Deep Memory Unrolled Networks For Solving Linear Imaging Problems

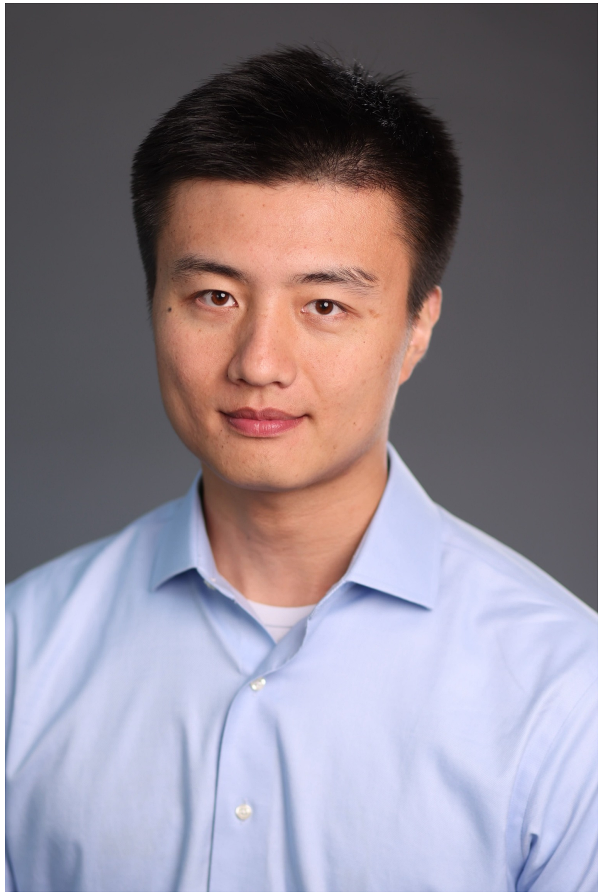
Eric Chen

Statistics & Data Science, Carnegie Mellon University

Sampling Theory and Applications, Vienna, 2025

Carnegie Mellon University

In Collaboration With



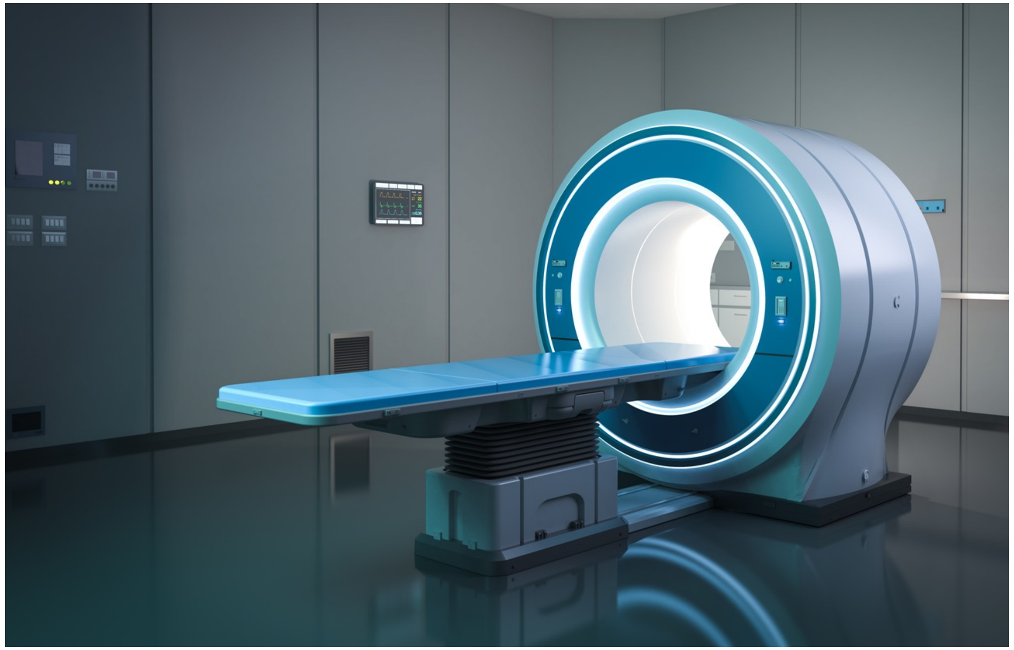
Xi Chen, Rutgers

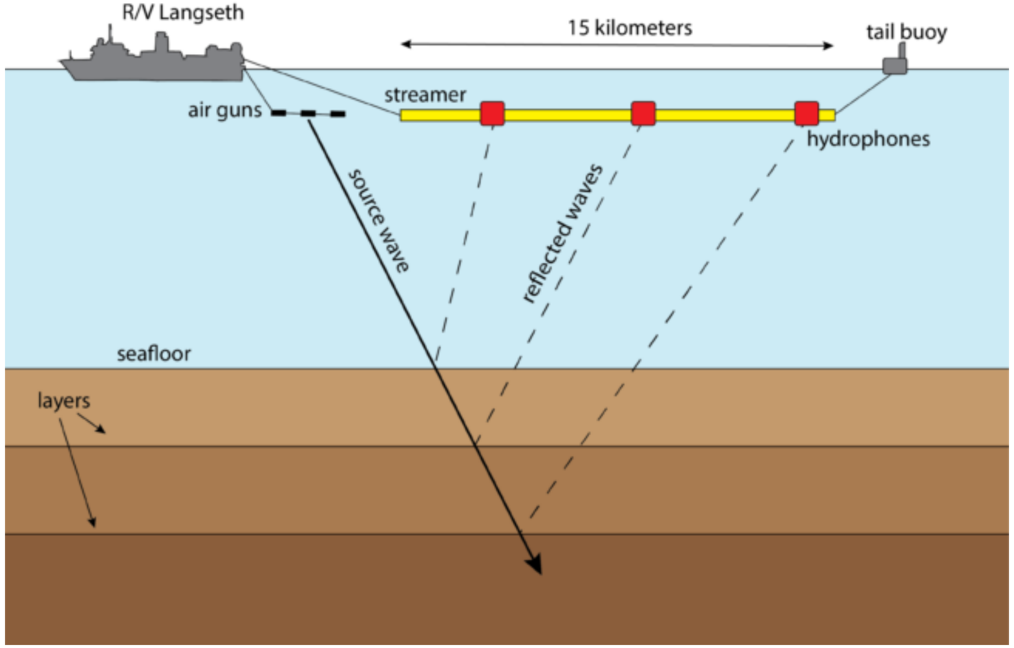
Shirin Jalali, Rutgers

Arian Maleki, Columbia

Linear Inverse Problems: Background

Modern Imaging Systems

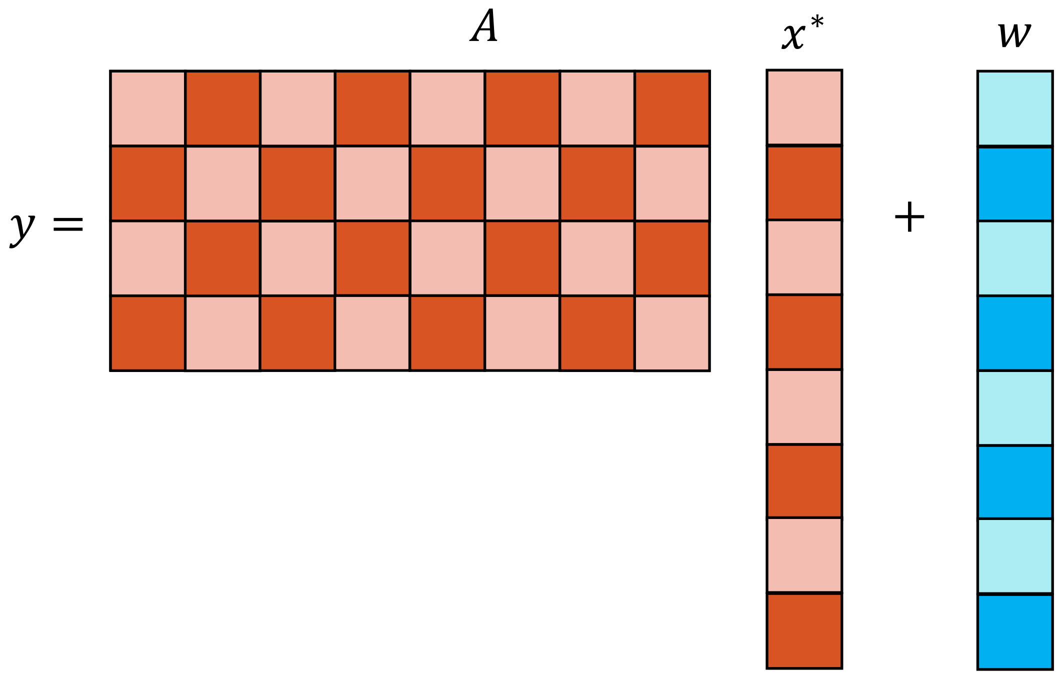




Application Domains: magnetic resonance imaging, seismic imaging, nuclear magnetic resonance...

The measurement is a linear function of the signal

Mathematical Formulation



- $A \in \mathbb{R}^{m \times n}$: the (underdetermined) measurement matrix
- w: measurement noise
- Goal: estimate x^* from observing y

Unrolled Networks: Motivation and Introduction

Projected Gradient Descent

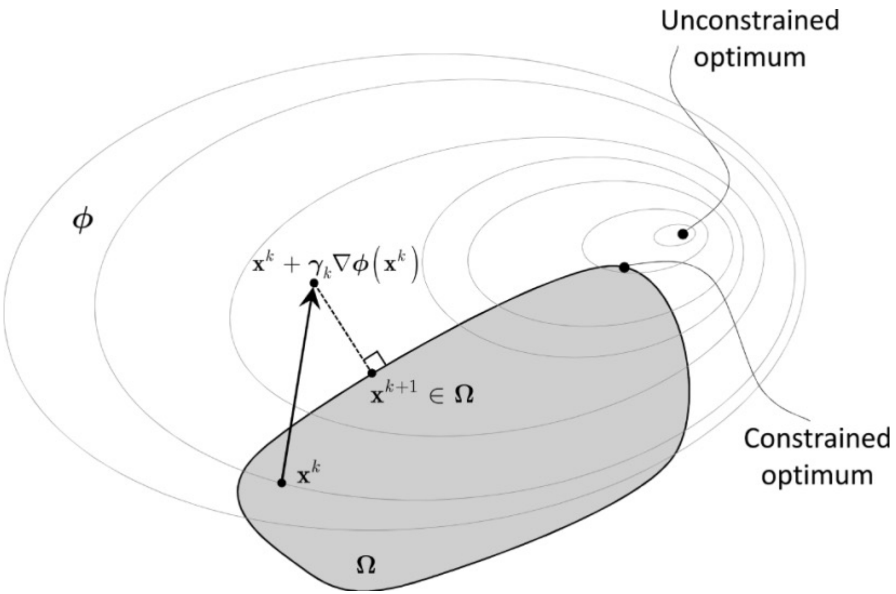
If all images of interest belong to a set $C \in \mathbb{R}^n$, then we may recover x^* by solving

$$\arg\min_{x \in C} ||y - Ax||_2^2$$

We can solve this using projected gradient descent

$$\tilde{x}^i = x^i + \mu_i A^T\left(y - Ax^i\right), \quad x^{i+1} = P_C(\tilde{x}^i)$$

- x^i: estimate of x^* in iteration i
- μ_i: step size in iteration i
- P_C: projection onto C, encoded with prior knowledge

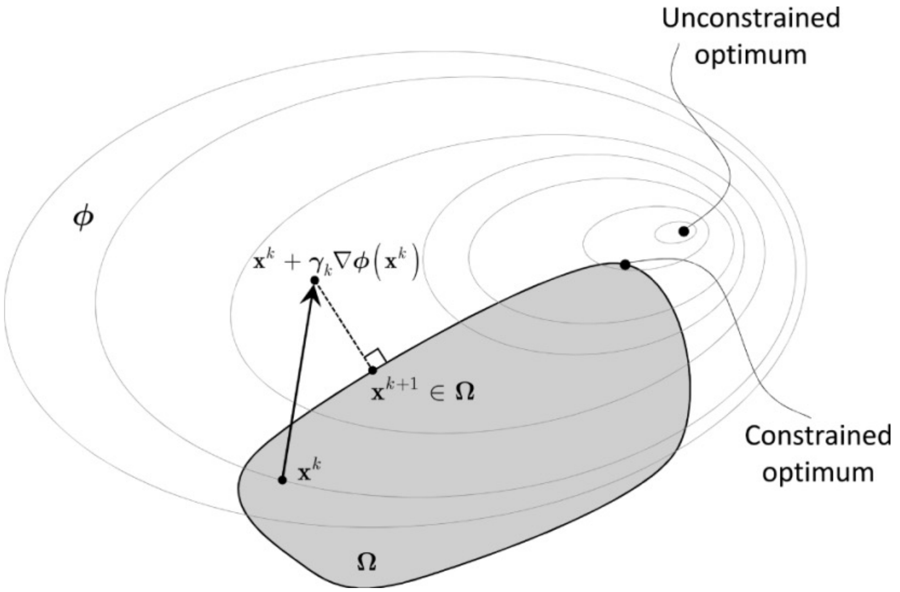


Challenges with Projected Gradient Descent

Projected Gradient Descent (PGD)

$$\tilde{x}^i = x^i + \mu_i A^T(y - Ax^i), \quad x^{i+1} = P_C(\tilde{x}^i)$$

- x^i: estimate of x^* in iteration i
- μ_i: step size in iteration i
- P_C: projection onto C, encoded with prior knowledge



Challenges:

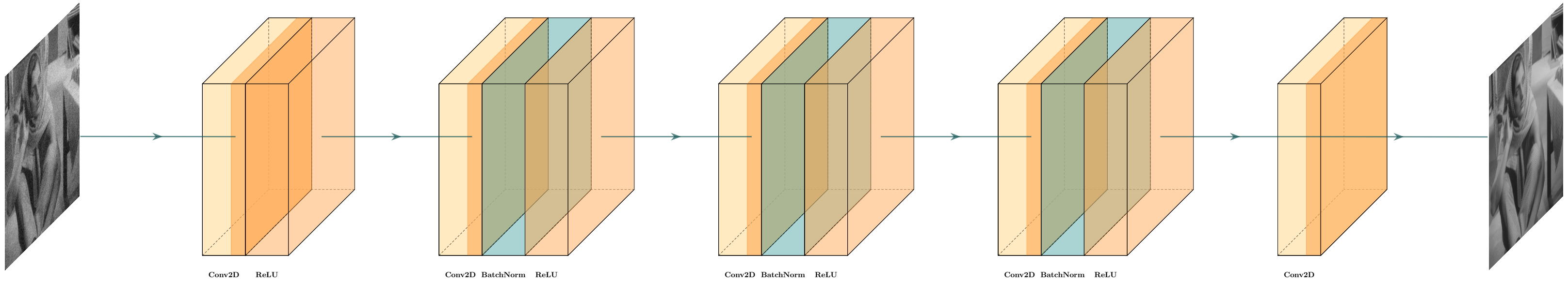
- C not known in practice
- Difficult to set step size μ_i

Unrolled Networks

We can stitch all the iterations of projected gradient descent together into one neural network:

$$NN(x) = P_n \circ GD_n \circ P_{n-1} \dots GD_2 \circ P_1 \circ GD_1(x)$$

- $GD_n(z) = z + \mu_n A^T(y - Az)$: gradient descent step
- P_n: projection step replaced by a mini-neural network



Advantages: projection and stepsize become learnable

Our Goals

Many design choices impact the reconstruction quality of an unrolled network:

- Loss function used for training
- Algorithm to unroll (gradient descent, Nesterov method, etc.)
- Projection operator (size and architecture of neural network)
- Number of unrolled steps

Using extensive simulations, we propose:

1. A stable loss function to train the network
2. Optimal unrolling strategy
3. Some other design choices

Deep Memory Unrolled Networks

What is a Good Training Loss?

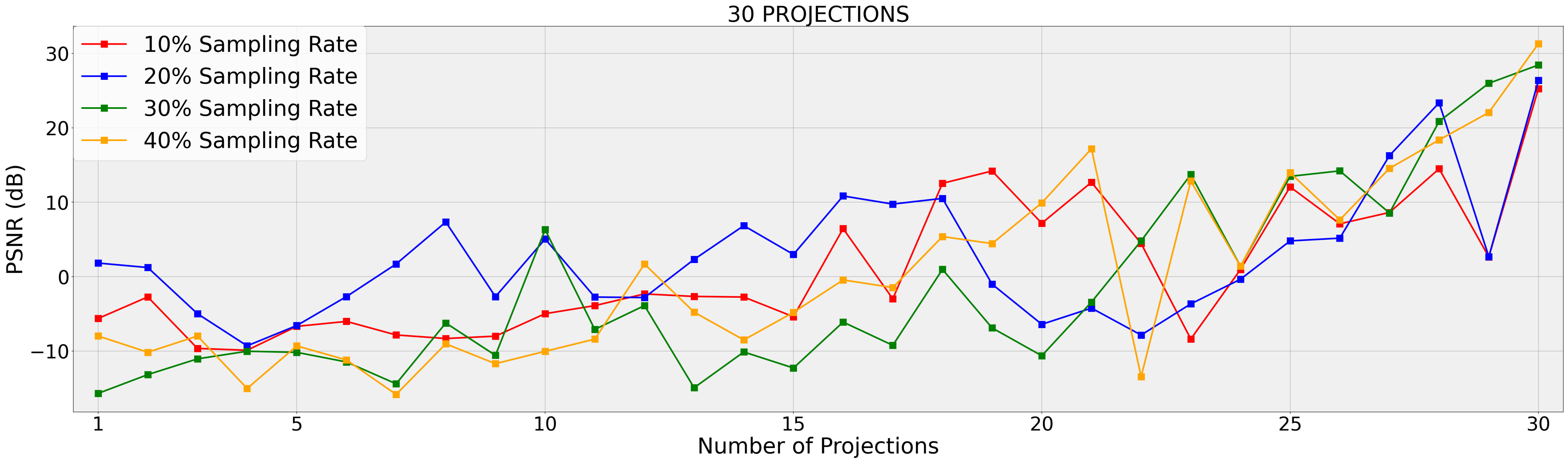
Previous works mainly considered the final projection:

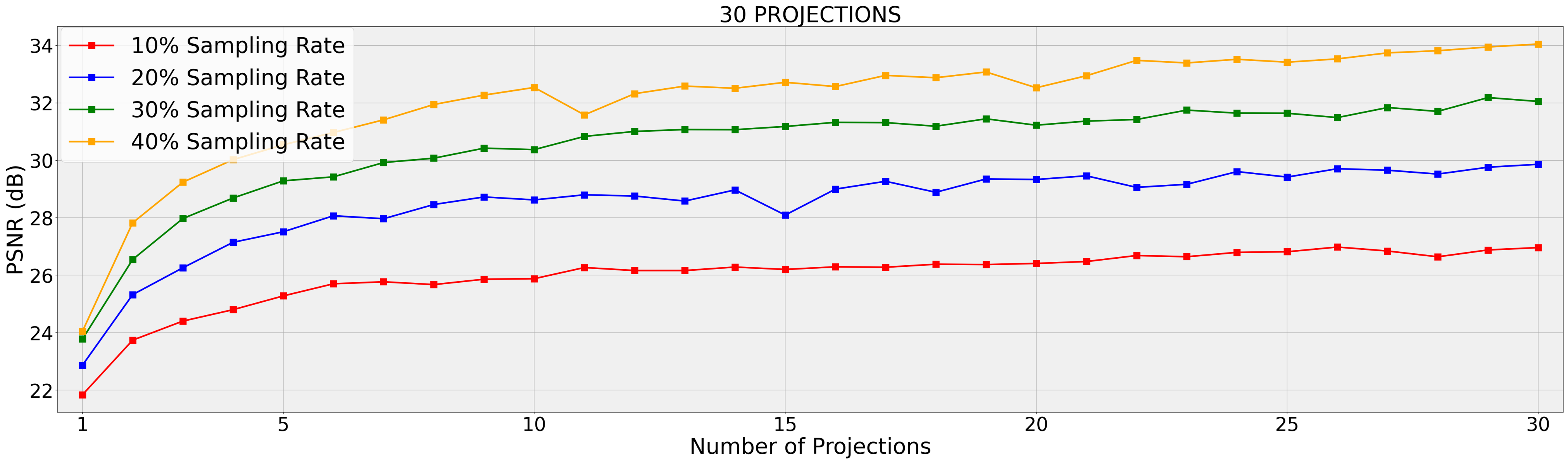
$$\mathcal{L}_{ll} = ||x^T - x||_2^2$$

Instead, we should encourage each projection to be good:

$$\mathcal{L}_I = \frac{1}{n} \sum_i ||x^i - x||_2^2$$

which gives us an intermediate loss function.


30 PROJECTIONS
10% Sampling Rate
20% Sampling Rate
30% Sampling Rate
40% Sampling Rate
PSNR (dB)
Number of Projections


30 PROJECTIONS
10% Sampling Rate
20% Sampling Rate
30% Sampling Rate
40% Sampling Rate
PSNR (dB)
Number of Projections

What is the Optimal Unrolling Algorithm?

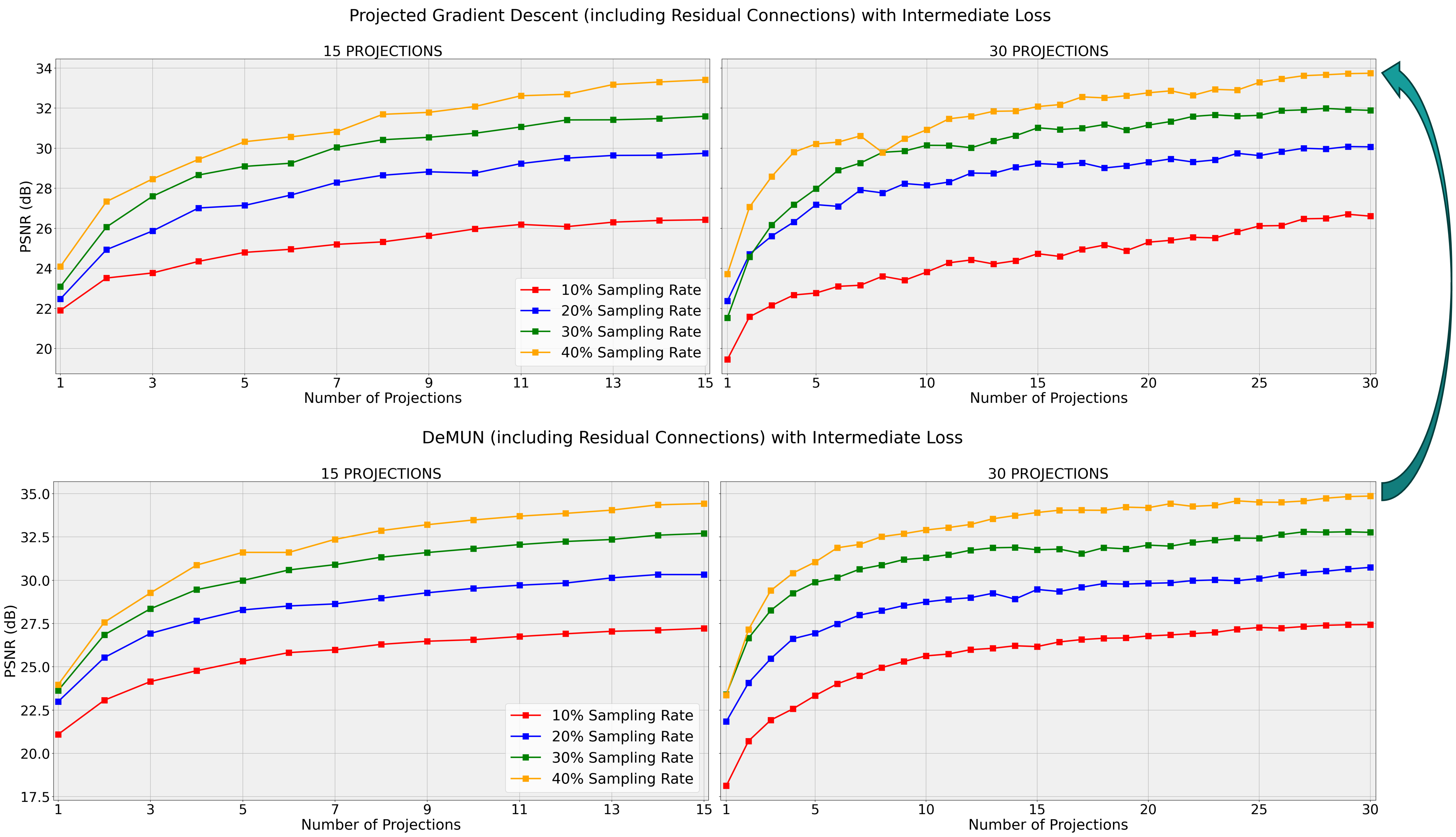
In each step of projected gradient descent, the update is given by

$$\tilde{x}^i = x^i + \mu_i A^T(y - Ax^i)$$

Instead, we can leverage the history of gradients, leading to the following update step

$$\tilde{x}^i = x^i + \sum_{j=0}^{i} \mu_j^i A^T(y - Ax^j)$$

which encapsulates other algorithms as special cases (PGD, AMP, Nesterov Method).


Projected Gradient Descent (including Residual Connections) with Intermediate Loss
15 PROJECTIONS
30 PROJECTIONS
PSNR (dB)
Number of Projections
10% Sampling Rate
20% Sampling Rate
30% Sampling Rate
40% Sampling Rate
DeMUN (including Residual Connections) with Intermediate Loss
15 PROJECTIONS
30 PROJECTIONS
PSNR (dB)
Number of Projections
10% Sampling Rate
20% Sampling Rate
30% Sampling Rate
40% Sampling Rate

THANK YOU!