MADA: Meta-Adaptive Optimizers through Hyper-gradient Descent

Kaan Ozkara[1], Can Karakus[2], Parameswaran Raman[2], Mingyi Hong[2,3], Shoham Sabach[2,4], Branislav Kveton[2], Volkan Cevher[2,5]

[1]UCLA [2]Amazon Web Services [3]University of Minnesota [4]Technion [5]EPFL

TLDR

- We introduce a unified optimizer framework that can generalize several known optimizers and dynamically learn the most suitable one during training.
- We parameterize the space of optimizers and dynamically search through it using hyper-gradient descent during training..
- Theoretically, we show that interpolations of optimizers might result in faster convergence (in constants) compared to individual components.
- Empirically, our method outperforms AdamW and other popular optimizers in many settings. In particular, MADA outperforms the competing methods in GPT-2 pre-training/fine-tuning on OpenWebText and Shakespeare datasets, in training ResNet and CNN architecture on CIFAR-10.

Motivation for a Unified Optimizer

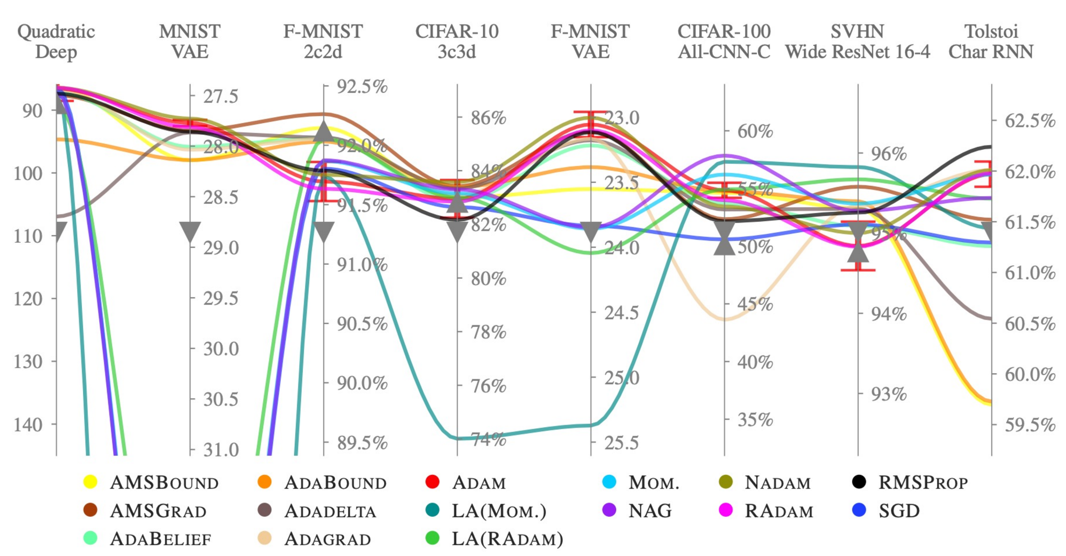


Figure 1 [1]. Different optimizers perform better in different tasks, Adam remains the most popular especially for LLMs.

Q: Can we find a meta optimizer that automatically chooses what optimizer to use during training?

A Unified Framework for Adaptive Optimizers

$$\boxed{x_t = x_{t-1} - \alpha_t \frac{m_t}{\sqrt{v_t} + \epsilon},}$$

where $\alpha_t > 0$ is learning rate and $\epsilon > 0$ is stability constant

Table 1. A unified framework to express adaptive optimizers.

Method	First-Order Moment	Second-Order Moment
Adam [Kingma and Ba, 2015]	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$	$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$
AMSGrad [Reddi et al., 2018]	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$	$\bar{v}_t = \beta_2 \bar{v}_{t-1} + (1-\beta_2) g_t^2$ $v_t = \max\{v_{t-1}, \bar{v}_t\}$
Adan [Xie et al., 2023]	$\bar{m}_t = \beta_1 \bar{m}_{t-1} + (1-\beta_1) g_t$ $n_t = \beta_3 n_{t-1} + (1-\beta_3)(g_t - g_{t-1})$ $m_t = \bar{m}_t + \beta_3 n_t$	$\hat{g}_t = g_t + \beta_3 (g_t - g_{t-1})$ $v_t = \beta_2 v_{t-1} + (1-\beta_2) \hat{g}_t^2$
Yogi [Zaheer et al., 2018]	$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$	$\hat{g}_t = v_{t-1} + g_t^2 \cdot \text{sign}(g_t^2 - v_{t-1})$ $v_t = \beta_2 v_{t-1} + (1-\beta_2) \hat{g}_t$

Parameterized Optimizers

Existing Optimizers $\xrightarrow[\text{via interpolations}]{\text{Parameterization}}$ Parameterized Optimizer

first-order moment

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$n_t = \beta_3 n_{t-1} + (1-\beta_3)(g_t - g_{t-1})$$
$$u_t = Lion(g_t, m_t^{lion}),$$

second-order moment

$$\hat{g}_t = g_t + \beta_3 (g_t - g_{t-1})$$
$$\tilde{g}_t^2 = c_{t-1}\hat{g}_t^2 + (1-c_{t-1})(\bar{v}_{t-1} + \hat{g}_t^2 \text{sign}(\hat{g}_t^2 - \bar{v}_{t-1}))$$
$$\bar{v}_t = \beta_2 \bar{v}_{t-1} + (1-\beta_2)\tilde{g}_t^2$$
$$\tilde{v}_t = (\bar{v}_t + (t-1)\tilde{v}_{t-1})/t$$
$$v_t = \rho \bar{v}_t + (1-\rho)\tilde{v}_t$$

update term

$$x_{t-1} = x_{t-1} - \lambda \alpha_t x_{t-1}$$
$$x_t = x_{t-1} - \alpha_t \Big(\gamma \frac{m_t + \beta_3 n_t}{\sqrt{v_t} + \epsilon} + (1-\gamma)\text{sign}(u_t)\Big),$$

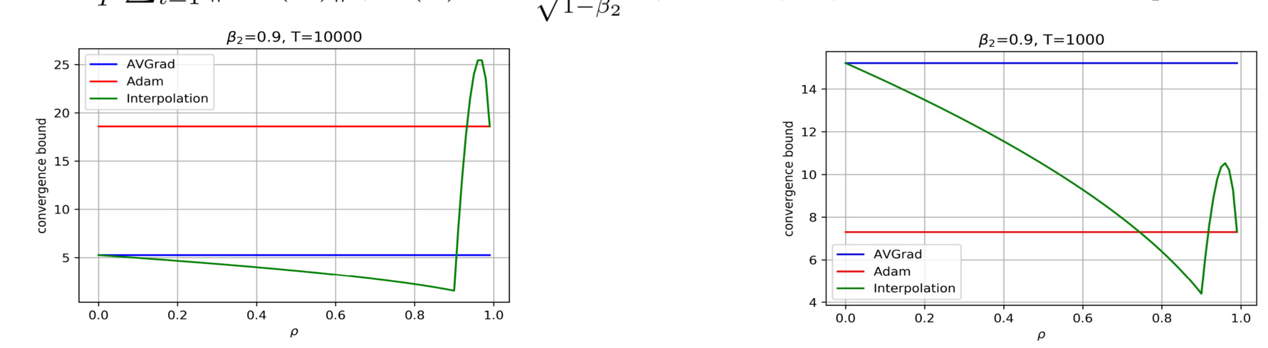
$\beta_1, \beta_2, \beta_3, c, \rho, \gamma$ are learnable parameters. Adam: $\beta_3 = 0, c = 1, \rho = 1, \gamma = 1$; AVGrad: $\beta_3 = 0, c = 1, \rho = 0, \gamma = 1$; Yogi: $\beta_3 = 0, c = 0, \rho = 1, \gamma = 1$; Lion: $\beta_3 = 0, c = 1, \rho = 1, \gamma = 0$; Adan: $c = 1, \rho = 1, \gamma = 1$.

Interpolated optimizers can have faster convergence than individual components:

Theorem 1 (Convergence of interpolation of AVGrad and Adam without momentum). *Under the assumptions above and $\alpha_t = \frac{\alpha}{\sqrt{t}}$ for some $\alpha > 0$, and for $\rho_t = \frac{\rho}{t}$ for a constant ρ:*

$$G_T \le E(T)\left[\frac{C_1}{T} + \frac{C_2 \ln(E(T))}{T} + C_3\Big[\ln\Big(\frac{\rho}{\beta_2}\Big)\Big]_+\right],$$

where, $G_T = \frac{1}{T}\sum_{t=1}^{T} \|\nabla F(x_t)\|^2$, $E(T) = \frac{\sqrt{\rho + (1-\rho)T}}{\sqrt{1-\beta_2}}$, and C_1, C_2, C_3 are constants independent of T, ρ, β_2.



MADA: Meta-Adaptive Optimizers

Algorithm 1 Pseudocode for a generic MADA

Input: A parameterized optimizer $\mathcal{O}_q$, where $q \in \mathcal{D}$, a hyper-learning rate α, number of total iterations T.
Init.: x_0 and q_0.

for t=1 to T **do**
  Sample f_t.
  Update the model parameters:
    $x_t = \mathcal{O}_{q_{t-1}}(x_{t-1})$.
  Update the optimizer coefficients:
    $q_t = \Pi_{\mathcal{D}}\left[q_{t-1} - \alpha \nabla_q f_t(x_{t-1})\right]$.
end for

Output: Model wights x_T.

- $\mathcal{O}_q$ denotes the parameterized optimizer with parameter set q, $\mathcal{D}$ denotes the domain of the parameters.
- At each iteration, we first update the model parameters using the meta-optimizer; and then we update the optimizer parameters through hyper-gradient descent.

Experiments

- **Language Model Experiments:** Pre-training GPT-2 (125M) on OpenWebText and a 10M model on Shakespeare, fine-tuning GPT-2 (1.5B) on Shakespeare. Measure perplexity on popular language datasets.
- **Vision Experiments:** Training ResNet-9 and 5 layer CNN on CIFAR-10.
- **Baselines:** Recently proposed adaptive optimizers, HyperAdam [2] where only β_1, β_2 are learned, SGD with momentum.

Training Curves

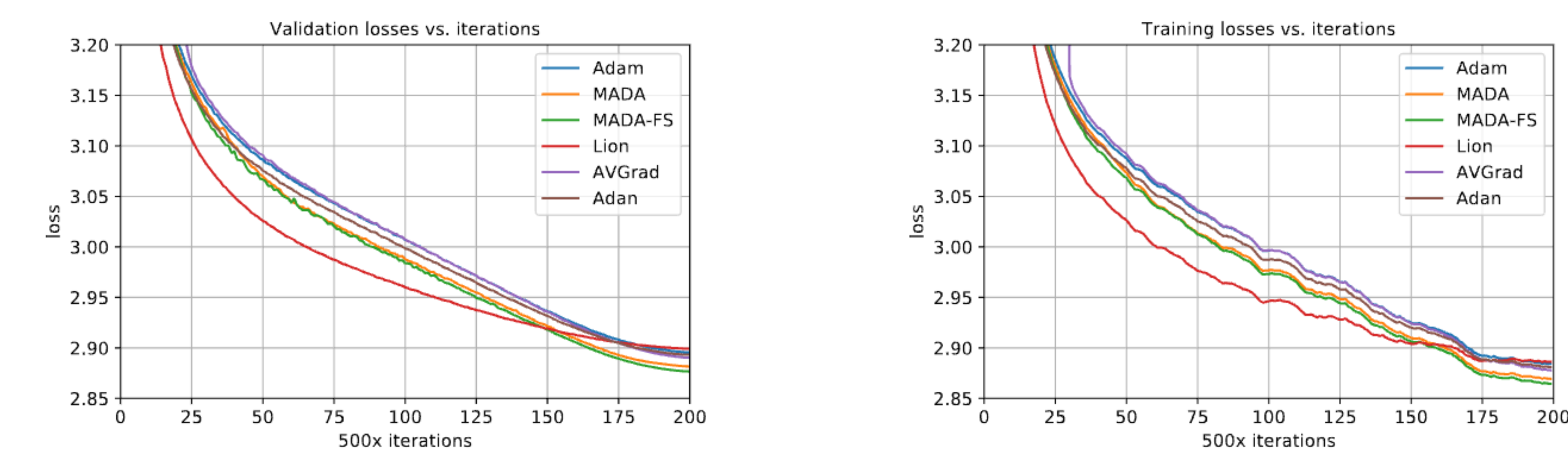


Figure 2 Validation and training losses of competing methods on OpenWebText for training GPT-2 (125M) model.

Validation Loss and Perplexity Tables

Table 2 Validation loss of MADA on OpenWebText vs other adaptive optimizer baselines.

Method	Validation Loss
Adam	2.8956
Adan	2.8896
HyperAdam	2.8950
Lion	2.8892
AVGrad	2.8895
MADA	**2.8806**
MADA-FS	**2.8766**
Poor initialization	
MADA$^-$	2.8921
Adam$^-$	2.9157

Table 3 Validation perplexities of competing methods on OpenWebText, Wikitext and Lambada datasets.

Method	OpenWebText	Wikitext	Lambada
Adam	18.0940	63.8544	77.3314
Adan	17.9863	63.5518	74.6970
HyperAdam	18.0843	61.7717	**72.6803**
Lion	17.9792	61.8661	75.3158
AVGrad	17.9840	64.2620	75.1317
MADA	**17.8249**	61.2513	74.2480
MADA-FS	**17.7544**	59.4086	**73.5623**
Poor initialization			
MADA$^-$	18.0317	**57.1613**	75.3550
Adam$^-$	18.4624	72.9017	79.1217

Table 4 Training losses after 2 epochs of fine-tuning on Shakespeare dataset.

Method	Training loss
MADA	**0.255**
Adam	0.276
HyperAdam	0.278

Table 5 Test accuracy of competing methods for CNN and ResNet models.

Method	5-layer CNN	ResNet-9
MADA	**66.12 ± 0.14**	**93.79 ± 0.11**
Adam	65.84 ± 0.11	93.73 ± 0.10
HyperAdam	65.80 ± 0.21	93.69 ± 0.06
Momentum SGD	65.97 ± 0.94	92.60 ± 0.10

Robustness

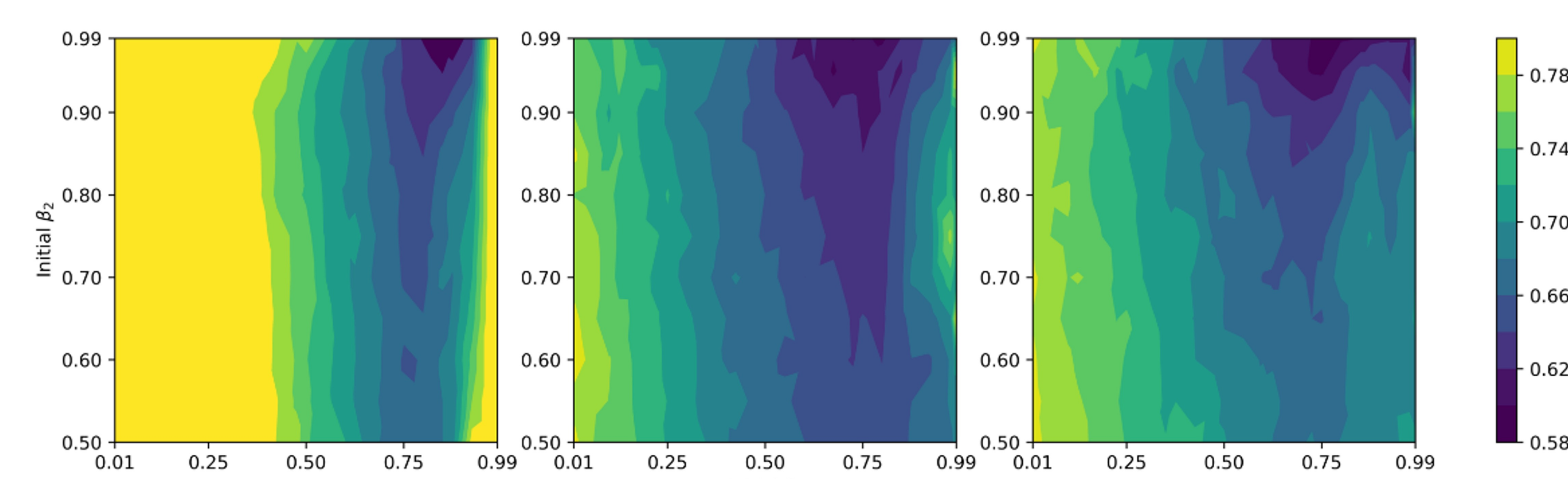


Figure 3 Final training losses of AdamW, MADA, HyperAdam with respect to different initial β_1, β_2 on Shakespeare dataset for training a 10M model. MADA yields lower loss for a wider region illustrating its robustness to initialization.

References

[1] Schmidt, Robin M., Frank Schneider, and Philipp Hennig. "Descending through a crowded valley-benchmarking deep learning optimizers." International Conference on Machine Learning. PMLR, 2021.

[2] Chandra, Kartik, et al. "Gradient descent: The ultimate optimizer." Advances in Neural Information Processing Systems 35 (2022)

Access to paper: