SliceFine

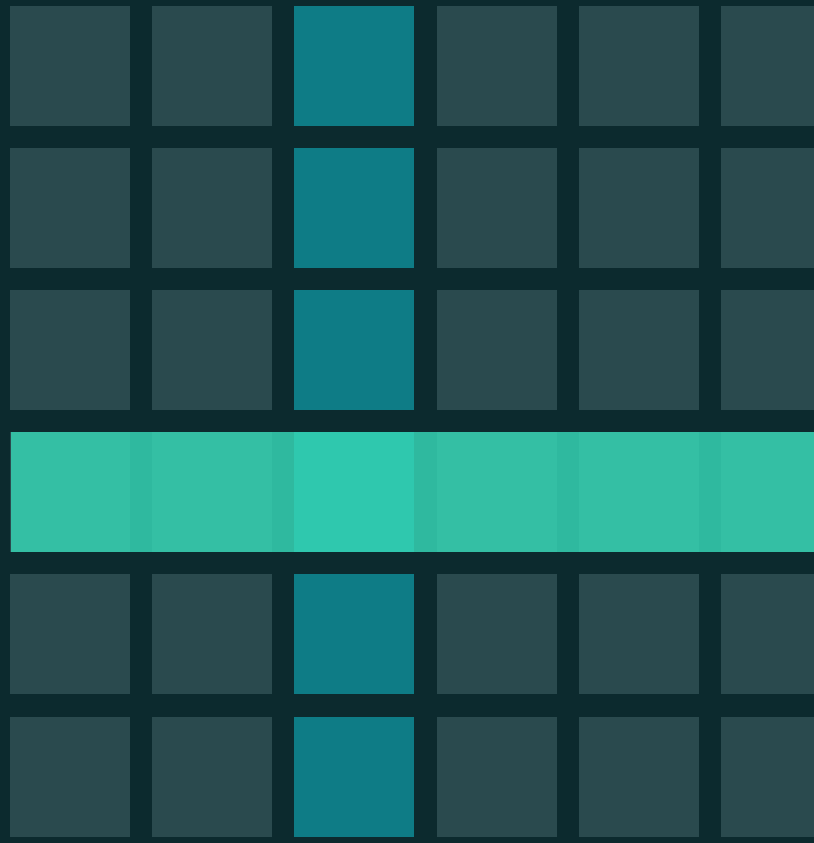
The Universal Winning-Slice Hypothesis for Pretrained Networks

Md Kowsher, Ali O. Polat, Ehsan Mohammady Ardehaly, Mehrdad Salehi, Zia Ghiasi, Prasanth Murali, Chen Chen

Meta · *University of Central Florida*

ICML 2026

github.com/facebookresearch/SliceFine



MOTIVATION

Do we really need to add parameters to adapt a model?

Pretrained models adapt to new tasks with tiny changes — yet every popular PEFT method bolts on new modules. Is that useful structure already inside the frozen weights?

Adapters · LoRA · prefixes

- Add new trainable modules on top of the frozen backbone
- Extra parameters, extra memory, extra inference cost
- Effective — but they don't explain why so little is needed

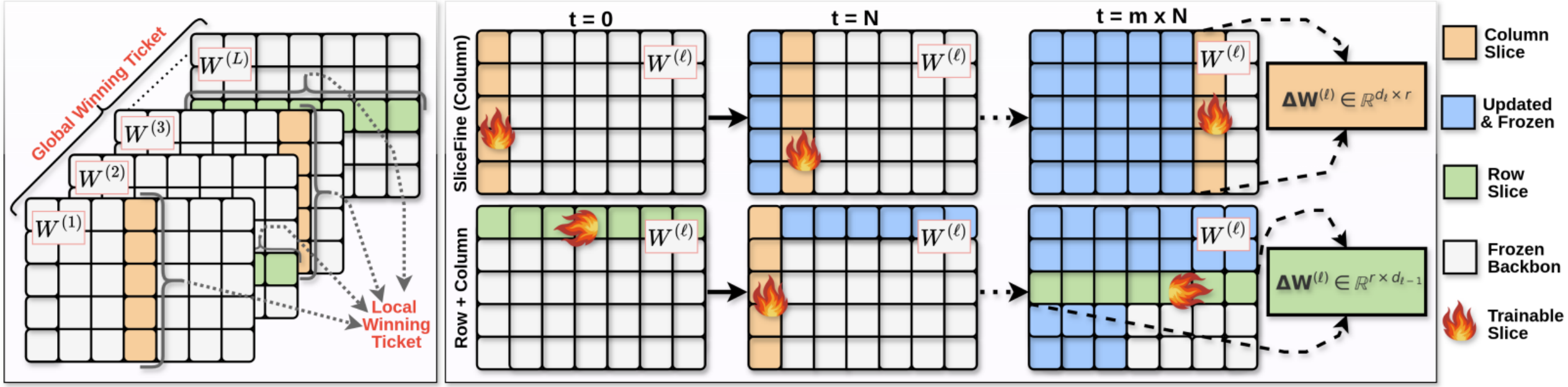
Reuse what's already there

- Fine-tune a small slice of the original weights
- Zero new parameters — dense and hardware-friendly
- Backed by a testable hypothesis (UWSH)

CORE IDEA

The Universal Winning-Slice Hypothesis

In a dense, pretrained network, any sufficiently wide slice of a weight matrix can be trained to lower the loss — and a few slices across layers match full fine-tuning.



WHY IT WORKS

Two properties make any slice a winner

1 · Spectral balance

Split a weight matrix into groups of rows / columns: their eigen-spectra are nearly identical. No slice is weak — which slice you pick barely matters.

inter-group dispersion $\rho \approx$ ***0.001***

2 · High task energy

The frozen backbone already concentrates task-relevant variance in a few directions (steep PCA / lazy NTK spectrum).

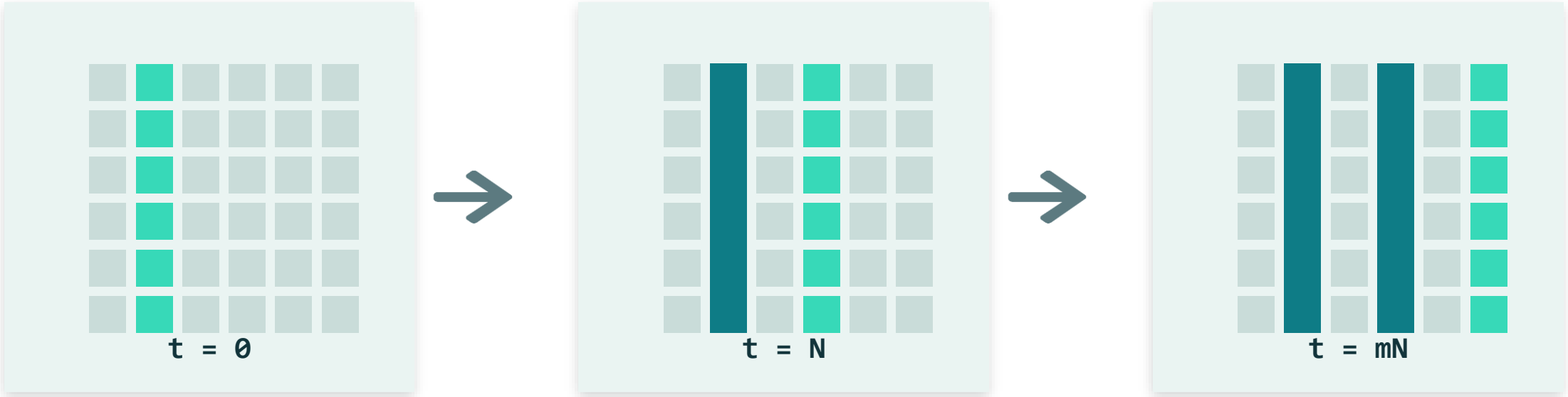
high explained variance → ***r = 1 often suffices***

Balanced strength + concentrated energy → any wide slice overlaps the task subspace, has a non-zero gradient, and lowers the loss.

METHOD

SliceFine: train one slice, then slide it

Update only the active slice; freeze it after N steps and activate a new position — block-coordinate descent across the matrix, with no parameters added.



active slice (trainable)

learned & frozen

frozen backbone

N ≈ 500 steps · 5–10 switches

RESULTS

Matches or beats strong PEFT baselines

LLM	Method	#TTPs	Commonsense Reasoning									Math Reasoning					
			BoolQ	PIQA	SIQA	H.Sw.	W.Gra.	ARCe	ARCc	OBQA	Avg.	M.Ar.	G.8K	A.S.	Se.Eq	S.MP	Avg.
LLaMA-3_{8B}	Prefix	38.09	70.85	81.91	78.66	79.98	75.11	74.40	59.88	73.08	74.23	87.29	71.66	84.24	82.93	54.20	76.06
	AdaLoRA	33.05	71.05	82.11	**82.27**	91.76	79.31	79.17	61.91	77.14	77.71	91.57	79.68	89.22	83.97	**63.51**	81.41
	VeRA	15.05	69.32	81.20	80.39	91.15	79.47	78.26	62.02	76.73	77.32	**92.16**	78.36	88.91	84.48	61.71	81.07
	LoRA	13.77	71.25	82.01	79.47	91.96	**80.29**	**79.83**	62.22	77.55	78.12	91.47	78.15	84.99	**84.96**	62.02	80.32
	RoCoFT	6.90	**72.47**	82.62	80.79	91.67	79.78	79.17	62.22	**79.37**	78.51	91.38	80.49	88.51	83.46	63.13	81.55
	HRA	**6.25**	72.27	**82.82**	80.29	91.97	79.37	79.07	62.52	**79.27**	78.44	92.18	80.29	89.42	83.36	62.83	81.62
	SliceFine-1R	**2.18**	70.88	81.16	79.12	90.19	78.04	78.58	61.95	78.73	77.33	91.65	79.85	88.36	**84.83**	62.55	81.45
	SliceFine-1C	**2.18**	70.94	81.08	78.95	90.98	78.00	77.56	61.14	77.70	77.04	91.44	78.80	87.21	82.72	61.73	80.38
	SliceFine-1RC	**2.18**	71.02	82.46	**81.26**	**92.61**	80.28	79.80	61.93	79.18	78.66	92.12	80.11	**89.77**	82.18	63.55	81.55
	SliceFine-5R	6.89	71.96	82.40	81.20	92.58	80.23	79.77	**62.88**	78.92	**78.74**	**92.15**	**81.05**	89.70	84.11	63.50	**82.08**
	SliceFine-5C	6.89	**72.49**	**82.86**	80.97	91.97	79.71	**79.82**	62.47	78.40	78.73	92.11	80.52	89.11	83.55	63.08	81.72
	SliceFine-5RC	6.89	72.01	82.45	81.25	**92.64**	80.27	**79.82**	**62.92**	78.97	**78.79**	92.10	**81.10**	**89.75**	84.16	**63.53**	**82.13**

Table 1: Commonsense and math reasoning with LLaMA-3B. SliceFine (shaded) rivals or surpasses baselines such as LoRA and AdaLoRA while using far fewer trainable parameters (#TTPs).

82.1%

best math-reasoning avg. — tops every baseline

1.25M

trainable params vs LoRA 13.8M · AdaLoRA 33M

88.9 / 73.1

VTAB-1K image & video avg. — also SOTA-level

Consistent across LLaMA-3B, Gemma-3-12B, DeepSeek-R1-8B, RoBERTa, ViT & VideoMAE.

EFFICIENCY

Smaller, faster, leaner — for free

0

new parameters
added (#APs = 0)

~18%

less peak
GPU memory

15-25%

higher training
throughput

42-50%

less wall-clock
training time

Throughput (iterations / sec — higher is better)

Structured slices keep dense GEMM kernels

Contiguous rows / columns stay hardware-friendly — unlike random masks, which match accuracy but lose speed to scattered memory access.

Takeaway

Adaptation doesn't need new modules — reuse the structure already in the weights.

A testable hypothesis

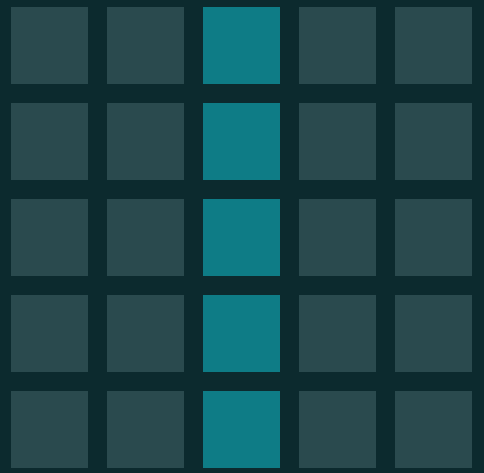
UWSH: any wide slice is a local winner; a few across layers form a global winner.

A parameter-free method

SliceFine trains sliding row/column slices — zero added parameters, dense compute.

Strong, broad results

Matches or beats LoRA / AdaLoRA across text, image & video with less memory and time.



github.com/facebookresearch/SliceFine