Use Coupled LSTM Networks to Solve Constrained Optimization Problems

Zheyu (Joe) Chen[*], Kin K. Leung[†], Shiqiang Wang[‡], Leandros Tassiulas[§], Kevin Chan[¶], Don Towsley[#]

[*†]Imperial College London, UK
[‡]IBM T.J. Watson Research Center, Yorktown Heights, NY, USA
[§]Yale University, New Haven, CT, USA
[¶]DEVCOM Army Research Lab, Adelphi, MD, USA
[#]University of Massachusetts, Amherst, MA, USA

Motivation

- Conventional gradient-based iterative algorithms may take time to converge
  - Gradient descent method

- To resolve various technical issues in the networks, it may need to repeated solve optimization problems with the same structure but different system parameters
  - To allocate the amount of resource r to a task for maximizing the utility

$$\max_{r} \frac{1}{1 + e^{-(r - R)}}$$

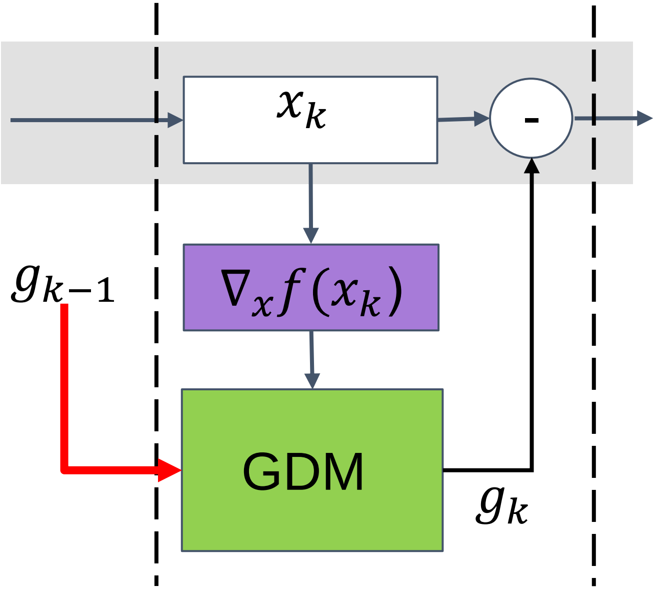
- It requires a new optimal solution when the system parameters (e.g., R) change
  - Requires re-run of optimization process

Desirable to have an approach to quickly produce solutions for a given optimization problem over a range of system parameters.

Reasons for choosing Long Short-Term Memory networks (LSTMs)

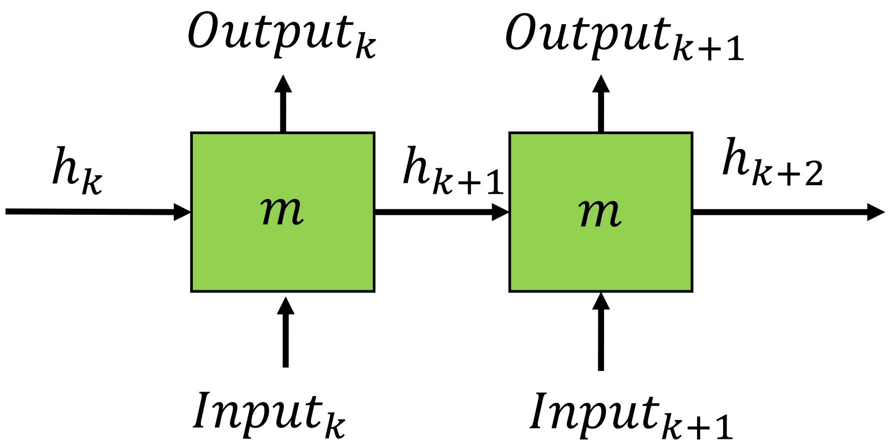
- Target: quickly generate solutions
  - Historical gradient information can help to converge
    - Gradient descent with momentum (GDM)

$$g_k = \gamma g_{k-1} + \alpha \nabla_x f(x_k)$$

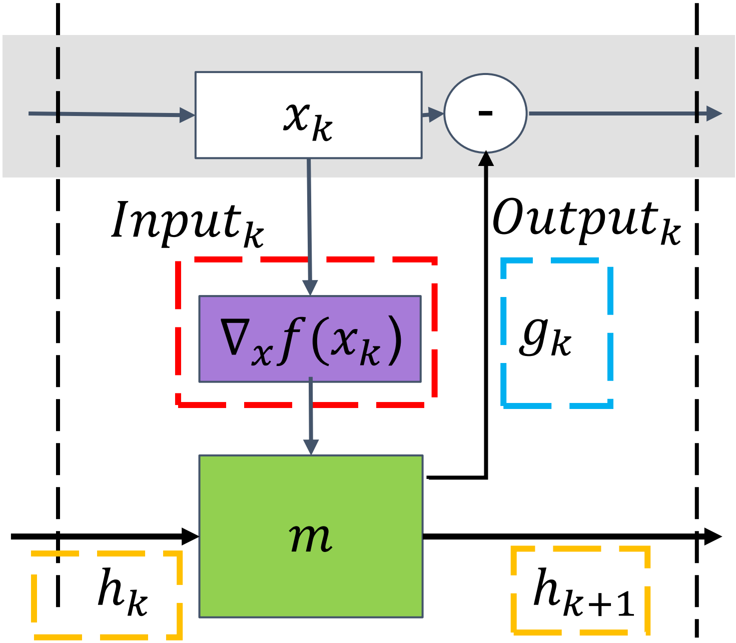


Iteration k

- Workflow of LSTMs



Iteration k



Historical gradient information

Solve Constrained optimization problems by CLSTMs

- For a constrained optimization problem

$$(P1)\ \min_{x} f(x)$$

$$\text{s.t. } h(x) \leq 0$$

- By introducing the Lagrange multiplier λ, we form the Lagrange function $J(x,\lambda)$ and the dual optimization problem $P2$

$$J(x,\lambda) = f(x) + \lambda h(x).$$

$$(P2)\ \max_{\lambda} J(\operatorname{argmin}_{x} J(x,\lambda), \lambda)$$

$$\text{s.t. } \lambda \geq 0$$

- To satisfy $\lambda \geq 0$ and avoid numerical issues, we define a 'smooth' projection function $\psi(\lambda) \geq 0, \forall \lambda$ to form $P3$

$$(P3)\ \max_{\lambda} J(\operatorname{argmin}_{x} J\big(x, \psi(\lambda)\big), \psi(\lambda))$$

❑ Solve Constrained optimization problems by CLSTMs

Assumption:
The strong duality holds (i.e., the duality gap is zero) for P1 and P2, and thus there exists at least a dual optimal λ^* and a primal optimal x^*

- ❑ According to the duality theory, P2 has the same optimal solution for P1 under the condition that the duality gap is zero

- ❑ *Theorem*: Having λ^* as the optimal solution for the problem P3 is equivalent to having u^* as the optimal solution for the problem P2, where $u^* = \psi(\lambda^*)$.

$$(P1)\ \min_x f(x)$$
$$\text{s.t. } h(x) \le 0$$

$$(P3)\ \max_\lambda J(\operatorname*{argmin}_x J\big(x, \psi(\lambda)\big), \psi(\lambda))$$

The proposed CLSTMs aims to find the optimal λ^* and x^* for P3

❑ Solve Constrained optimization problems by CLSTMs

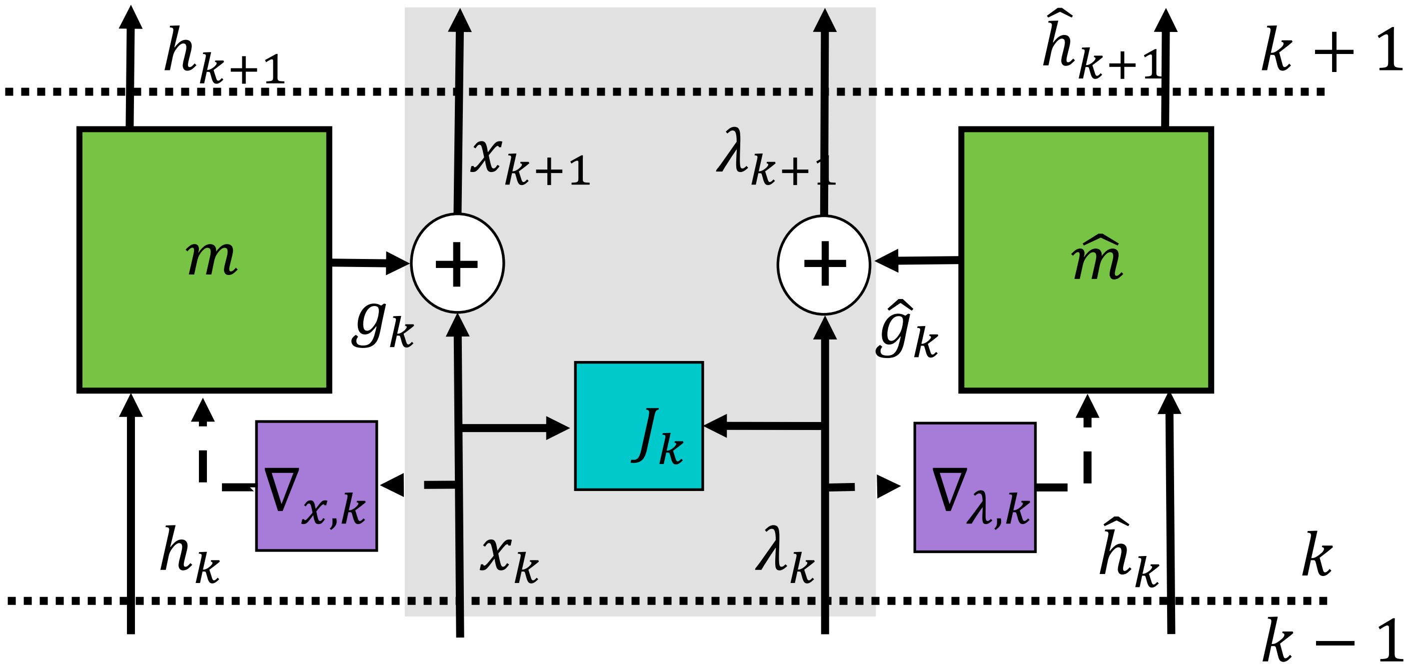
❑ During the inference process, two coupled LSTMs, m and $\hat{m}$, are used to find the optimal x and λ, respectively, by iterations:

$$\begin{bmatrix} \hat{g}_k \\ \hat{h}_{k+1} \end{bmatrix} = \hat{m}(\nabla_\lambda J(x_k, \psi(\lambda_k)), \hat{h}_k, \hat{\phi}),$$

$$\lambda_{k+1} = \lambda_k + \hat{g}_k,$$

$$\begin{bmatrix} g_k \\ h_{k+1} \end{bmatrix} = m(\nabla_x J(x_k, \psi(\lambda_{k+1})), h_k, \phi),$$

$$x_{k+1} = x_k + g_k,$$



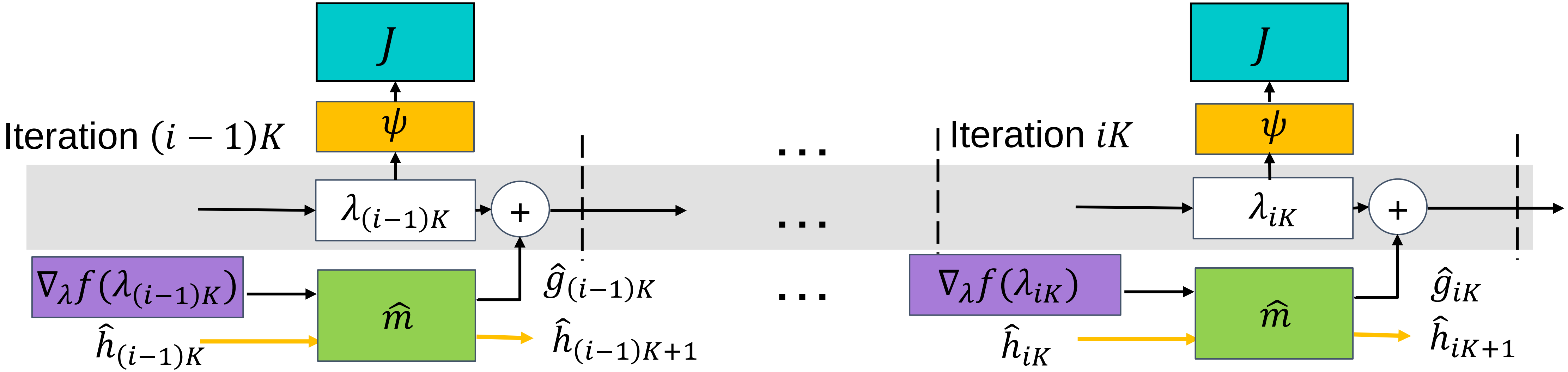
$\nabla_{\lambda,k} = \nabla_\lambda J(x_k, \psi(\lambda_k))$ $\nabla_{x,k} = \nabla_x J(x_k, \psi(\lambda_{k+1}))$

❑ Training of CLSTMs

- ❑ In each iteration, x and λ are updated
- ❑ After K iterations (i.e., one frame), the parameters ϕ_i and $\hat{\phi}_i$ of the LSTM m and $\hat{m}$ are updated to minimize the loss functions:

$$L(\phi_i) = E_f[\sum_{k=(i-1)K}^{iK} w_k\, J(x_k, \psi(\lambda_{k+1}))]$$

$$\hat{L}(\hat{\phi}_i) = -E_f[\sum_{k=(i-1)K}^{iK} \hat{w}_k\, J(x_k, \psi(\lambda_k))]$$



Selection of Projection function $\psi(\lambda)$

- To avoid numerical issues (e.g., during calculating gradients), selection criteria for the projection function are:
  - $\psi(\lambda) \in [0, \infty)$ for all $\lambda \in R$
  - $\psi(\lambda)$ should be differentiable everywhere
  - The derivative of the projection function becomes a non-zero constant, which can be different from 1, when $\lambda \rightarrow \infty$ and -∞
  - The value of the two constants should not be too small or large
- An example of $\psi(\lambda)$ where a can be any even number including 2

$$\psi(\lambda) = \begin{cases} -a\lambda - (a-1), & if\ \lambda < -1 \\ \lambda^a, & if\ -1 \leq \lambda \leq 1 \\ a\lambda - (a-1), & if\ \lambda > 1 \end{cases}$$

Numerical Study: Resource Allocation

- The resource-allocation problem is to allocate cluster resources to competing jobs for maximizing the sum of job utilities.

N	The number of jobs
C	The amount of available resource
r_n	The amount of resource allocated to the job n
R_n	The resource requirement of the job n
$u_n(r_n)$	The utility function given the allocated resource r_n
α, β	Two parameters to set the minimum and maximum amount of resource requirement of job n

$$\max_{r_1,\ldots,r_N} \sum_{n=1}^{N} u_n(r_n)$$

$$s.t. \sum_{n=1}^{N} r_n \leq C$$

$$r_n \geq \alpha R_n, \forall n$$

$$r_n \leq \beta R_n, \forall n$$

❑ Experiment Setup

- ❑ Consider using 5 machines to provide CPU resource to 10 competing jobs
  - In each problem scenario, the amount of available CPU resource and the CPU requirements of jobs are randomly selected from the Alibaba cluster trace
- ❑ Algorithm implementation
  - Each LSTM of the CLSTMs has the layer with 20 neural units.
  - Python and Tensorflow 2.1
  - Evaluated on an Ubuntu 20.04 LTS server with a NVIDIA TITAN XP graphics card

- ❑ Training process uses 5,120 problem scenarios
- ❑ Inference (evaluation process) by the trained CLSTMs
  - 1,000 problem scenarios
  - 2,000 iteration steps for each scenario

❑ Experiment Setup

❑ Metrics: Relative Accuracy

$$\alpha = 1 - \frac{|\hat{f} - f|}{|f|},$$

$\hat{f}$: the optimal value of objective function found by the CLSTMs or the baselines

f: the true optimal value of objective function generated by the fmincon (i.e., provided by the Optimization-toolbox in Matlab R2016)

❑ Mean relative accuracy is the average of the relative accuracy over the 1,000 problem scenarios

❑ Two Baseline Approaches for Comparison

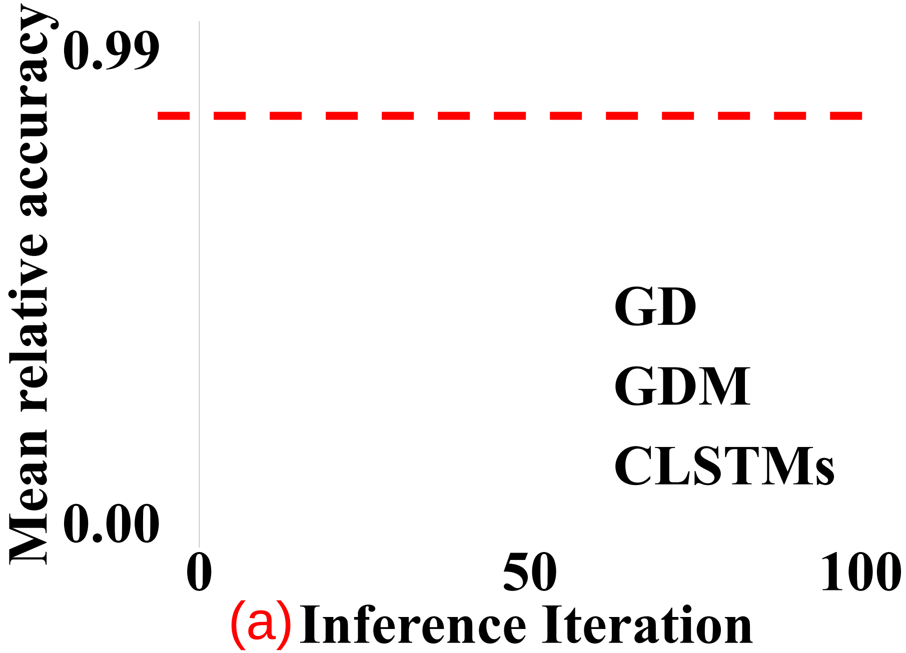
- Gradient descent (GD)
- Gradient descent with momentum (GDM)
- Baseline approach parameters are selected by exhaustively evaluating various parameter combinations

❑ Improvements by the CLSTMs

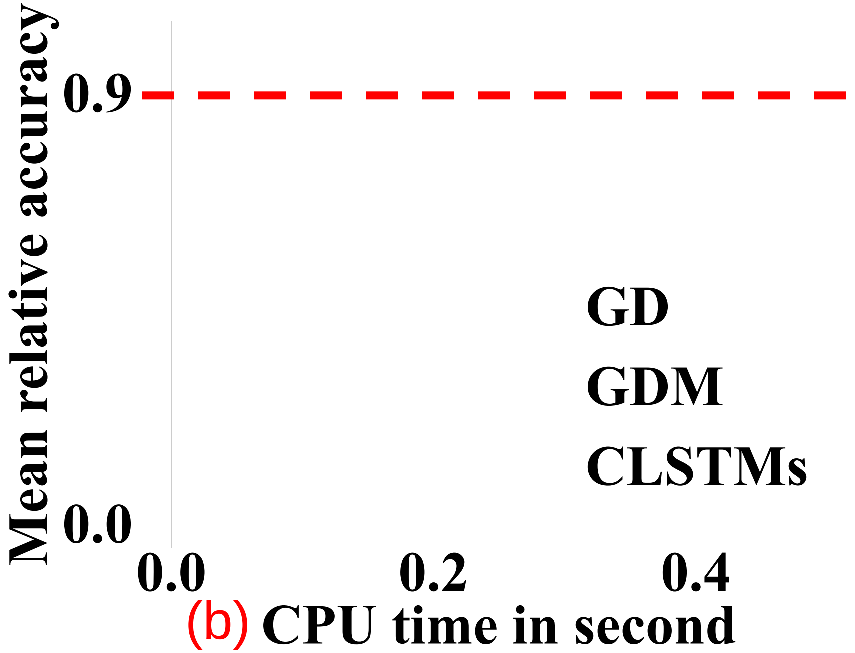
❑ Convex utility functions

$$u_n(r_n) = -\mu_n\left(\frac{r_n}{R_n} - 1\right)^2 + \frac{r_n}{R_n}$$

The mean relative accuracy ($\pm$ one standard deviation) over (a) 100 iterations and (b) CPU time in seconds



(a)



(b)

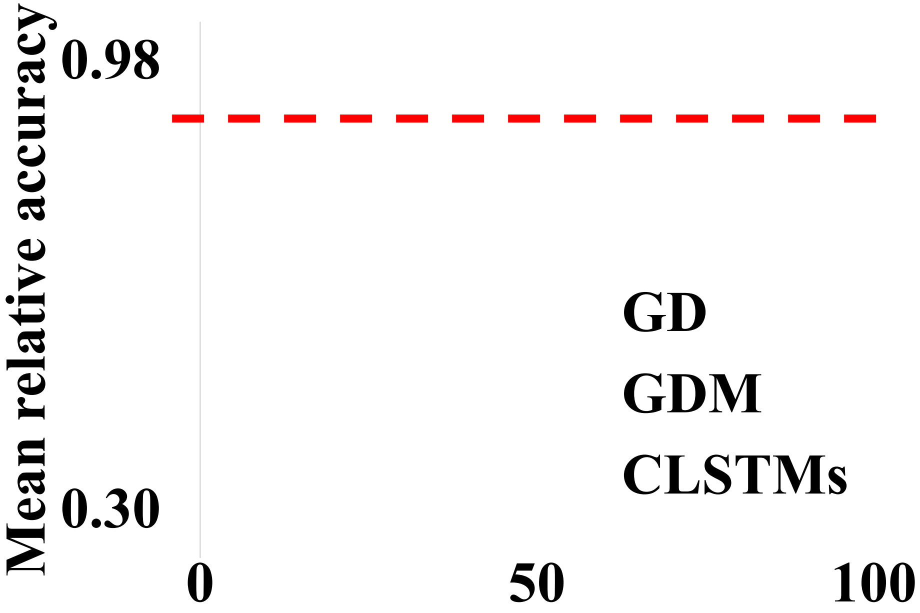
The iteration and CPU time consumption for achieving 90% mean relative accuracy are reduced by 86% and 56% when compared with the GDM, respectively

❑ Improvements by the CLSTMs

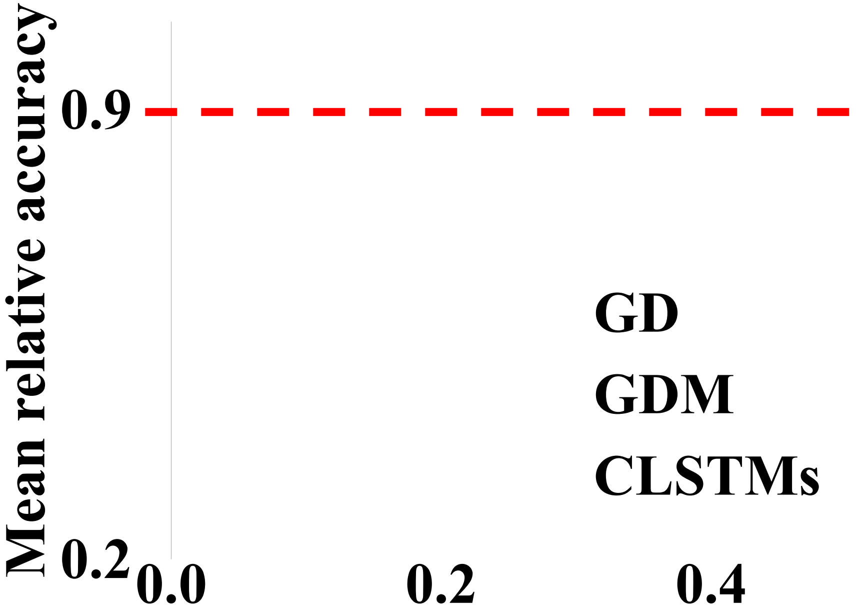
❑ Nonconvex utility functions

$$u_n(r_n) = \frac{1}{1 + e^{-\mu_n(r_n - R_n)}}$$

The mean relative accuracy ($\pm$ one standard deviation) over (a) 100 iterations and (b) CPU time in seconds



(a) **Inference Iteration**



(b) **CPU time in second**

The iteration and CPU time consumption for achieving 90% mean relative accuracy are reduced by 81% and 33% when compared with the GDM, respectively

❑ Impact of projection functions

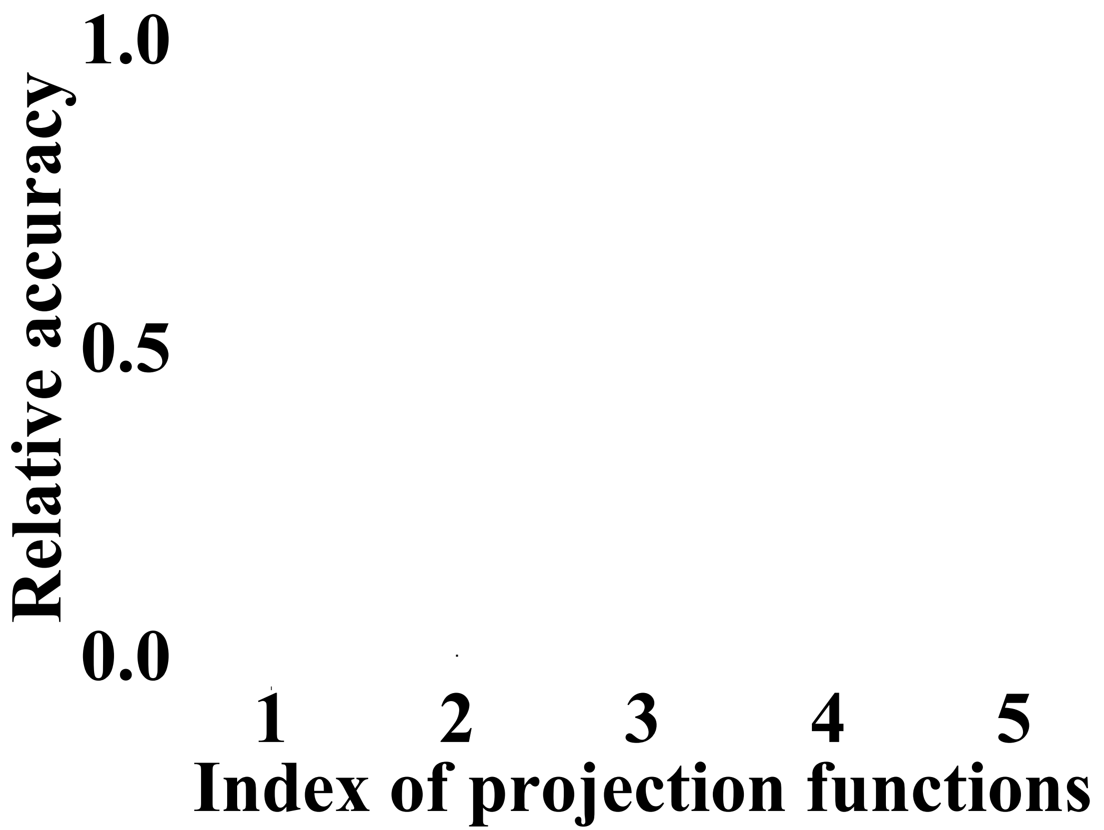
❑ Consider five different projection functions

(1) $\psi(\lambda) = |\lambda|$

(2) $\psi(\lambda) = \frac{1}{2}\left(\sqrt{\lambda^2 + 0.25} + \lambda\right)$

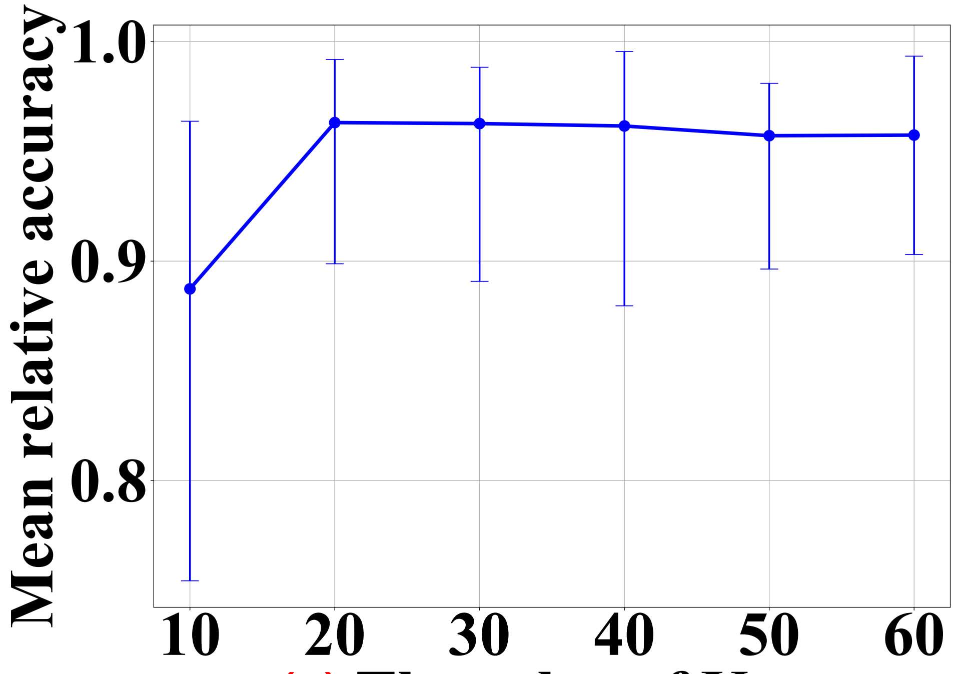
$$\psi(\lambda) = \begin{cases} -a\lambda - (a-1), & if\ \lambda < -1 \\ \lambda^a, & if\ -1 \leq \lambda \leq 1 \\ a\lambda - (a-1), & if\ \lambda > 1 \end{cases}$$

(3) a=2; (4) a=4; (5) a=6

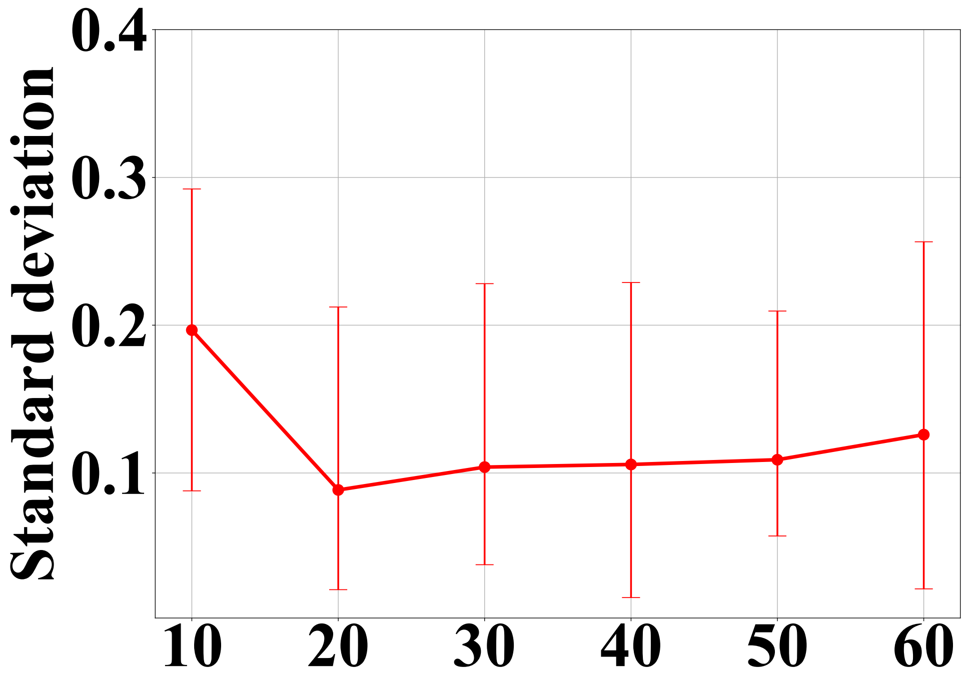


❑ The lower whisker, the bottom of the box, the red horizontal line, the top of the box and the upper whisker represent the 5th, 25th, 50th, 75th and 95th percentile of the relative accuracy, respectively

❑ Robustness: Impact of K



(a) The value of K



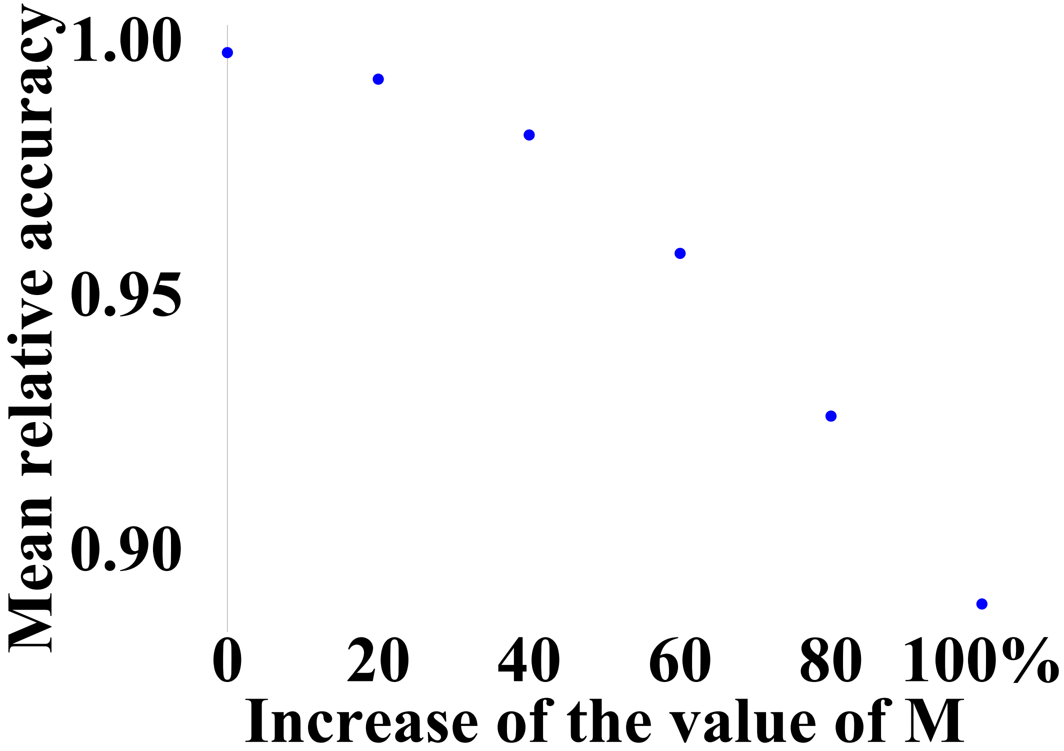
(b) The value of K

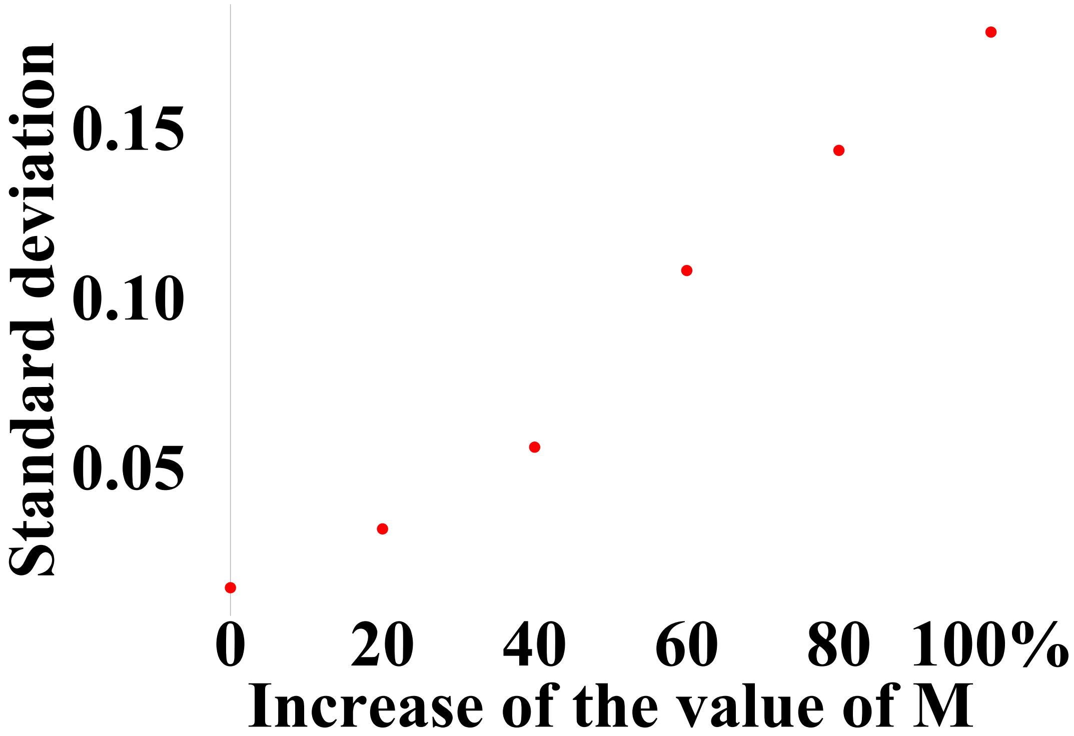
- ❑ Each K value (the parameter in the loss functions) is used to train five CLSTMs with randomly initialized weights for neural networks
- ❑ Fig (a) and Fig (b) present the average, maximum and minimum of the mean relative accuracy and the standard deviation of relative accuracy for five CLSTMs using the same K value, respectively

❑ Robustness: Impact of M

$$u_n(r_n) = -\mu_n\left(\frac{r_n}{R_n} - 1\right)^2 + \frac{r_n}{R_n}$$

- ❑ μ_n is randomly selected from the uniform distribution in the range of [0.001, M)
- ❑ M is set to 1 to generate the training dataset,
- ❑ M is increased by 0, 20, 40, 60, 80 and 100% when generating the six datasets for the evaluation





❑ Robustness: Large numbers of variables and constraints

❑ Nonconvex utility functions $u_n(r_n) = \frac{1}{1 + e^{-\mu_n(r_n - R_n)}}$

The impact of the number of variables on the relative accuracy after 10 iterations and the number of iterations/the CPU time consumed when the mean relative accuracy achieves 0.97

Num of jobs in each problem scenario	Num of variables	Num of constraints	Mean relative accuracy/standard deviation	Num of iterations	CPU time (second)
10	10	21	0.90/0.09	68	1.2
50	50	101	0.94/0.04	66	1.2
70	70	141	0.92/0.06	62	1.0
90	90	181	0.92/0.05	70	1.2
100	100	201	0.96/0.03	66	1.2

Thank you!

Questions?