北京大学
PEKING UNIVERSITY

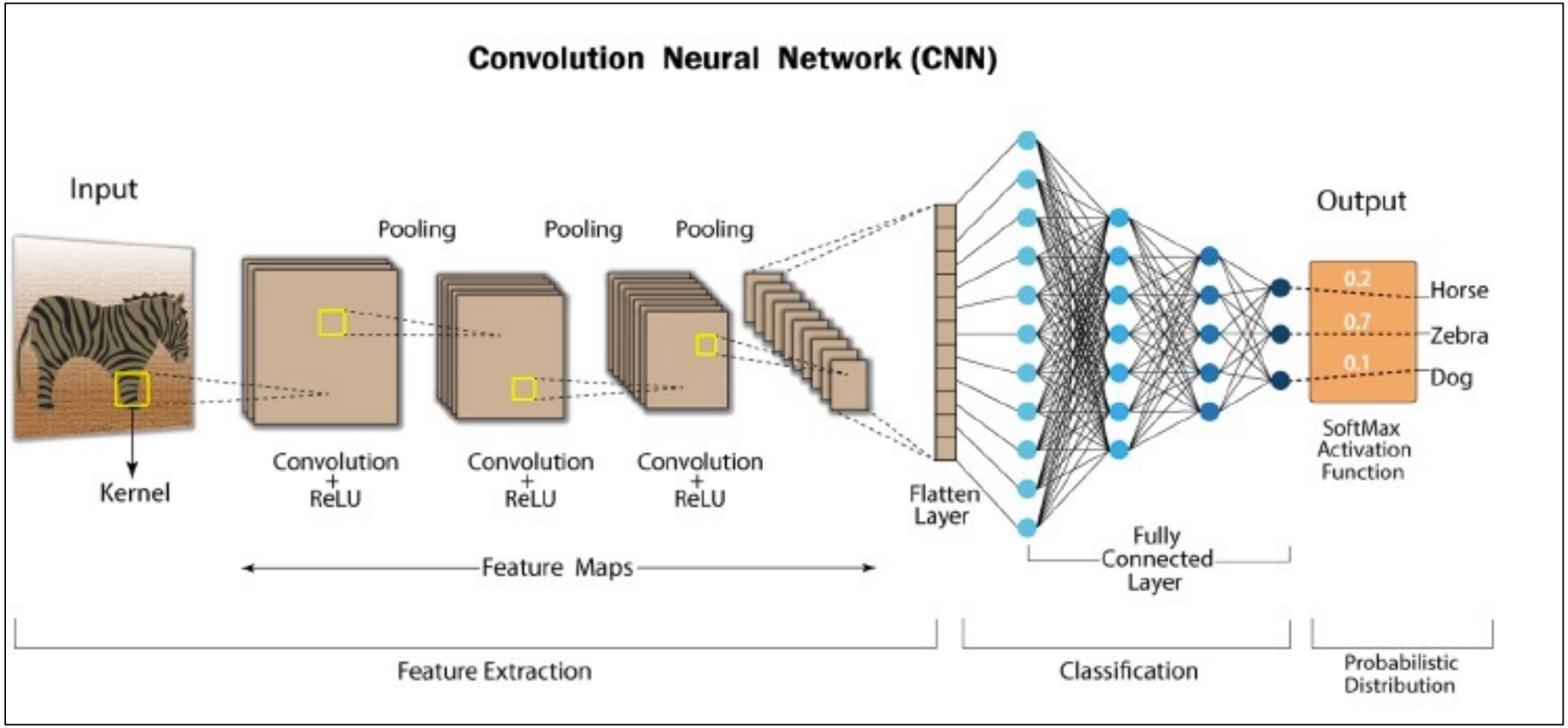
Lecture 12-3: Learning to learn

Kun Yuan

Dec. 5, 2023

Contents

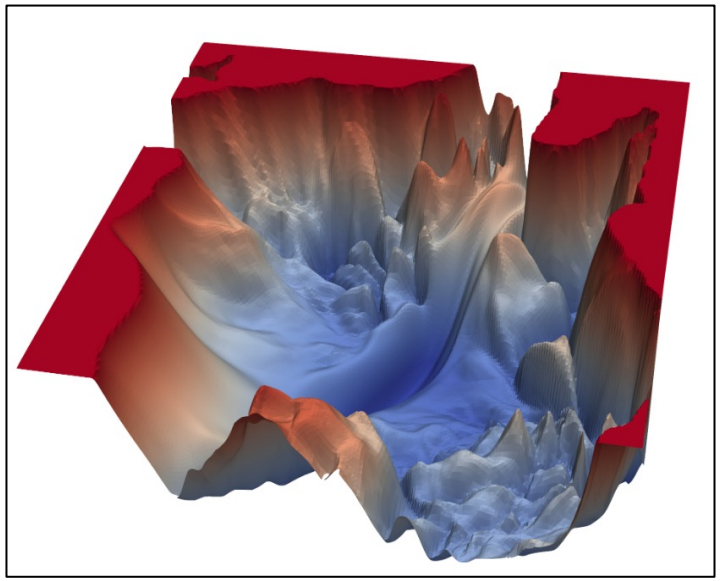
Deep neural network is notoriously difficult to train



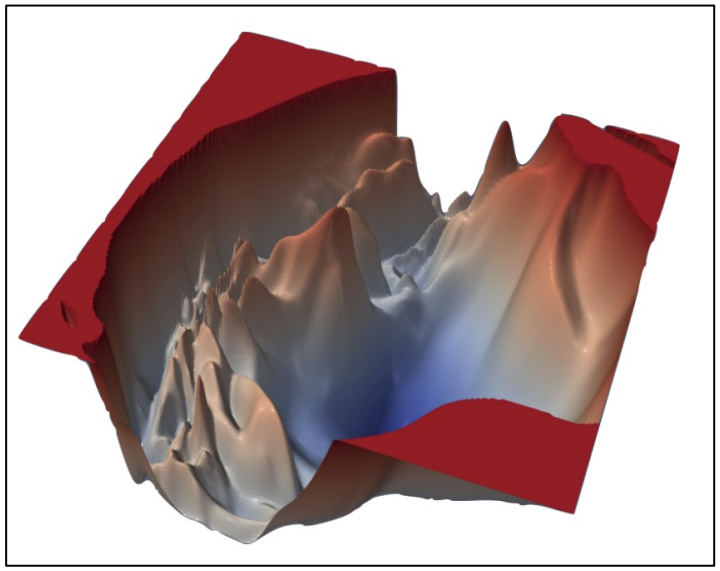
Source: Analyticsvidhya.com

Deep neural network is notoriously difficult to train

VGG-56



VGG-110



Source: Li et. al., Visualizing the loss landscape of neural nets., NeurIPS 2018

How to optimize neural network efficiently?

北京大学 PEKING UNIVERSITY

- SGD
- Momentum SGD / Nesterov accelerated SGD
- AdaGrad/RMSprop/AdaBound/Adam/AdamW

All these algorithms are designed by hands. They are not designed automatically.

Are these algorithms optimal? Probably not. Are there better algorithms? Probably yes.

How to design better algorithms? We do not know yet.

A brand new idea: Learn-to-Optimize

北京大学 PEKING UNIVERSITY

- Replace hand-designed update rules with a learned update rule

- Classical gradient-based algorithm

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t) .$$

- Learned gradient-based algorithm [1]

$$\theta_{t+1} = \theta_t + g_t(\nabla f(\theta_t), \phi) .$$

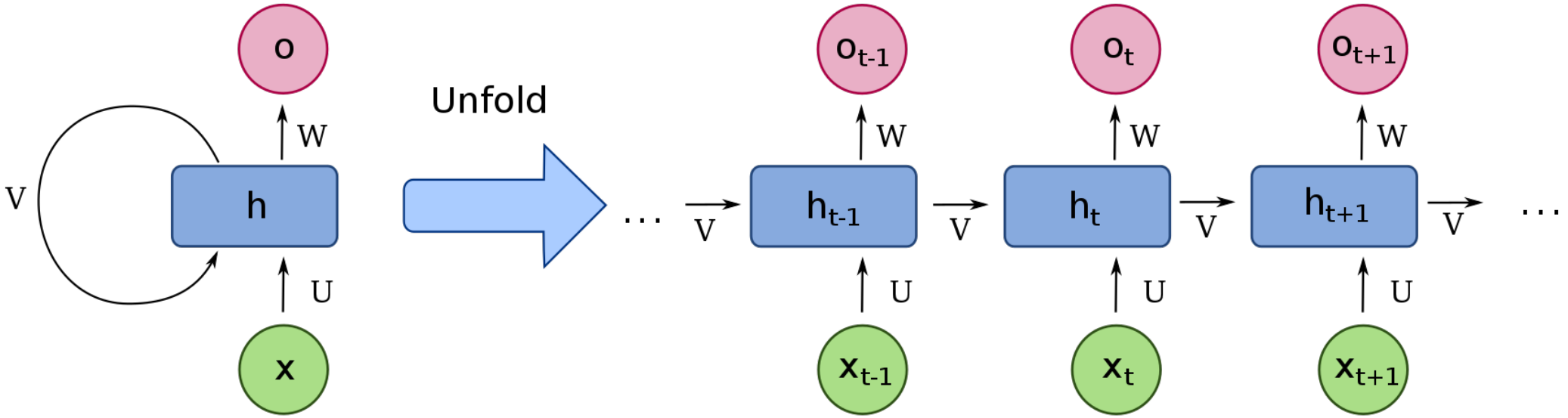
g_t is a mapping to be learned. It is parameterized by ϕ . When taking a gradient, it outputs a better update direction

[1] Marcin Andrychowicz et.al., Learning to learn by gradient descent by gradient descent, NeurIPS 2016

Review RNN

- RNN is to handle message sequences

$$h_t = p(Ux_t + Vh_{t-1})$$
$$o_t = q(Wh_t)$$



Learned optimizer

- We can use RNN to learn a “good” update direction per iteration
- For each update, RNN takes in a gradient $\nabla f(\theta_t)$, outputs a learned update direction g_t
- Then the optimizee updates θ

$$h_t = p(U\nabla f(\theta_t) + Vh_{t-1})$$
$$g_t = q(Wh_t)$$

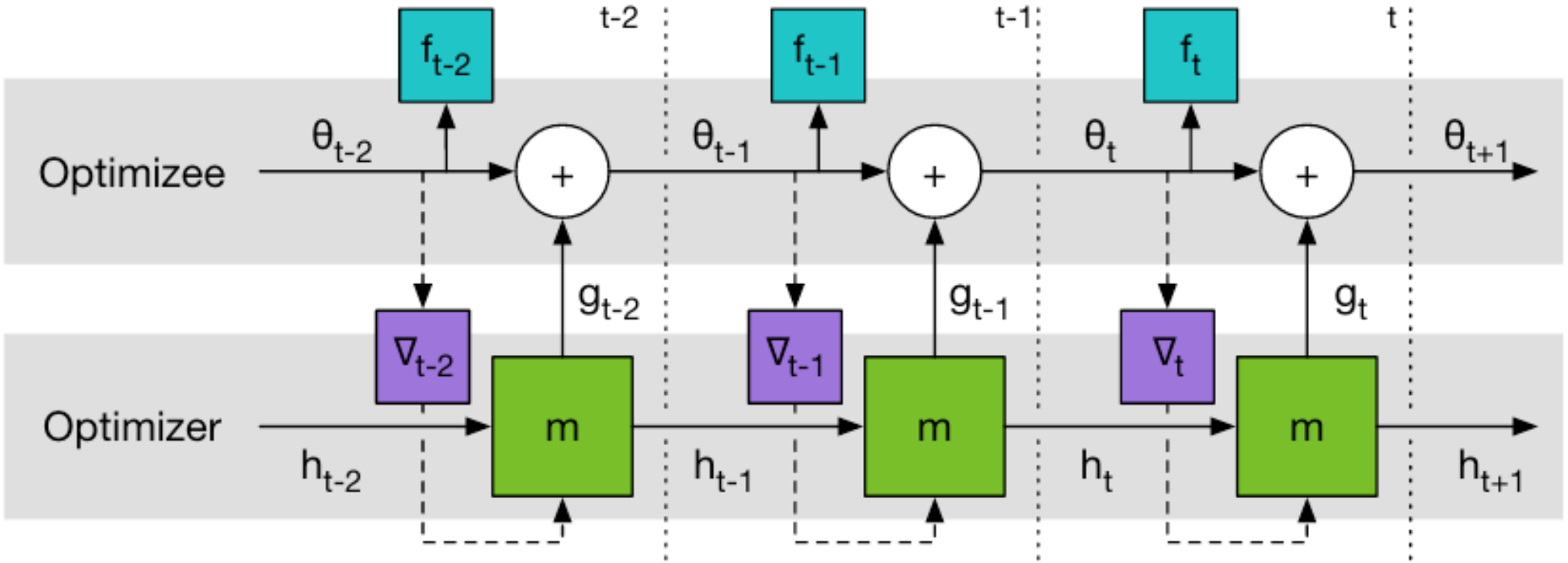
RNN; U, V, W are to be learned

$$\theta_{t+1} = \theta_t + g_t$$

Optimizee

Learned optimizer

$$h_t = p(U\nabla f(\theta_t) + Vh_{t-1})$$
$$g_t = q(Wh_t)$$
$$\theta_{t+1} = \theta_t + g_t$$



How to train optimizer?

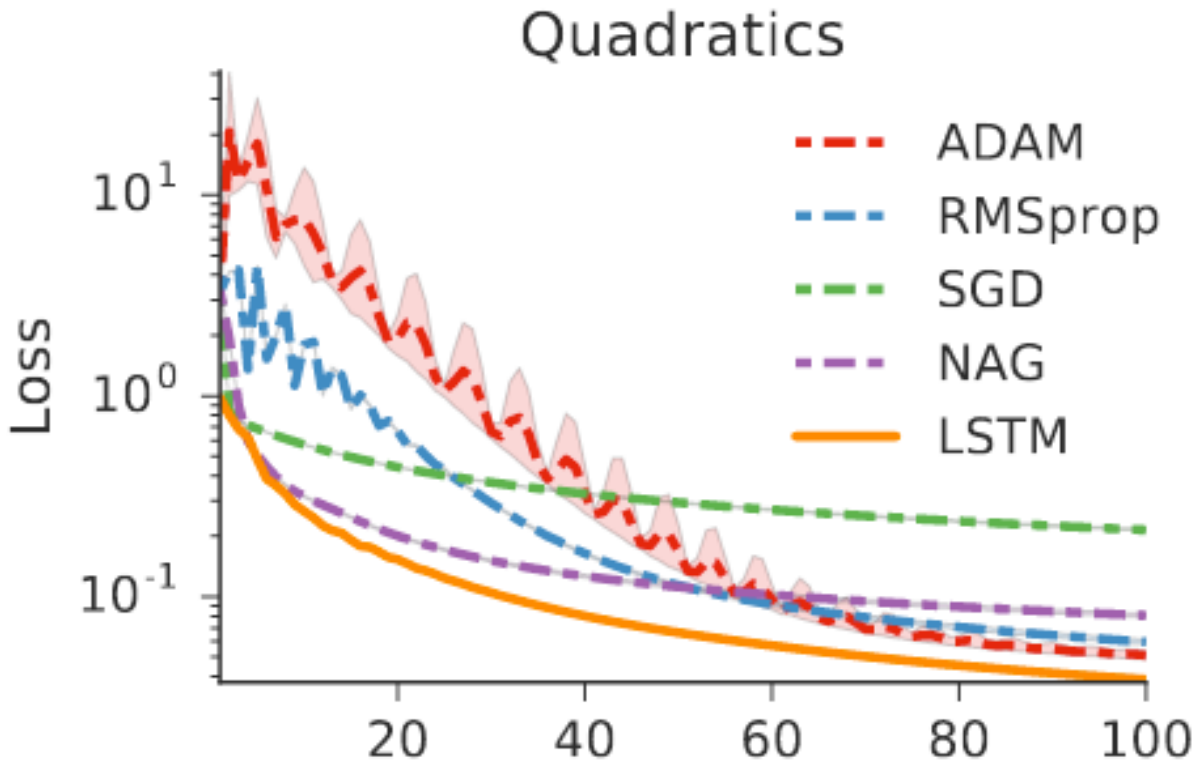
- Loss function: $\mathcal{L}(\phi) = \mathbb{E}_f \left[\sum_{t=1}^{T} w_t f(\theta_t) \right]$ for some predefined w_t

- Learn ϕ by stochastic gradient descent

  - Sample a function f. Compute forward propagation $L(\phi)$
  - Compute stochastic gradient as $\partial L(\phi)/\partial\phi$
  - Update ϕ by stochastic gradient descent
  - Repeat the above procedure until convergence

Experiments

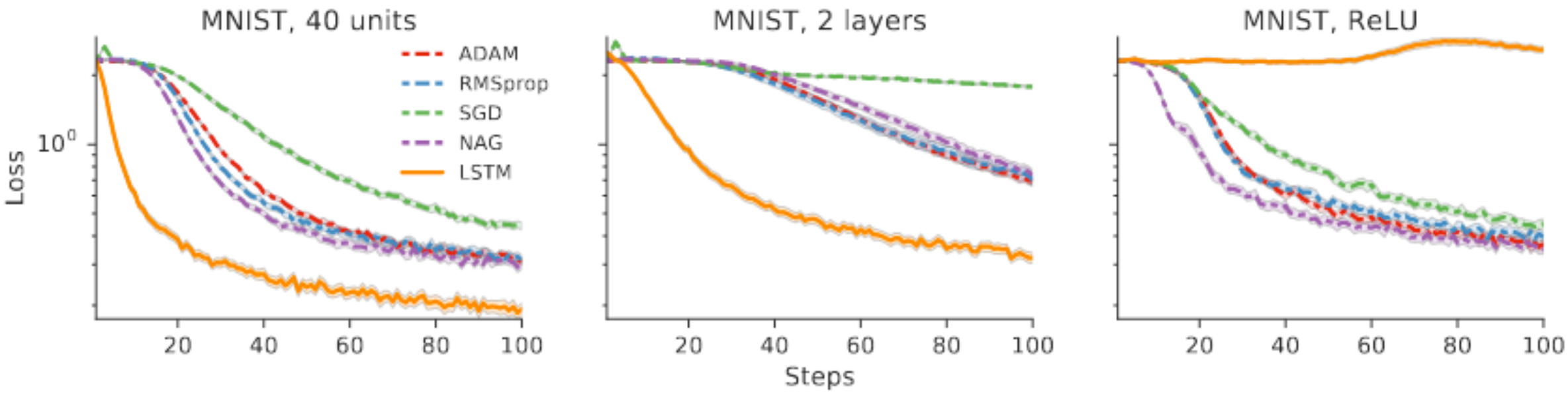
- Quadratic functions: $f(\theta) = \|W\theta - y\|_2^2$



Experiments

北京大学 PEKING UNIVERSITY

- MNIST: train on a neural network with 20 hidden units



Experiments

- Neural arts: train with only 1 style with low-resolution; test with different style images and high-resolution

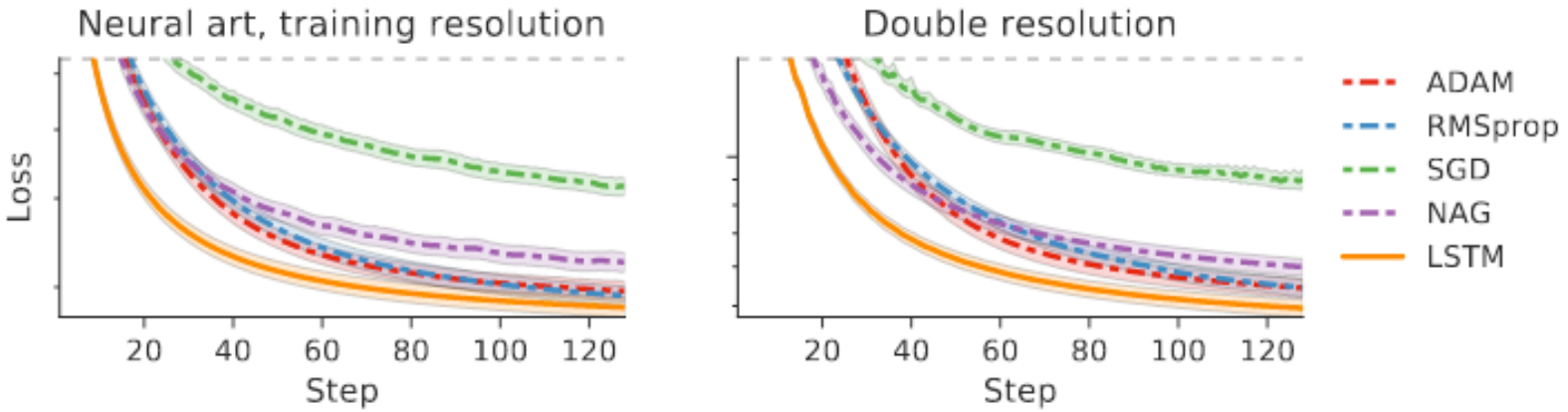


Figure 8: Optimization curves for Neural Art. Content images come from the test set, which was not used during the LSTM optimizer training. Note: the y-axis is in log scale and we zoom in on the interesting portion of this plot. **Left:** Applying the training style at the training resolution. **Right:** Applying the test style at double the training resolution.

Experiments

北京大学 PEKING UNIVERSITY

- Neural arts: train with only 1 style with low-resolution; test with different style images and high-resolution

Figure 9: Examples of images styled using the LSTM optimizer. Each triple consists of the content image (left), style (right) and image generated by the LSTM optimizer (center). **Left:** The result of applying the training style at the training resolution to a test image. **Right:** The result of applying a new style to a test image at double the resolution on which the optimizer was trained.

北京大学
PEKING UNIVERSITY

Learning Fast Approximation of Sparse Coding

Gregor and LeCun

References: Gregor and LeCun ICML 2010; Chen, Liu, Wang, and Yin NeurIPS 2018

Sparse coding

This paper aims to recover a sparse vector x^* from its noisy linear measurements:

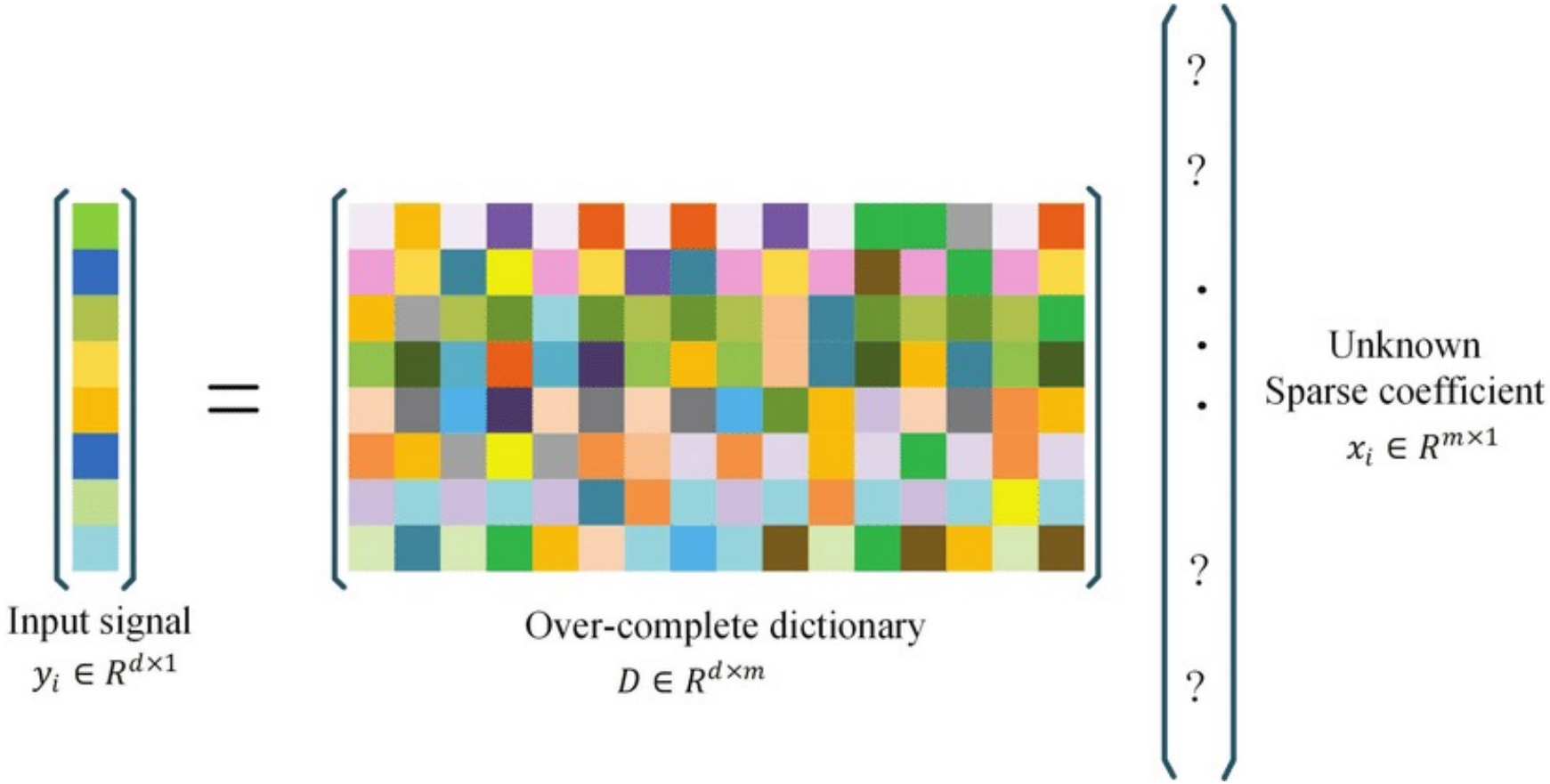
$$b = Ax^* + \varepsilon, \tag{1}$$

where $b \in \mathbb{R}^m$, $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $\varepsilon \in \mathbb{R}^m$ is additive Gaussian white noise, and we have $m \ll n$. (1) is an ill-posed, highly under-determined system. However, it becomes easier to solve if x^* is assumed to be sparse, i.e. the cardinality of support of x^*, $S = \{i | x_i^* \neq 0\}$, is small compared to n.

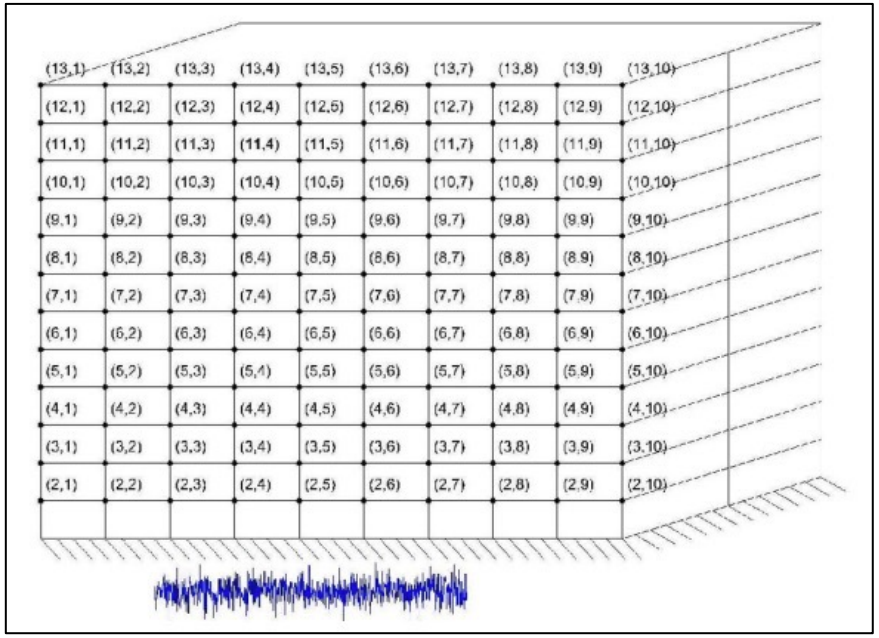
- The above problem can be solved via the LASSO model

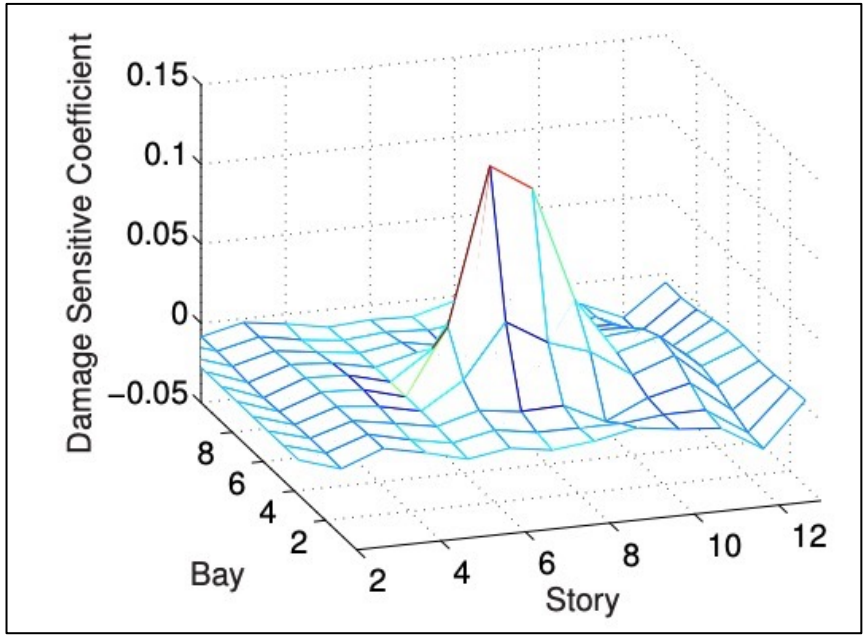
$$\underset{x}{\text{minimize}} \frac{1}{2}\|b - Ax\|_2^2 + \lambda\|x\|_1$$

Sparse coding



Sparse coding application





- N locations and M sensors in which M << N

$$b = Ax + \epsilon, \text{ where } x \in \mathbb{R}^N \text{ and } b \in \mathbb{R}^M$$

- **Events are sparse**

ISTA algorithm

$$\underset{x}{\text{minimize}}\ \frac{1}{2}\|b - Ax\|_2^2 + \lambda\|x\|_1$$

- ISTA algorithm

$$x^{k+1} = \eta_{\lambda/L}\left(x^k + \frac{1}{L}A^T(b - Ax^k)\right), \quad k = 0, 1, 2, \ldots$$

where η_θ is the soft-thresholding function $\eta_\theta(x) = \text{sign}(x)\max(0, |x| - \theta)$

- Let $W_1 = \frac{1}{L}A^T$, $W_2 = I - \frac{1}{L}A^T A$, $\theta = \frac{1}{L}\lambda$. The above recursion becomes

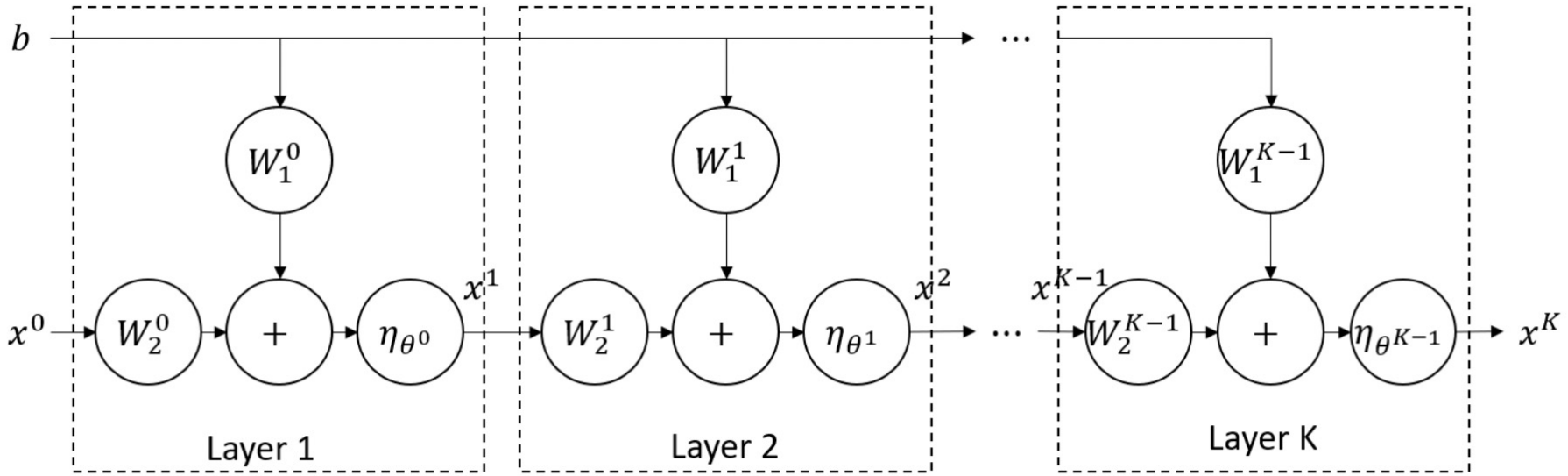
$$x^{k+1} = \eta_\theta\ (W_1\, b + W_2\, x^k), \quad k = 0, 1, \cdots, K-1,$$

- Very similar to RNN with hidden states x and parameters W1, W2, and θ

Learned ISTA algorithm

- We unfold the RNN network into K layers

$$x^{k+1} = \eta_{\theta^k}(W_1^k b + W_2^k x^k), \quad k = 0, 1, \cdots, K-1,$$



(b) Unfolded learned ISTA Network.

Learned ISTA algorithm

北京大学 PEKING UNIVERSITY

- Different from ISTA where no parameters is learnable, LISTA is treated as a specially structured neural network and trained with SGD

- Given training dataset $\{(x_i^*, b_i)\}_{i=1}^N$ sampled from some distribution $\mathcal{P}(x, b)$

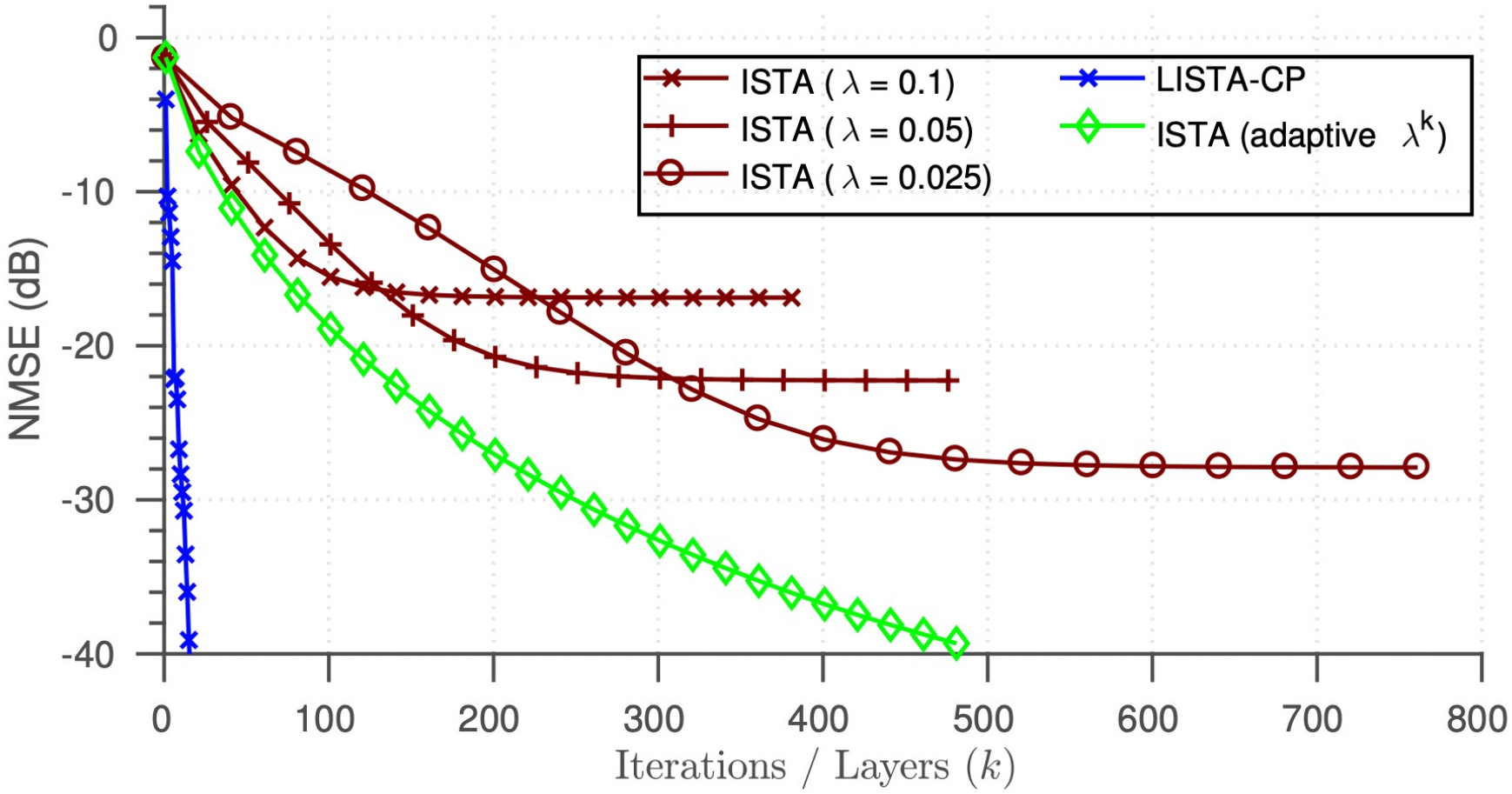
- Let parameters $\Theta = \{(W_1^k, W_2^k, \theta^k)\}_{k=0}^{K-1}$ are subject to learning

- The training is modeled as

$$\underset{\Theta}{\text{minimize}}\, \mathbb{E}_{x^*,b} \left\| x^K\left(\Theta, b, x^0\right) - x^* \right\|_2^2.$$

K = 10 ~ 20

Remarkably surpassing ISTA



Summary

北京大学 PEKING UNIVERSITY

- We reviewed two families of learning-to-learn algorithms:
  - Learning the optimization algorithm by a black-box RNN
  - Parameterize the optimization algorithm and unfold it with DNN

- Promising directions:
  - Learn-to-IntOpt
  - Learn-to-DistOpt

北京大学
PEKING UNIVERSITY

Thank you!