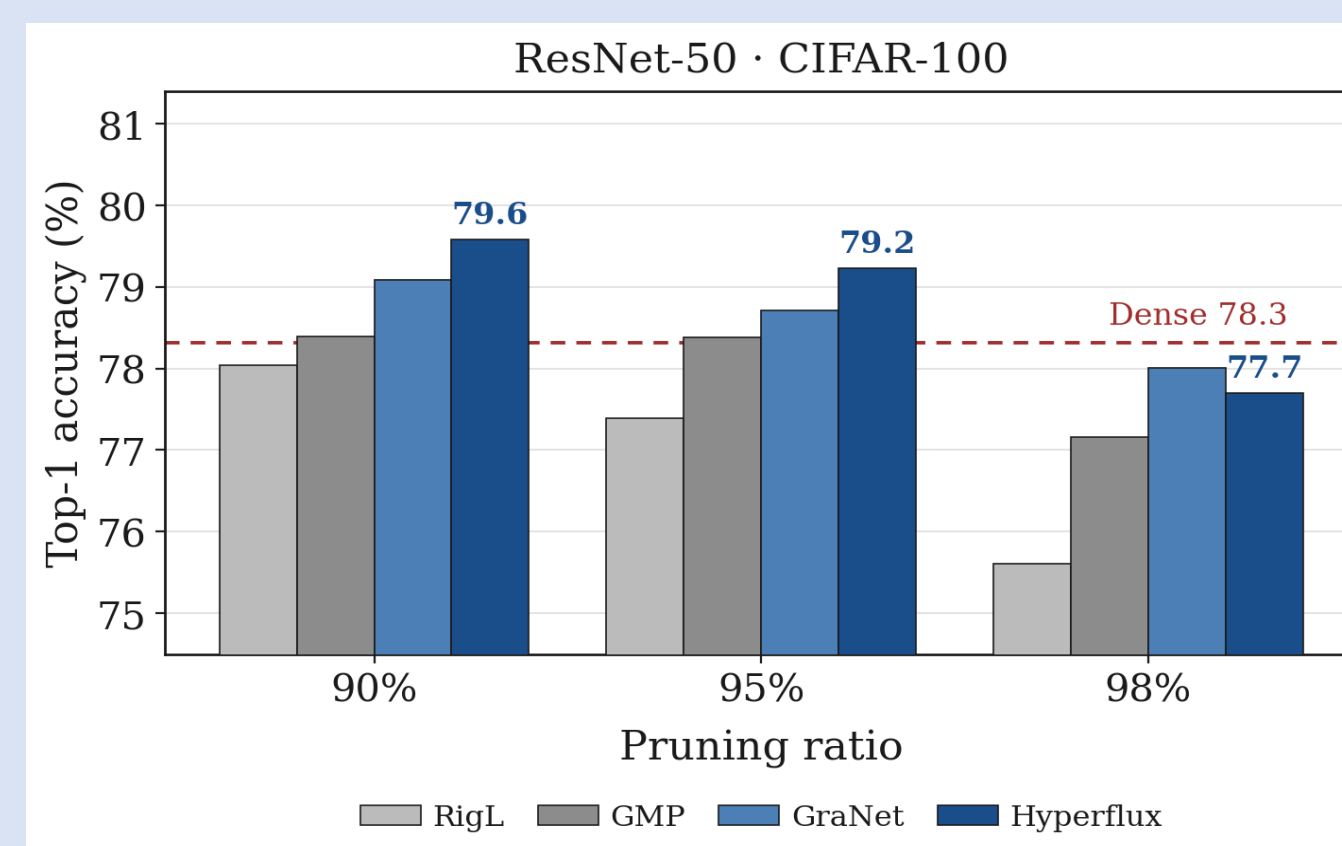


Fig. 7. CIFAR-100 accuracy.

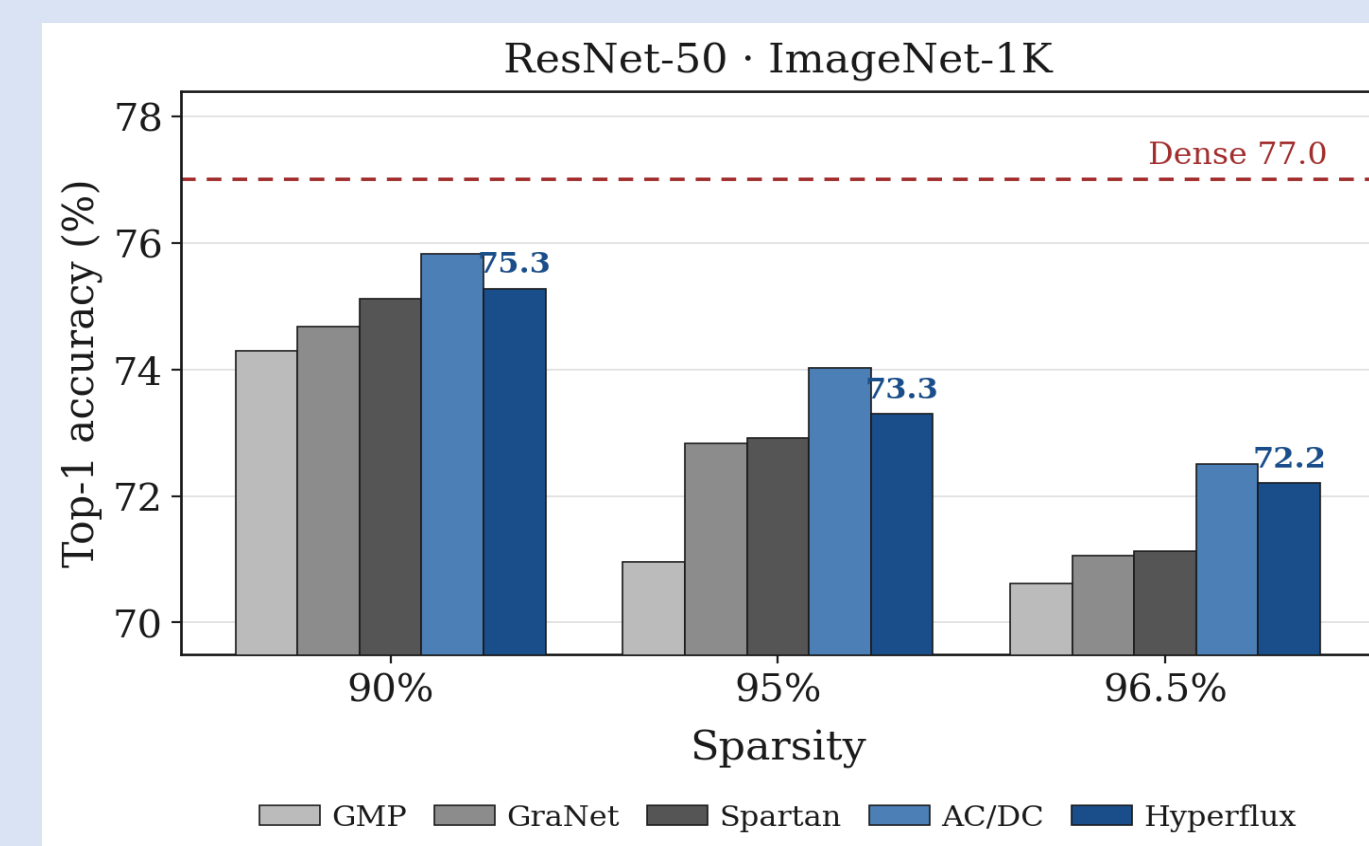


Fig. 8. ImageNet ResNet-50 accuracy.

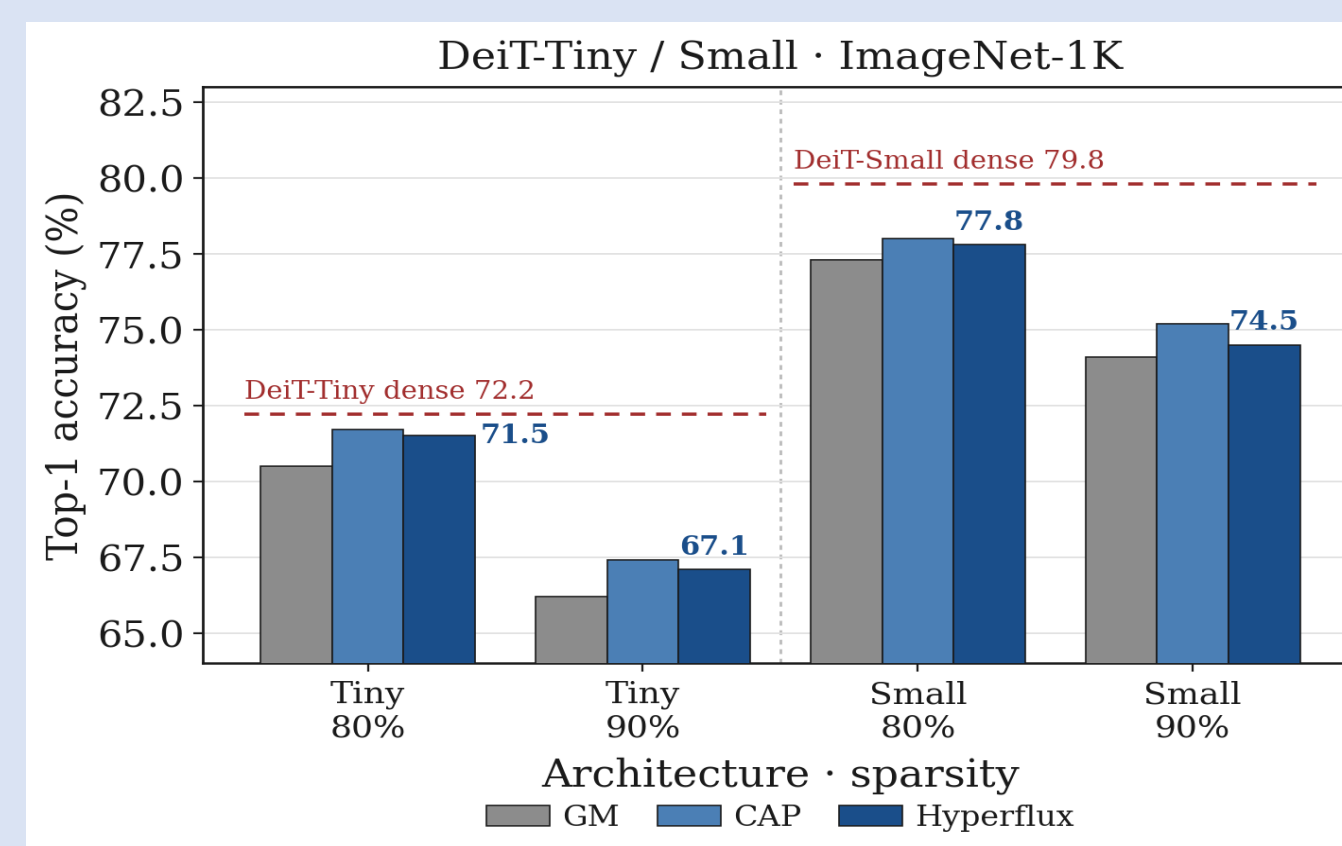


Fig. 9. DeiT-Tiny / Small accuracy.

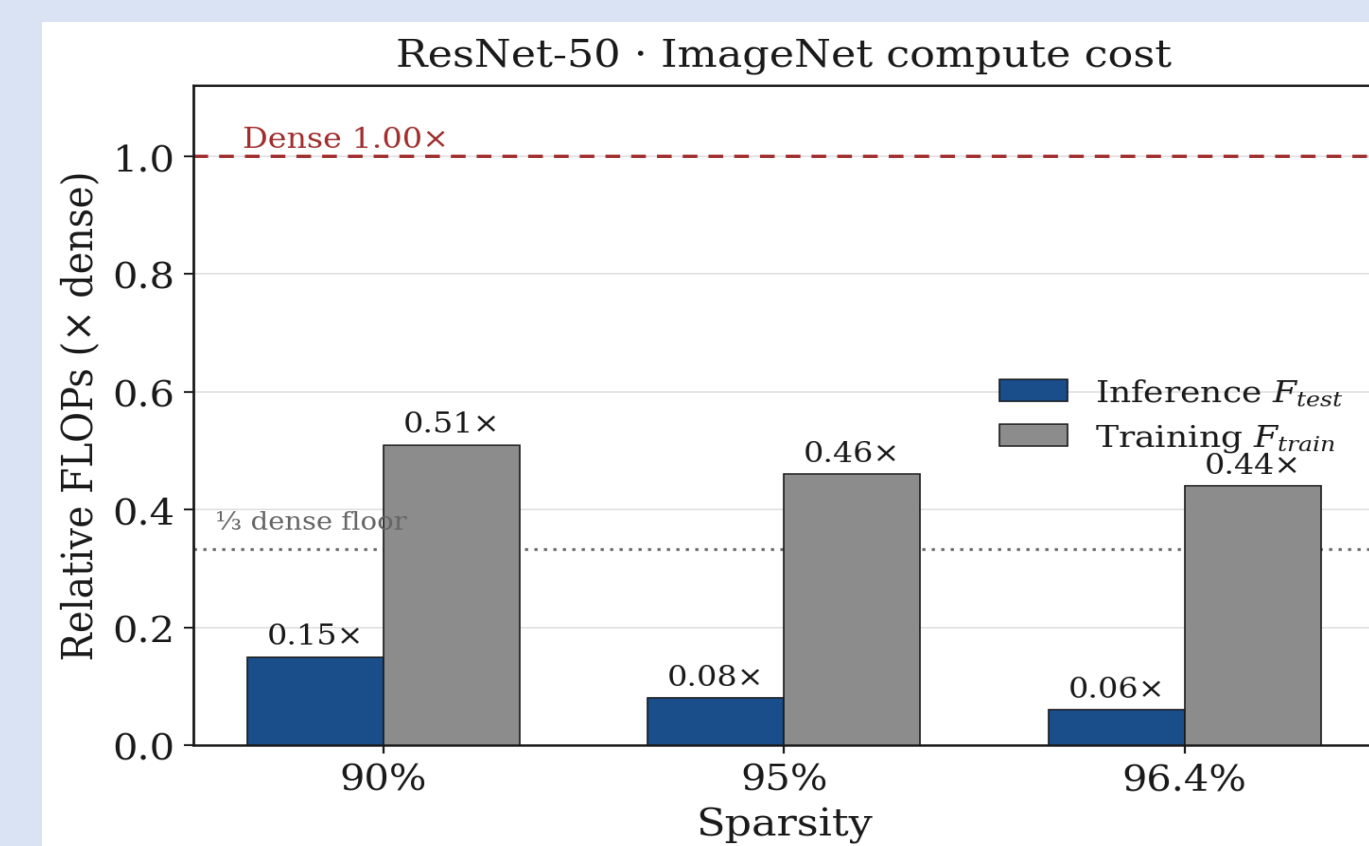


Fig. 10. ImageNet compute cost.

7. Conclusions & future work

Hyperflux turns pruning from an empirical black box into a predictable flux–pressure system, understandable at both the weight and network level, while staying competitive across CNNs and ViTs. Future work extends the pressure–density power law, building on the Neural Pruning Law Hypothesis [2504.05349v3] (Bărbulescu et al.), and tests Hyperflux on reinforcement learning.

Strengths

- Interpretable at weight & network level
- Precise sparsity targeting
- Regrowth at any training step
- Scales to CNNs & ViTs

Limitations

- $\geq \frac{1}{3}$ dense training FLOPs
- Extra hyper-parameters to tune
- Regrowth can drift final sparsity
- Untested on RL / LLMs