Data-Driven Algorithms for Gaussian Measurement Matrix Design in Compressive Sensing

Presentation Video

Yang Sun and Jonathan Scarlett, National University of Singapore

Abstract

What about?

- **Data-driven algorithms for learning compressive sensing measurement matrices**

- **Common approach**: Ubiquitous i.i.d. Gaussian design
- **Ours**: Place more energy on the “most important” parts of the signal

Setup

Compressive Sensing Introduction

Recover a target signal $\mathbf{x} \in \mathbb{R}^n$ via linear measurements of the form

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{z}$$

- $\mathbf{y} \in \mathbb{R}^\ell$ is the measurement vector
- $\mathbf{A} \in \mathbb{R}^{\ell \times n}$ is the measurement matrix
- $\mathbf{z} \in \mathbb{R}^\ell$ is the additive noise
- High-dimensional regime $\ell \ll n$ with structure on $\mathbf{x}$

Setup

Compressive Sensing Introduction

- Given both $\mathbf{A}$ and $\mathbf{y}$
- Estimate $\hat{\mathbf{x}}$ that ideally closely approximates $\mathbf{x}$.
- Per-entry mean squared error (MSE) criterion for correctness:

$$\mathrm{MSE}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n} \|\mathbf{x} - \hat{\mathbf{x}}\|^2$$

Objective

In this paper, we focus on the design of $\mathbf{A} \in \mathbb{R}^{\ell \times n}$

- Each row of $\mathbf{A}$ independently drawn from the same zero-mean Gaussian distribution

- But different variances $(v_1, \ldots, v_n)$

  $\sum_{i=1}^{n} v_i = nP$ for some predefined "power level" P

- Assume we have access **to training data** $\mathbf{x}_1, \ldots, \mathbf{x}_m$

- **Our aim is to choose the values of $(v_1, \ldots, v_n)$ using the training data**

Method

Two main proposed algorithms

- **Variance-Proportional Sampling**

  Simple variance-proportional sampling (i.e., more energy at locations where the signal tends to vary more)

- **Iteratively-Learned Power Allocation**

  Iteratively up-weigh and down-weigh the variance values according to performance on training data

Variance-Proportional Sampling

Intuitively…

- If the signals always take near-identical values at certain entries, then it is wasteful to allocate a large amount of power to them

Variance-Proportional Sampling

Mathematically…

- Compute the empirical variance $\hat{\sigma}_i^2$ of each signal entry i (with respect to the training data)

- Choose the value such that

$$v_i = nP \cdot \frac{\hat{\sigma}_i^2}{\sum_{j=1}^{n} \hat{\sigma}_j^2}$$

Iteratively-Learned Power Allocation

Intuitively…

- Given the estimate $\hat{\mathbf{x}}$
- Up-weigh the v_i whose corresponding entries i were estimated the least accurately
- Down-weigh those that were estimated the most accurately

Iteratively-Learned Power Allocation

Mathematically…

- Proportion parameter $\alpha \in (0, \frac{1}{2})$
- Update weight $\{\lambda_t\}_{t=1}^{T}$ with $\lambda_t > 0$
- Number of iterations $T > 0$
- Mini-batch size $B > 0$

Iteratively-Learned Power Allocation

Mathematically…

- Initialize $(v_1, \ldots, v_n) = (P, \ldots, P)$
- For iterations $t = 1$ to T:
  1. Generate a random matrix $\mathbf{A}$ according to the current $(v_1, \ldots, v_n)$
  2. MSE of $(\hat{\mathbf{x}}, \mathbf{x})$ of each signal entry averaged over B training data
  3. Top αn MSE values: $v_i = v_i e^{-\lambda_t}$;

     Bottom αn MSE values: $v_i = v_i e^{\lambda_t}$
  4. Rescale $(v_1, \ldots, v_n)$ such that $\sum_{i=1}^{n} v_i = nP$

Experiment

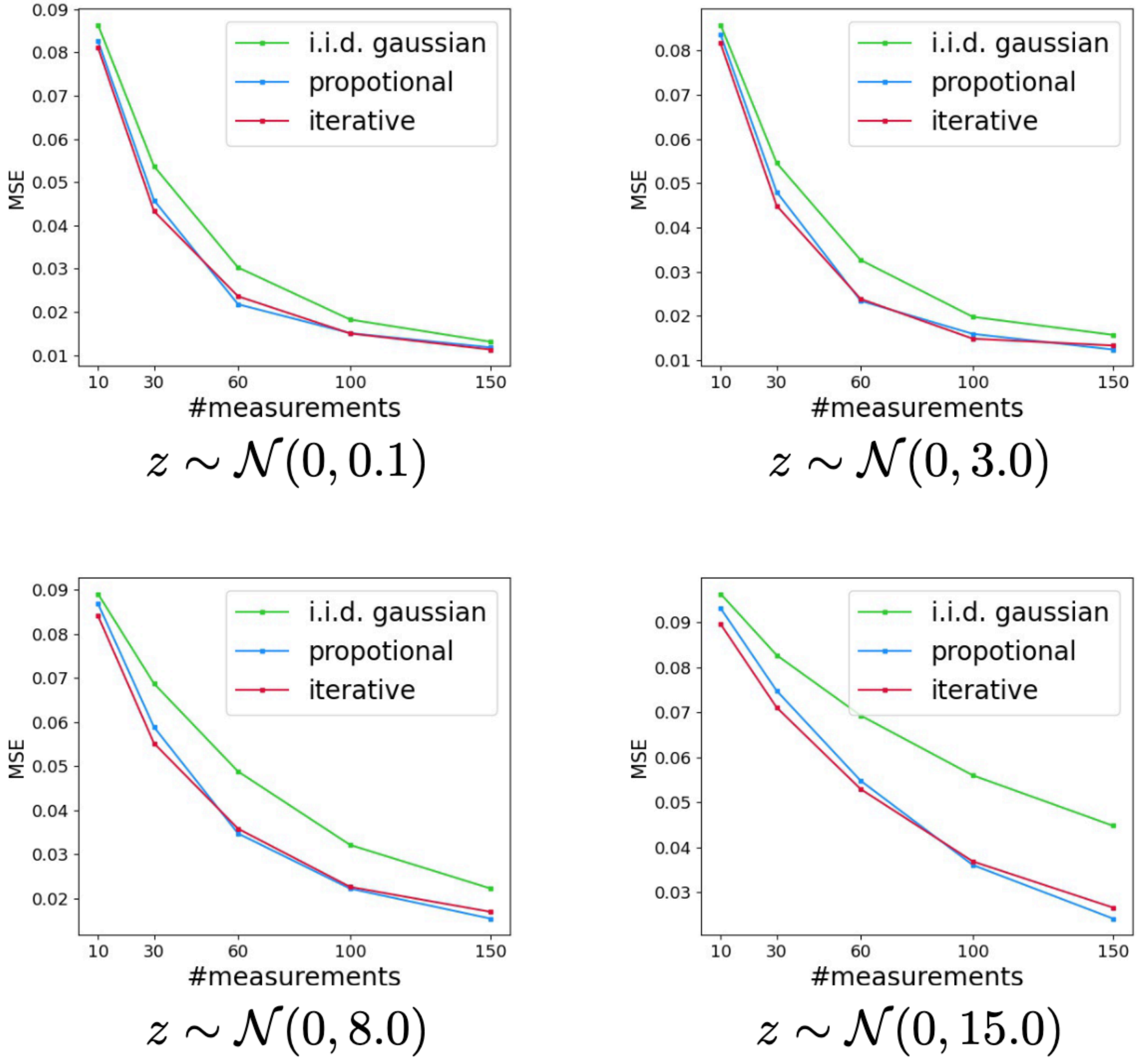
Baseline and Data

- Power level $P = 1$: Baseline $\mathbf{A}$ having i.i.d. standard Gaussian entries
- MNIST (28×28)
- Cropped CelebA-HQ $(128 \times 128 \times 3)$
- Synthetic one-dimensional non-uniform sparse signal (1000×1)

Result

MNIST performance figure

- 50000 training data for empirical variance $\hat{\sigma}^2$ **(Proportional)**
- Batch size $B = 20$ for each iteration **(Iterative)**
- 300 testing data to produce the average results **(Both)**
- MSE reduced visibly

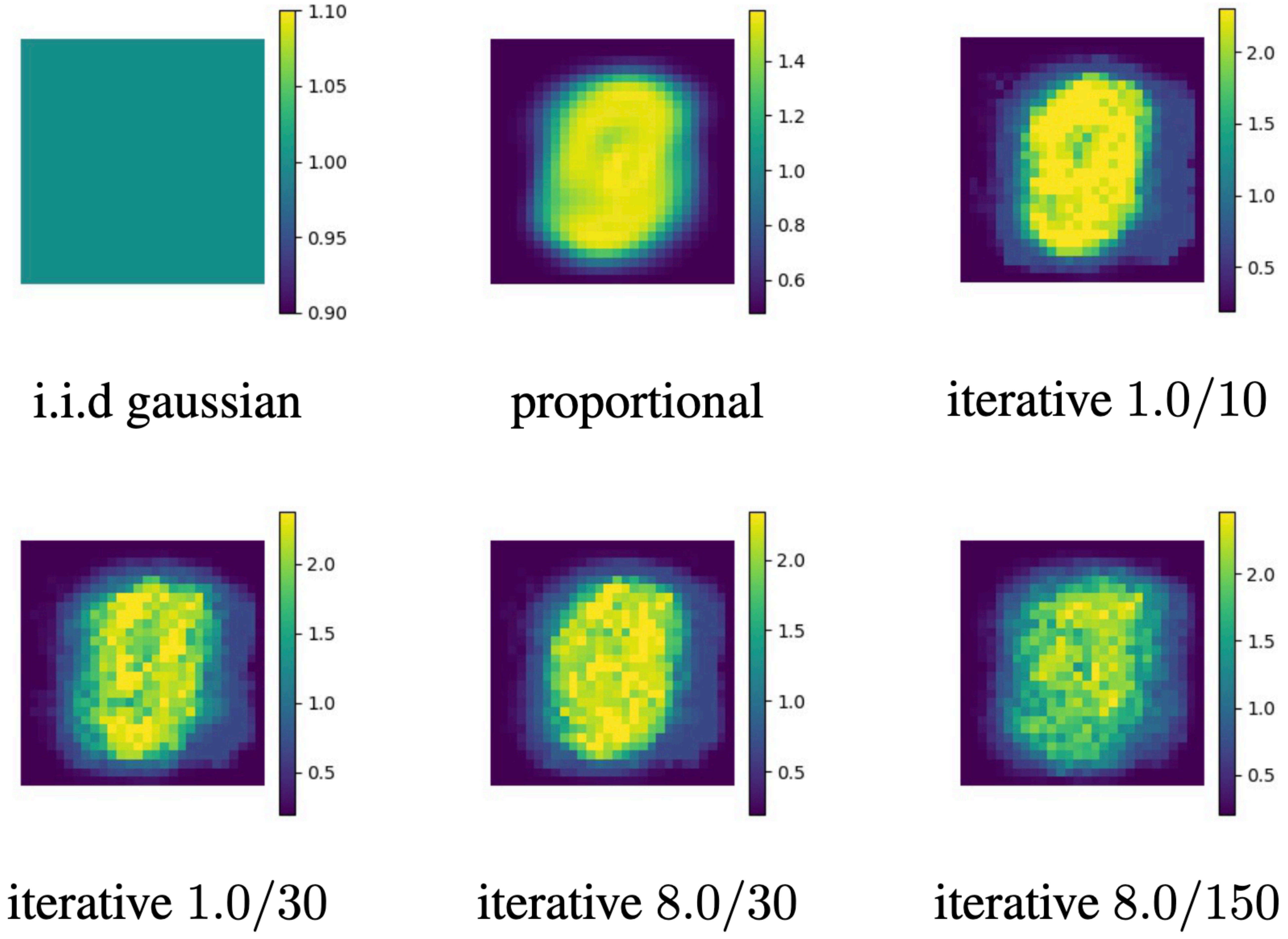


Recovery performance on MNIST data

Result

MNIST variance map

- Naturally place more energy at the locations where the MNIST number strokes lie
- As ℓ increases, the variance map become less uniform around the centre of the image

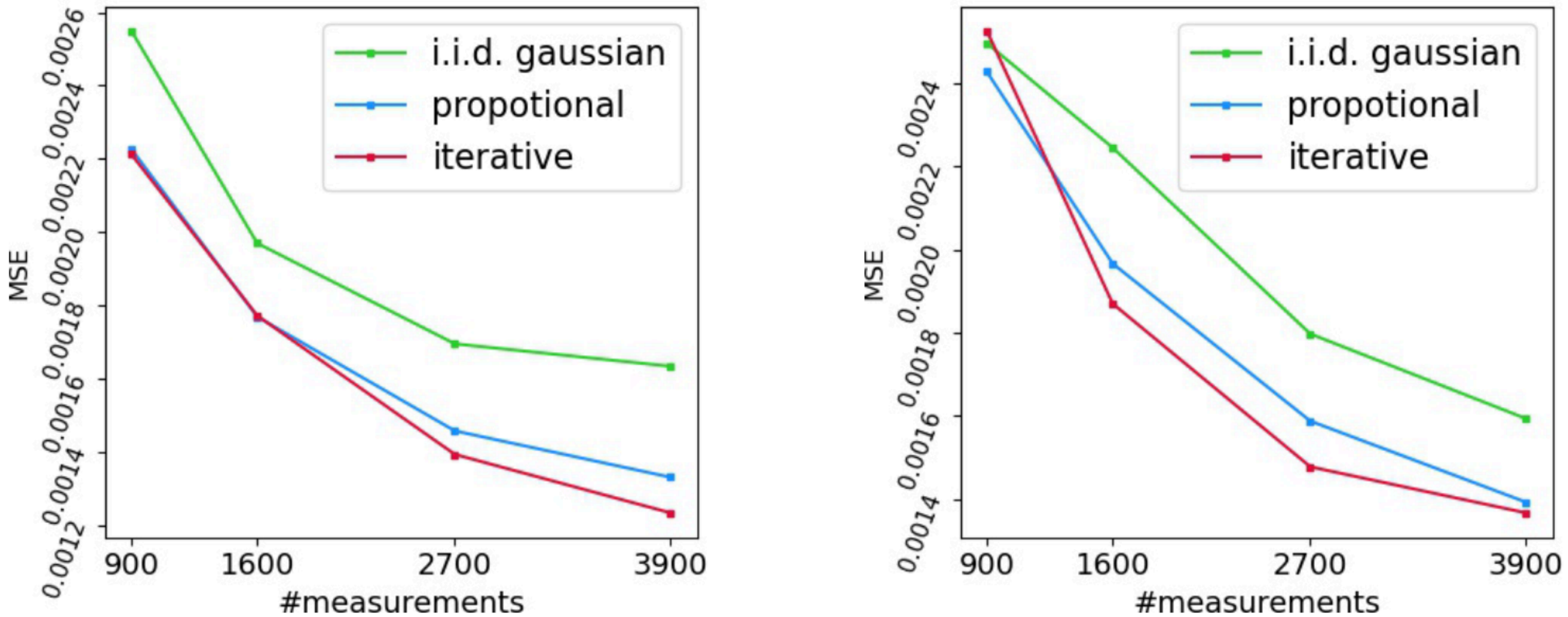


Variance map for MNIST data
(Numbers after iterative approach: the variance of z / number of measurement ℓ)

Result

CelebA-HQ performance figure

- 2000 training data to compute the empirical variance $\hat{\sigma}^2$ **(Proportional)**
- Mini-batch size $B = 1$ for each iteration **(Iterative)**
- 3 testing data to produce the average results **(Both)**
- MSE again lower then i.i.d. Gaussian



$z \sim \mathcal{N}(0, 8.0)$ $z \sim \mathcal{N}(0, 16.0)$

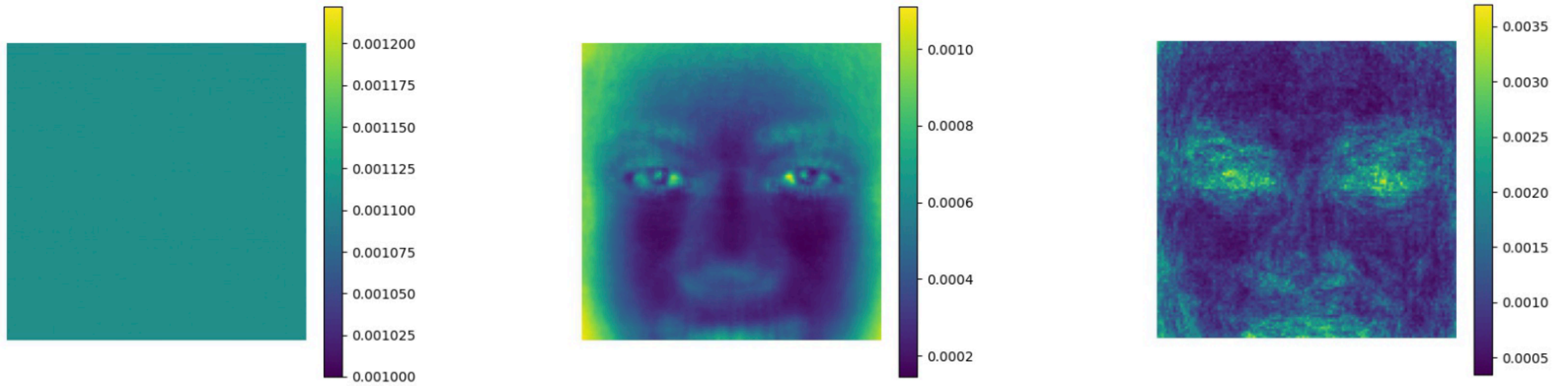
Recovery performance on cropped CelebA data

Result

CelebA-HQ variance map

- **Proportional** places too much energy around the corners

- **Iterative** instead focuses on the most “ambiguous” regions, e.g., the eyes, noses, and lips



i.i.d gaussian | proportional | iterative 16.0/900

Variance map generated for CelebA-HQ

Result

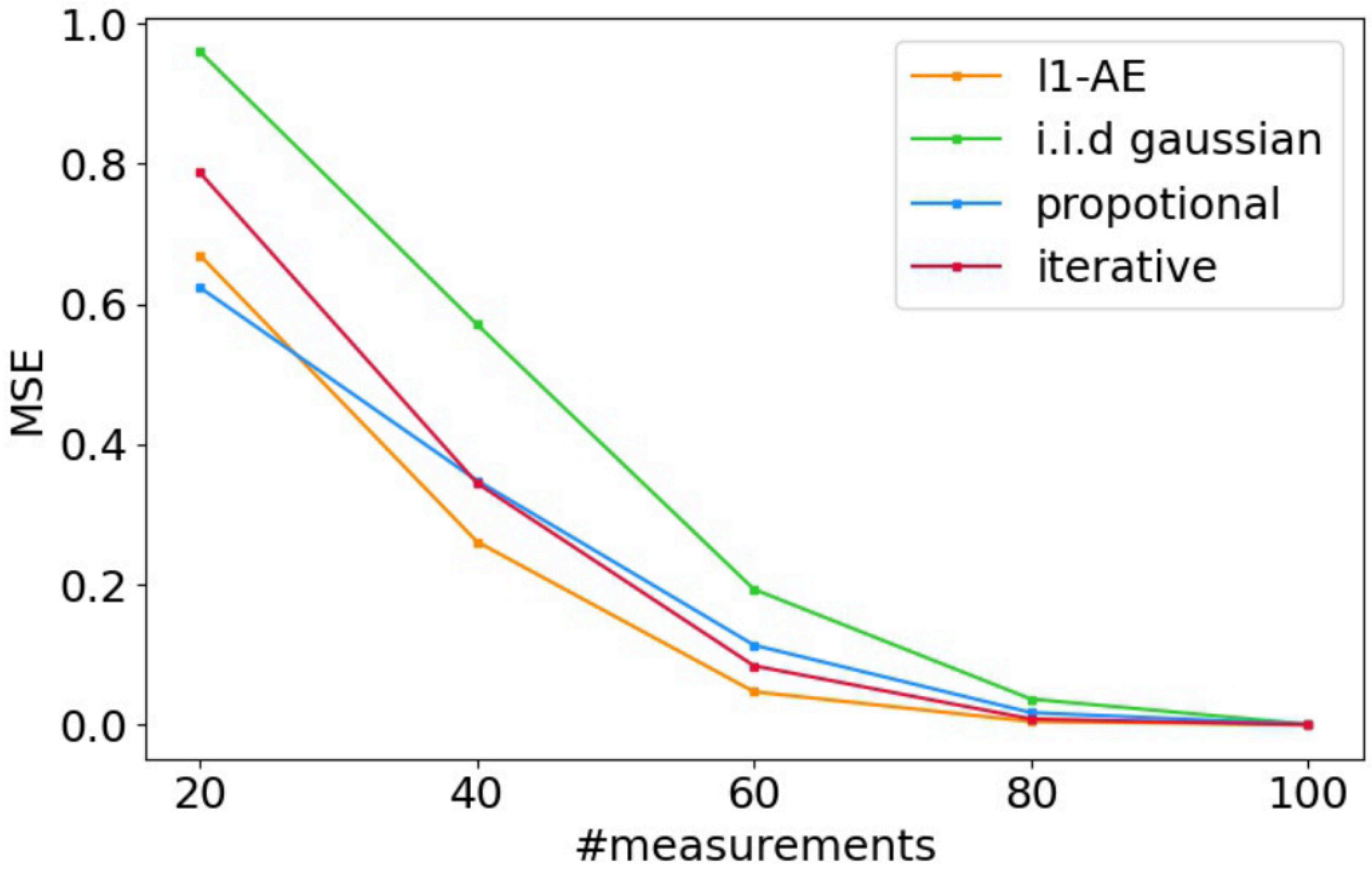
Synthetic data

- One-dimensional synthetic vector
- The i-th entry of the vector is non-zero with probability $\min\left\{1, \frac{c}{i}\right\}$ and uniform in $[0,1]$
- Compare against a deep learning based auto-encoder approach called $\ell_1 - AE$ proposed in *Shanshan Wu et al. (2019)*

Result

Synthetic performance figure

- 6000 training data to compute the empirical variance $\hat{\sigma}^2$ **(Proportional)**
- Batch size $B = 300$ for each iteration **(Iterative)**
- 2000 testing data to produce the average results **(Both)**
- We attain lower MSE and fall only slightly short of $\ell_1 - AE$

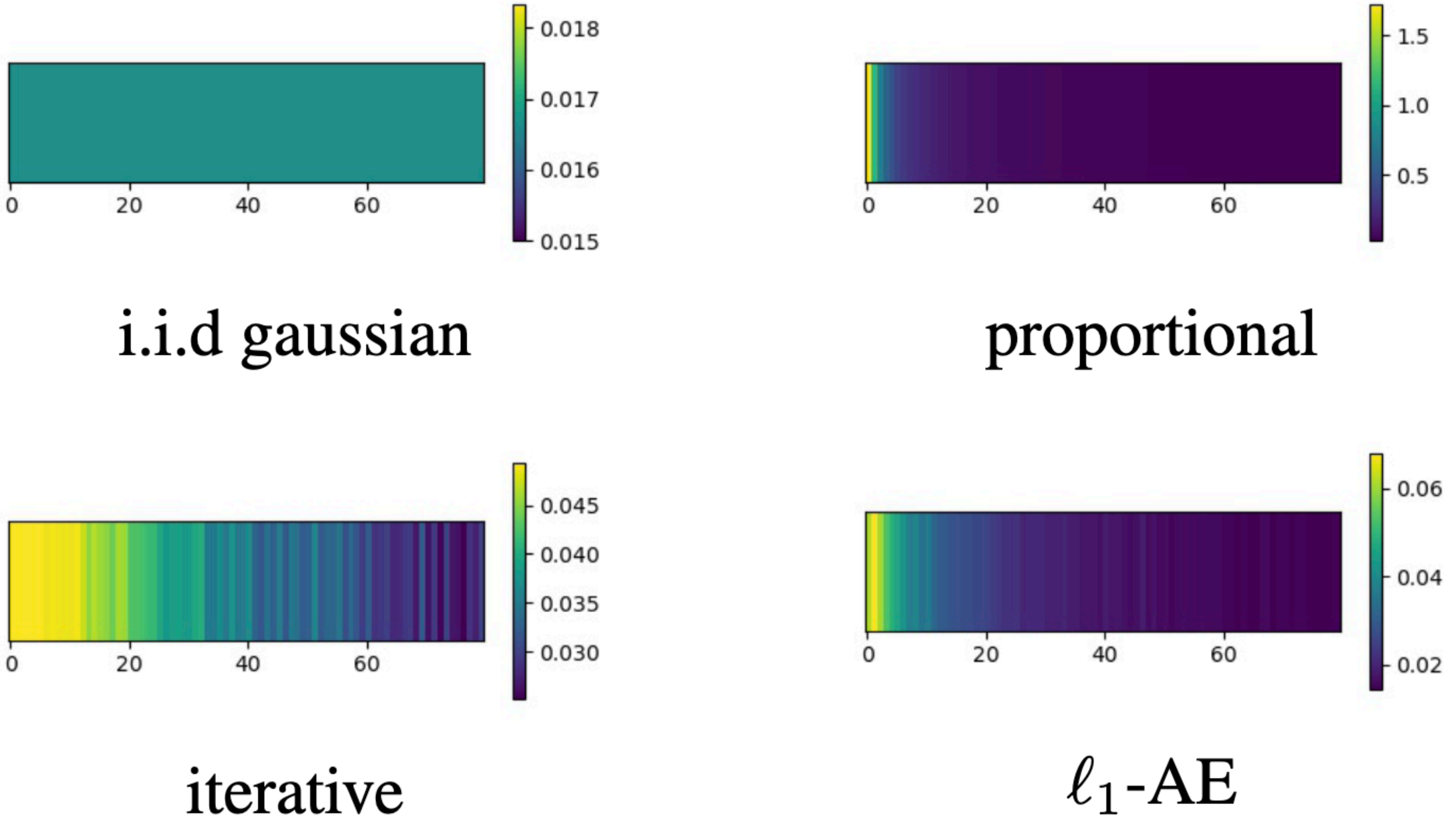


Recovery performance on synthetic data (noiseless)

Result

Synthetic variance map

- Similar pattern in proportional and $\ell_1 - AE$



First 80 entries of the variance map for synthetic data

Conclusion

- Two data-driven algorithms for learning power allocations in Gaussian compressive sensing measurement matrices
- Experimentally outperform standard i.i.d. Gaussian measurements under both sparse and generative priors
- Future work: Identify further areas for improvement and move towards application-driven experimental settings