Greedy Alignment Principle

for Optimizer Hyperparameter Selection

ComputerVisionLab
Seoul National University

NEURAL INFORMATION PROCESSING SYSTEMS

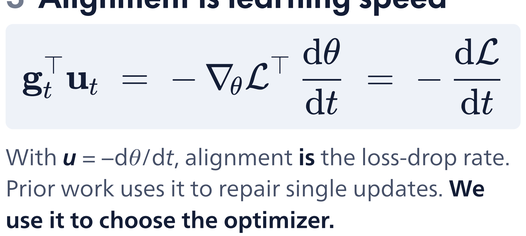
Jaerin Lee and Kyoung Mu Lee
Computer Vision Lab, Seoul National University
ironjr@snu.ac.kr

jaerinlee.com

Optimizers update model parameters. Why not let models update optimizer hyperparameters?

1 Momentum is still swept

Momentum β sets how much gradient history to keep. Its best value changes with task and model, so we **sweep**: one full run each.



2 An optimizer is a filter



gradients g noisy, correlated — **optimizer Q** impulse response Q_k — **updates u** filtered output

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

SGD+Momentum $\quad Q_k \;=\; \eta\,\beta^k\,I$

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

3 Alignment is learning speed

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

With $\boldsymbol{u} = -\mathrm{d}\theta/\mathrm{d}t$, alignment **is** the loss-drop rate. Prior work uses it to repair single updates. **We use it to choose the optimizer.**

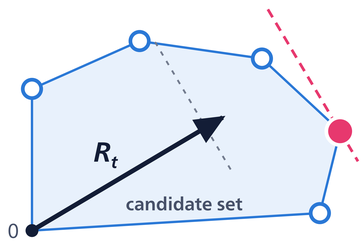
4 The Greedy Alignment Principle

On average, the loss-drop rate is an **inner product** between the optimizer filter Q and the gradient autocorrelation R:

$$\mathbb{E}\big[\mathbf{g}_t^\top\,\mathbf{u}_t\big] \;=\; \langle Q,\,R_t\rangle_{\mathcal{H}}, \qquad R_{t,k} = \mathbb{E}\big[\mathbf{g}_t\,\mathbf{g}_{t-k}^\top\big]$$

So at every step, pick the filter that best matches the autocorrelation of the gradients:

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



Q★ wins the candidate best aligned with R

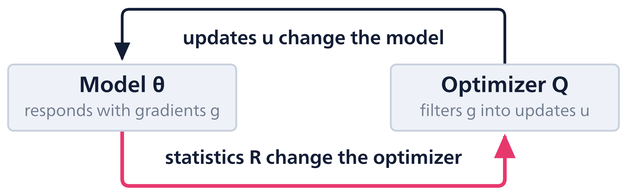
other candidates one filter per β

dashed: equal ⟨Q, R⟩

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

5 Two problems, solved together

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



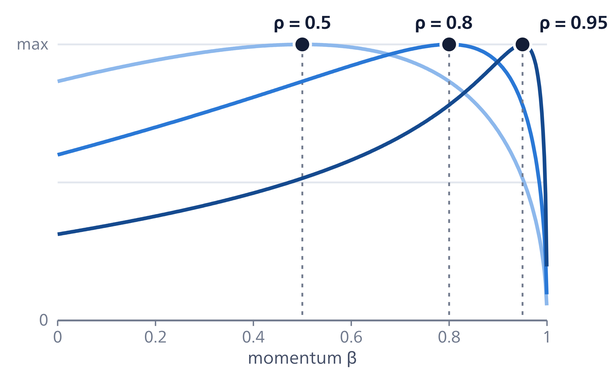
We maximize its drop rate $\quad \max_{Q\in\mathcal{Q}} \langle Q, R_t\rangle_{\mathcal{H}} \;=\; -\,\mathbb{E}\left[\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}t}\right]$

6 Momentum = gradient autocorrelation

For SGD+Momentum: one score per candidate β.

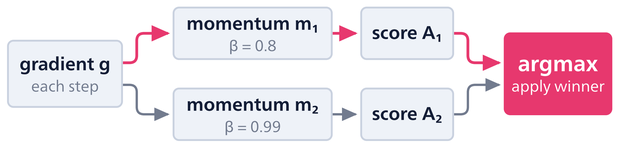
$$\beta_t^\star \;=\; \arg\max_{\beta}\ \sqrt{1-\beta^2}\ \mathbb{E}\big[\mathbf{g}_t^\top\,\mathbf{m}_{\beta,t}\big]$$

Score vs. β each curve peaks at β = ϱ



The score peaks at $\boldsymbol{\beta\star = \varrho}$: the best momentum is the temporal coherence of the gradients. Adam and AdamW follow the same rule.

7 K-Switch: two extra lines

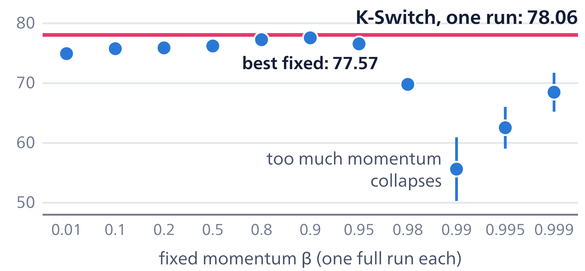



```
g = grad(loss, θ)              # every step
for k in 1..K:                 # K candidates
  m[k] = β[k] * m[k] + g
  A[k] = sqrt(1 - β[k]²) * dot(g, m[k])
k⋆ = argmax(A)
θ = θ - η * m[k⋆]
```

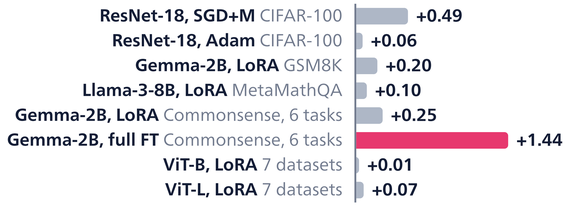
***K* = 2** is enough: two endpoints, fixed in advance, that bracket the usual range.

8 One run matches a full sweep

CIFAR-100, ResNet-18, SGD+M test accuracy (%), 10 runs



Accuracy gain over the best swept baseline in points



Baselines: oracle pick from 6–11 value sweeps. Most gaps are within one std: **parity with the sweep**.

Cost of tuning K-Switch against a sweep

4.6–7.0×	**≤ +7%**	**≤ +2%**
less tuning cost than a 6–8 run sweep	**memory (LoRA)** full model +40–65%	**step time (LoRA)** full model +14–30%

9 The best β moves during training

Share of steps picking β = 0.8 over 0.99 SGD+M

