OPTIMIZING BINARY AUTOENCODERS USING AUXILIARY COORDINATES

Ramin Raziperchikolaei and **Miguel Á. Carreira-Perpiñán.** EECS, UC Merced, USA

ICML@NYC
International Conference on Machine Learning

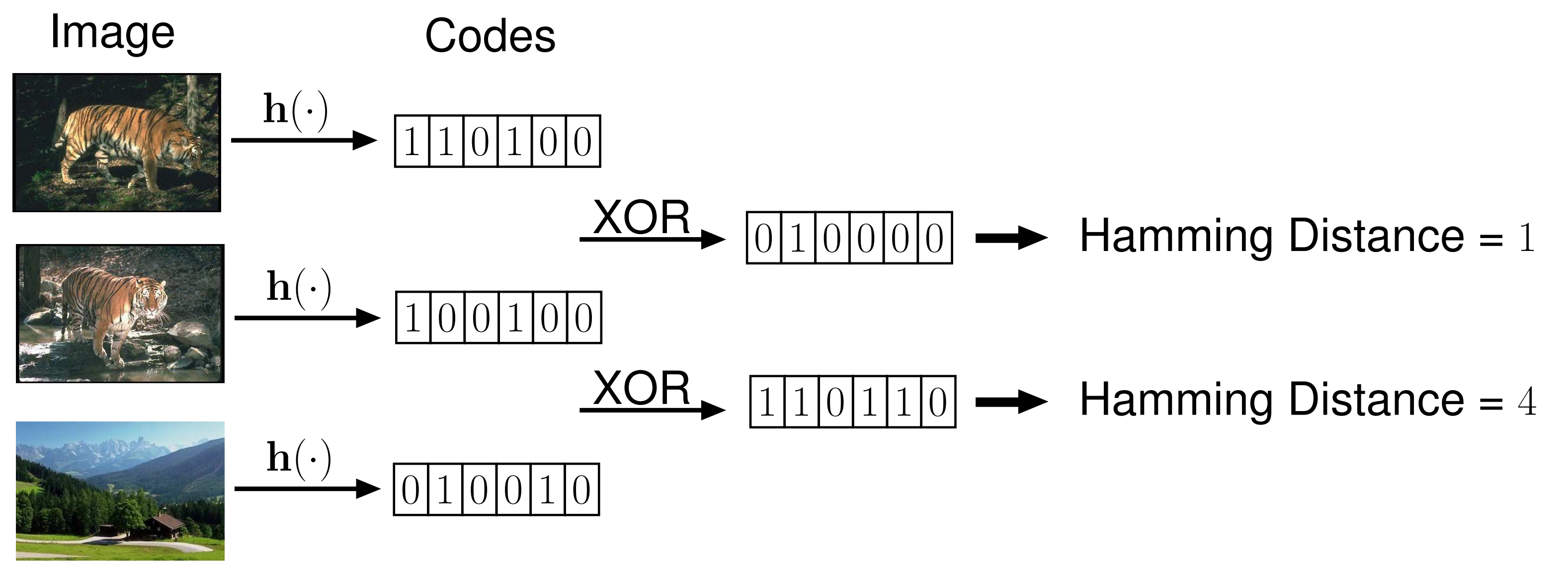
1 Binary hash functions for fast image retrieval

In K nearest neighbors problem, there are N training points in D-dimensional space (usually $D > 100$) $\mathbf{x}_i \in \mathbb{R}^D, i = 1,\dots,N$ and the goal is to find the K nearest neighbors of a query point $\mathbf{x}_q \in \mathbb{R}^D$.

- Exact search in the original space is $\mathcal{O}(ND)$ in time and space.

A binary hash function $\mathbf{h}$ takes as input a high-dimensional vector $\mathbf{x} \in \mathbb{R}^D$ and maps it to an L-bit vector $\mathbf{z} = \mathbf{h}(\mathbf{x}) \in \{0,1\}^L$. The search is done in this low-dimensional, binary space.

- The main goal is preserving the neighborhood, i.e., assign (dis)similar codes to (dis)similar patterns.



Finding K nearest neighbors in the Hamming space is more efficient:

- Time and space complexities would be $\mathcal{O}(NL)$ instead of $\mathcal{O}(ND)$.
- Hamming Distance can be computed efficiently and fast using hardware operations.

Suppose that $N = 10^9$, $D = 500$ and $L = 64$

Search in	Space	Time
original space	2 TB	1 hour
Hamming space	8 GB	10 seconds

2 Difficulties in optimizing the objective function

Many hashing papers formulate an objective function of the hash function $\mathbf{h}$ or of the binary codes.
If the hash function $\mathbf{h}$ was a continuous function, one could compute derivatives over the parameters of $\mathbf{h}$ and then apply a nonlinear optimization method.

In binary hashing, optimization is much more difficult:

- the hash function must output binary values, hence the problem is not just generally nonconvex, but also nonsmooth.
- While the gradients of the objective function do exist wrt $\mathbf{W}$, they are zero nearly everywhere.

Most hashing methods use a suboptimal, "filter" approach:

1. Relax the binary constraints and solve a continuous problem.
2. Binarize the continuous codes by finding a threshold.
3. Fit L classifiers to (patterns $\mathbf{x}$, codes $\mathbf{z}$) to obtain the hash function $\mathbf{h}$.

This is suboptimal because optimizing real codes and then projecting them onto the binary space is not the same as optimizing the codes in the binary space.

We seek an optimal, "wrapper" approach: optimize the objective function jointly over linear mappings and thresholds, respecting the binary constraints while learning $\mathbf{h}$.

3 Our hashing model: Binary Autoencoder

We consider binary autoencoders as our hashing model:

$$E_{\text{BA}}(\mathbf{h},\mathbf{f}) = \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{h}(\mathbf{x}_n))\|^2 \quad \text{s.t.} \quad \mathbf{h}(\mathbf{x}_n) \in \{0,1\}^L.$$

- The encoder $\mathbf{h}\colon \mathbf{x} \rightarrow \mathbf{z}$ maps a real vector $\mathbf{x} \in \mathbb{R}^D$ onto a low-dimensional binary vector $\mathbf{z} \in \{0,1\}^L$ (with $L < D$).
- The decoder $\mathbf{f}\colon \mathbf{z} \rightarrow \mathbf{x}$ maps $\mathbf{z}$ back to $\mathbb{R}^D$ in an effort to reconstruct $\mathbf{x}$.

We use the method of auxiliary coordinates (MAC), a generic approach to optimize nested functions. First, we convert the problem for $E_{\text{BA}}(\mathbf{h},\mathbf{f})$ into an equivalent constrained problem:

$$\min_{\mathbf{h},\mathbf{f},\mathbf{Z}} \sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 \quad \text{s.t.} \quad \begin{array}{l} \mathbf{z}_n = \mathbf{h}(\mathbf{x}_n) \in \{0,1\}^L \\ n = 1,\dots,N. \end{array}$$

that is not nested, where $\mathbf{z}_n$ is the auxiliary coordinate for the output of $\mathbf{h}(\mathbf{x}_n)$. Now we apply the quadratic-penalty method:

$$E_Q(\mathbf{h},\mathbf{f},\mathbf{Z};\mu) = \sum_{n=1}^{N} \left(\|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2 + \mu \|\mathbf{z}_n - \mathbf{h}(\mathbf{x}_n)\|^2 \right) \quad \text{s.t.} \quad \begin{array}{l} \mathbf{z}_n \in \{0,1\}^L \\ n = 1,\dots,N. \end{array}$$

where we start with a small μ and increase it slowly. To optimize E_Q we apply alternating optimization:

- Over $\mathbf{f}$ for fixed $\mathbf{Z}$: $\sum_{n=1}^{N} \|\mathbf{x}_n - \mathbf{f}(\mathbf{z}_n)\|^2$. With a linear decoder this is a straightforward linear regression with data $(\mathbf{Z},\mathbf{X})$.
- Over $\mathbf{h}$ for fixed $\mathbf{Z}$: $\min_{\mathbf{h}} \sum_{n=1}^{N} \|\mathbf{z}_n - \mathbf{h}(\mathbf{x}_n)\|^2$. This separates for each bit $l = 1\dots L$. The subproblem for each bit is a binary classification problem with data $(\mathbf{X},\mathbf{Z}_{\cdot l})$.
- Over $\mathbf{Z}$ for fixed $(\mathbf{h},\mathbf{f})$: $\min_{\mathbf{z}_n} e(\mathbf{z}_n) = \|\mathbf{x} - \mathbf{f}(\mathbf{z}_n)\|^2 + \mu \|\mathbf{z}_n - \mathbf{h}(\mathbf{x})\|^2$. This is a binary optimization on NL variables, but it separates into N independent optimizations each on only L variables.

Although the problem over each z_n is binary and NP-complete, a good or even exact solution may be obtained:

- For small $L \lesssim 16$, this can be solved *exactly* by Enumeration, at a worst-case runtime cost $\mathcal{O}(L^2 2^L)$. We know the solution will be near $\mathbf{h}(\mathbf{x})$ which is very helpful in accelerating the search.
- For larger L, we use alternating optimization over groups of g bits. We find in our experiments that it finds near-global optima *if using a good initialization*.

Advantages of optimizing BA using MAC:

- It respects the binary constraints and optimizes the objective over binary codes and hash function jointly.
- It introduces significant parallelism in optimization over codes and functions.
- The individual steps in alternating optimization are (reasonably) easy to solve.

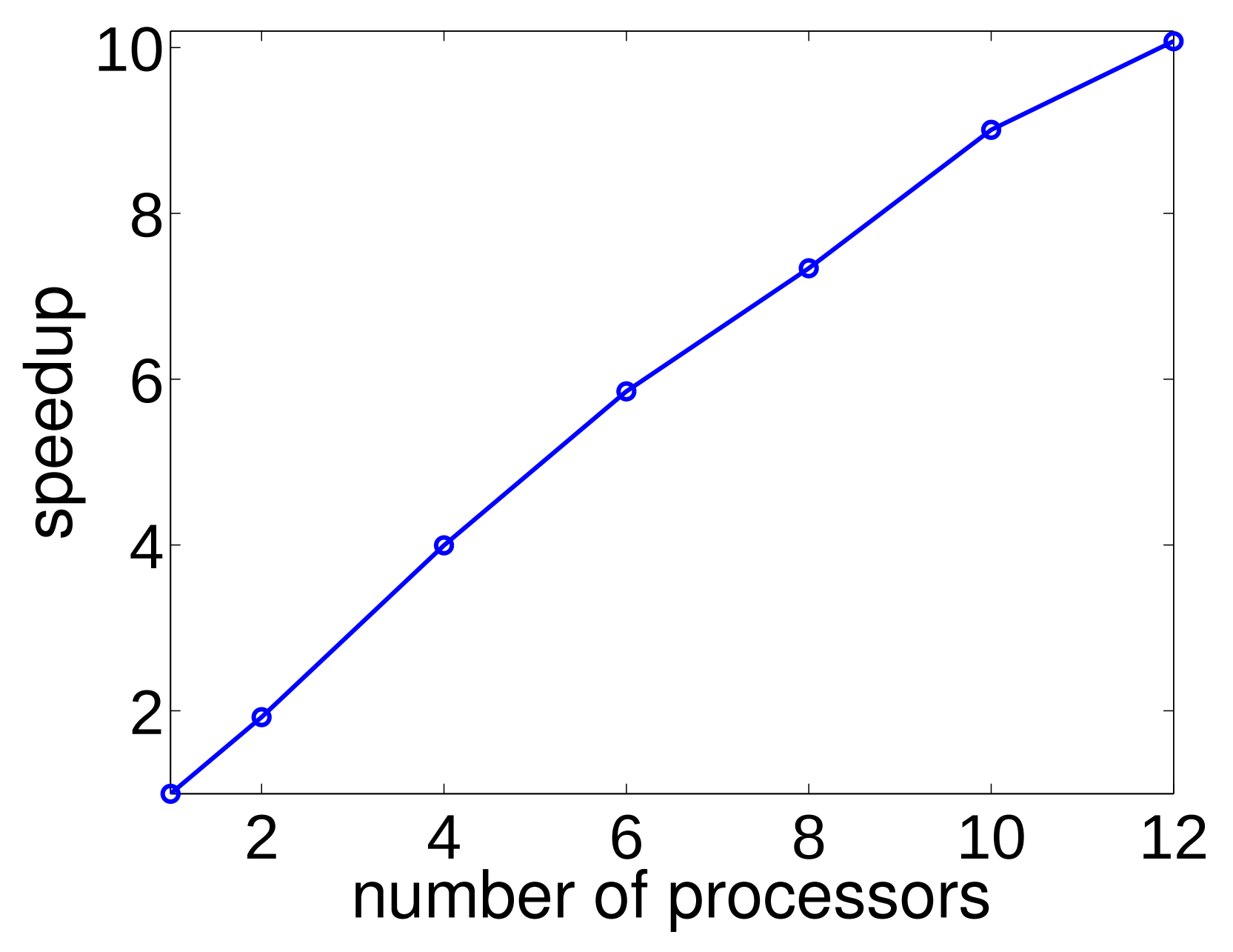
Theoretical results:

Theorem 0.1. *Assume the steps over $\mathbf{h}$ and $\mathbf{f}$ are solved exactly by finding their unique global minimum. Then the MAC algorithm for the binary autoencoder stops at a finite μ.*

Theorem 0.2. *Let $e(\mathbf{z}) = \|\mathbf{x} - \mathbf{f}(\mathbf{z})\|^2 + \mu\|\mathbf{z} - \mathbf{h}(\mathbf{x})\|^2$. Then: (1) A global minimizer $\mathbf{z}^*$ of $e(\mathbf{z})$ is at a Hamming distance from $\mathbf{h}(\mathbf{x})$ of $\frac{1}{\mu}\|\mathbf{x} - \mathbf{f}(\mathbf{h}(\mathbf{x}))\|^2$ or less. (2) If $\mu > \|\mathbf{x} - \mathbf{f}(\mathbf{h}(\mathbf{x}))\|^2$ then $\mathbf{h}(\mathbf{x})$ is a global minimizer.*

4 Experiments

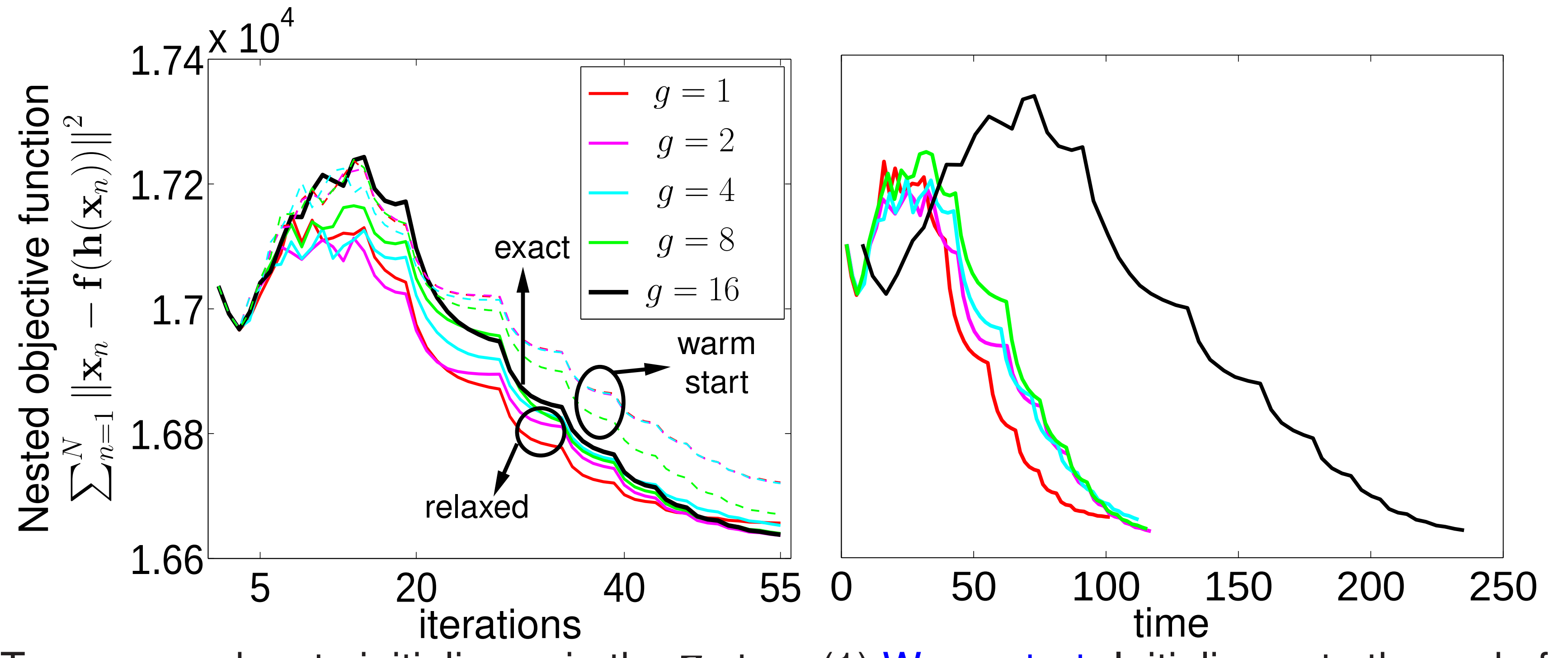
Parallel processing:



The algorithm is highly parallel:

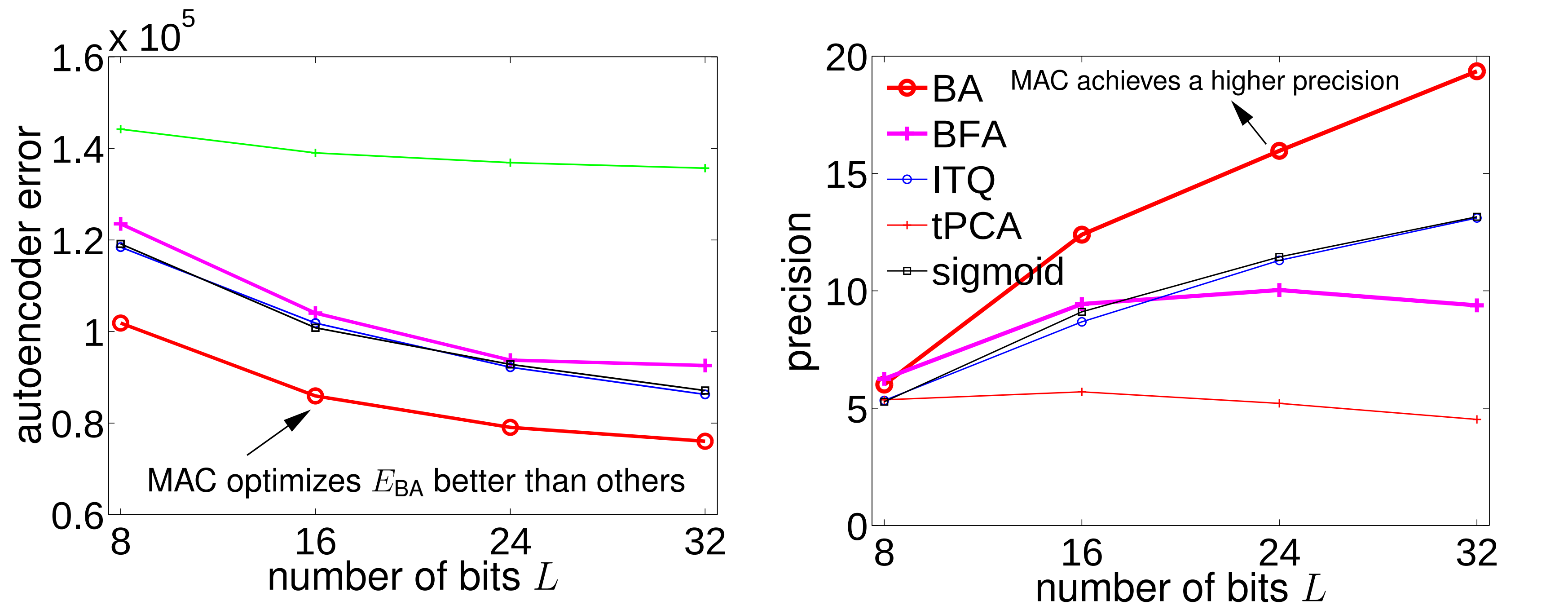
- We train BA on CIFAR dataset.
- We use the Matlab Parallel Processing Toolbox with up to 12 processors and simply replace "for" with "parfor" loops.
- For fixed $\mathbf{Z}$ we have $L{+}1$ independent problems for each of the single bit hash functions, and for $\mathbf{f}$.
- For fixed $\mathbf{h}$ and $\mathbf{f}$ we have N independent optimization problems each over L binary variables.

Alternating optimization and initialization in the $\mathbf{Z}$-step:



Two approaches to initialize $\mathbf{z}_n$ in the $\mathbf{Z}$ step: (1) Warm start. Initialize $\mathbf{z}_n$ to the code found in the previous iteration's $\mathbf{Z}$ step, (2) Solve the relaxed problem on $\mathbf{z}_n \in [0,1]^L$ and then truncate it.The latter achieves better local optima than using warm starts.
Inexact $\mathbf{Z}$ steps achieve solutions of similar quality than exact steps but much faster. Best results occur for $g \approx 1$ in alternating optimization.

How much does respecting the binary constraints help?



NUS-WIDE-LITE dataset, $N = 27\,807$ training/ $27\,808$ test images. Comparison between the methods that minimize the binary autoencoder objective function
BA achieves lower reconstruction error and better precision using MAC than using a suboptimal optimization as in tPCA (truncates codes at zero), ITQ (finds the best rotation matrix), and sigmoid (relaxes the step function to a sigmoid in training).

Work supported by NSF award IIS–1423515