How Low Can We Go: Trading Memory for Error in Low-Precision Training

Chengrun Yang*, Ziyang Wu*, Jerry Chee, Christopher De Sa, Madeleine Udell

Cornell University

Modern neural networks are expensive to train

Training increasingly larger models entails significant computational costs:

- \$80k - \$1.6M to train a BERT model with 1.5-billion parameters
- over \$4.6M to train GPT-3 using a Tesla V100 cloud instance

Memory as a major bottleneck

Heavy memory consumption in training deep models entails surging cost. Specifically, there are three types of memory in deep learning:

- model memory (parameters)
- activation memory (layer outputs; **usually dominate**)
- Optimizer memory (gradients, momentum, ...)

Low-precision (LP) training

Using low-precision representations for network weights, activations, optimizer:

- **(+)** reduces **memory usage**
- **(+)** accelerates **computation**
- **(+)** saves **energy**

but **(−)** introduces **quantization error**.

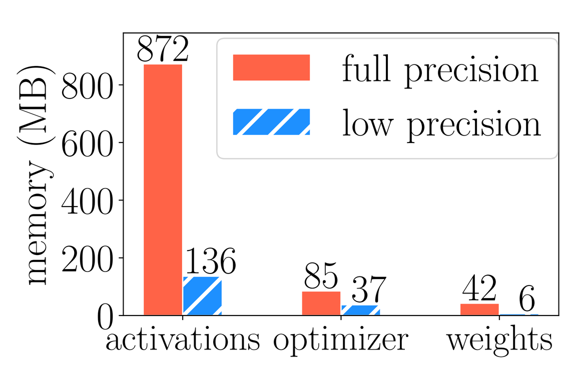
We focus on floating-point (FP) numbers in this work. An example of 8-bit FP format:

7	6	5	4	3	2	1	0
sign	3-bit exponent			4-bit mantissa			

Figure 1: An 8-bit FP format representing $(-1)^{\text{sign}} \cdot 2^{\text{exponent}-7} \cdot 1.b_3b_2b_1b_0$.

Mixed-precision training [1] is a popular and effective training paradigm:

- use lower precision formats (e.g. FP16) for network weights and activations
- use higher precision formats (e.g. FP32) for the optimizer
- define *low-precision configuration* = {lower precision format, higher precision format}

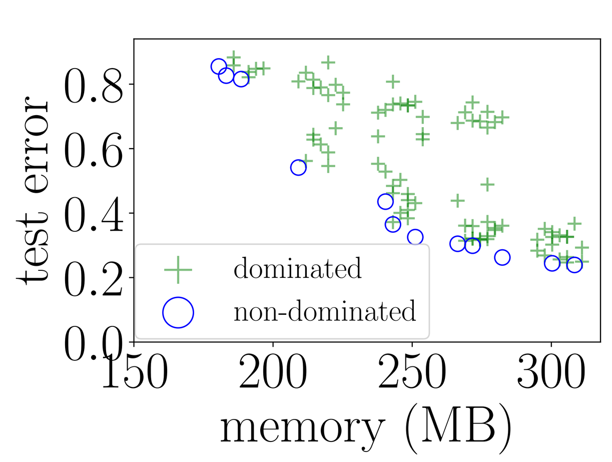


Tradeoff between cost and error

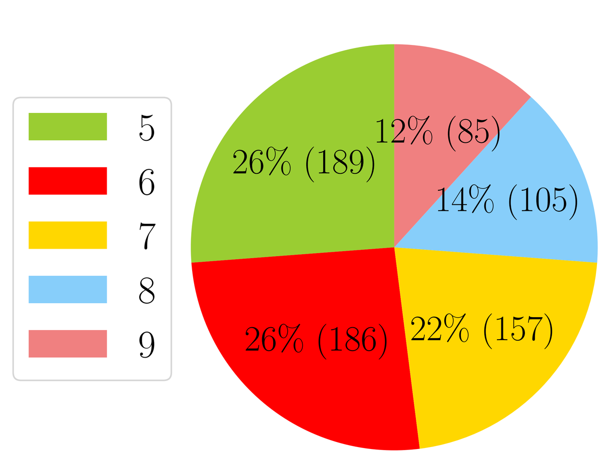
Fundamental challenge: how to select a proper low-precision configuration?

Low-precision training requires careful tuning to reduce cost and control error.

- **hyperparameters**: what are the best low-precision formats?
- **goal**: efficiently pick the best low-precision configuration under a memory budget.
- **key**: identify the *Pareto frontier* (i.e. set of non-dominated low-precision configurations), which characterizes the tradeoff between memory and error.



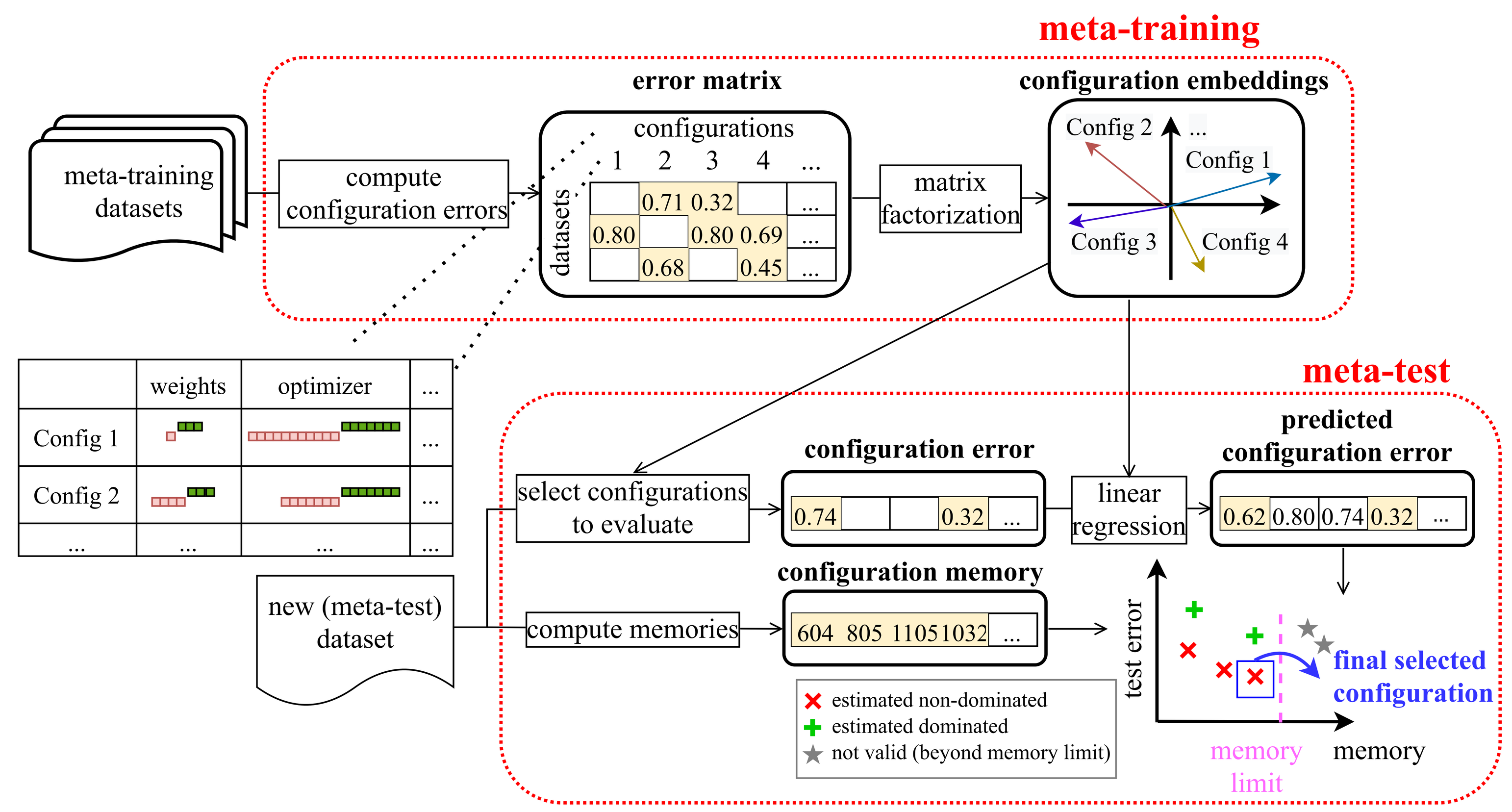
(a) error vs memory tradeoff on CIFAR-10, across 99 LP configurations



(b) # activation bits in non-dominated configurations, on 87 image datasets

PEPPP: Pareto Estimation to Pick the Perfect Precision

We propose PEPPP, a novel AutoML system that studies the error-memory tradeoff in low-precision training and facilitates inference without exhaustive search:



PEPPP contains two main stages:

- **meta-training**: find the Pareto frontiers of a collection of related tasks.
- **meta-test**: efficiently estimate the Pareto frontier of a novel task, based on information learned on previous tasks.

PEPPP Methodology

Meta-training: learn from related tasks

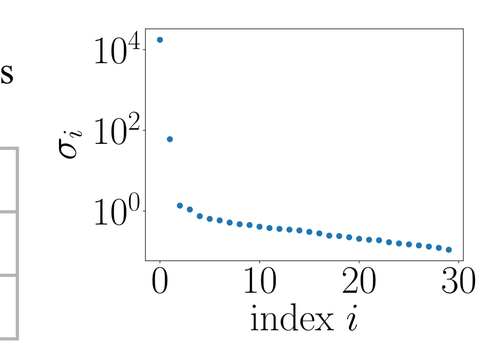
1. Collect the error and memory matrices:

datasets	a	b	c	d
1	0.85	0.71	0.32	0.28
2	0.80	0.80	0.80	0.69
3	0.80	0.68	0.46	0.45

(a) error matrix E (columns: low-precision configurations)

datasets	a	b	c	d
1	181	269	272	295
2	180	269	272	295
3	590	924	927	1032

(b) memory matrix M (MB) (columns: low-precision configurations)



(c) singular value decay

With a specific model (e.g., ResNet-18),

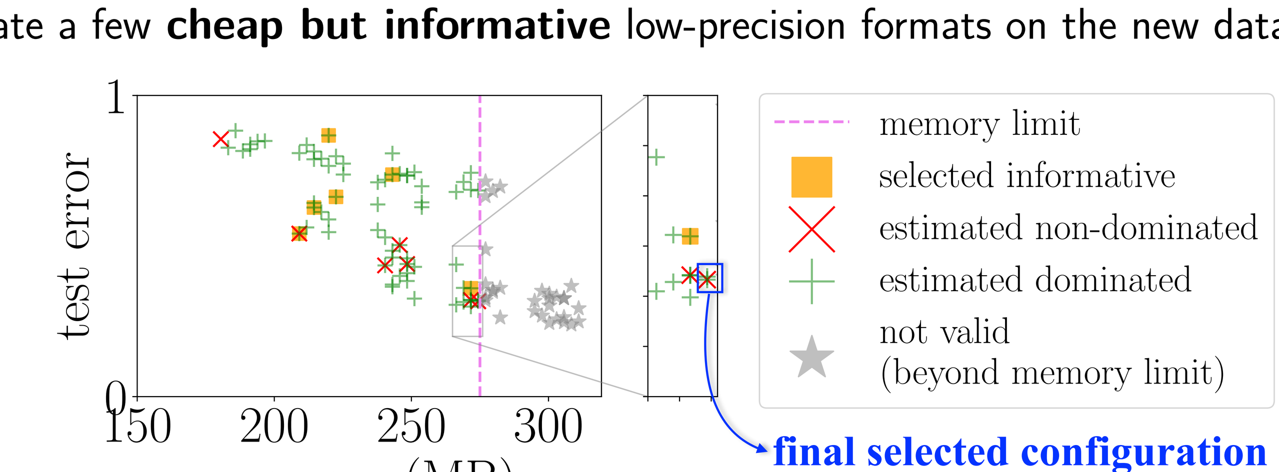
- E_{ij}: test error of the j-th low-precision configuration on the i-th dataset
- M_{ij}: training memory with j-th low-precision configuration on the i-th dataset

2. Obtain embeddings for datasets and LP formats by **matrix factorization** (MF)

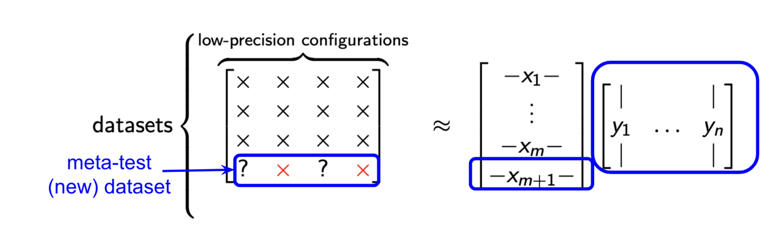
$$\text{datasets}\left\{\underbrace{\begin{bmatrix} \times & \times & \times & \times \\ \times & \times & E & \times \\ \times & \times & \times & \times \end{bmatrix}}_{\text{low-precision configurations}}\right. \approx \begin{bmatrix} -x_1- \\ X \\ -x_m- \end{bmatrix} \begin{bmatrix} | & & | \\ y_1 & Y & y_n \\ | & & | \end{bmatrix}$$

Meta-test: efficiently extrapolate to new tasks

- Evaluate a few **cheap but informative** low-precision formats on the new dataset:



- **Selection method**: experiment design with matrix factorization (ED-MF)



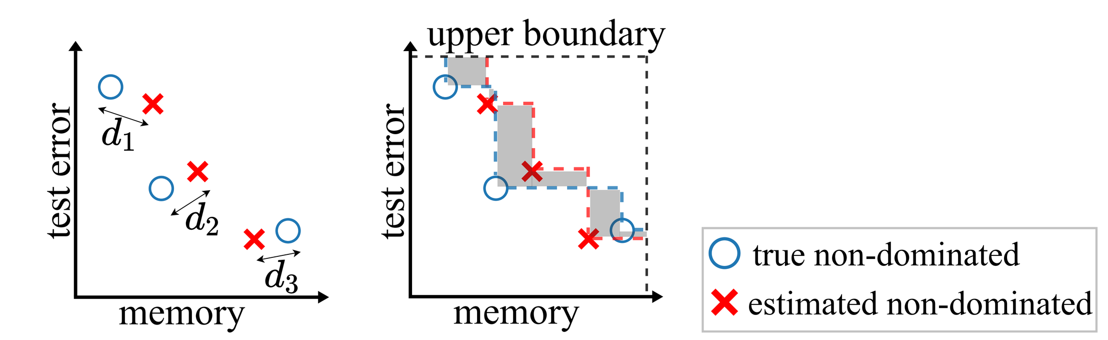
(a) Linear regression to avoid exhaustive search.
×: configurations selected by ED-MF
?: configurations whose performance is predicted

$$\begin{aligned} \text{minimize} \quad & \log\det \underbrace{\Big(\sum_{j\in S} y_j y_j^\top\Big)^{-1}}_{\text{variance of the meta-test dataset embedding}} \\ \text{subject to} \quad & |S| \le \ell \\ & S \subseteq [d] \end{aligned}$$

(b) D-optimal experiment design

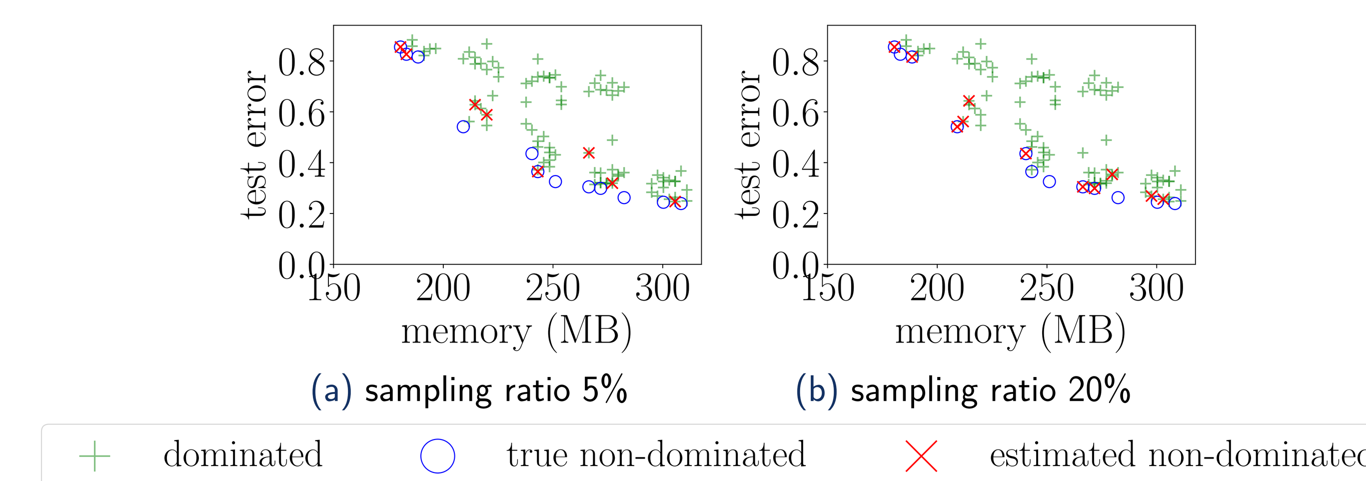
Experimental results

Setup: 87 vision datasets on 99 different low-precision configurations, across 6 architectures.
Evaluation metrics: (a) convergence, (b) HyperDiff.



(a) convergence (b) HyperDiff

Meta-training: more evaluations improve prediction on missing configurations.



(a) sampling ratio 5% (b) sampling ratio 20%

Meta-test: ED-MF consistently outperforms competing methods:

- BO-MF: Bayesian optimization with matrix factorization
- BO-full: Bayesian optimization with full meta-training data
- Random-MF: random selection with matrix factorization
- random high-memory: randomly select the LP configuration with highest memory

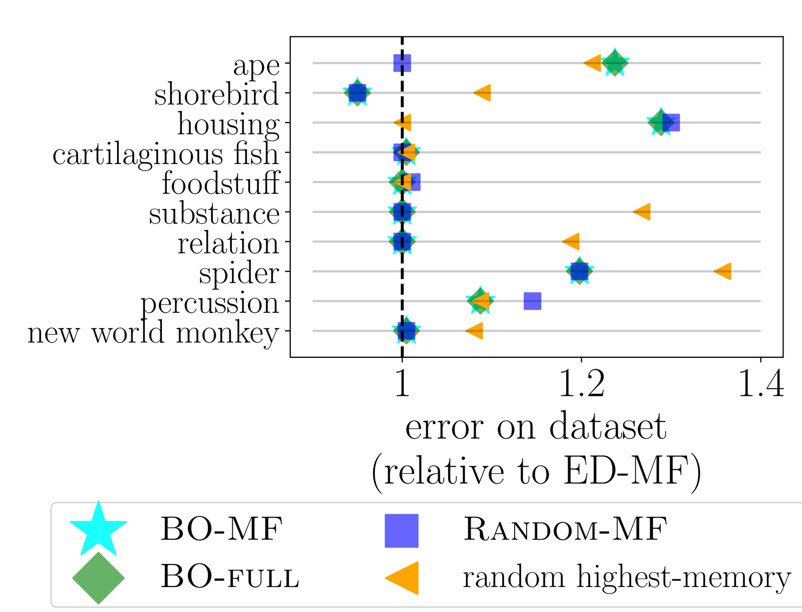


Figure 7: Relative performance with respect to ED-MF in meta-test. ED-MF outperforms in most cases.

Thanks!

- Chengrun Yang: cy438@cornell.edu
- Ziyang Wu: zw287@cornell.edu

Bibliography

[1] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.