ComputerVisionLab
Seoul National University

NEURAL INFORMATION PROCESSING SYSTEMS

Greedy Alignment Principle

for Optimizer Hyperparameter Selection

Jaerin Lee and Kyoung Mu Lee
Computer Vision Lab, Seoul National University

NeurIPS 2026, arXiv:2512.06370

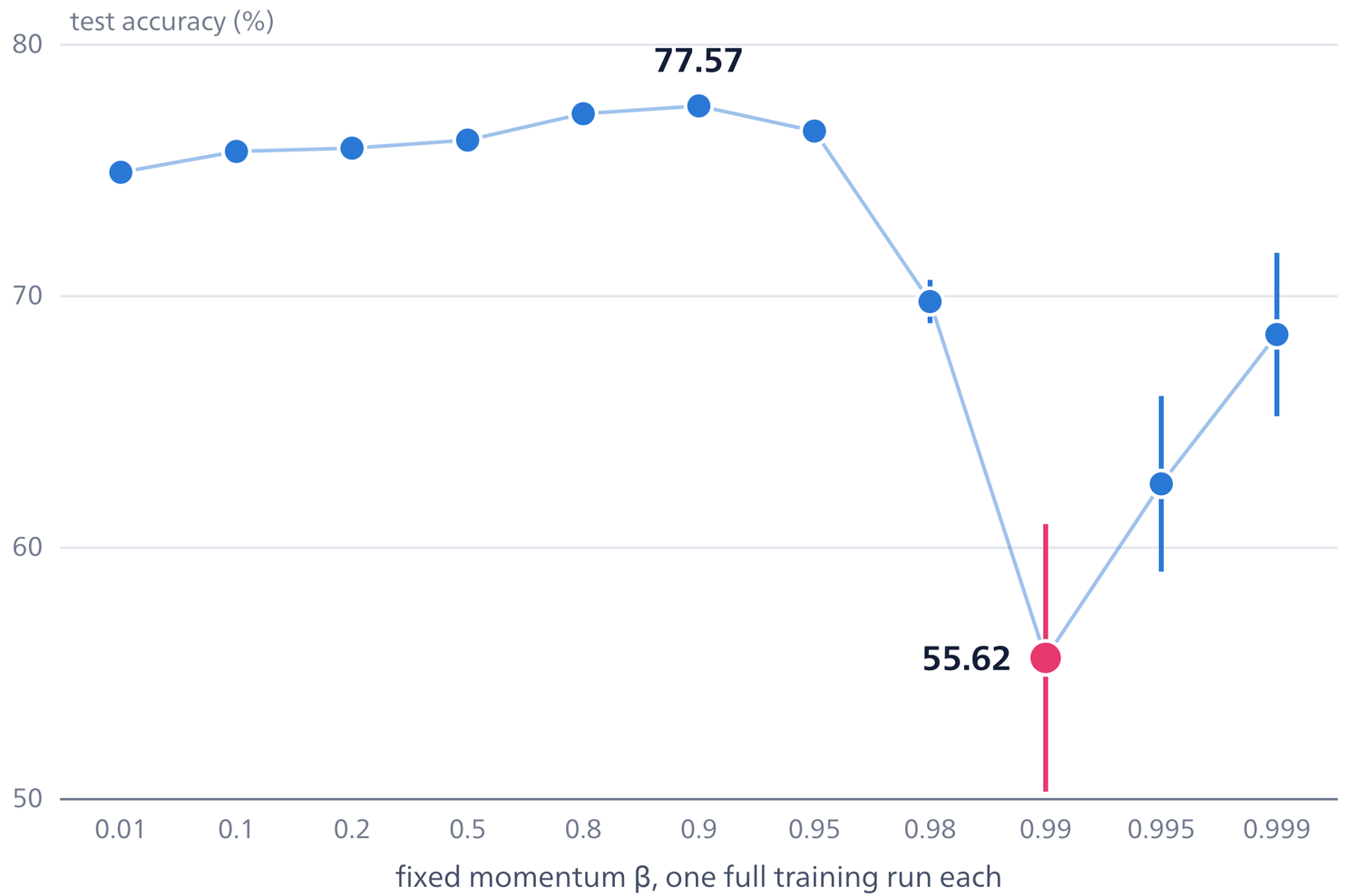
jaerinlee.com

Optimizers update model parameters.

Why not let models update optimizer hyperparameters?

Picking the momentum β is still trial and error

SGD+Momentum, ResNet-18 on CIFAR-100. Test accuracy for each fixed β; dot = mean, bar = ± std of 10 runs.



$$\mathbf{m}_t = \beta\,\mathbf{m}_{t-1} + \mathbf{g}_t$$

Momentum β sets how much of the past update is kept.

77.57

best accuracy, at **β = 0.9**

55.62

at **β = 0.99**, about 22 points lower

11

values tried, each a **full training run**

A sweep pays one full run per candidate

Fixed-β sweep



11 full runs, keep 1

What we want

β chosen again at every step



1 run, no sweep

And the answer does not transfer: the best value is 0.9 here, 0.8 with Adam, and 0.95 or 0.5 for Gemma-2B with LoRA or full fine-tuning.

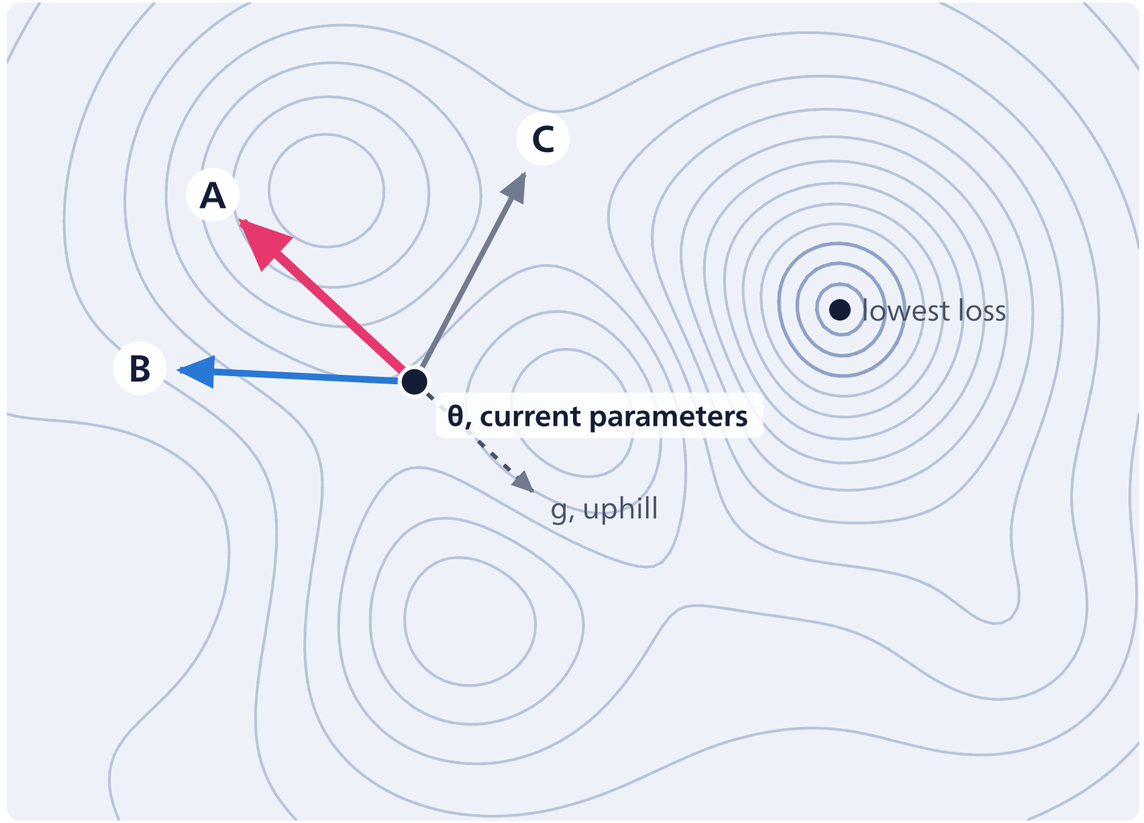
Alignment is the speed of learning

An update ***u*** is the speed of the parameters: $\mathrm{d}\theta/\mathrm{d}t = -\boldsymbol{u}$. The gradient ***g*** = ∇L points uphill.

$$\mathbf{g}_t^\top \mathbf{u}_t \;=\; -\nabla_\theta \mathcal{L}^\top \frac{\mathrm{d}\theta}{\mathrm{d}t} \;=\; -\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}t}$$

To let the model choose, we need a score. By the chain rule, alignment is the rate at which the loss drops.

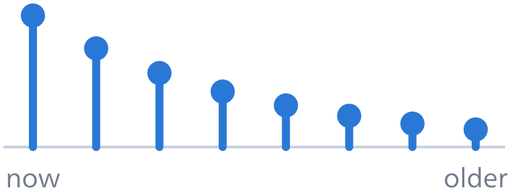
Prior work uses it to repair single updates. **We use it to choose the optimizer.** It is a local rule: step A is the fastest way down from here, not a path to the lowest loss.



alignment with –g, for three candidate steps –u of equal length



Background: every update rule unrolls into a filter

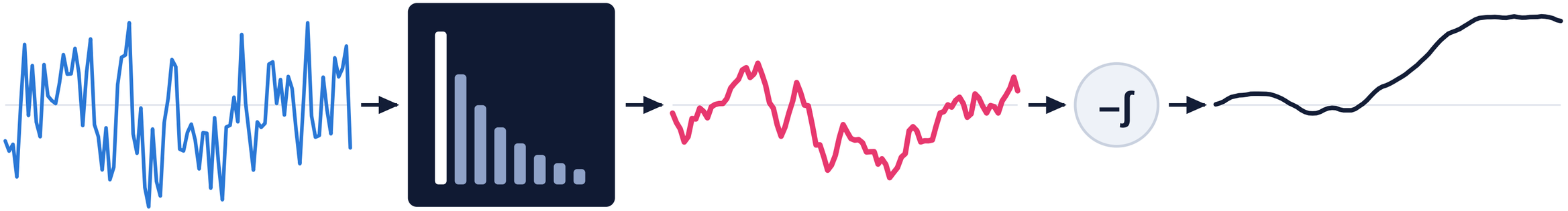
the update rule you write	the same rule, unrolled over time	weight on each past gradient
SGD		
$\mathbf{u}_t = \eta\, \mathbf{g}_t$	$\mathbf{u}_t = \eta\, \mathbf{g}_t$	
SGD+Momentum		
$\mathbf{m}_t = \beta\, \mathbf{m}_{t-1} + \mathbf{g}_t, \quad \mathbf{u}_t = \eta\, \mathbf{m}_t$	$\mathbf{u}_t = \eta\, \left(\mathbf{g}_t + \beta\, \mathbf{g}_{t-1} + \beta^2 \mathbf{g}_{t-2} + \cdots\right)$	
Adam		
$\boldsymbol{\mu}_t = \beta_1 \boldsymbol{\mu}_{t-1} + (1-\beta_1)\mathbf{g}_t, \quad \mathbf{u}_t = \eta\, \boldsymbol{\mu}_t / \mathbf{c}_t$	$\mathbf{u}_t = \frac{\eta(1-\beta_1)}{\mathbf{c}_t}\left(\mathbf{g}_t + \beta_1 \mathbf{g}_{t-1} + \beta_1^2 \mathbf{g}_{t-2} + \cdots\right)$	

Every first-order update is a weighted sum of past gradients. Gradients are the input, updates the output, and the weights are a causal filter: the optimizer.

The signal view of gradients follows Grokfast (Lee et al., 2024). $\boldsymbol{c}_t$ is Adam's per-coordinate scale, $\sqrt{(\boldsymbol{v}_t + \varepsilon)}$.

An optimizer is a filter on the gradient stream

An optimizer is one object: a filter Q. Choosing it means choosing its weights.



gradients g
noisy, correlated

optimizer Q
impulse response

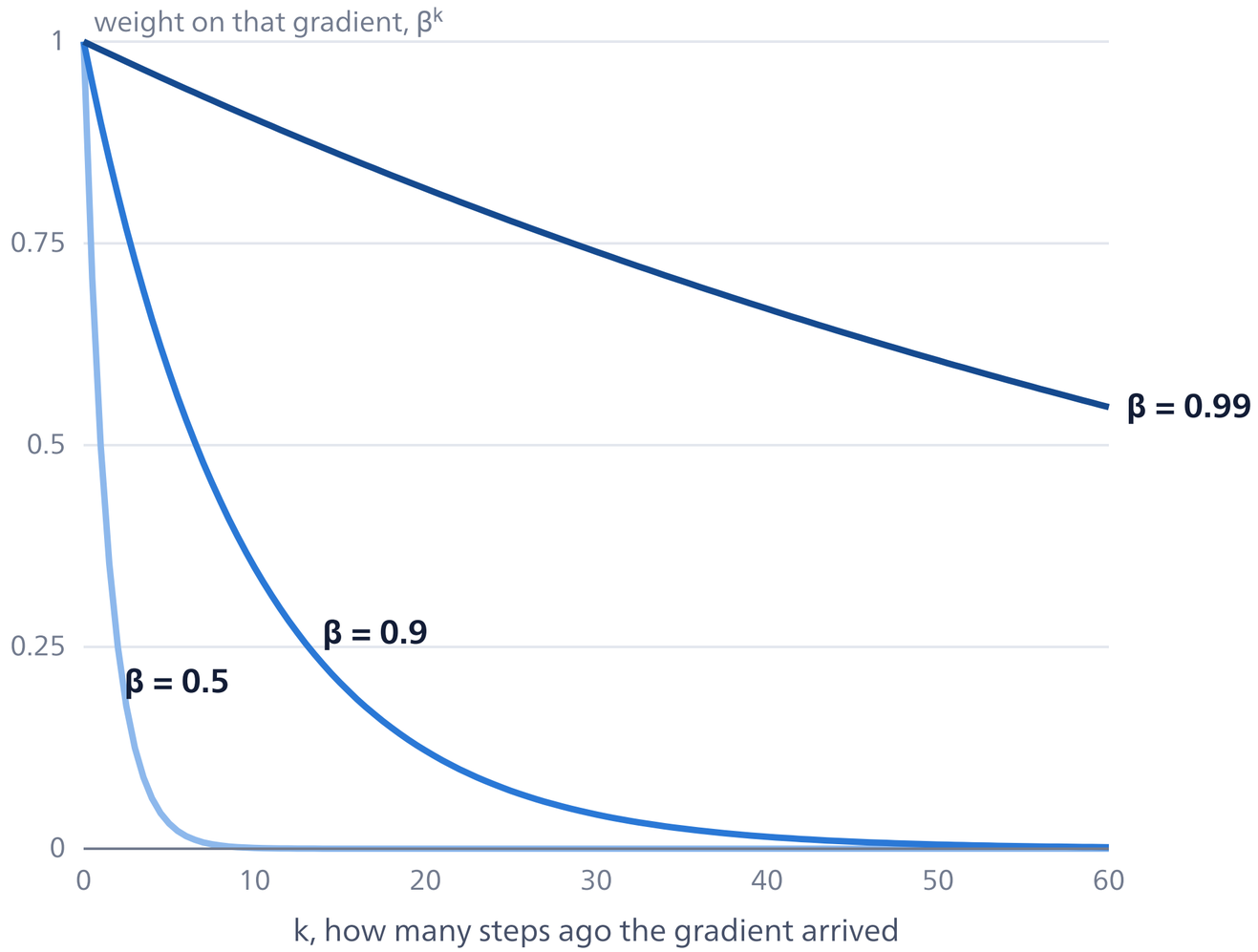
updates u
speed of the parameters

add up

parameters θ
θ ← θ – u

$$\mathbf{u}_t \;=\; (Q\,\mathbf{g})_t \;=\; \sum_{k=0}^{\infty} Q_k\,\mathbf{g}_{t-k}, \qquad\qquad \theta_{t+1} \;=\; \theta_t - \mathbf{u}_t$$

Momentum is a decaying impulse response



SGD+Momentum

$$Q_k \;=\; \eta\,\beta^k\,I$$

Adam

$$Q_k \;=\; \eta\,(1-\beta_1)\,\beta_1^k\;\mathrm{diag}(\mathbf{c})^{-1}$$

$\mathbf{c} = \sqrt{(\boldsymbol{v} + \varepsilon)}$ is Adam's per-coordinate scale.

A gradient k steps old keeps weight β^k.
Small β forgets quickly; large β retains.

Learning power is an inner product

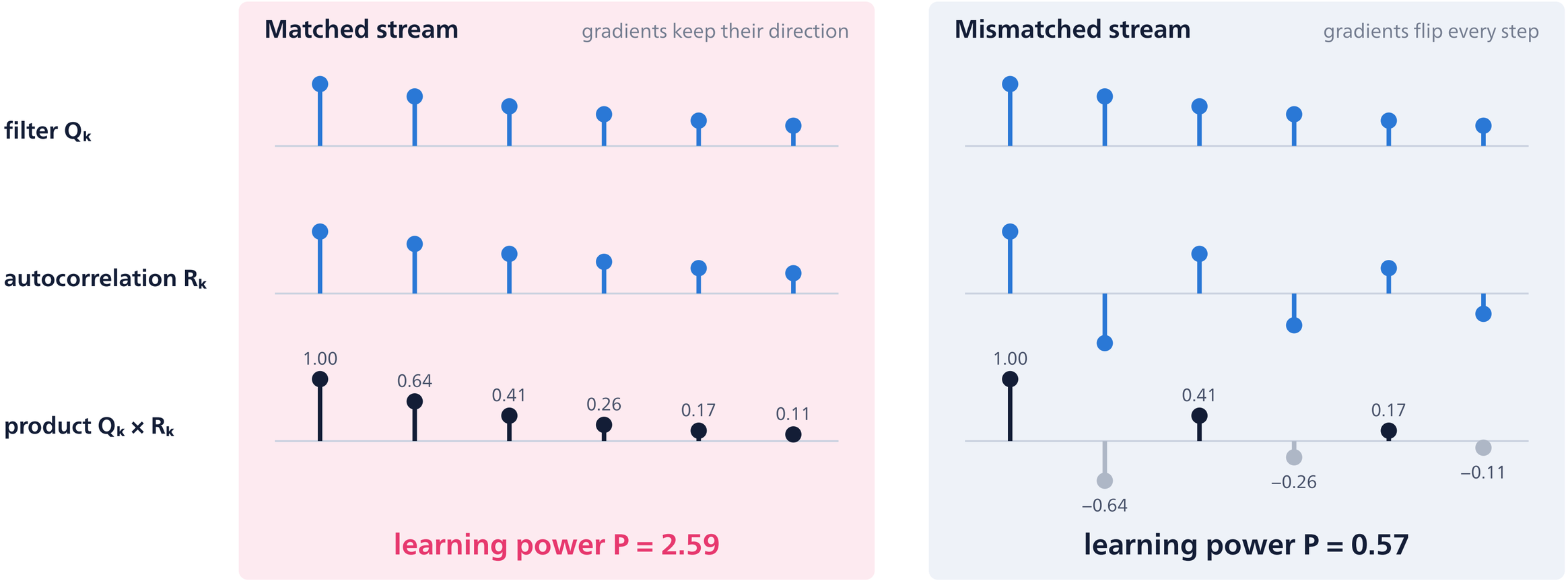
$$P_t(Q) \;=\; \mathbb{E}\big[\mathbf{g}_t^\top \mathbf{u}_t\big] \;=\; \sum_{k=0}^{\infty} \operatorname{Tr}\big(Q_{t,k}^\top R_{t,k}\big) \;=\; \langle Q_t,\, R_t \rangle_{\mathcal{H}}$$

Learning power $P \;=\; -\,\mathbb{E}\Big[\dfrac{\mathrm{d}\mathcal{L}}{\mathrm{d}t}\Big]$ how fast the loss falls, on average

Gradient autocorrelation $R_{t,k} \;=\; \mathbb{E}\big[\mathbf{g}_t\, \mathbf{g}_{t-k}^\top\big]$ today's gradient against the one *k* steps ago

The same filter helps one stream and hurts another

Scalar example over lags 0 to 5: the momentum filter $Q_k = 0.8^k$ meets two gradient streams.



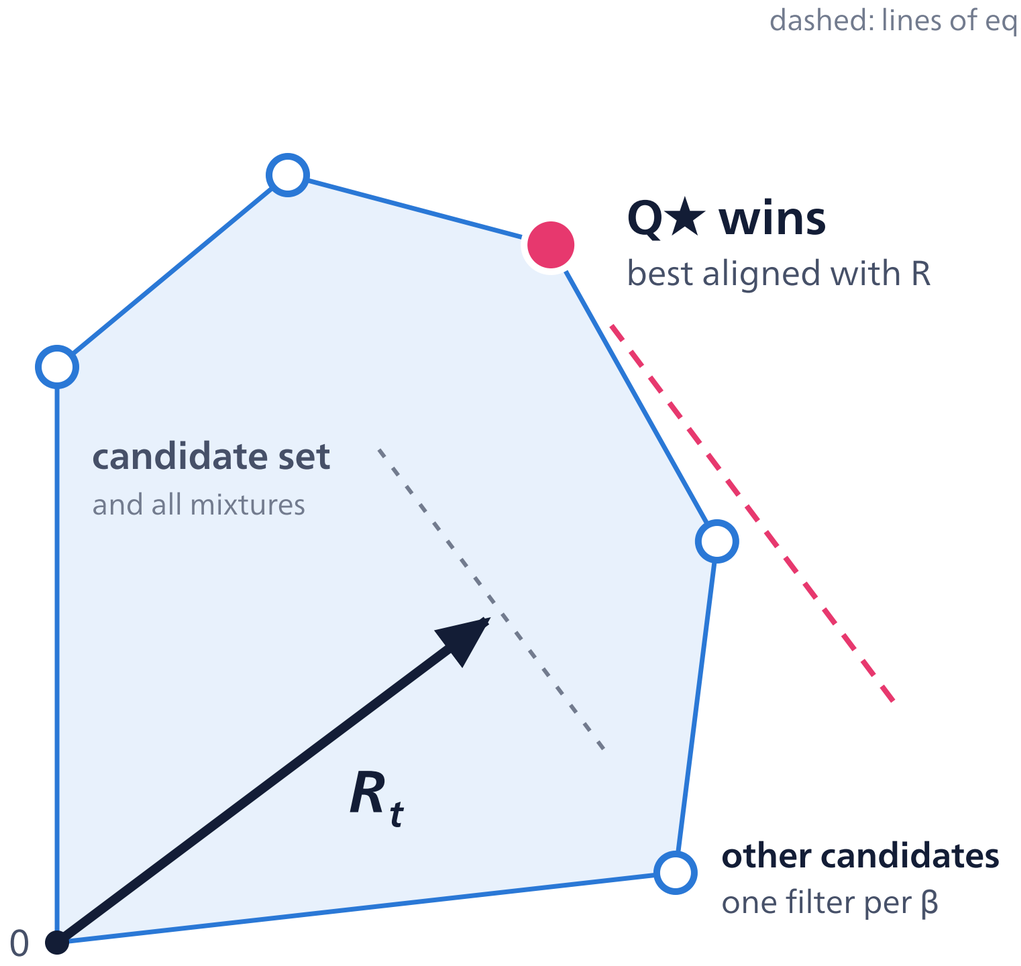
Plain SGD scores $P = 1.00$ on both. Momentum helps only while gradients stay alike.

The Greedy Alignment Principle

$$Q_t^\star \;=\; \arg\max_{Q \in \mathcal{Q}} \langle Q,\, R_t \rangle_{\mathcal{H}}$$

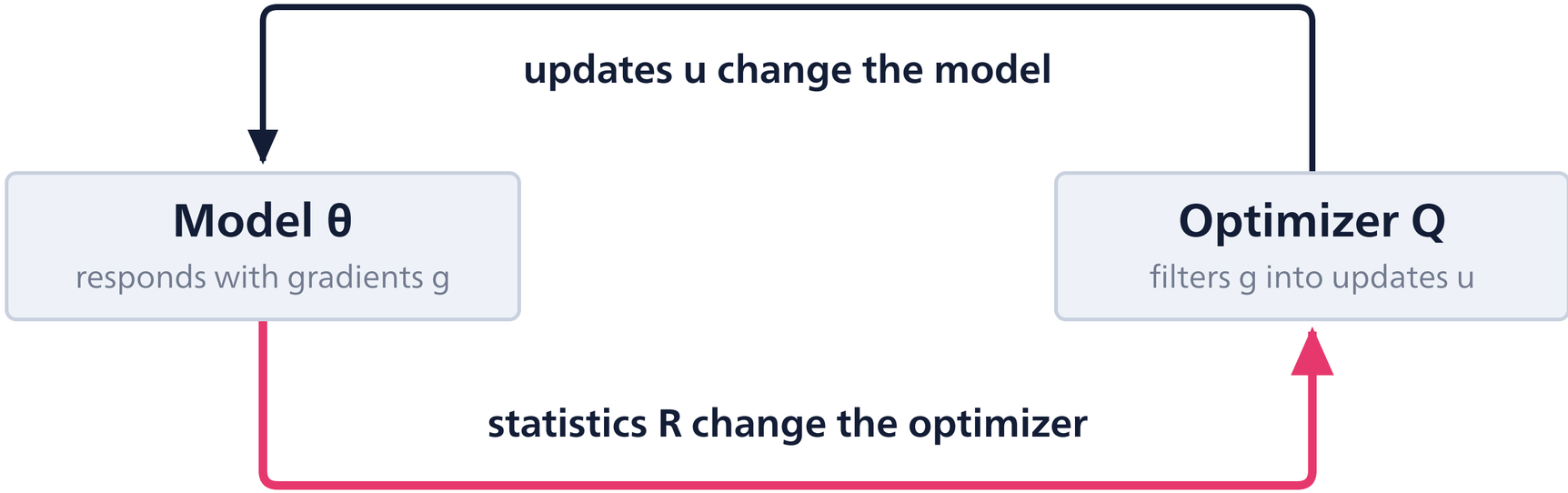
At every step, pick the filter that best matches the autocorrelation of the gradients.

The objective is linear and the candidate set is convex, so the optimum sits at a **vertex**. Scoring a finite list of candidates is **exact**, not a heuristic.



Two problems, solved together

Training minimizes the loss $\min_{\theta} \mathcal{L}(\theta)$



We maximize its drop rate $\max_{Q \in \mathcal{Q}} \langle Q, R_t \rangle_{\mathcal{H}} = -\mathbb{E}\left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}t}\right]$

What the theory guarantees

Theorem 2.2 Greedy optimal first-order optimizers

Let $\mathcal{Q} \subset \mathcal{H}$ be a compact convex set of causal filters with $0 \in \mathcal{Q}$. For every $R \in \mathcal{H}$:

(i) $P^\star(R) = \max_{Q \in \mathcal{Q}} \langle Q, R \rangle_{\mathcal{H}}$ is attained, finite and sublinear.

A best optimizer **always exists**.

(ii) $Q^\star \in \partial_R P^\star(R)$, and $Q^\star = \nabla_R P^\star(R)$ if it is unique.

We can **construct** it: it is the slope of the best learning power.

(iii) $\left| P^\star(R) - P^\star(\hat{R}) \right| \le \|R - \hat{R}\|^{\mathrm{sym}}_{\mathcal{Q}^\circ}$ for all $R, \hat{R} \in \mathcal{H}$.

Noisy mini-batch statistics move the answer **only a little**.

Corollary 2.3 Family-hull reduction

For a bounded family $\mathcal{F}_\Phi = \{Q_\phi : \phi \in \Phi\}$ and $\mathcal{Q}_\Phi = \mathrm{cl}\,\mathrm{co}(\mathcal{F}_\Phi \cup \{0\})$:

$$\sup_{Q \in \mathcal{Q}_\Phi} \langle Q, R \rangle_{\mathcal{H}} = \left(\sup_{\phi \in \Phi} \mathbb{E}\left[\mathbf{g}_t^\top (Q_\phi \mathbf{g})_t \right] \right)_+$$

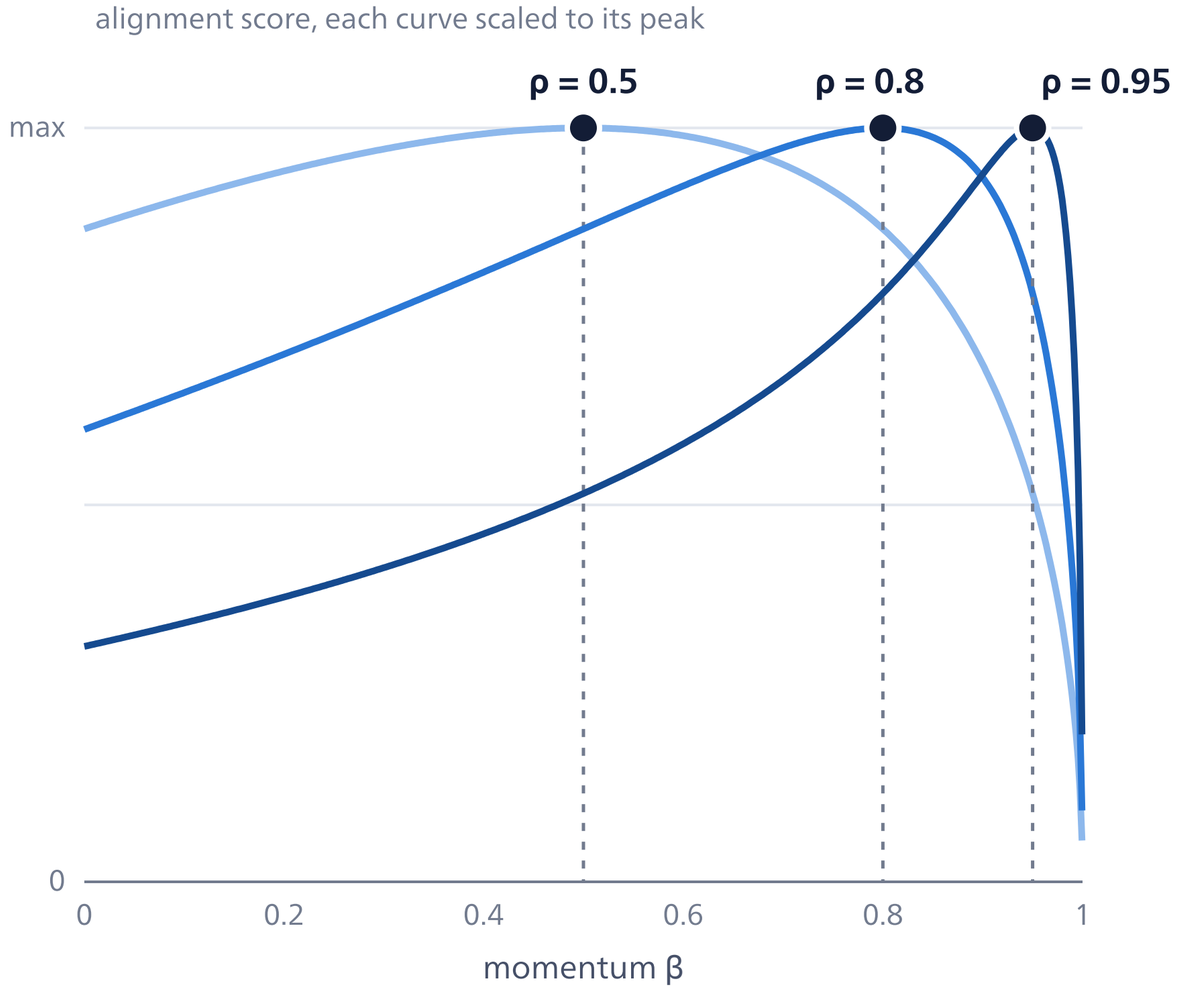
Scoring each hyperparameter setting and keeping the best is **exact**, not an approximation.

Momentum = gradient autocorrelation

$$\beta_t^\star = \arg\max_\beta \sqrt{1-\beta^2}\ \mathbb{E}\left[\mathbf{g}_t^\top \mathbf{m}_{\beta,t}\right]$$

If gradient correlation decays as ϱ^k, the score peaks exactly at $\beta\bigstar = \varrho$.

The factor $\sqrt{(1-\beta^2)}$ keeps the contest fair: it removes the pull toward no momentum or infinite momentum.



The same rule for Adam and AdamW

$$(\beta_1^\star, \beta_2^\star)_t \;=\; \arg\max_{\beta_1,\beta_2} \sqrt{\frac{1+\beta_1}{1-\beta_1}}\; \frac{1}{\sqrt{\sum_j c_j^{-1}}}\; \mathbb{E}\left[\mathbf{g}_t^\top \mathbf{u}_{\beta_1,\beta_2,t}\right]$$

Update	$\mathbf{u} \;=\; \boldsymbol{\mu}_{\beta_1} \odot \mathbf{c}^{-1}, \qquad c_j = \sqrt{v_{\beta_2,j} + \epsilon}$
What changes	Adam measures each parameter's step with **its own ruler** c_j. The choosing rule stays the same.

μ and ***v*** are Adam's first and second moment estimates. The score uses the gradient-driven update only; decoupled weight decay is applied identically to every candidate.

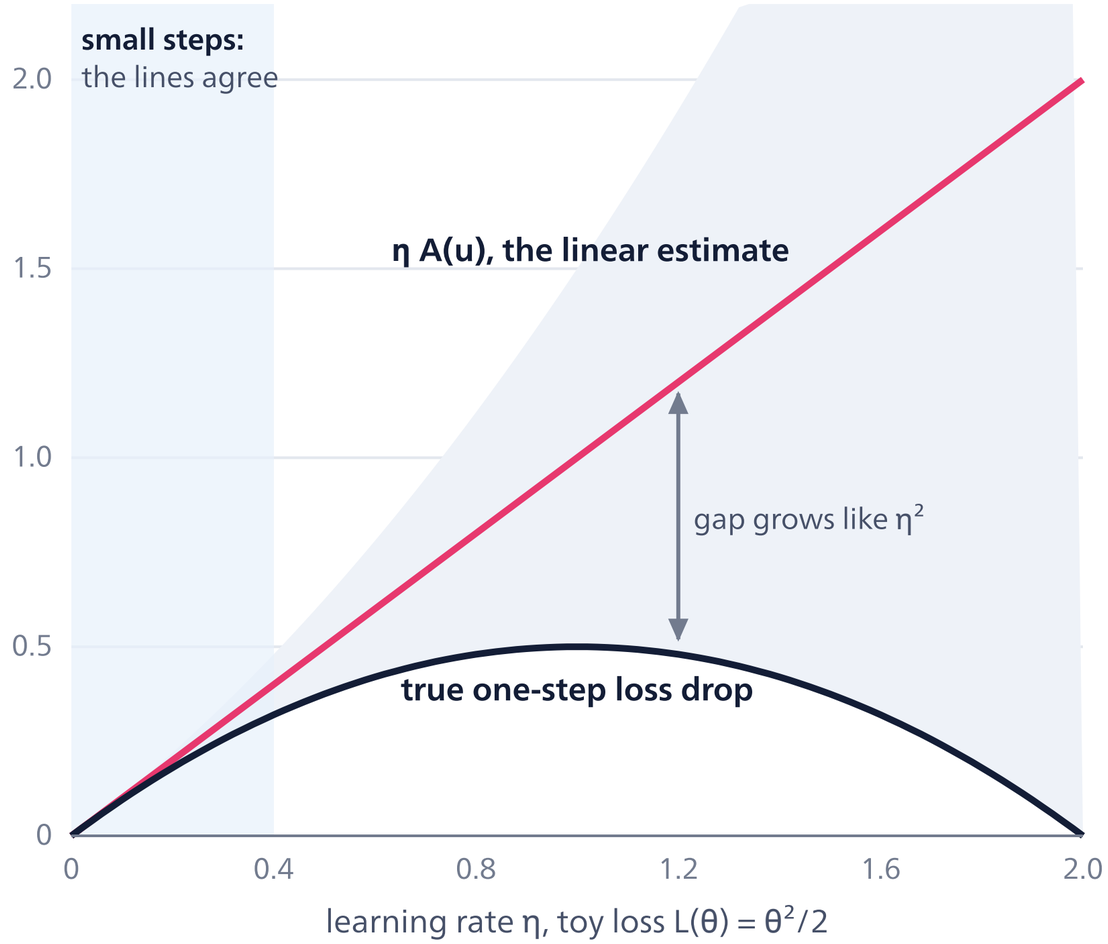
One greedy step is safe when the step is small

$$\left| D_\eta(\mathbf{u}) - \eta\, \mathbf{g}^\top \mathbf{u} \right| \;\le\; \frac{L_s \eta^2}{2} \, \|\mathbf{u}\|^2$$

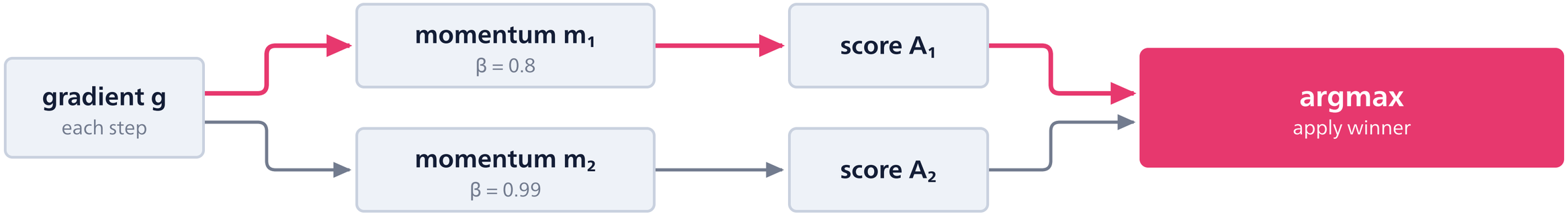
$D_\eta(\boldsymbol{u}) = L(\theta) - L(\theta - \eta \boldsymbol{u})$ is the true loss drop after one step; L_s is the smoothness of the loss.

Alignment is the true one-step loss drop up to a term in η^2.

Halving the step cuts the error fourfold, so the greedy pick is near-optimal whenever $\eta\|\boldsymbol{u}\|$ is small.



K-Switch: two extra lines in your optimizer




```
g = grad(loss, θ)          # every step
for k in 1..K:             # K candidates
  m[k] = β[k] * m[k] + g
  A[k] = sqrt(1 - β[k]²) * dot(g, m[k])
k⋆ = argmax(A)
θ = θ - η * m[k⋆]
```

Keep *K* momentum buffers, score each at every step, apply the best aligned.

***K* = 2** is enough: two endpoints, fixed in advance, that bracket the usual range.

K-Switch is exact, and robust to noisy scores

Corollary 4.1 Exact finite-candidate K-Switch

For $P_t(Q) = \mathbb{E}\big[\mathbf{g}_t^\top (Q\mathbf{g})_t\big]$ and normalized candidates $\tilde{Q}_{k,t}$ with $(\tilde{Q}_{k,t}\mathbf{g})_t = a_{k,t}\mathbf{u}_{k,t}$:

$$\sup_{Q \in \mathrm{co}\{\tilde{Q}_{k,t}\}_{k=1}^{K}} P_t(Q) \;=\; \max_k P_t(\tilde{Q}_{k,t}) \;=\; \max_k \mathbb{E}\big[a_{k,t}\, \mathbf{g}_t^\top \mathbf{u}_{k,t}\big]$$

No blend of candidates beats the single best one. K-Switch is the **exact** greedy optimum, not a heuristic.

Proposition 4.2 Online stability

Let $\hat{A}_k$ estimate A_k with $\max_k |\hat{A}_k - A_k| \le \varepsilon$, $\hat{k} \in \arg\max_k \hat{A}_k$, $k^\star \in \arg\max_k A_k$.

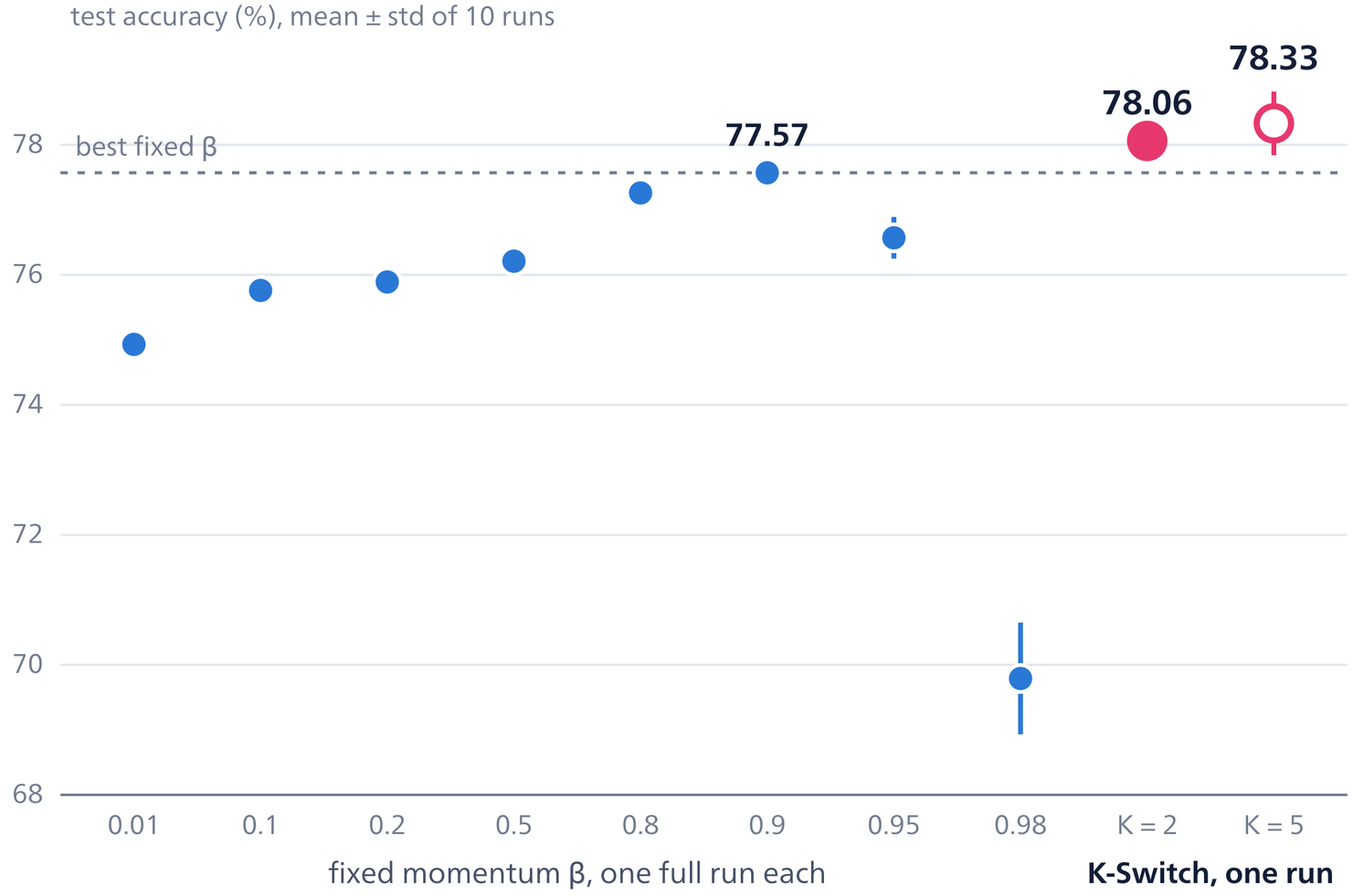
Then $A_{k^\star} - A_{\hat{k}} \;\le\; 2\varepsilon$.

If the best candidate is unique with a gap $> 2\varepsilon$, then $\hat{k} = k^\star$.

If mini-batch scores are off by at most ε, the pick loses **at most 2ε**. A clear enough lead gives exactly the right pick.

One run beats every fixed β

ResNet-18 on CIFAR-100, SGD+Momentum. K = 2 switches between β = 0.8 and 0.99; alone, β = 0.99 reaches only 55.62.



78.06

K-Switch, two options, one run

77.57

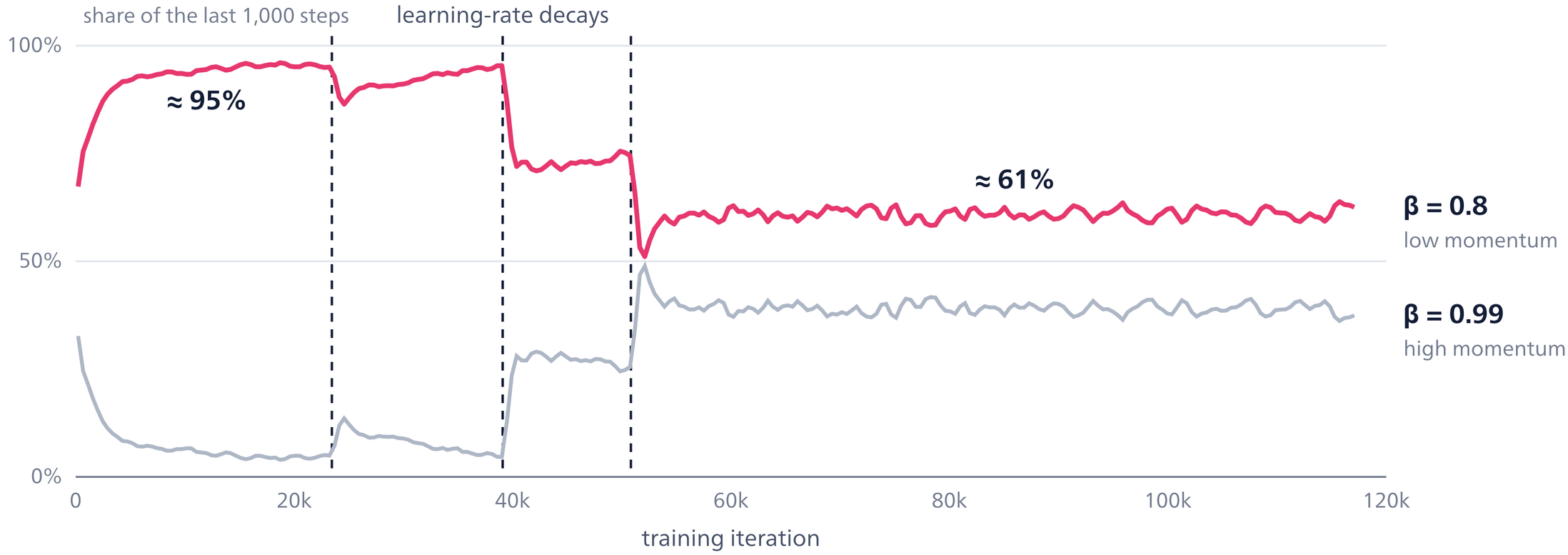
best fixed β, picked from 11 runs

78.33

K-Switch with five options

The preferred β moves during training

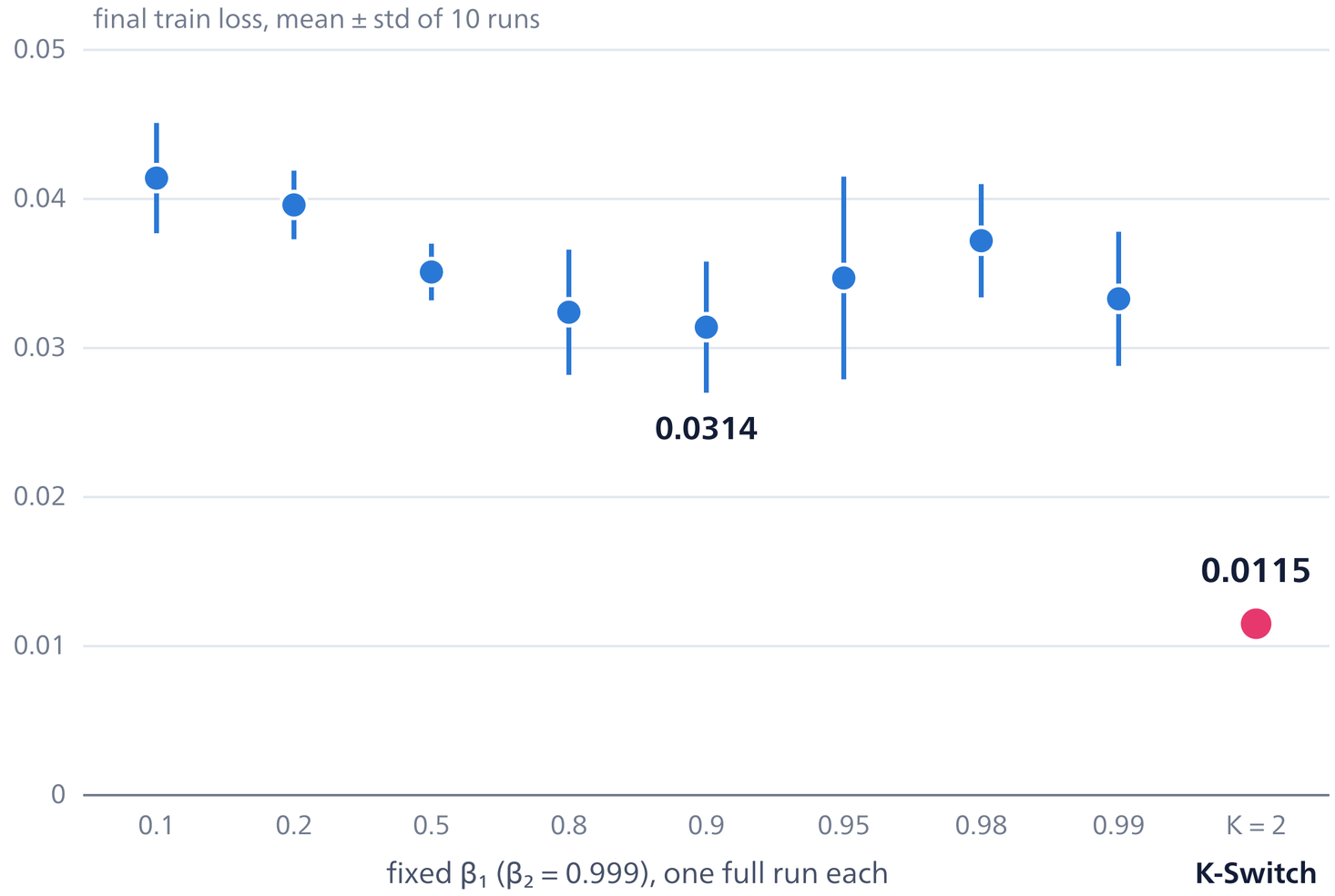
Inside the SGD+Momentum run of the previous slide: which of the two options K-Switch picked, over training.



The preference jumps at every learning-rate decay. No single fixed β suits the whole run.

Adam: same accuracy, far lower training loss

ResNet-18 on CIFAR-100, Adam. K = 2 switches between $\beta_1 = 0.8$ and 0.999.



2.7×

lower train loss than **any** fixed β_1

73.26

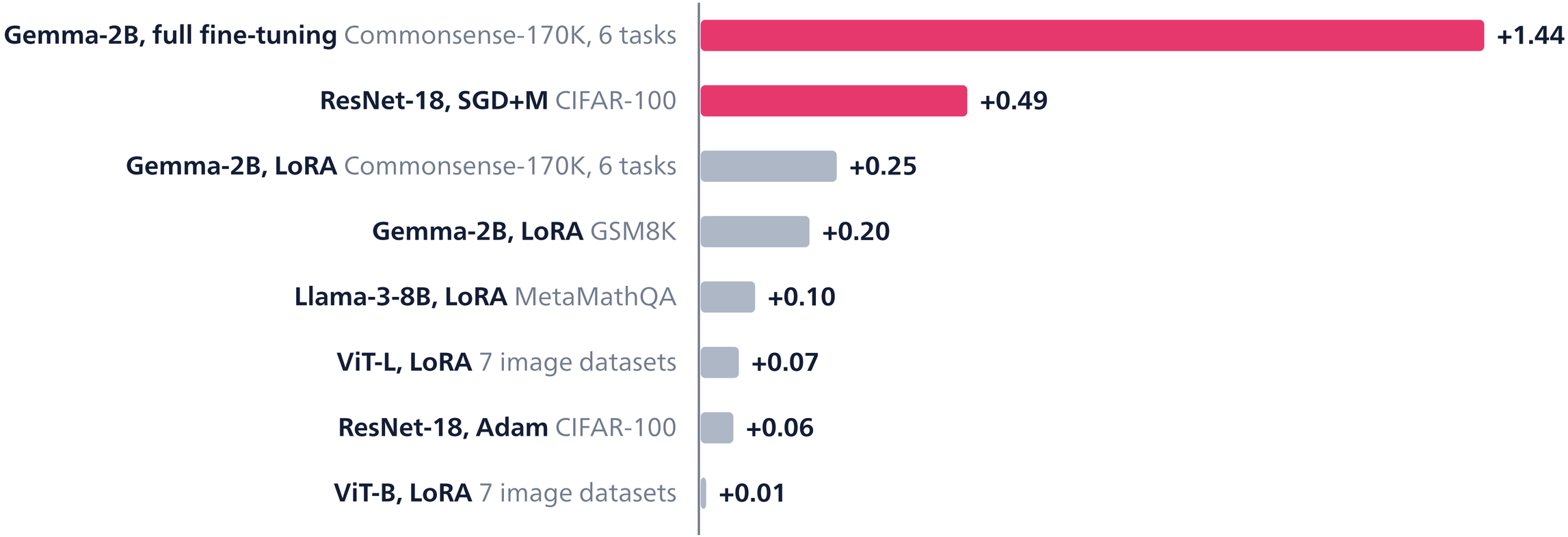
test accuracy, K-Switch

73.20

best fixed $\beta_1 = 0.8$, within the noise

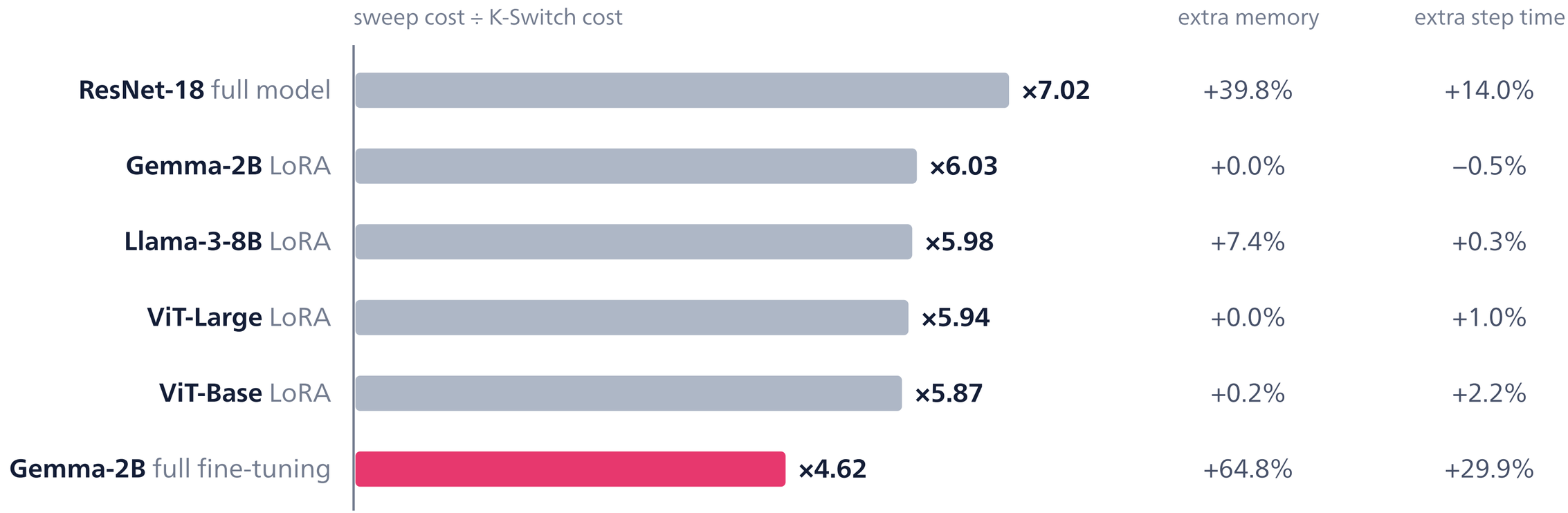
Across models: parity with the sweep, and two clear wins

Accuracy gain of K-Switch over the best swept baseline, in points. Pink: gain larger than the run-to-run spread.



Baselines are oracle picks from a sweep. K-Switch chooses during one run and matches them.

One K-Switch run replaces a 6 to 8 run sweep



Tuning cost drops by at least 4.6×. Overhead is near zero with LoRA and moderate for full models.

speedup ≈ number of β candidates ÷ (1 + step overhead), for example 6 ÷ 1.299 = 4.62. Candidates as counted in the paper: 8 for ResNet-18, 6 elsewhere.

Limits of the principle

Greedy and first-order	It maximizes today's expected loss-drop rate. No global convergence or validation-loss guarantee.
You still pick the family	It tunes inside the candidate set you supply. It does not choose the optimizer class.
Adam is approximated	Adam is nonlinear, so it is treated as a local time-varying filter.
Cost grows with K	Each candidate keeps its own state: up to +65% memory and +30% step time for full models.

Greedy Alignment Principle

1. An optimizer is a filter on the gradient stream.

2. Learning power is how well that filter matches the gradient autocorrelation.

3. K-Switch picks momentum at every step: one run instead of a sweep.

ironjr@snu.ac.kr

arXiv:2512.06370

jaerinlee.com