— ICML 2026

Unifying Stacking and Cascading for Efficient Ensemble Inference

LazyStack

Ashwin Colaço, Sharad Mehrotra, Michael De Lucia, Kevin Hamlen, Murat Kantarcioglu, Latifur Khan, Ananthram Swami, Bhavani Thuraisingham, Unnat Jain

UC Irvine · Army Research Laboratory · UT Dallas · Virginia Tech

ICML 2026 · Forty-third International Conference on Machine Learning

— THE PROBLEM

Ensembles are accurate — but you pay for every model

Combining many models lifts accuracy. But running all of them on every input is expensive, and models differ wildly in cost.

Within a single ensemble, the most expensive model can cost **3–11×** the cheapest (vision), **17×** (LLMs), or even more on tabular tasks.

The two levers, in tension

Accuracy ↑ wants *more* models

Cost ↓ wants *fewer* models

Most inputs are easy and need just one or two. A few are hard and need the whole ensemble.

— EXISTING APPROACH · 1

Stacking — run all models, fuse their predictions

input · leopard

MobileNetV2 8.2 ms	→ jaguar	0.38
ResNet-18 12.4 ms	→ leopard	0.52
EfficientNet-B0 15.1 ms	→ leopard	0.71
ResNet-50 21.3 ms	→ tiger	0.46
EfficientNet-B4 28.7 ms	→ leopard	0.88
ViT-B/16 30.5 ms	→ leopard	0.83

→

meta-learner

↓

leopard

`cost = 116.2 ms`

Accurate: the meta-learner exploits agreement and per-model reliability. **But you pay the full cost on every input.**

— EXISTING APPROACH · 2

Cascading — exit on single-model confidence (or fall through)

EASY INPUT · EXITS AT MODEL 2

easy input · sunflower

MobileNetV2	~~sunflower~~	0.72	< θ
ResNet-18	**sunflower**	0.91	EXIT ✓
EfficientNet-B0	—	0.00	—
ResNet-50	—	0.00	—
EfficientNet-B4	—	0.00	—

prediction: **sunflower** ✓ — 20.6 ms

HARD INPUT · FALLS THROUGH

hard input · leopard

MobileNetV2	~~jaguar~~	0.38	< θ
ResNet-18	~~leopard~~	0.52	< θ
EfficientNet-B0	~~leopard~~	0.71	< θ
ResNet-50	~~tiger~~	0.46	< θ
EfficientNet-B4	**leopard**	0.83	no exit

prediction: **leopard** ✓ — 85.7 ms · full!

Easy inputs exit early on a single confident model. On **hard** inputs no single model crosses θ — the cascade runs *everything* (full ensemble cost), and earlier predictions are still thrown away.

— THE GAP

No method learns both the order and the aggregation

METHOD	EXECUTION ORDER	AGGREGATION ACROSS MODELS
Cost Cascade	cheapest-first heuristic	none — last model only
ABC	accuracy-order heuristic	fixed averaging
FrugalGPT	cost-order heuristic	none — quality judge
RouteLLM	routes to one model	none — no sequence
LazyStack	**learned (MDP) ✓**	**learned (substacker) ✓**

— OUR IDEA · LAZYSTACK

Progressive stacking — same hard input, earlier exit

same hard input · leopard

MobileNetV2 8.2 ms	THIS MODEL → **jaguar** 0.38	SUBSTACKER · ALL SO FAR	agg 0.38
ResNet-18 12.4 ms	THIS MODEL → **leopard** 0.52	SUBSTACKER · ALL SO FAR	agg 0.61
EfficientNet-B0 15.1 ms	THIS MODEL → **leopard** 0.71	SUBSTACKER · ALL SO FAR	EXIT ✓

+ 2 idle ResNet-50 · EfficientNet-B4 — never run

decision uses
aggregated conf

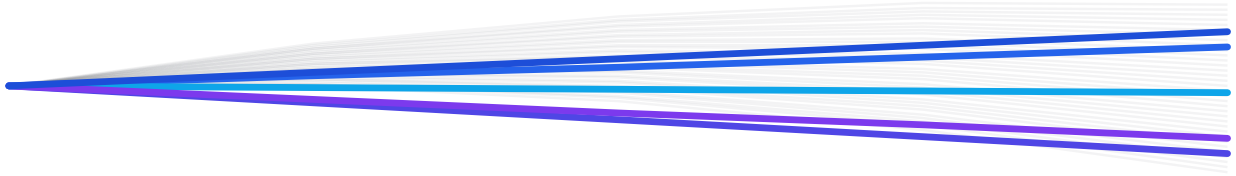
leopard

cost = 35.7 ms

A learned **substacker** re-aggregates *all* predictions so far. Three models — none individually confident — together exceed θ. Exit in **35.7 ms** instead of the cascade's full 85.7 ms.

— WHY IT'S TRACTABLE

Samples concentrate on a few execution paths



Out of 2^k possible orderings, just a handful are taken in practice.

Across every benchmark we tested, the top 3–8 trajectories already cover 95%+ of inputs. We only need to learn aggregators for those paths — not all 2^k.

how it works

— METHOD

Two offline stages, one online loop.

— OVERVIEW

Discover order → learn aggregators → exit early

OFFLINE · 1

Trajectory Discovery

An **MDP** learns which model to run next and when to stop — discovering the trajectories that matter.

OFFLINE · 2

Prefix Meta-Learning

Train lightweight **substackers** for the **prefixes** of those trajectories — not all 2^k subsets.

ONLINE

Early-Exit Inference

Follow the policy, aggregate with the matching substacker, **exit** once confident.

— STAGE 1 · OFFLINE

Trajectory discovery as an MDP

State `(E, h, k)`

executed-set **E** · entropy bin **h** · last predicted class **k**

Actions

`EXEC(Mⱼ)` run a model · `STOP`

Reward

$\beta \cdot \mathbb{1}[\text{correct}] - \Sigma\ \text{cost}$

Pay per model run; get paid for being right. Solved by **value iteration**.

Roll out the policy → keep the top trajectories (each ≥ 2% of inputs).

— WHAT THE MDP LEARNS

The cheapest model first is not optimal

Cheapest-first (cost cascade) — **70.6 ms**

MobileNetV2	ResNet-18	ResNet-50	EfficientNet-B4
8.2 ms	12.4 ms	21.3 ms	28.7 ms
			EXIT ✓

LazyStack (MDP-discovered) — **36.4 ms**

EfficientNet-B0	ResNet-50	ViT-B/16	EfficientNet-B4
15.1 ms	21.3 ms	30.5 ms	28.7 ms
	EXIT ✓		

total cost to a confident answer

The MDP discovers that a **costlier but more confident** first model can be cheaper overall.

Higher confidence triggers an earlier exit, more than paying back the extra cost. No cheapest-first heuristic finds this.

— STAGE 2 · OFFLINE

Train aggregators only for the prefixes that occur

2^k

all subsets — intractable

→

$O(k)$

prefixes of discovered trajectories

A trajectory $[M_2, M_5, M_1]$ contributes prefixes $[M_2]$, $[M_2, M_5]$, $[M_2, M_5, M_1]$.

Trajectories **share prefixes** → only **~12–18** substackers in practice.

— TWO VARIANTS

Specialized substackers, or one masked stacker

LAZYSTACK-SUB

One MLP per prefix

Input = concatenated probs of executed models.

+0.23–0.33% accuracy (specialization)

14–22 stacker artifacts (8–12 MB total).

vs

LAZYSTACK-MASK

One shared MLP + binary mask

Unexecuted slots zero-filled; a mask says which ran.

up to 79.6× on NSL-KDD · simpler deploy

A single model artifact to ship.

— ONLINE · INFERENCE

Run, aggregate, exit when confident

input · leopard

ResNet-50
21.3 ms
→ tiger
0.46

EfficientNet-B0
15.1 ms
→ leopard
0.71

EfficientNet-B4
28.7 ms
→ leopard
0.88

+ 3 idle ResNet-18 · MobileNetV2 · ViT-B/16 — never run

`exit · 3/6 models · prediction leopard · aggregated 0.88`

Illustrative input. θ is calibrated on validation to retain ~97% of full-ensemble accuracy.

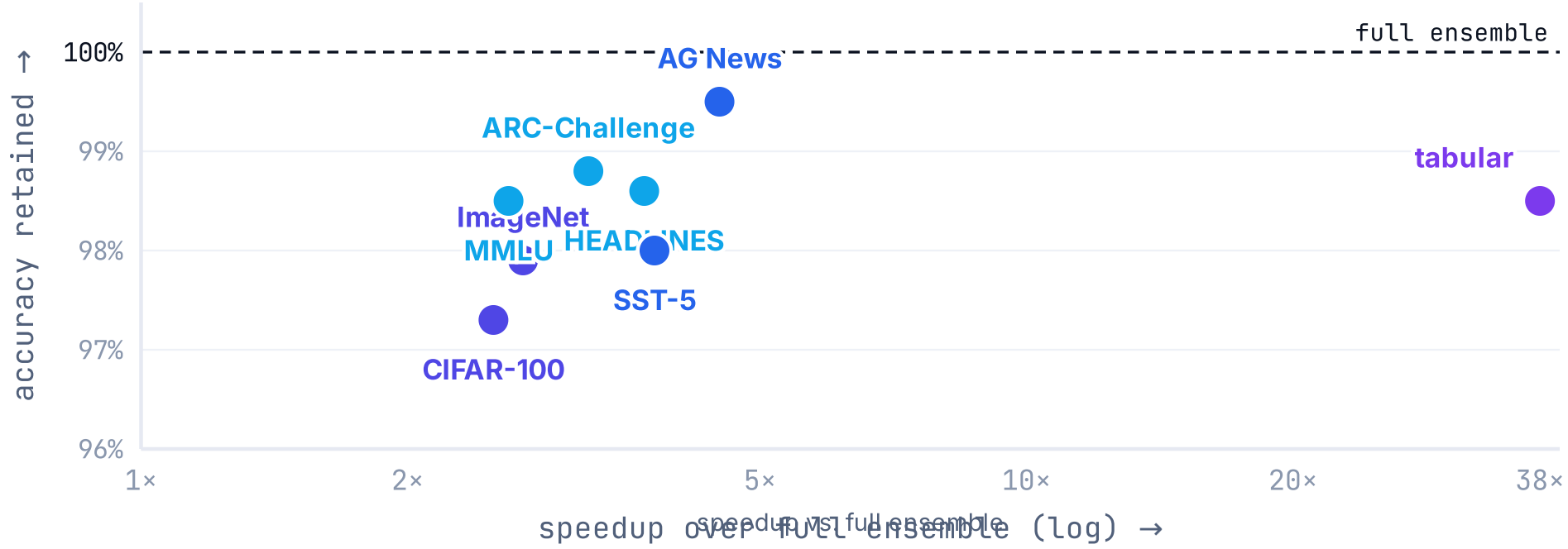
does it work

— EVALUATION

8 datasets, 4 domains. Vision · tabular · text · LLM routing.

— OVERVIEW · 8 BENCHMARKS

97%+ of full-ensemble accuracy across every benchmark

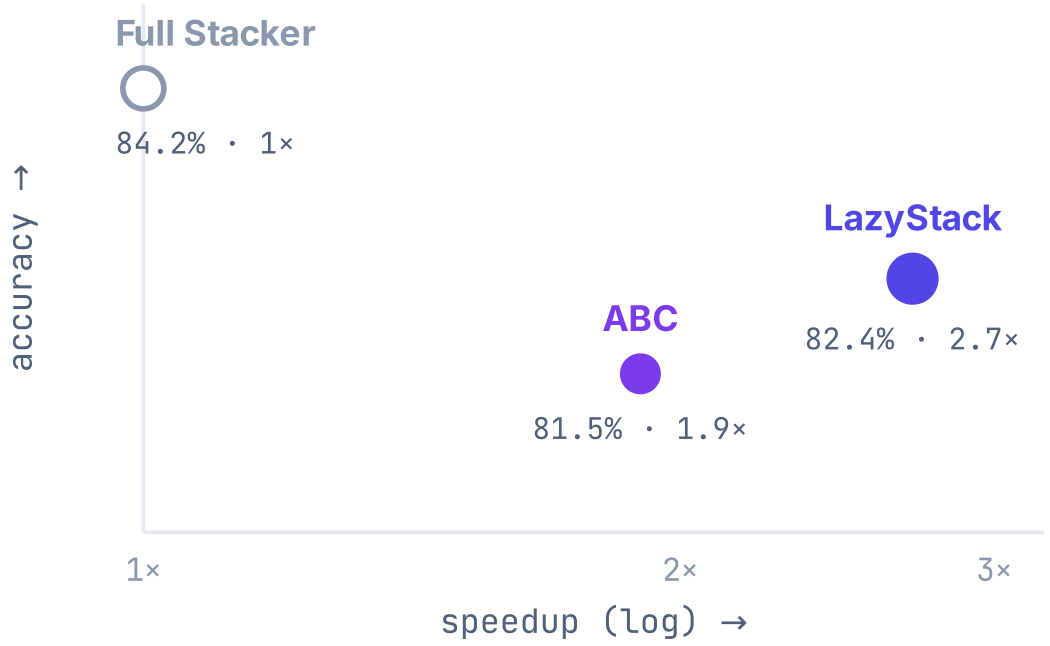


vision · LLM routing · text · tabular

Every benchmark retains 97%+ of full-ensemble accuracy while using only ~2–3 models per input on average. Speedup ranges from 2.5× on many-class vision to 38× on tabular (cost ratio 65×).

SPOTLIGHT · IMAGENET-1K

More accurate than the baseline, near the full ensemble



LazyStack: **82.4%** at 2.7× — beats ABC on both axes.

The full ensemble caps at 84.2% (1×). LazyStack keeps almost all of it while running ~2.7 of 8 models per image.

SAME IDEA · LLM ROUTING

Stop at the smallest model that's confident

ARC-CHALLENGE

A candle in a sealed jar of CO_2 goes out almost immediately. Which best explains why?

A. fuel ran out

B. jar absorbed heat

C. no O_2 to combust

D. CO_2 reflected light

Qwen2.5-7B 48 ms	→ **A**	0.43
LLaMA-3.1-8B 52 ms	→ **C**	0.61
Qwen2.5-32B 180 ms	→ **C**	0.82
LLaMA-3.1-70B 820 ms	→ —	0.00
Qwen2.5-72B 890 ms	→ —	0.00

exit · 3/5 models · prediction **C** · aggregated 0.82

Illustrative cascade. Two large 70B+ models stay idle; the substacker exits at 280 ms instead of running the full ~2 s ensemble.

— GENERALITY

The pattern holds across every domain

DATASET	DOMAIN	LAZYSTACK	FULL STACKER	SPEEDUP
ImageNet-1K	vision	82.4%	84.2%	2.7×
CIFAR-100	vision	75.9%	78.0%	2.5×
MMLU	LLM routing	77.2%	—	2.6×
ARC-Challenge	LLM routing	88.2%	—	3.2×
HEADLINES	LLM routing	64.3%	—	3.7×
AG News	text	95.3%	95.8%	4.5×
SST-5	text	87.4%	89.2%	3.8×
NSL-KDD	tabular	76.8%	78.0%	38×

LLM Full Stacker numbers omitted; paper reports 97%+ retention at 2.6–3.9× speedup. Speedups for LLMs are relative to always-70B.

— STRONG CLAIM

Black-box LazyStack matches a white-box method

IMAGENET · MATCHED SPEEDUP	ACCESS	ACC	SPEEDUP
Gatekeeper ($M_2 \rightarrow M_8$)	white-box	81.6%	2.7×
LazyStack	**black-box**	**82.4%**	**2.7×**

CIFAR-100: 75.9% vs 75.8% at comparable speed.

At the same speedup, LazyStack beats Gatekeeper by **+0.8%** — despite Gatekeeper having white-box access to fine-tune internals.

LazyStack sees only output probabilities. Black-box access is not a handicap on vision.

ABLATION · IMAGENET

Both learned components are essential — and superadditive

Remove either and speedup drops; remove both and you collapse to running the whole ensemble. Same pattern across MMLU, AG News, and the tabular task.

— TAKEAWAYS

Run the ensemble lazily

1. Progressive stacking turns ensemble accuracy into cascade cost.
2. Samples concentrate on a few trajectories — so learning is tractable.
3. Works black-box across vision, text, tabular, and LLM routing.

Honest limits

- Needs labeled validation data.
- State space $O(2^N \cdot B \cdot C)$ → function approximation past ~15 models.
- Many classes spread confidence thin → smaller speedups.
- Generative LLM tasks remain open.