JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

Transformer-Based Learned Optimization

WED-AM-357


Erik Gärtner[1,2]


Luke Metz[1]


Mykhaylo Andriluka[1]

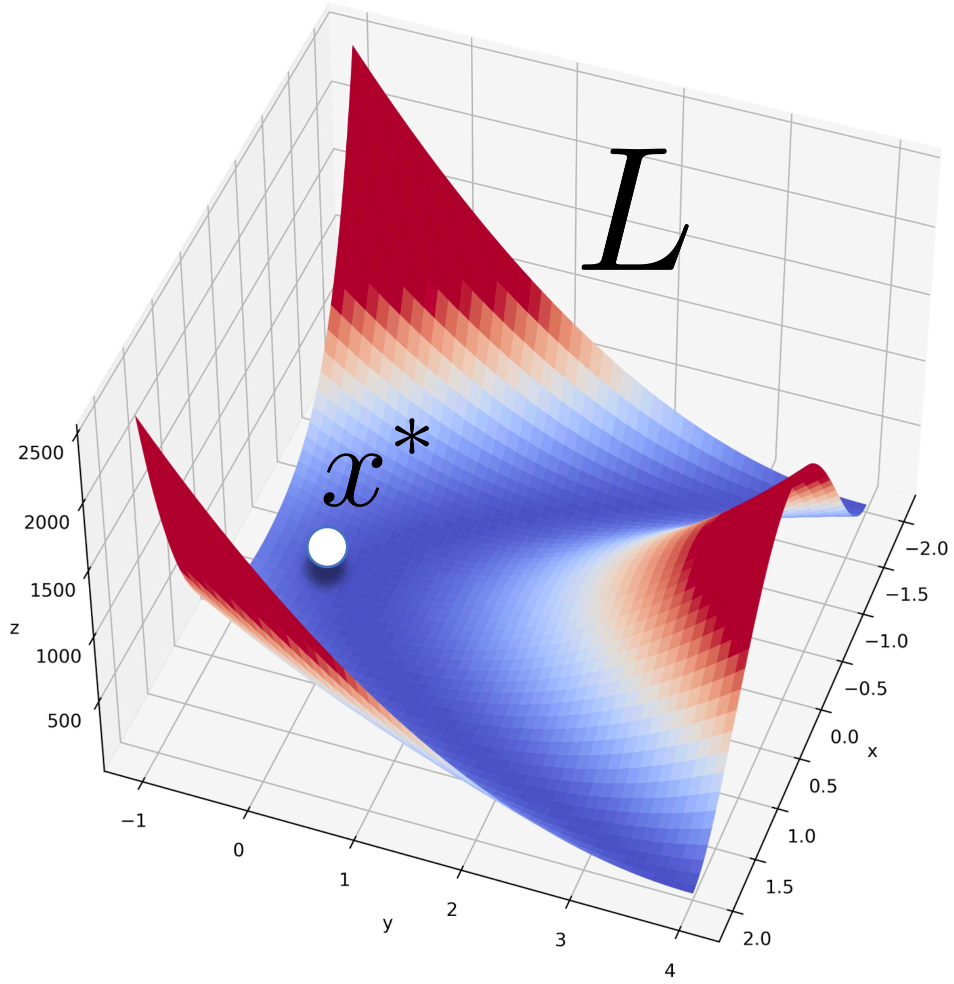

C. Daniel Freeman[1]


Cristian Sminchisescu[1]

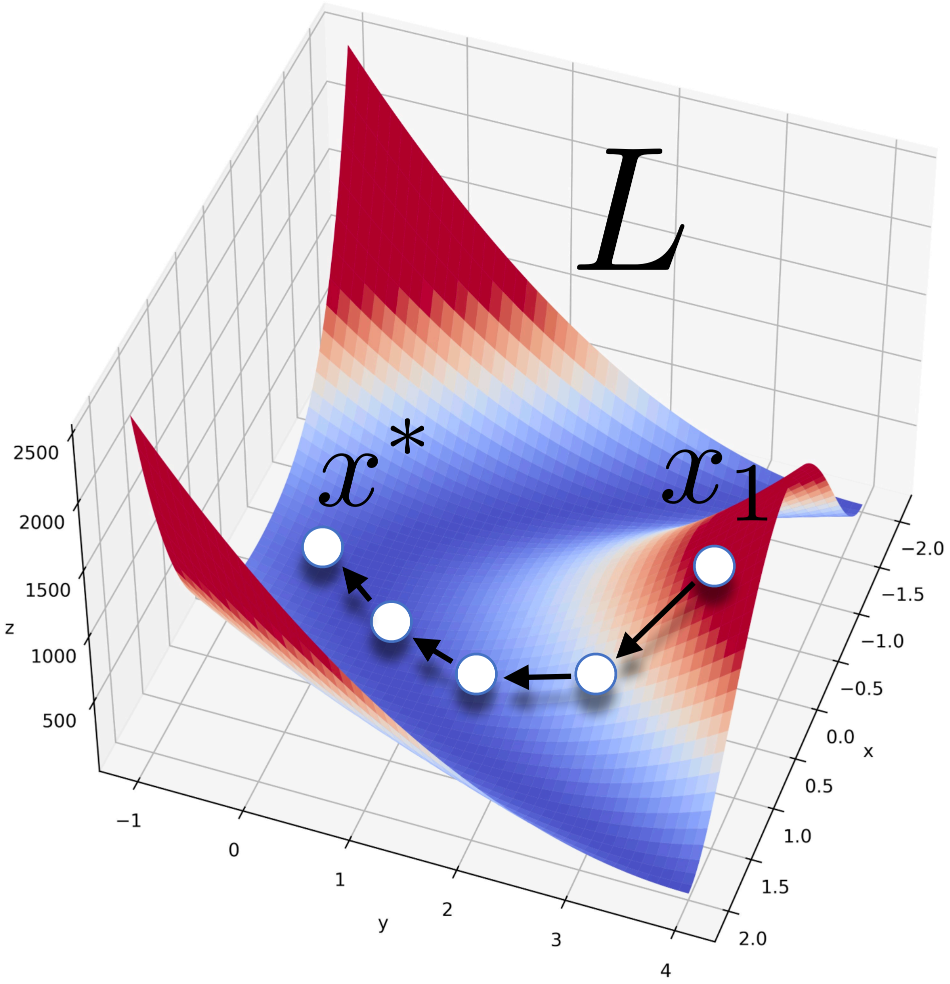
[1]Google Research

[2]Lund University

Introduction



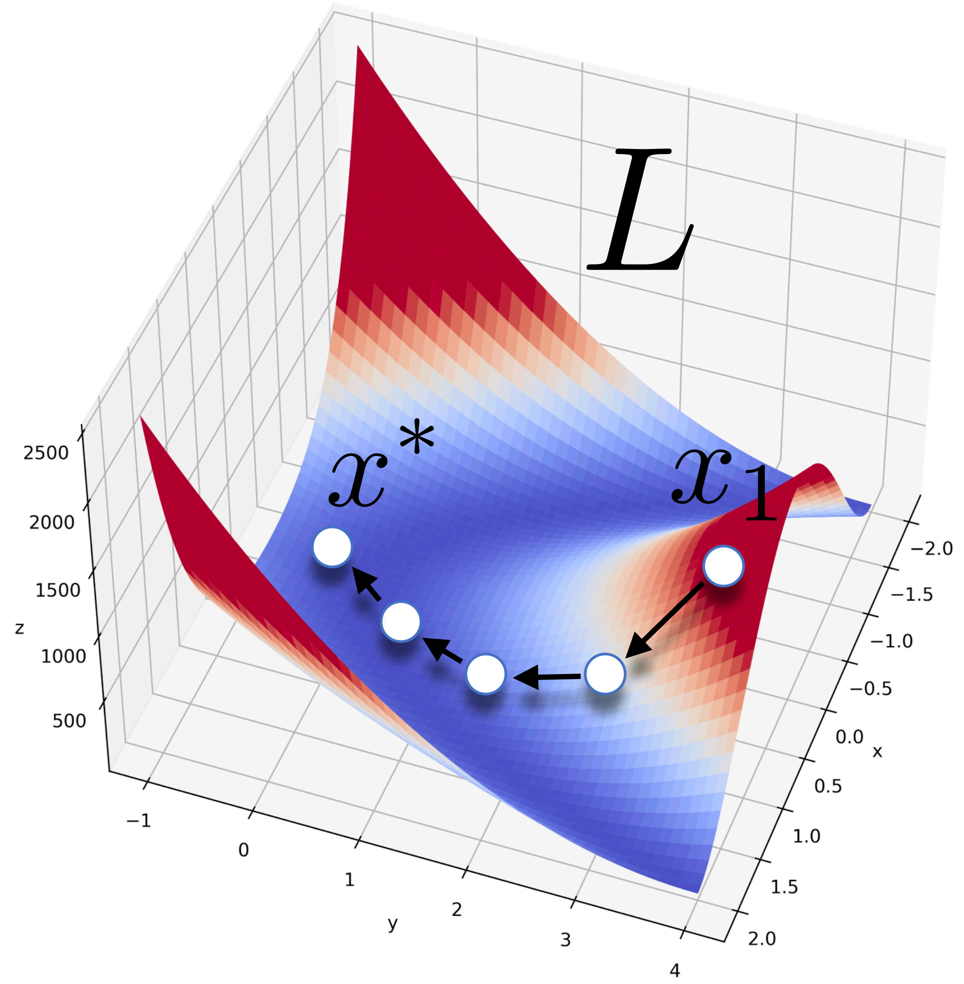
Introduction



Iterative gradient-based optimization:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - U(\nabla_{\leq k} L(\mathbf{x}_{1:k}))$$

Introduction



Iterative gradient-based optimization:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - U(\nabla_{\leq k} L(\mathbf{x}_{1:k}))$$

Gradient descent:

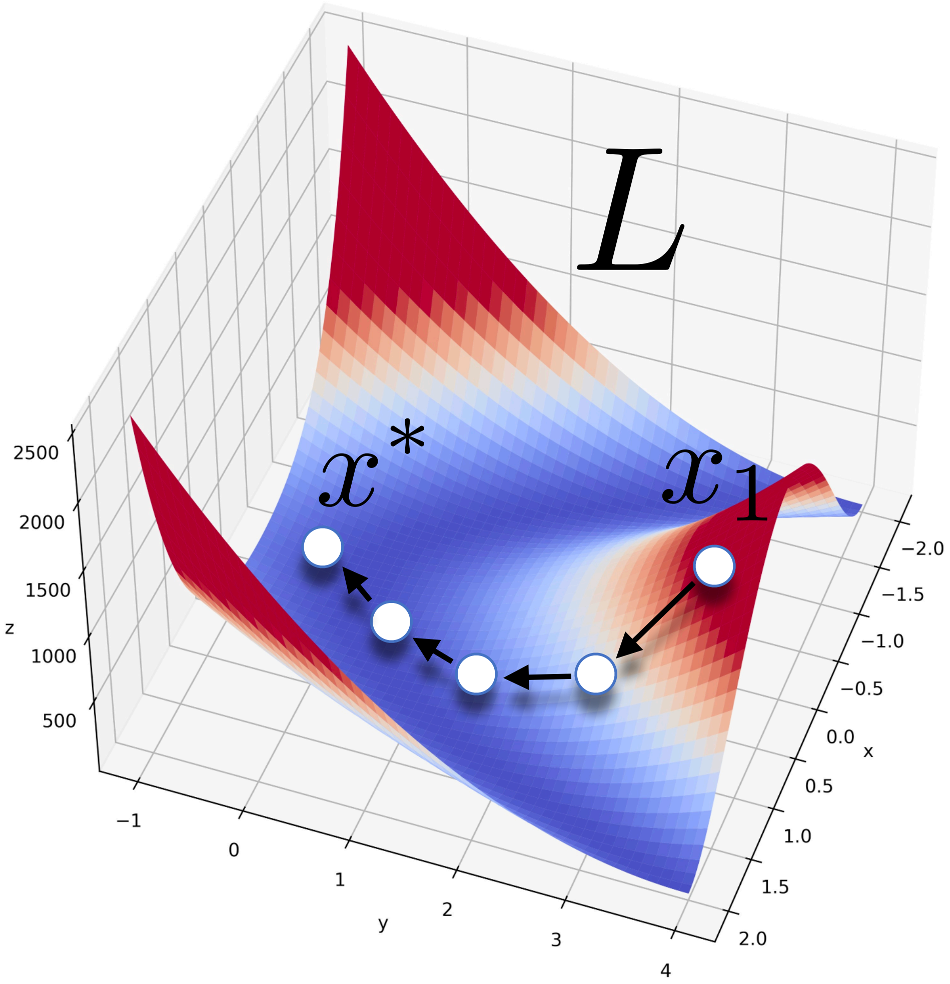
$$U_{\text{gd}}(\nabla_{\leq k} L(\mathbf{x}_{1:k})) = \alpha \nabla L(\mathbf{x}_k)$$

α - learning rate

Gradient descent with momentum: $\mu \in (0, 1)$

$$U_{\text{gdm}}(\nabla_{\leq k} L(\mathbf{x}_{1:k})) = \alpha \sum_{n=1}^{k} \mu^{k-1} \nabla L(\mathbf{x}_n)$$

Our approach



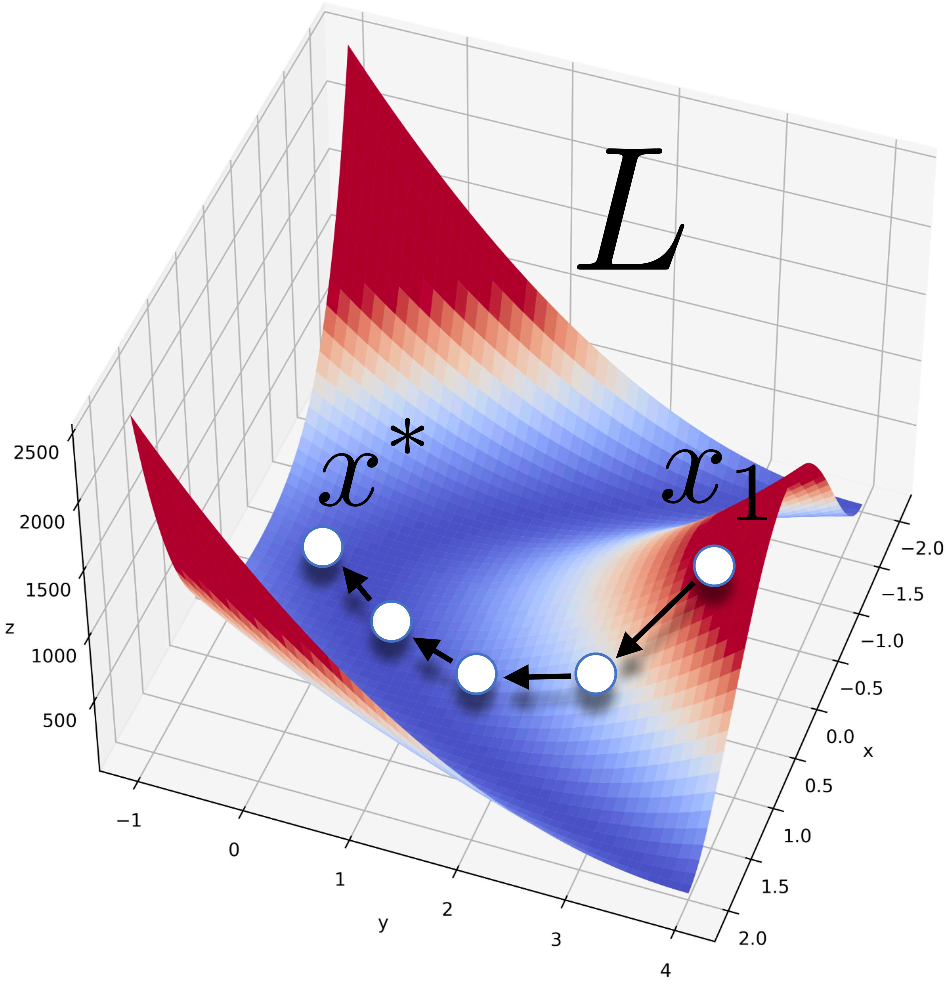
Iterative gradient-based optimization:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - U(\text{features}\,(L, \mathbf{x}_{1:k})|\theta\,)$$

Neural network trained on a set of optimization problems

θ - parameters of the learned optimizer (=meta-parameters)

Our approach



Iterative gradient-based optimization:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - U\left(\text{features}\,(L, \mathbf{x}_{1:k})|\theta\right)$$

Neural network trained on a set of optimization problems

θ - parameters of the learned optimizer (=meta-parameters)

$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$

Preconditioning matrix

Per-parameter descent direction

Related work

- U is a multi-layer perceptron (MLP), same U is applied to every dimension of x

L. Metz, C. D. Freeman, J. Harrison, N. Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. LLA'22

L. Metz, N. Maheswaranathan, J. Nixon, D Freeman, and J. Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. ICML'19

Related work

- U is a multi-layer perceptron (MLP), same U is applied to every dimension of x

L. Metz, C. D. Freeman, J. Harrison, N. Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. LLA'22

L. Metz, N. Maheswaranathan, J. Nixon, D Freeman, and J. Sohl-Dickstein. Understanding and correcting pathologies in the training of learned optimizers. ICML'19

✓ Dimensionality of the problem can change at test time

✓ Parsimonious with respect to number of meta-parameters

✗ Does not allow coordinated updates across multiple dimensions

Related work

- U is a MLP operating on the entire vector x

Jie Song, Xu Chen and Otmar Hilliges, Human body model fitting by learned gradient descent. ECCV'20

✓ Couples update across multiple dimensions

✗ Training and test optimization problems must have the same dimensionality

Our approach

- ✓ Couples update across multiple dimensions
- ✓ Dimensionality of the problem can change at test time

- ✗ Expensive update step with complexity that is quadratic in the number of dimensions

Related work

- VeLO: concurrent work with our approach, uses RNN to couple updates across dimensions, trained and evaluated on training neural networks

L. Metz, J. Harrison, C. D. Freeman, A. Merchant, L. Beyer, J. Bradbury, N. Agrawal, B. Poole, I. Mordatch, A. Roberts, J. Sohl-Dickstein, VeLO: Training Versatile Learned Optimizers by Scaling Up, arXiv:2211.09760

✓ Couples update across multiple dimensions

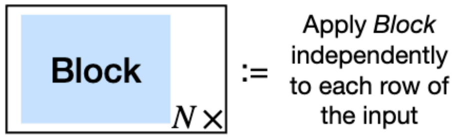
✓ Dimensionality of the problem can change at test time

Our approach

$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$

Our approach

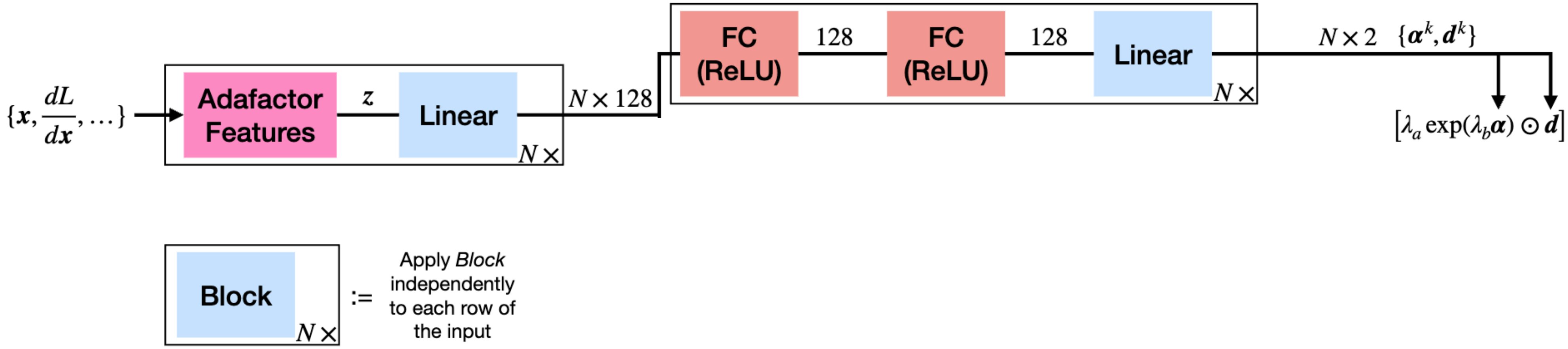
$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$



N. Shazeer and M. Stern, Adafactor: Adaptive Learning Rates with Sublinear Memory Cost, ICML'18

Our approach: per-parameter update

$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$

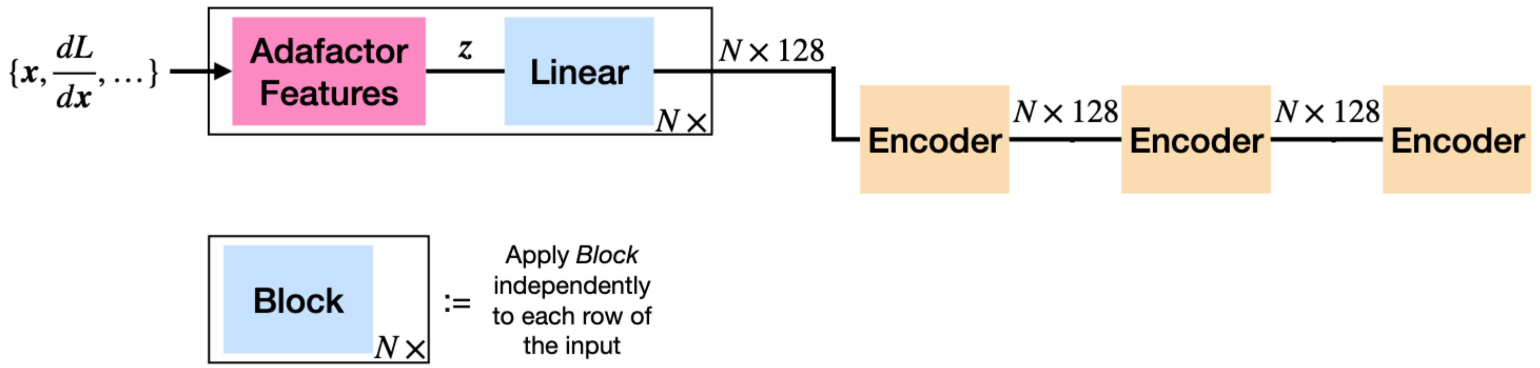


L. Metz, C. D. Freeman, J. Harrison, N. Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. LLA'22

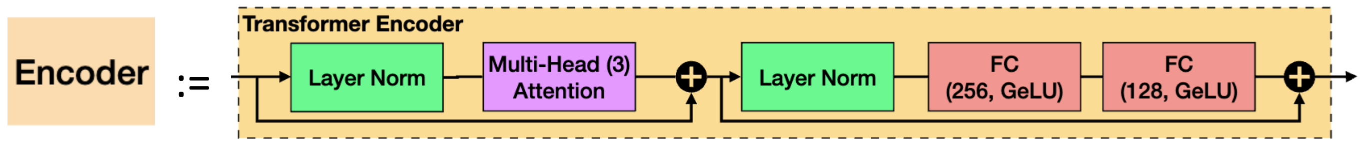
N. Shazeer and M. Stern, Adafactor: Adaptive Learning Rates with Sublinear Memory Cost, ICML'18

Preconditioning matrix

$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$



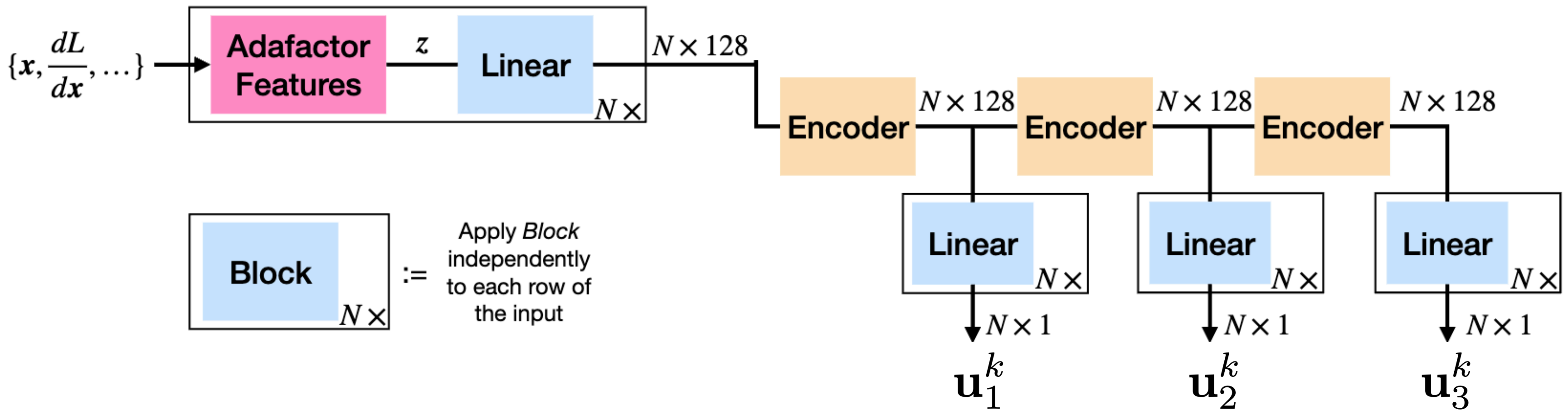
Transformer encoder



A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is All you Need, NeurIPS’17

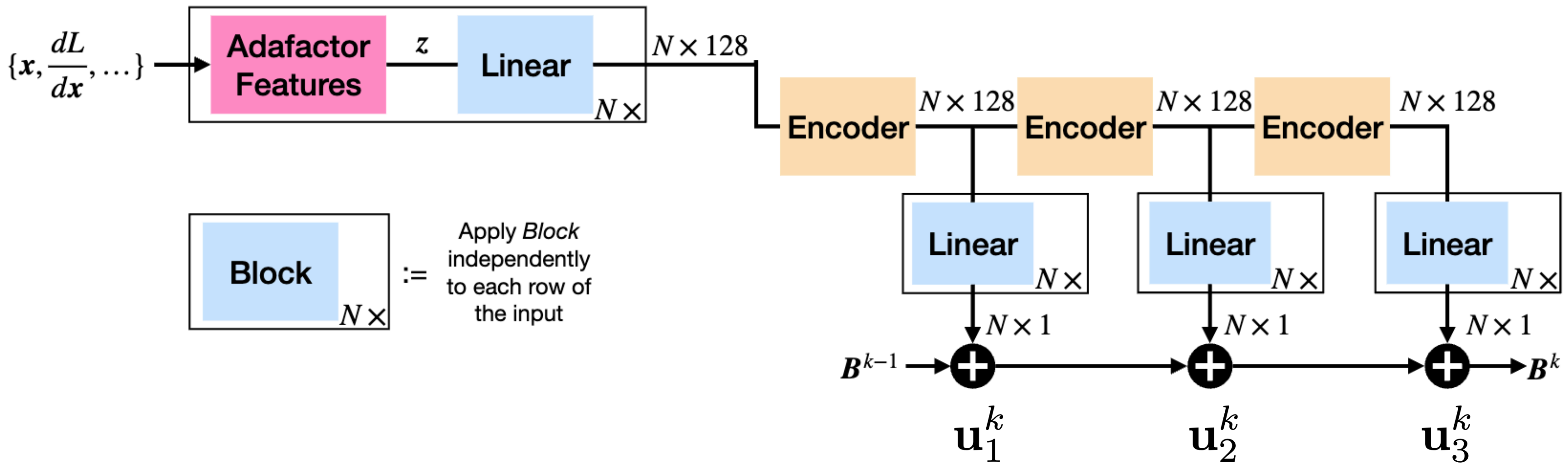
Preconditioning matrix

$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$



Preconditioning matrix

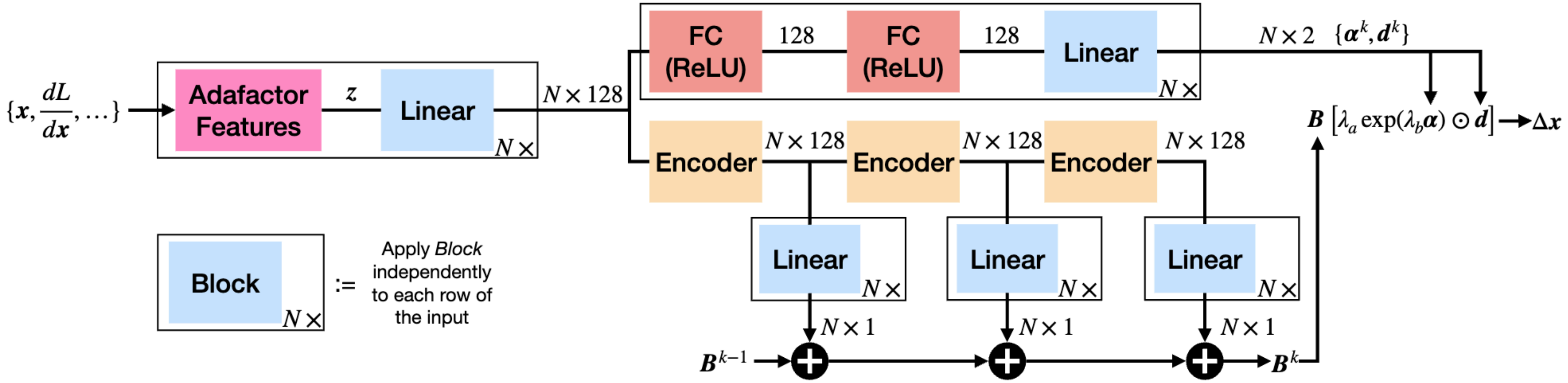
$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$



$$\tilde{\mathbf{B}}^{k+1} = \mathbf{B}^k + \sum_{l=1}^{L} \mathbf{u}_l^k (\mathbf{u}_l^k)^\top, \quad \mathbf{B}^{k+1} = \tilde{\mathbf{B}}^{k+1} / \|\tilde{\mathbf{B}}^{k+1}\|$$

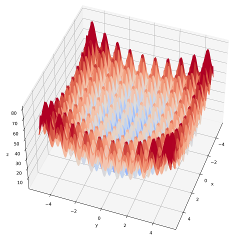
Our approach

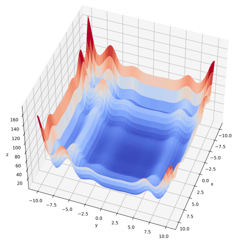
$$U^k(\cdot|\theta) = \mathbf{B}^k(\cdot|\theta)\mathbf{s}^k(\cdot|\theta)$$

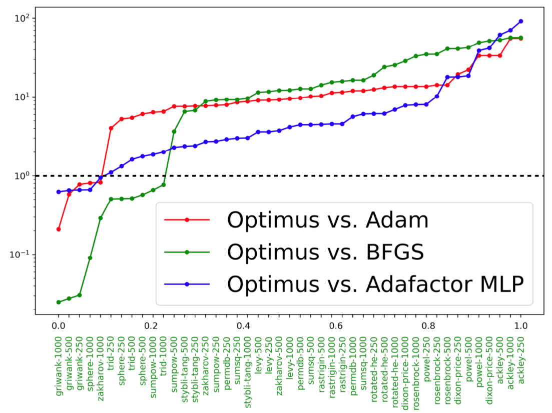


Results

Setting 1:
Classic objective functions used for evaluation of optimization algorithms

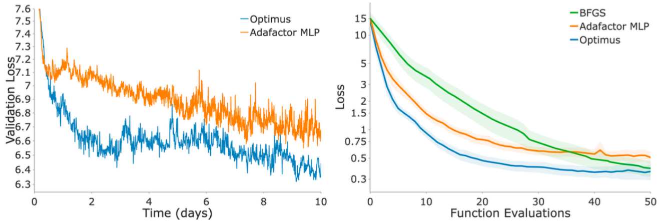






10x reduction in the number of update steps for half of the classical optimization problems

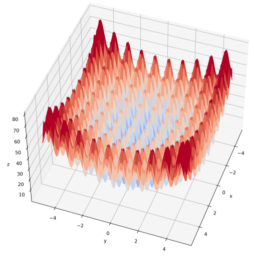
Setting 2:
Physics-based articulated pose reconstruction from video



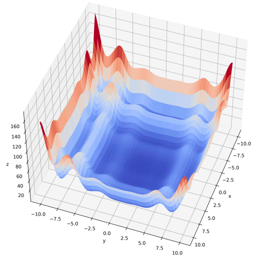
- Same accuracy as BFGS while requiring half of the objective function evaluations
- Faster convergence at training time

Results on classic objective functions

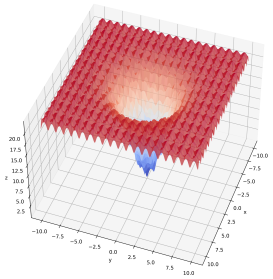
- We use a dataset of 15 classic objective function used for evaluation of optimization algorithms
  - Training set: dimensions 2-100
  - Test set: dimensions 250, 500, 1000
  - 45 test functions in the test set
  - Randomly shift each function to diversify the train/test sets



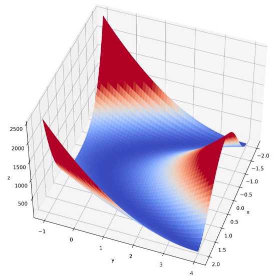
Rastrigin



Levy



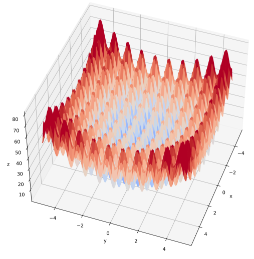
Ackley



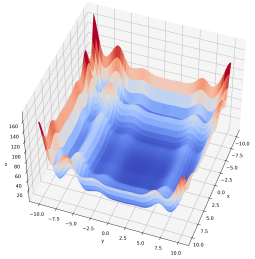
Rosenbrock

Results on classic objective functions

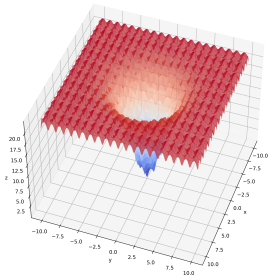
- We use a dataset of 15 classic objective function used for evaluation of optimization algorithms
    - Training set: dimensions 2-100
    - Test set: dimensions 250, 500, 1000
    - 45 test functions in the test set
    - Randomly shift each function to diversify the train/test sets
- Baselines
    - Gradient descent with momentum (GD-M), tuned learning rate
    - Adam, tuned learning rate
    - BFGS
    - AdafactorMLP [Metz et al., 2022]



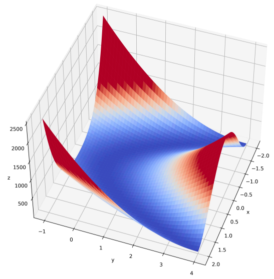
Rastrigin



Levy



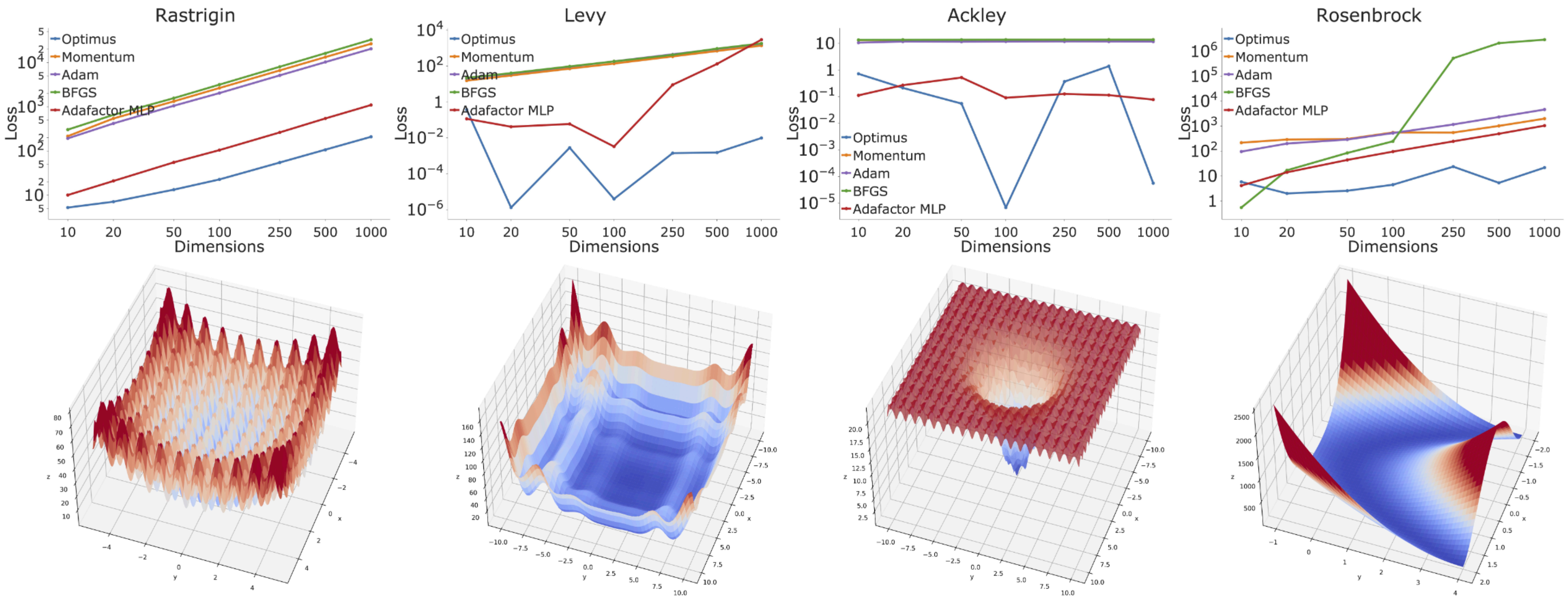
Ackley



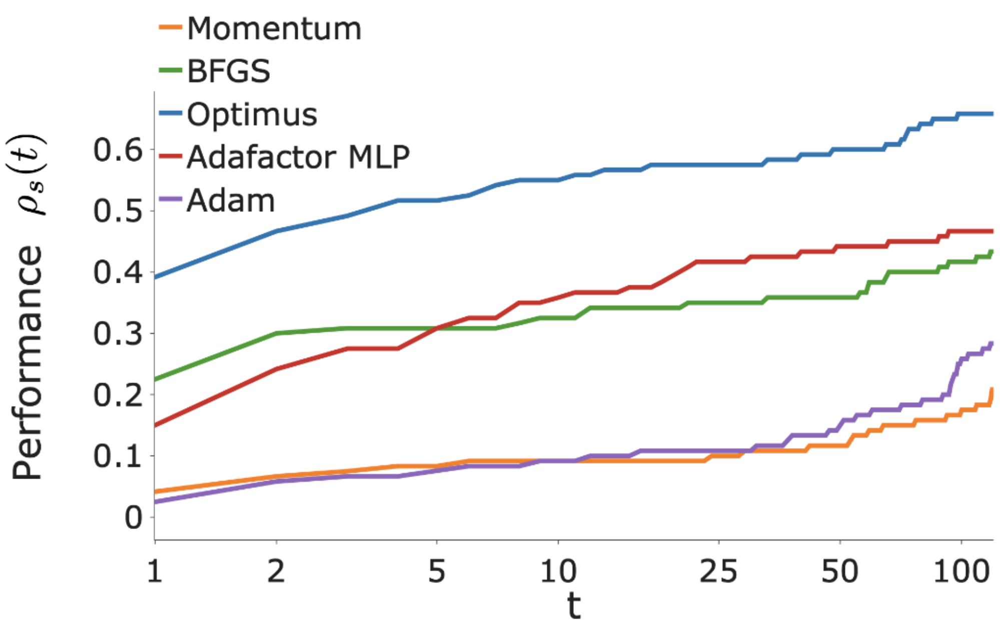
Rosenbrock

L. Metz, C. D. Freeman, J. Harrison, N. Maheswaranathan, and Jascha Sohl-Dickstein. Practical tradeoffs between memory, compute, and performance in learned optimizers. LLA'22

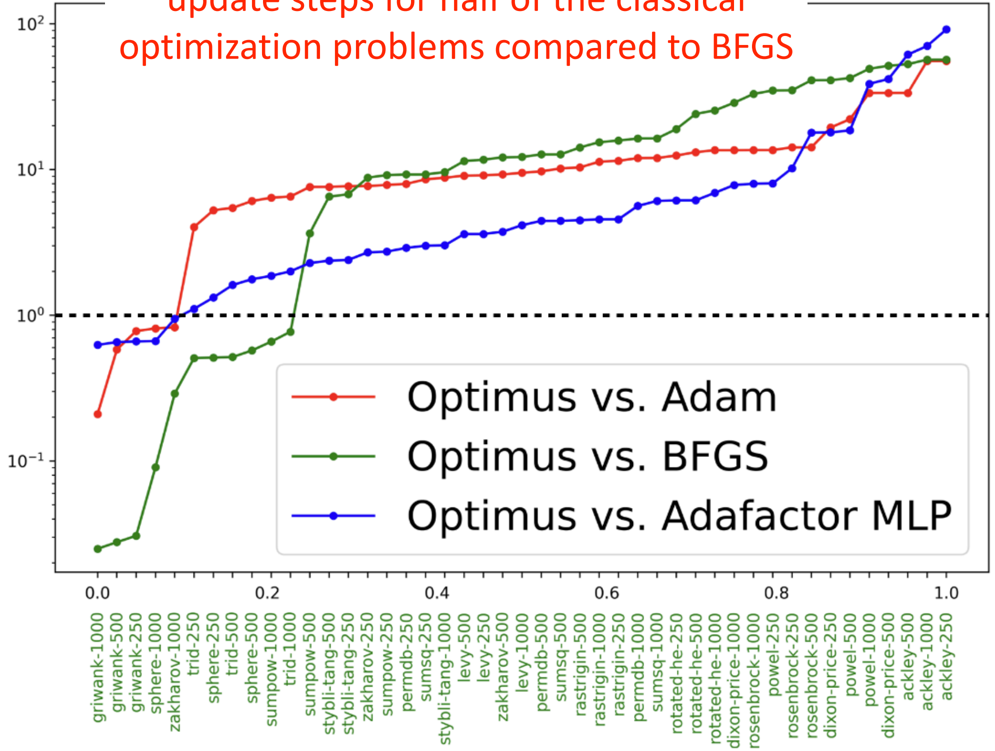
Results on classic objective functions



Results on classic objective functions

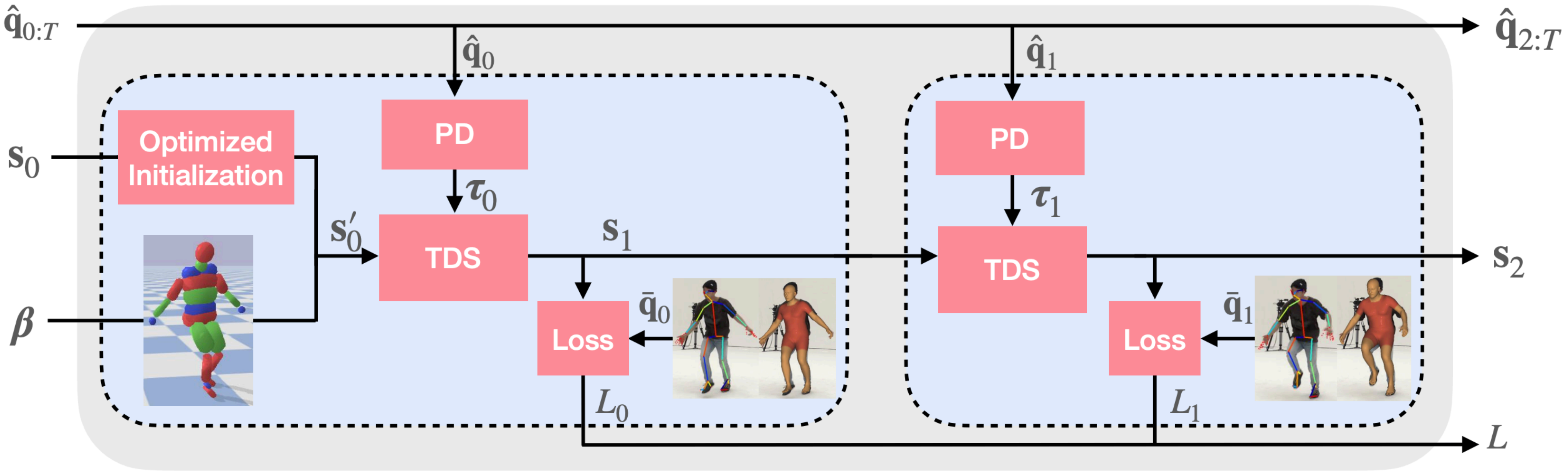


At least a 10x reduction in the number of update steps for half of the classical optimization problems compared to BFGS



V. Beiranvand, W. Hare and Y. Lucet. Best practices for comparing optimization algorithms. Optimization and Engineering, 2017

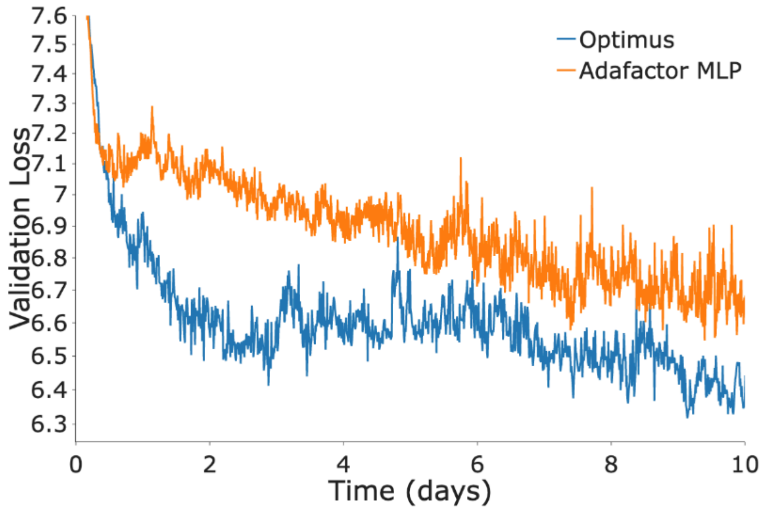
Results on physics-based human motion reconstruction



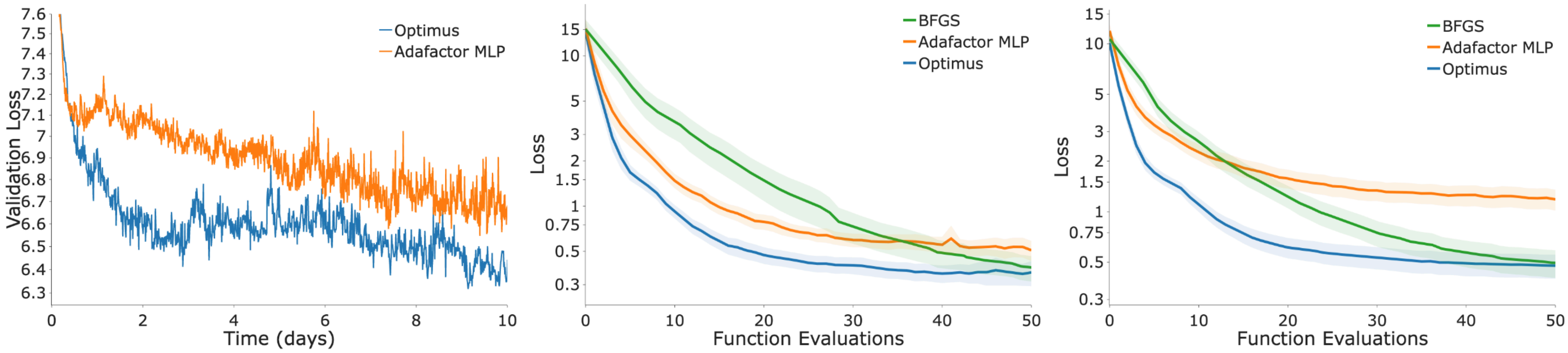
Task: minimize a loss function dependent on the output of physics simulation with respect to initial state $\mathbf{s}_0$ and control trajectory $\hat{\mathbf{q}}_{0:T}$

E. Gärtner, M. Andriluka, E. Coumans, C. Sminchisescu, Differentiable Dynamics for Articulated 3d Human Motion Reconstruction, CVPR'22

Results on physics-based human motion reconstruction



Results on physics-based human motion reconstruction



Results on physics-based human motion reconstruction

Model	MPJPE-G	MPJPE	MPJPE-PA	MPJPE-2d	TV	Foot skate
VIBE [20]	207.7	68.6	43.6	16.4	0.32	27.4
PhysCap [38]	-	97.4	65.1	-	-	-
SimPoE [53]	-	**56.7**	**41.6**	-	-	-
Shimada et al. [37]	-	76.5	58.2	-	-	-
Xie et al. [51]	-	68.1	-	-	-	-
DiffPhy [15]	139.1	82.1	55.9	13.2	0.21	7.2
Optimus (Ours)	**138.6**	82.8	57.0	13.2	**0.20**	**6.5**

Optimus requires on average half of the objective function evaluations compared to DiffPhy

Thank you!