Optimizer Benchmarking Needs to Account for Hyperparameter Tuning

Prabhu Teja S [*,1,2] Florian Mai [*,1,2] Thijs Vogels [2]
Martin Jaggi [2] François Fleuret [1,2]

[1]Idiap Research Institute, [2]EPFL, Switzerland
[*]Equal Contribution

prabhu.teja, florian.mai@idiap.ch

The problem of optimizer evaluation

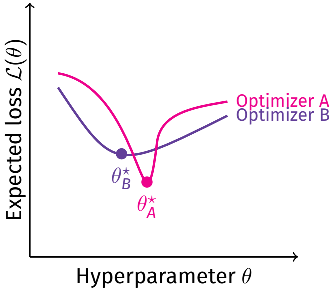


Figure: Two optimizers A & B with hyperparameter θ. Which one do we prefer in practice?

The problem of optimizer evaluation

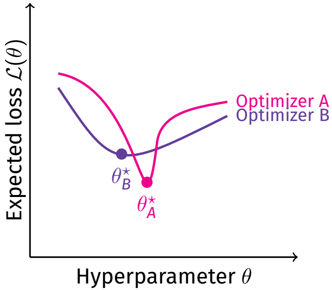
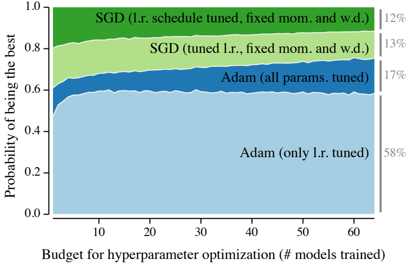


Figure: Two optimizers A & B with hyperparameter θ. Which one do we prefer in practice?

1. The absolute performance of the optimizer $\rightarrow \mathcal{L}(\theta_A^\star), \mathcal{L}(\theta_B^\star)$
2. Difficulty of finding good hyperparameter configuration $\approx \theta_A^\star, \theta_B^\star$.

The Problem of Optimizer Evaluation: SGD vs Adam

1. SGD often achieves better peak performance than Adam in previous literature
2. We take into cognizance the cost of automatic Hyperparameter Optimization (HPO), and find:



Our method eliminates human biases arising from manual hyperparameter tuning.

prabhu.teja, florian.mai@idiap.ch

Revisiting the notion of an Optimizer

Definition

An optimizer is a pair $\mathcal{M} = (\mathcal{U}_\Theta, p_\Theta)$, which applies its update rule $\mathcal{U}(S_t; \Theta)$ at each step t depending on its current state S_t.
Its hyperparameters $\Theta = (\theta_1, \ldots, \theta_N)$ have a prior probability distribution $p_\Theta : (\Theta \to \mathbb{R})$ defined.

p_Θ should be specified by the optimizer designer,
e.g., Adam's $\epsilon > 0$ and close to $0 \implies \epsilon \sim$ Log-uniform$(-8, 0)$

Algorithm 1 Benchmark with 'expected quality at budget'

input: optimizer O, cross-task hyperparameter prior p_Θ, task T, tuning budget B
Initialize $list \leftarrow [\,]$.
for R repetitions **do**
  Perform random search with budget B:
  – $S \leftarrow$ sample B elements from p_Θ.
  – $list \leftarrow [\textsc{best}(S), \ldots list]$.
return $\textsc{mean}(list)$, $\textsc{var}(list)$, or other statistics

Calibrated task independent priors p_Θ

Optimizer	Tunable parameters	Cross-task prior
SGD	Learning rate Momentum Weight decay Poly decay (p)	??
Adagrad	Learning rate	
Adam	Learning rate β_1, β_2 ϵ	

Calibrated task independent priors p_Θ

Optimizer	Tunable parameters	Cross-task prior
SGD	Learning rate Momentum Weight decay Poly decay (p)	??
Adagrad	Learning rate	
Adam	Learning rate β_1, β_2 ϵ	

- Sample a large number of points and their performance from a large range of admissible values
- Maximum Likelihood Estimate (MLE) of the prior's parameters using the top 20% performant values from the previous step.

Calibrated task independent priors p_Θ

Optimizer	Tunable parameters	Cross-task prior
SGD	Learning rate	Log-normal(-2.09, 1.312)
	Momentum	$\mathcal{U}[0, 1]$
	Weight decay	Log-uniform(-5, -1)
	Poly decay (p)	$\mathcal{U}[0.5, 5]$
Adagrad	Learning rate	Log-normal(-2.004, 1.20)
Adam	Learning rate	Log-normal(-2.69, 1.42)
	β_1, β_2	1- Log-uniform(-5, -1)
	ϵ	Log-uniform(-8, 0)

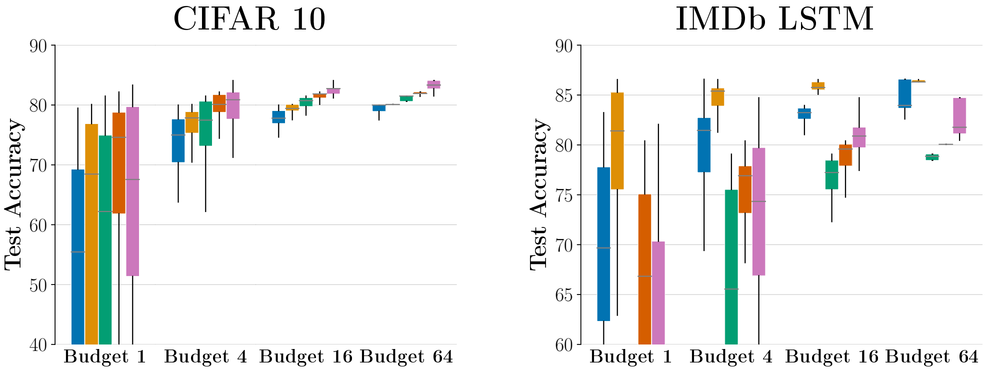
- Sample a large number of points and their performance from a large range of admissible values
- Maximum Likelihood Estimate (MLE) of the prior's parameters using the top 20% performant values from the previous step.

The importance of Recipes

Optimizer label	Tunable parameters
SGD-$M^C W^C$	SGD(γ, μ=0.9, λ=10^{-5})
SGD-M^CD	SGD(γ, μ=0.9, λ=10^{-5}) + Poly Decay(p)
SGD-MW	SGD(γ, μ, λ)
Adam-LR	Adam(γ, β_1=0.9, β_2=0.999, ϵ=10^{-8})
Adam	Adam($\gamma, \beta_1, \beta_2, \epsilon$)

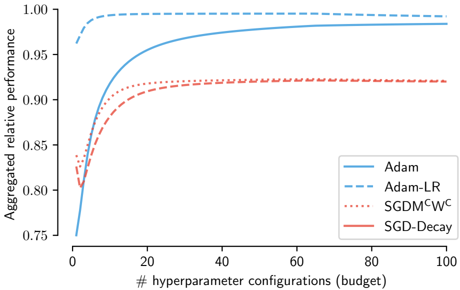
SGD(γ, μ, λ) is SGD with γ learning rate, μ momentum, λ weight decay coefficient. Adagrad(γ) is Adagrad with γ learning rate, Adam($\gamma, \beta_1, \beta_2, \epsilon$) is Adam with learning rate γ, momentum parameters β_1, β_2, and normalization parameter ϵ

Performance at a budget



Performance of **Adam-LR**, **Adam**, **SGD-M^{C}W^C**, **SGD-MW**, **SGD-M^{C}D** at various hyperparameter search budgets

Summarizing our findings



Summary statistics:

$$S(o, k) = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \frac{o(k, p)}{\max_{o' \in \mathcal{O}} o'(k, p)},$$

where $o(k, p)$ denotes the expected performance of optimizer $o \in \mathcal{O}$ on test problem $p \in \mathcal{P}$ after k iterations of hyperparameter search.

Our findings

1. Support the hypothesis that adaptive gradient methods are easier to tune than non-adaptive methods
   - The substantial value of the adaptive gradient methods, specifically Adam, is its amenability to hyperparameter search.

Our findings

1. Support the hypothesis that adaptive gradient methods are easier to tune than non-adaptive methods
   - The substantial value of the adaptive gradient methods, specifically Adam, is its amenability to hyperparameter search.
2. Tuning optimizers' hyperparameters apart from the learning rate becomes more useful as the available tuning budget goes up.
   - Even with relatively large tuning budget, tuning only the learning rate of Adam is the safer choice, as it achieves good results with high probability.

THANK YOU