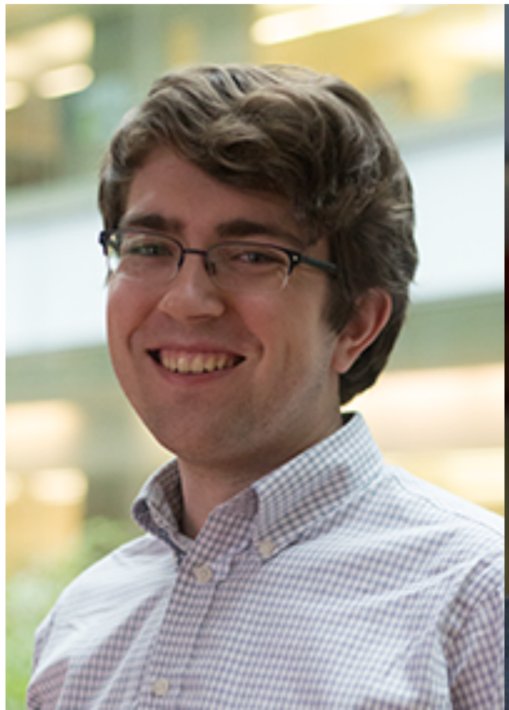
Model Adaptation for Inverse Problems in Imaging

Rebecca Willett, University of Chicago

Davis Gilton, UW-Madison

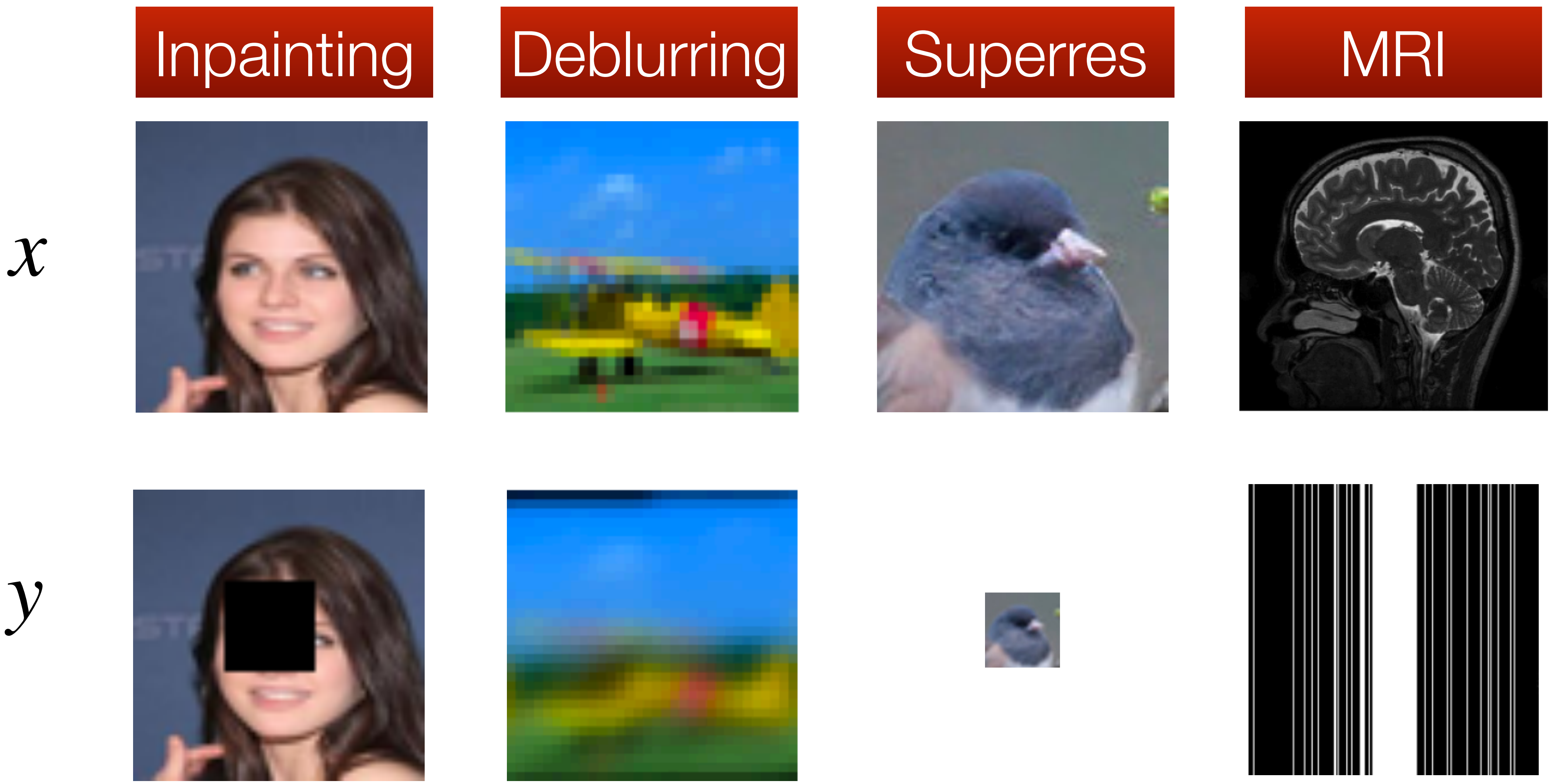
Greg Ongie, Marquette



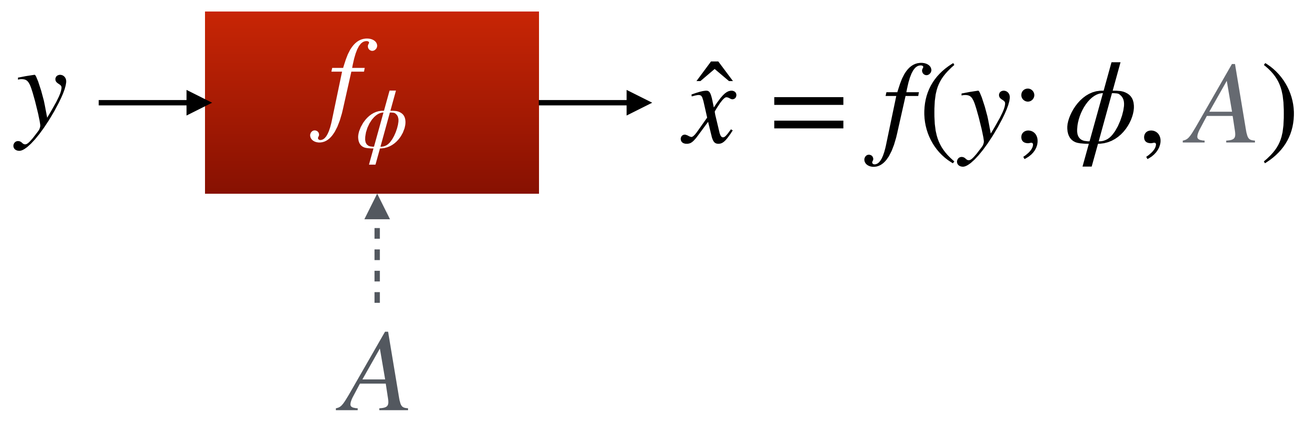
Inverse problems in imaging

Observe: $y = Ax + \varepsilon$

Goal: Recover x from y

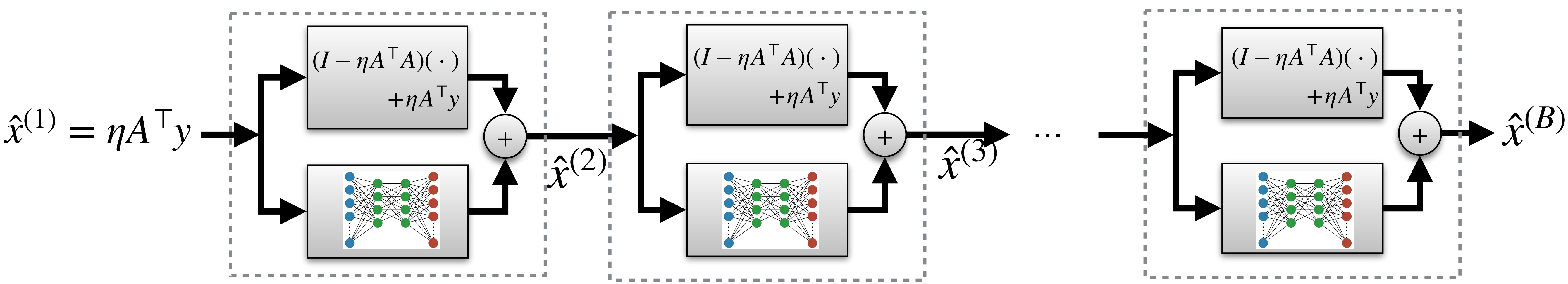


Learned solutions to inverse problems



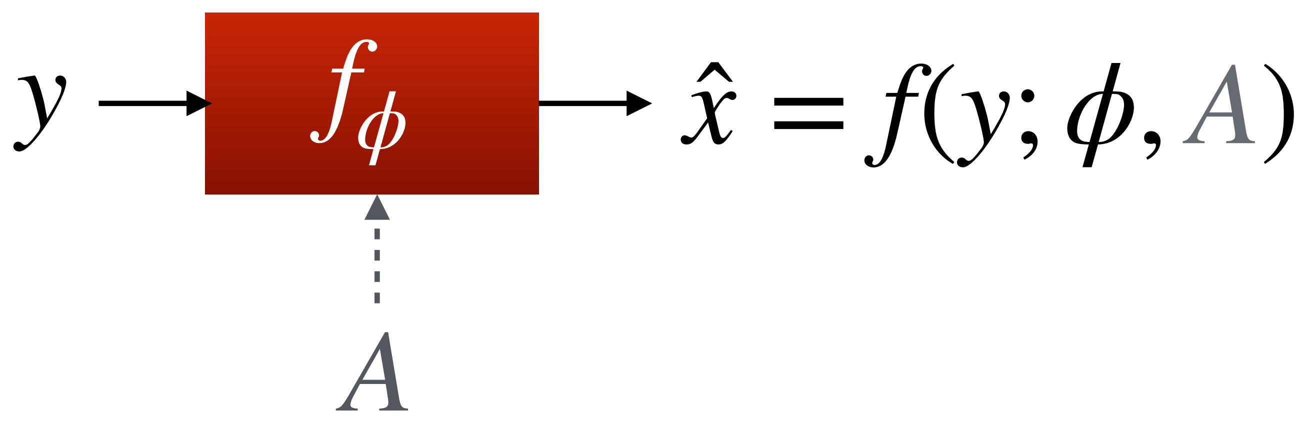
Neural network with parameters ϕ (and optionally A) inputs y and outputs an estimate $\hat{x}$.

Example: unrolled gradient decent for objective $\frac{1}{2}\|y - Ax\|_2^2 + r(x)$



Arridge, Maass, Öktem, Schönlieb, 2019
Ongie, Jalal, Metzler, Baraniuk, Dimakis, Willett, 2020

Learned solutions to inverse problems



Neural network with parameters ϕ (and optionally A) inputs y and outputs an estimate $\hat{x}$.

What if we train for A_0, but then at deployment have data corresponding to A_1 **(model drift)**?

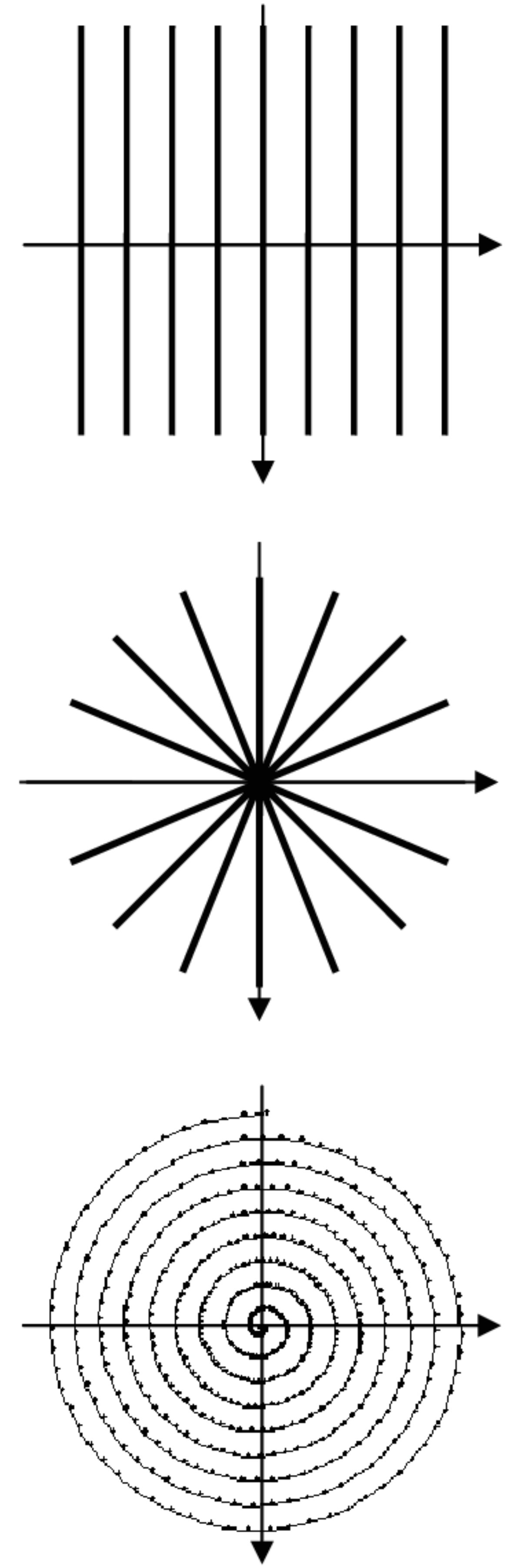
Arridge, Maass, Öktem, Schönlieb, 2019
Ongie, Jalal, Metzler, Baraniuk, Dimakis, Willett, 2020

Motivating example: MRI

- Substantial variation in the forward model
    - Cartesian vs. non-Cartesian k-space sampling trajectories,
    - different undersampling factors,
    - different number of coils and coil sensitivity maps,
    - magnetic field inhomogeneity maps…

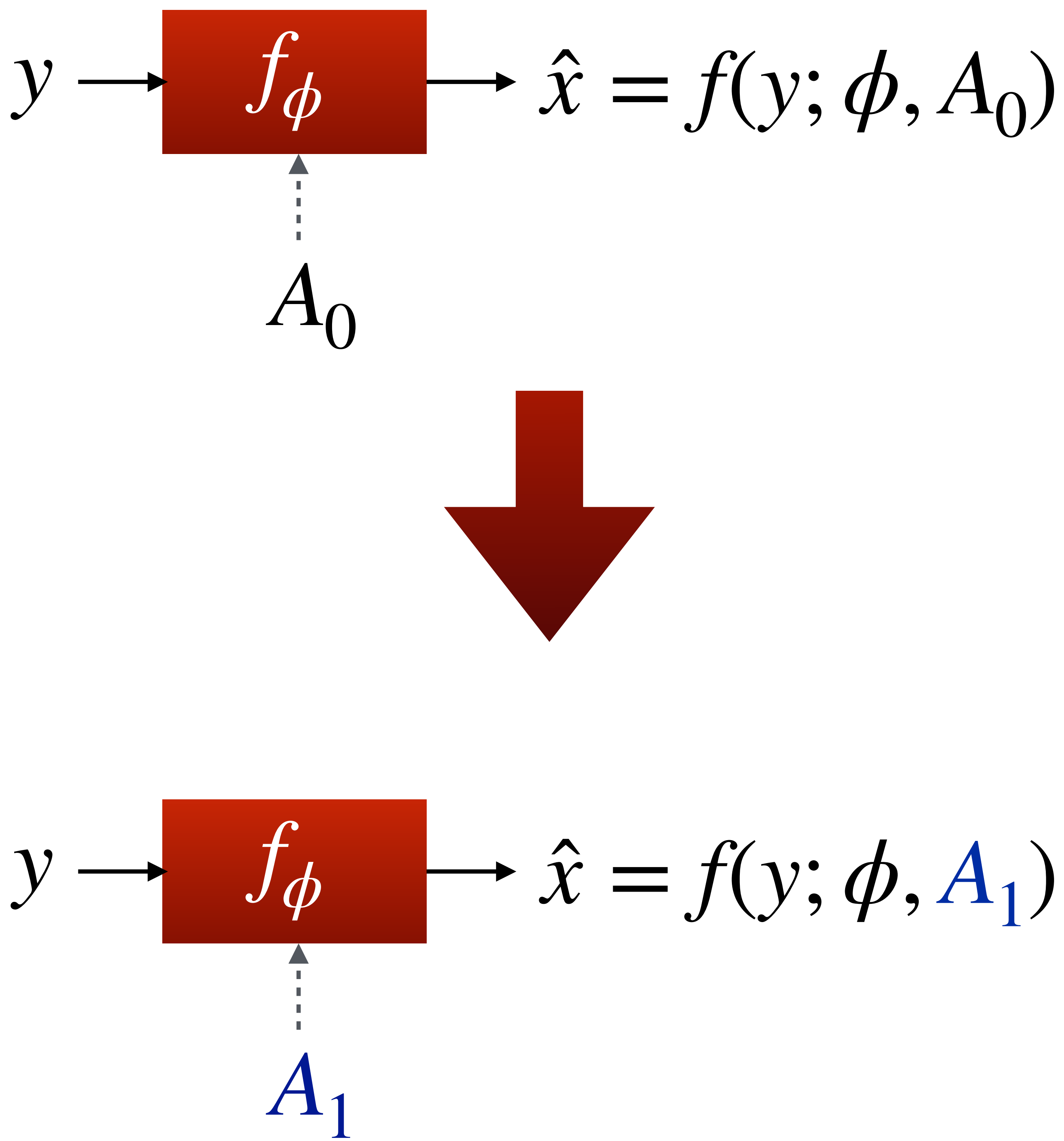
- Network trained for one of these forward models may perform poorly even a slightly different setting (e.g., from four-fold to three-fold undersampling of k-space)

- Training a new network from scratch may not always be feasible after deployment due to a lack of access to ground truth images (e.g. privacy concerns)



Antun, Renna, Poon, Adcock, and Hansen 2020

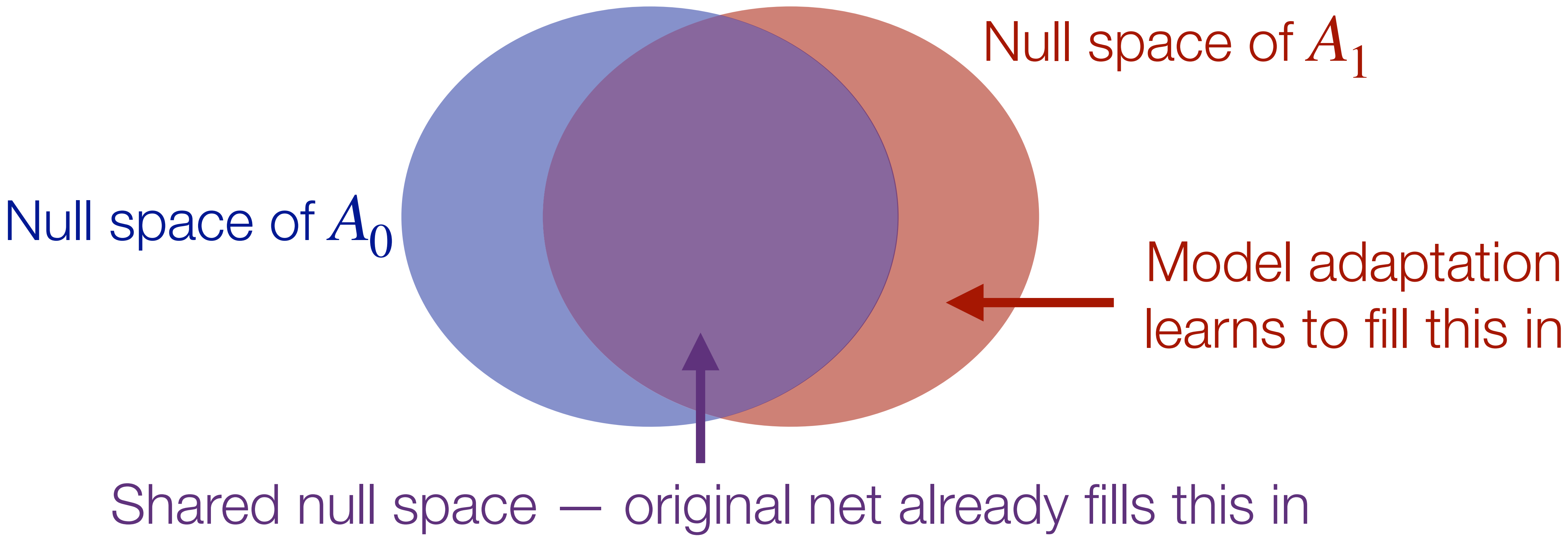
Naïve reconstruction with known model drift



A null space perspective

- A learned reconstruction network trained with data corresponding to A_0 essentially learns how to fill in images in the null space of A_0

- If we then get data from A_1 with a different null space, then we haven't learned how to fill in its null space



Our approach:

train a reconstruction network for a known forward model

then adapt to new forward model

without access to ground truth images,

and without knowing the exact parameters of the new forward model

Basic setup

- At train time, we have collection of training data $\{x_i^{(0)}, y_i^{(0)}\}$ for $i = 1,\ldots,n_0$, all corresponding to same (perhaps unknown) A_0:

$$y_i^{(0)} = A_0 x_i^{(0)} + \varepsilon_i^{(0)}$$

- After deployment, we have a collection of ***calibration data*** $\{y_i^{(1)}\}$ for $i = 1,\ldots,n_1$, all corresponding to same (perhaps unknown) A_1:

$$y_i^{(1)} = A_1 x_i^{(1)} + \varepsilon_i^{(1)}$$

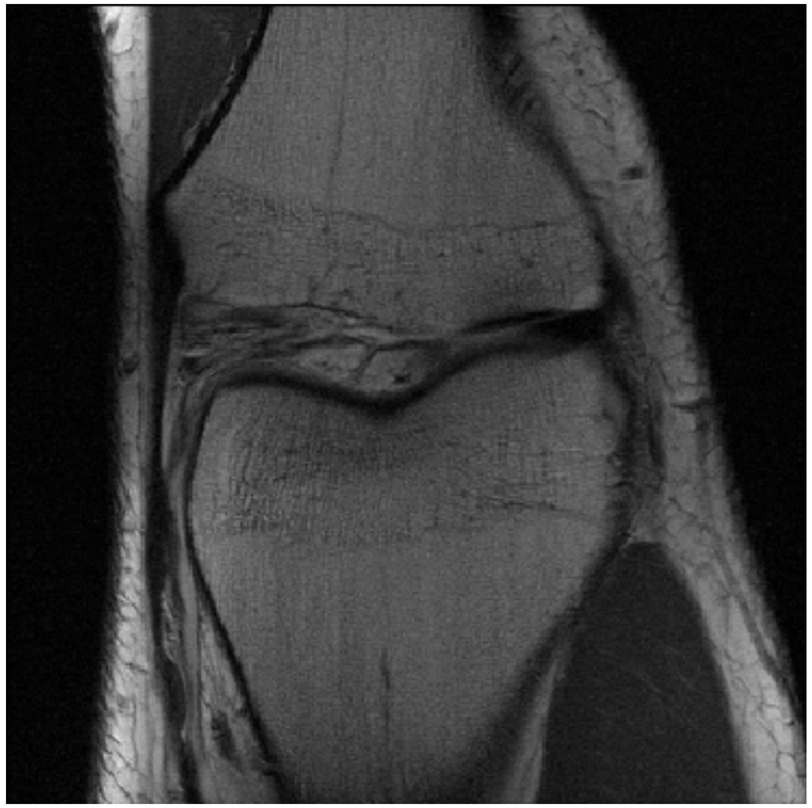
Model vs. distribution drift and model vs. domain adaptation

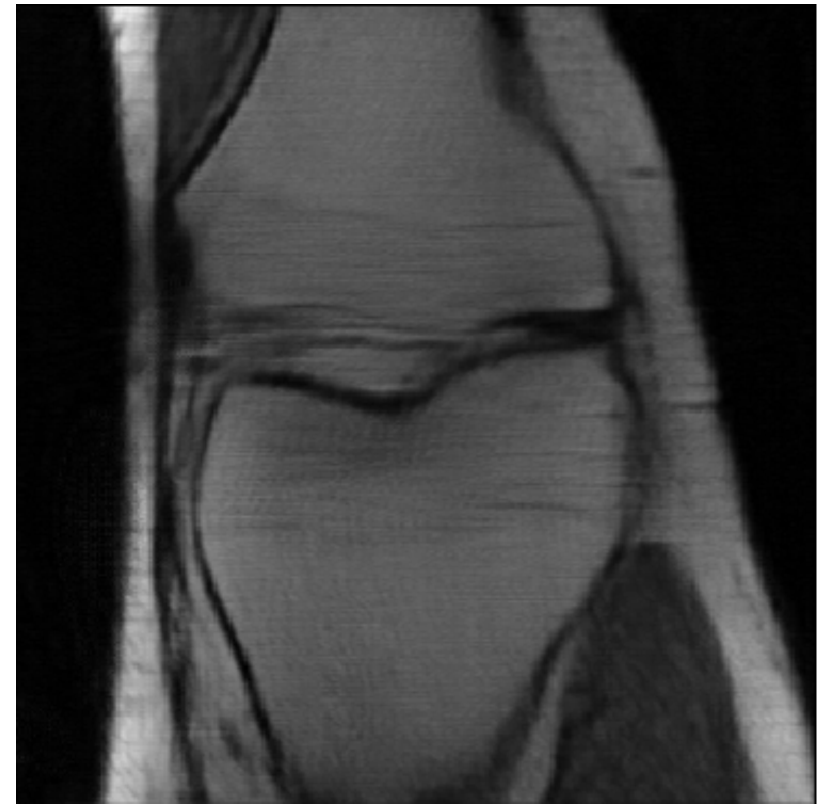
- **Distribution drift:** $p(X, Y)$ changes in unknown way between train and deployment
- **Model drift:** $p(Y|X)$ changes in known or partially known way between train and deployment (know have change in linear relationship)
- **Domain adaptation:** First train with many samples $(x_i^{(0)}, y_i^{(0)}) \sim p_0$, then adapt using few samples from $(x_i^{(1)}, y_i^{(1)}) \sim p_1$
- **Model adaptation:** First train with many samples $(x_i^{(0)}, y_i^{(0)}) \sim p_0$ and, then adapt using few samples from $(?, y_i^{(1)}) \sim p_1$

Han, Yoo, Kim, Shin, Sung, and Ye, 2018

The Big Picture

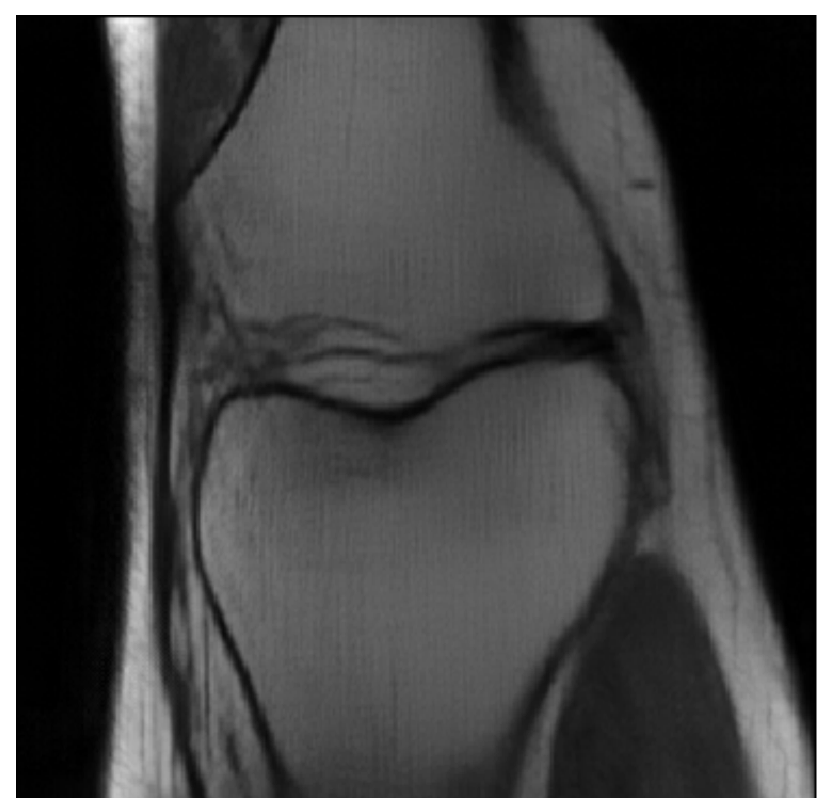
Original Image





Train for $y = A_0 x + \varepsilon$; test on $y = A_0 x + \varepsilon$

Train for $y = A_0 x + \varepsilon$; test on $y = A_1 x + \varepsilon$ no model adaptation



Train for $y = A_0 x + \varepsilon$; test on $y = A_1 x + \varepsilon$ **with** model adaptation

The Big Picture

Original Image

Train for $y = A_0 x + \varepsilon$; test on $y = A_0 x + \varepsilon$

PSNR = 31

Train for $y = A_0 x + \varepsilon$; test on $y = A_1 x + \varepsilon$ no model adaptation.

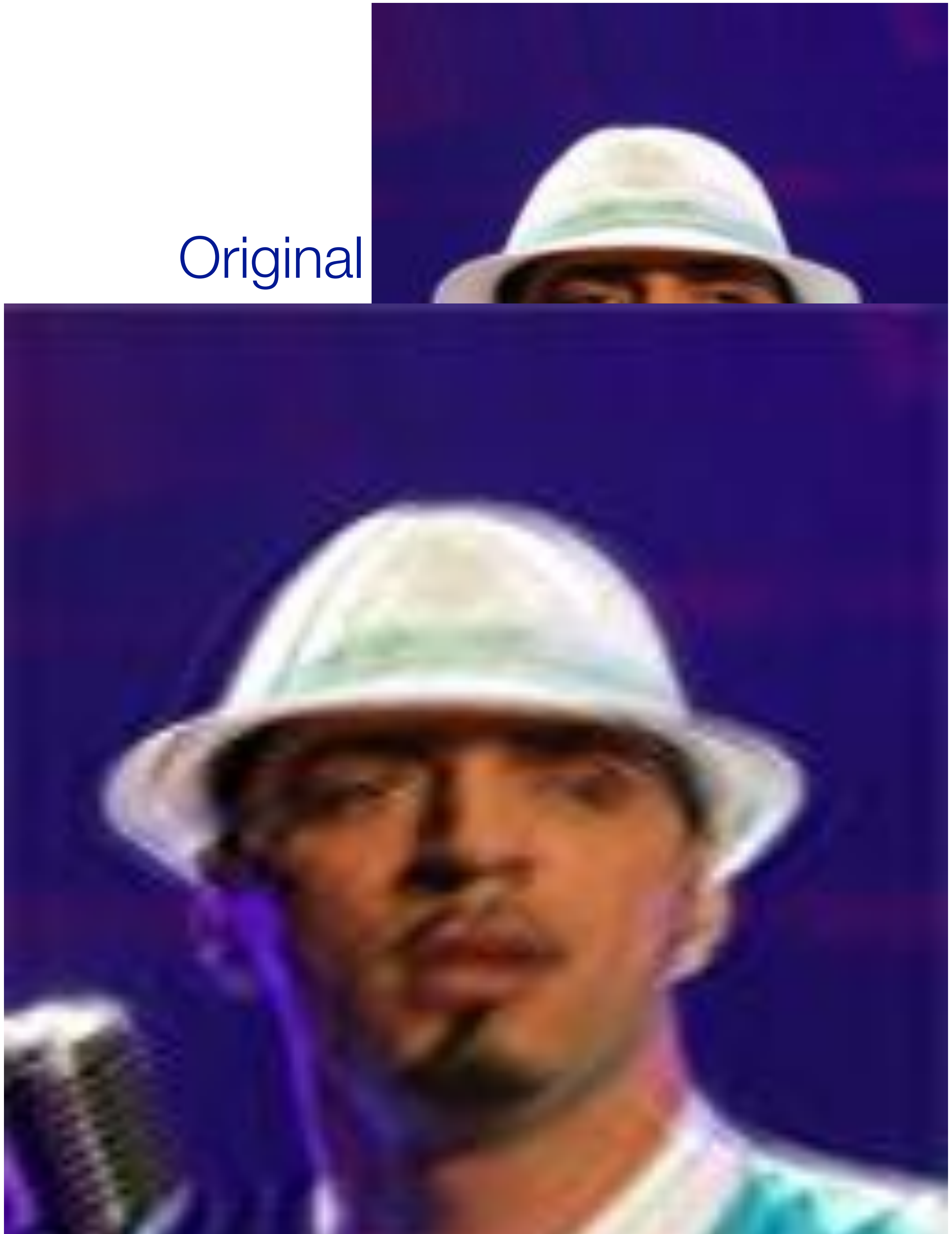
PSNR = 21

Train for $y = A_0 x + \varepsilon$; test on $y = A_1 x + \varepsilon$ **with** model adaptation.

PSNR = 30

The Big Picture

Original



Train for

$y = A_0 x + \varepsilon$;

test on

$y = A_0 x + \varepsilon$

PSNR = 31

Train for

$y = A_0 x + \varepsilon$;

test on

$y = A_1 x + \varepsilon$

with model adaptation.

PSNR = 30

The Big Picture

Original

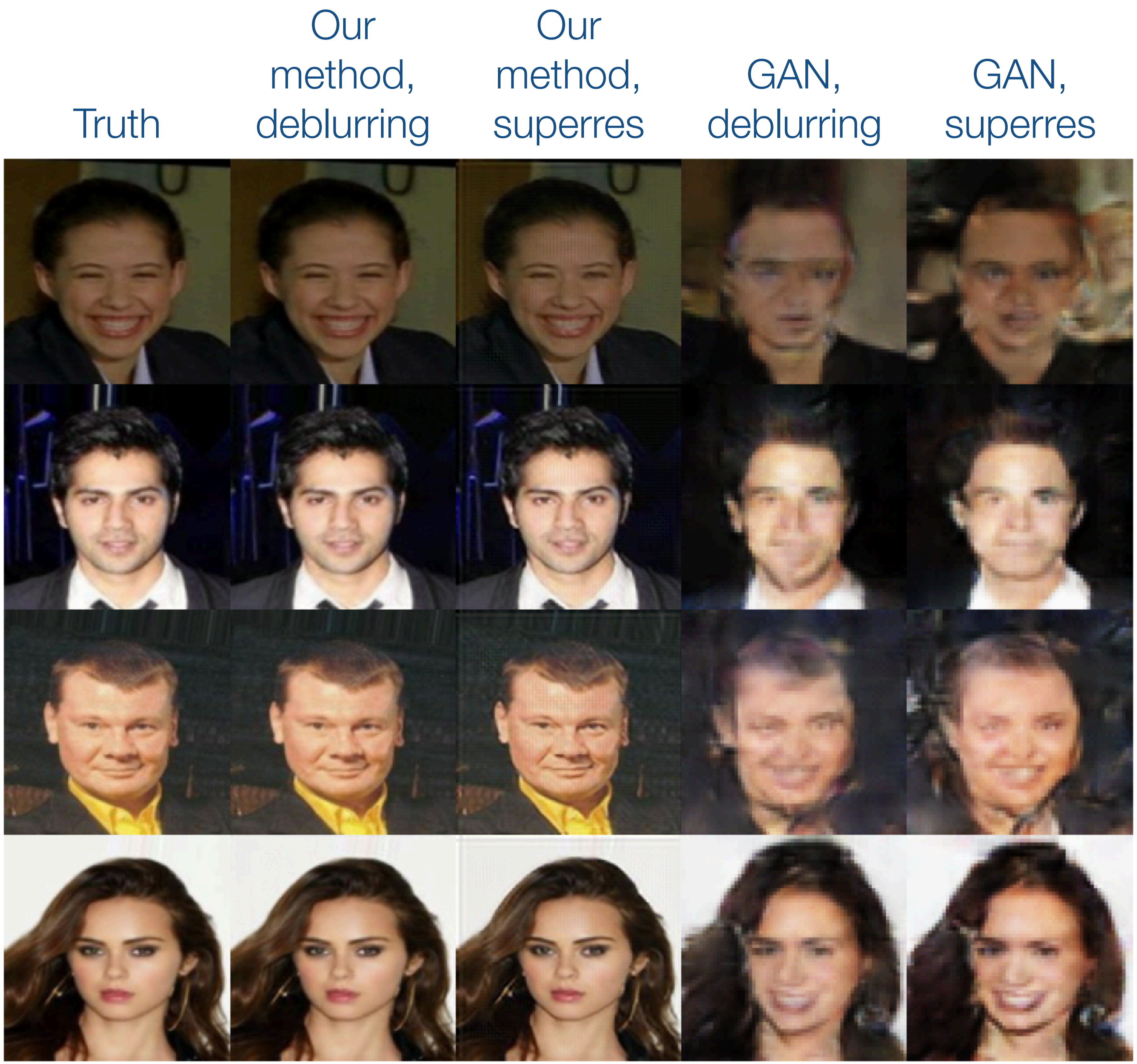
Train for
$y = A_0 x + \varepsilon$;
test on

Why not use generative models?

- Learn a generative model for images x

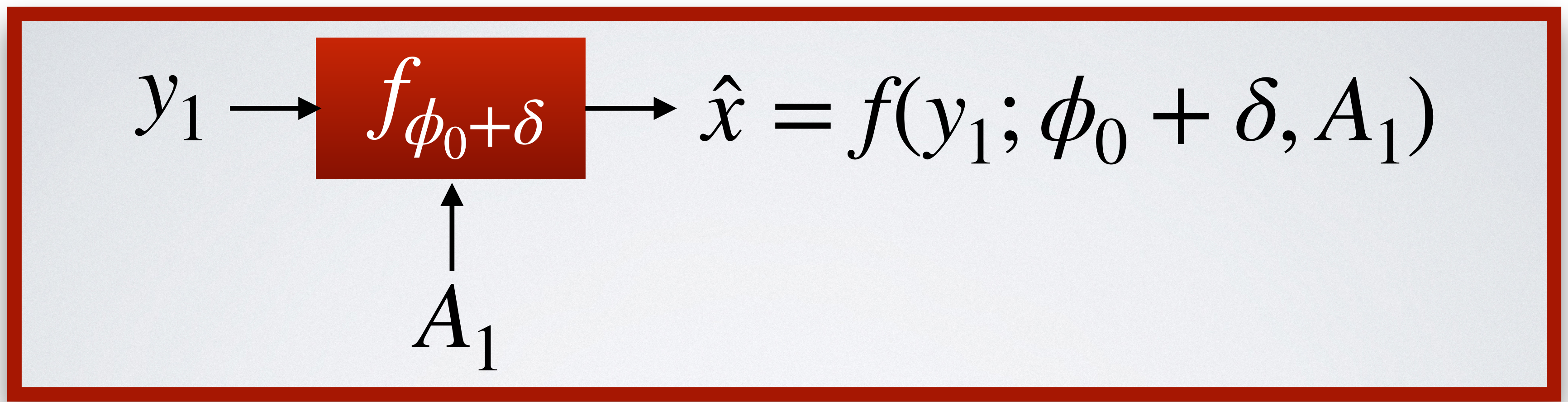
- Given A_1, generate training samples (x_i, y_i) by generating x_i from model and setting $y_i = A_1 x_i$, then train a reconstruction network

- Generative models can't learn whole distribution



Anirudh, Thiagarajan, Kailkhura, Bremer 2018

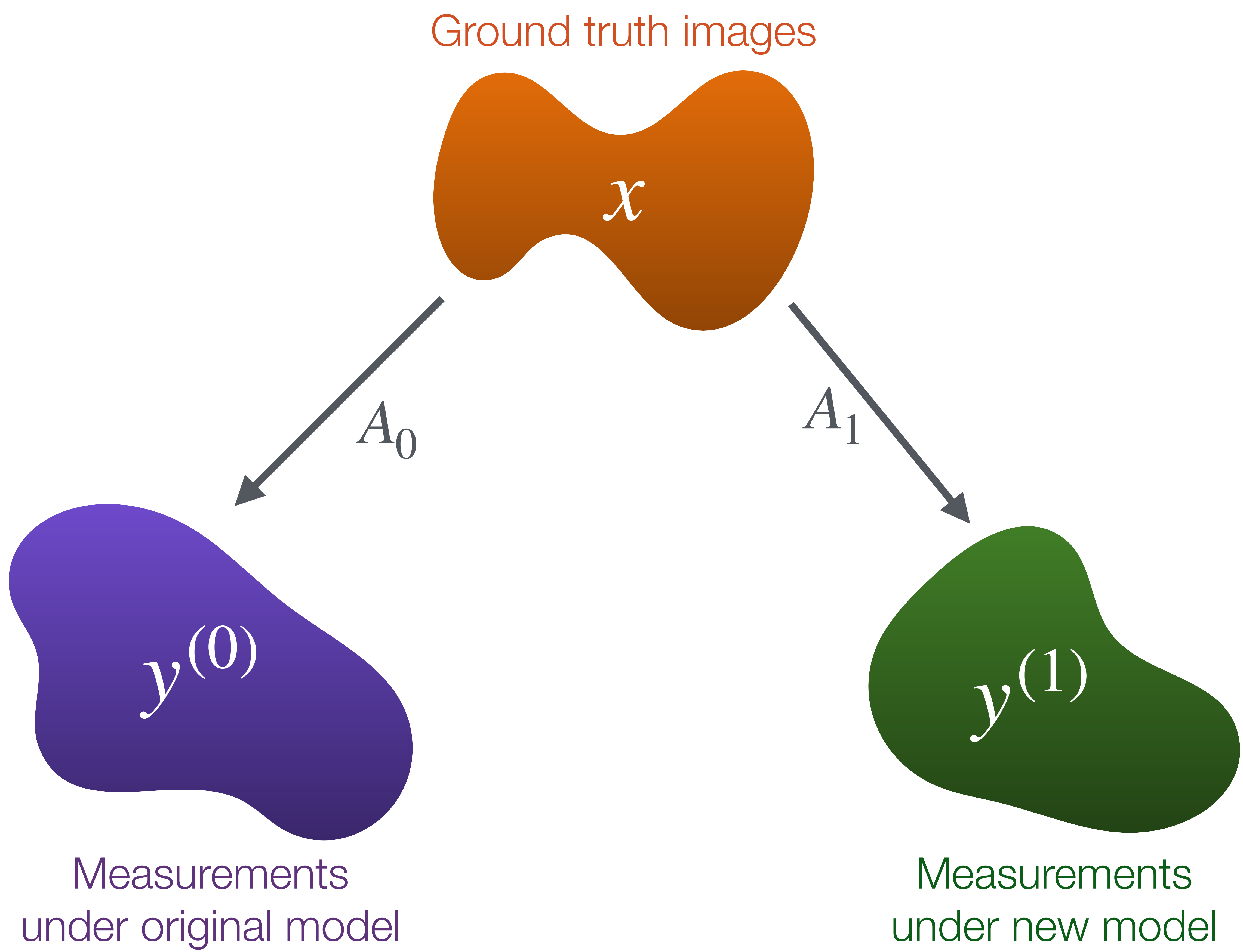
Approach 1: Parameterize and Perturb (P&P)

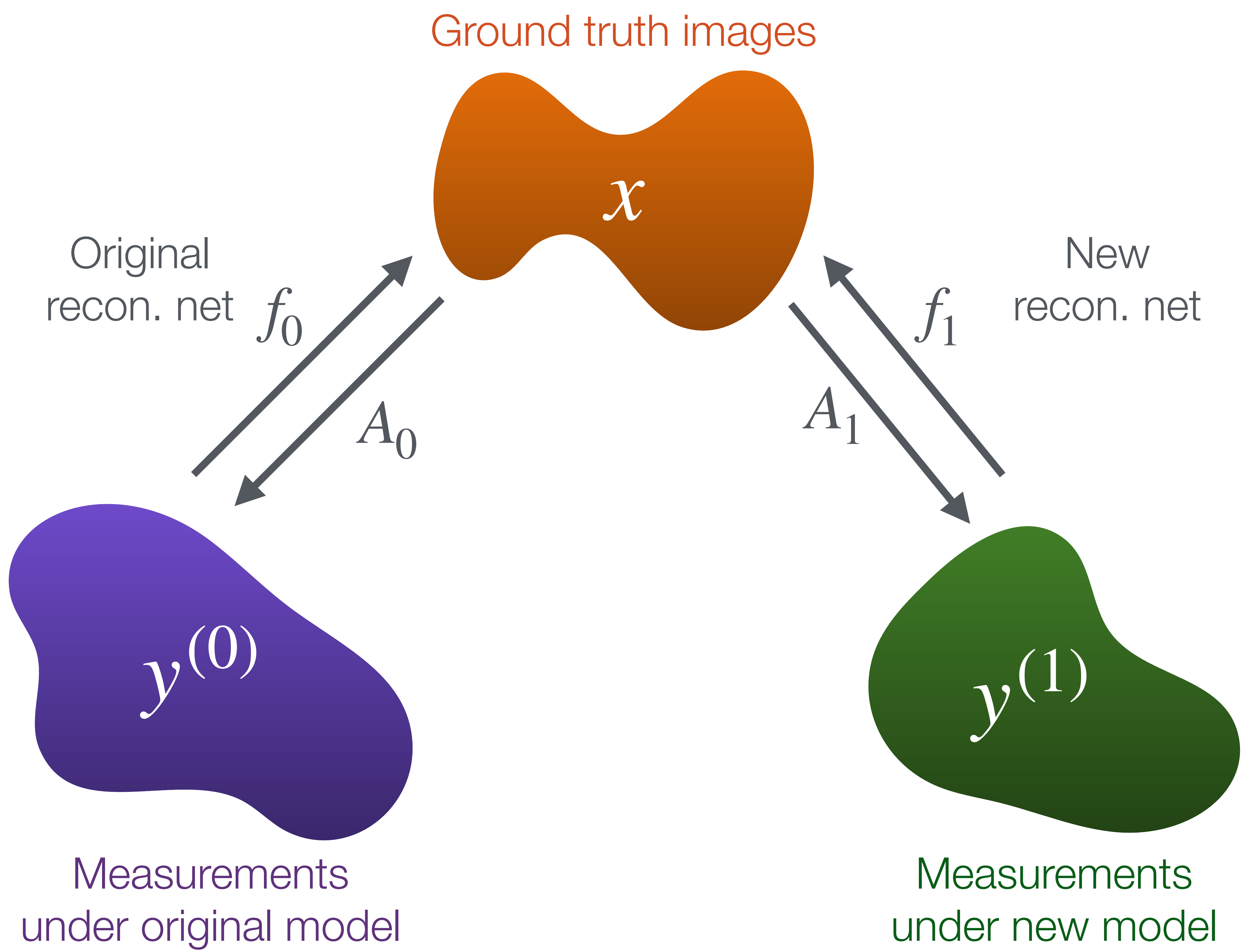


Basic idea: use calibration data to perturb parameters of original reconstruction network

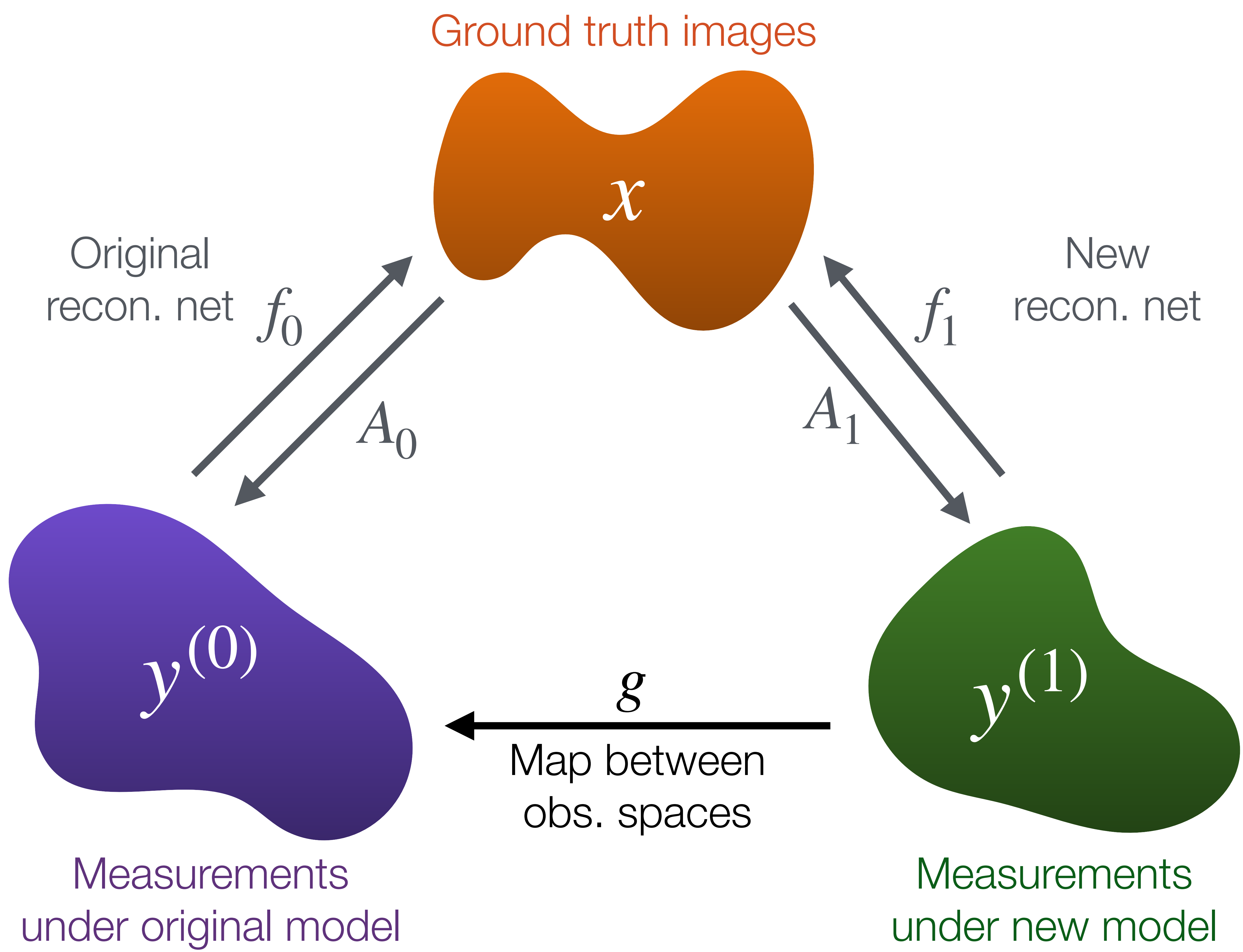
$$\hat{\delta} = \arg\min_{\delta} \sum_{i=1}^{n_1} \|y_i^{(1)} - A_1 \hat{x}_i(\delta)\|_2^2 + \lambda \|\delta\|_2^2$$

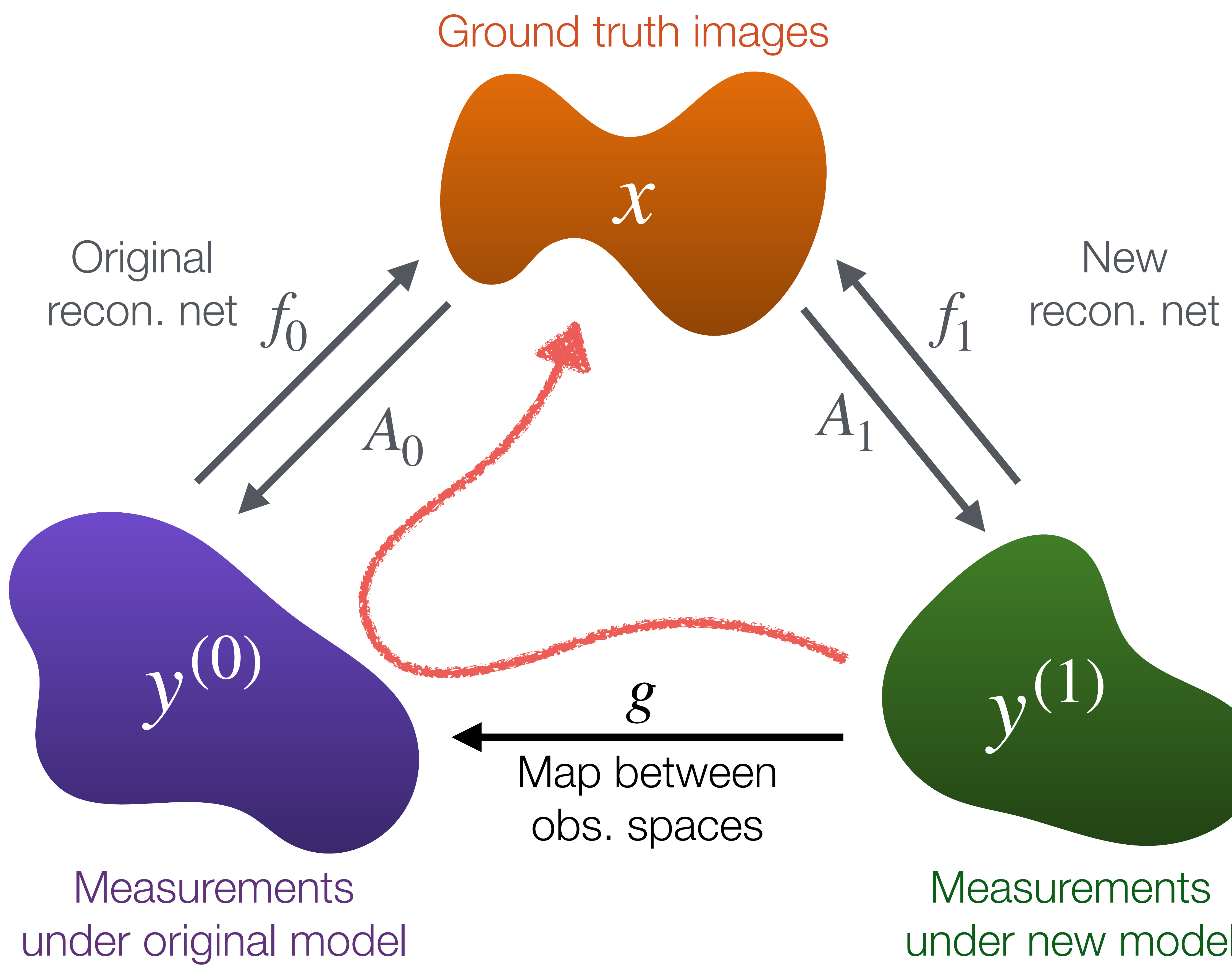
$$\text{subject to } \hat{x}_i(\delta) = f(y_i^{(1)}; \phi_0 + \delta, A_1)$$


Ground truth images
x
A_0
A_1
$y^{(0)}$
$y^{(1)}$
Measurements
under original model
Measurements
under new model

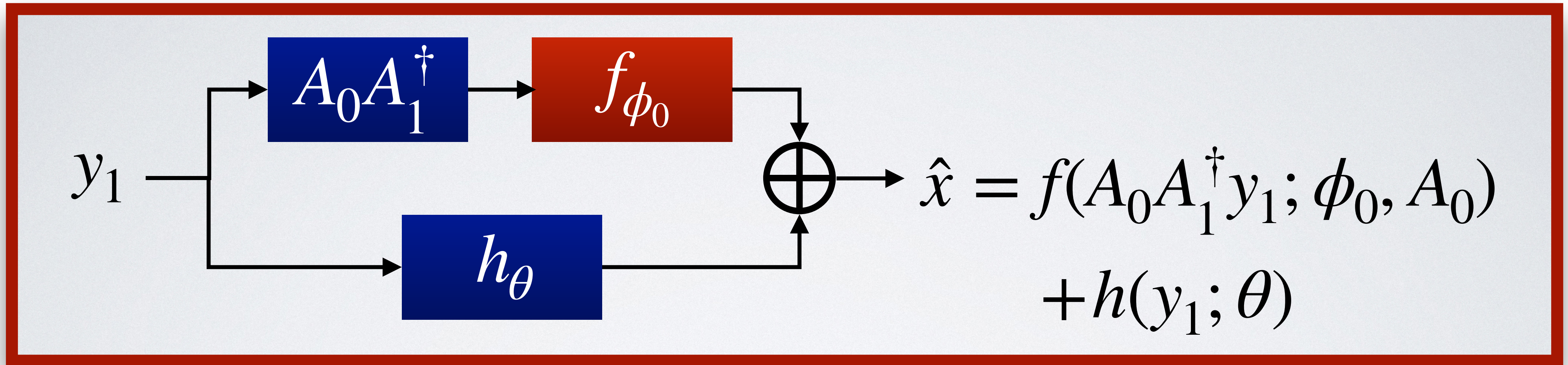


Ground truth images
x
Original
recon. net
f_0
A_0
New
recon. net
f_1
A_1
$y^{(0)}$
$y^{(1)}$
Measurements
under original model
Measurements
under new model


Ground truth images
x
Original
recon. net
f0
A0
New
recon. net
f1
A1
y(0)
g
Map between
obs. spaces
y(1)
Measurements
under original model
Measurements
under new model


Ground truth images
x
Original
recon. net
f_0
A_0
New
recon. net
f_1
A_1
$y^{(0)}$
$y^{(1)}$
g
Map between
obs. spaces
Measurements
under original model
Measurements
under new model

Approach 2: Condition and Correct (C&C)

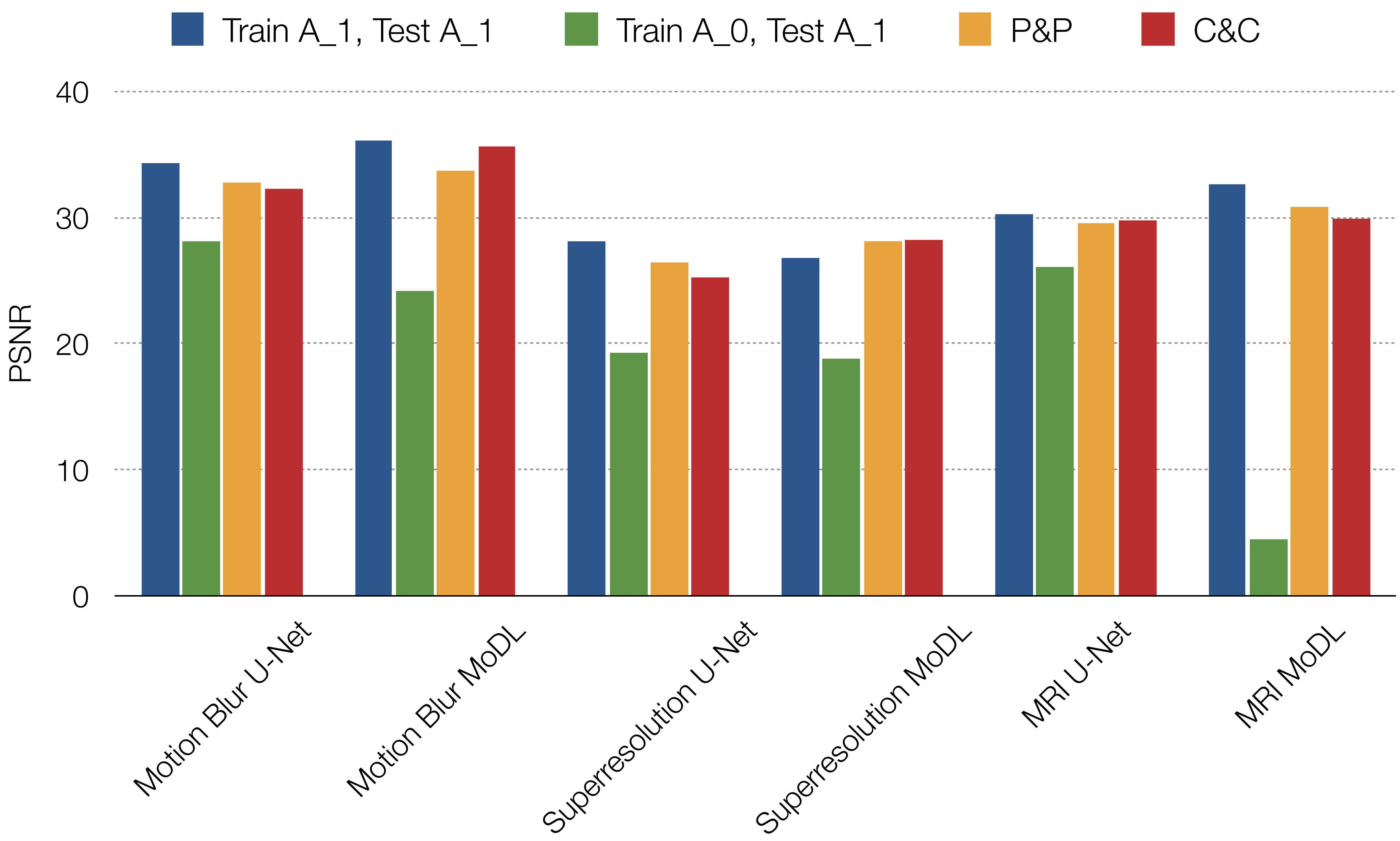


- Map $y^{(1)} \to \hat{y}^{(0)}$ (condition): $\hat{y}^{(0)} = g(y^{(1)}) = A_0A_1^\dagger y^{(1)}$
- Use original reconstruction network to get initial estimate
- Correct errors with network $h(\,\cdot\,;\theta)$

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n_1} \|y_i^{(1)} - A_1\hat{x}_i(\theta)\|_2^2 + \lambda\|h(y_i^{(1)};\theta)\|_2^2$$

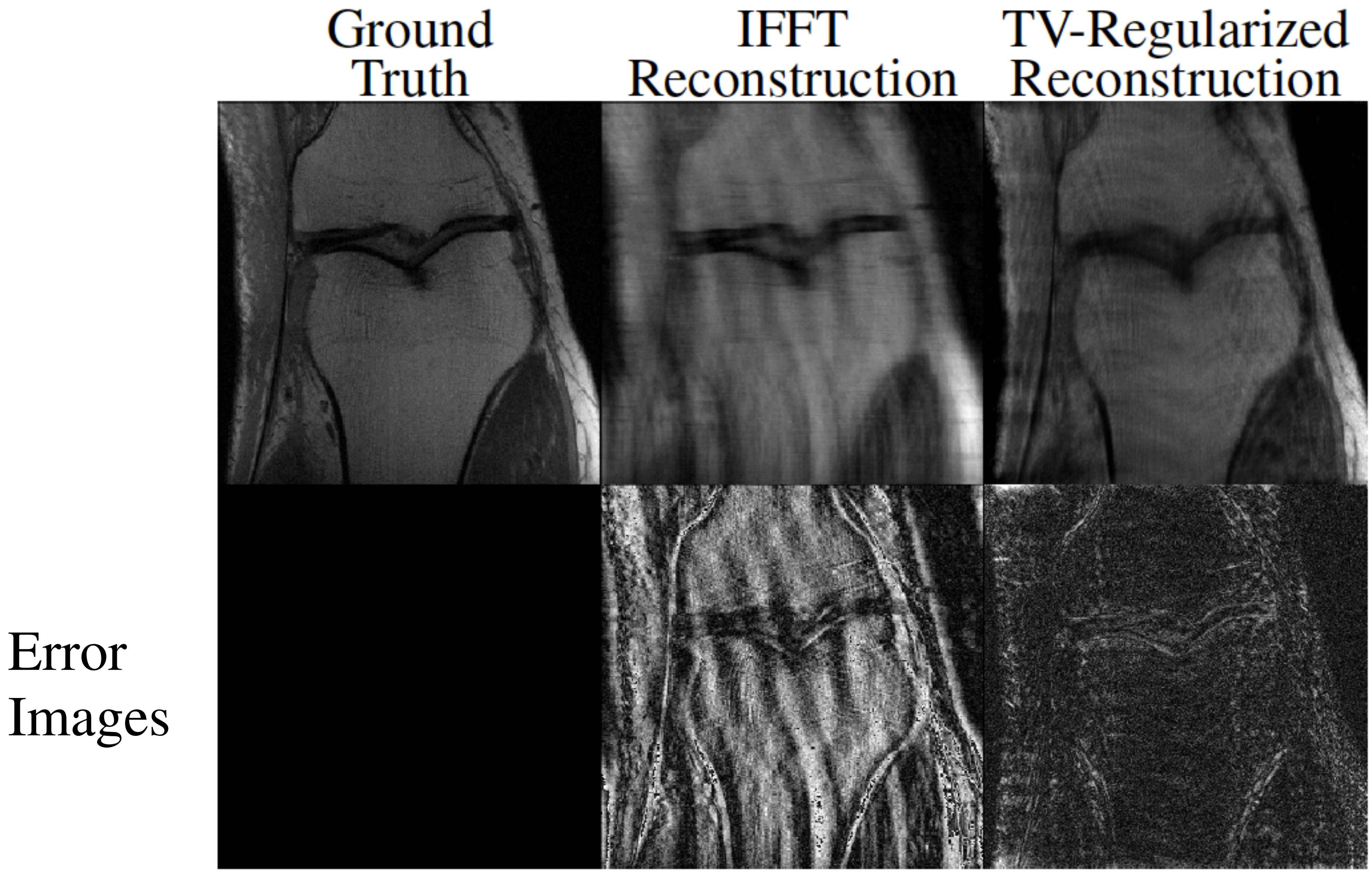
$$\text{subject to } \hat{x}_i(\theta) = f(\hat{y}_i^{(0)};\phi_0, A_0) + h(y_i^{(1)};\theta)$$

Performance with known A_1



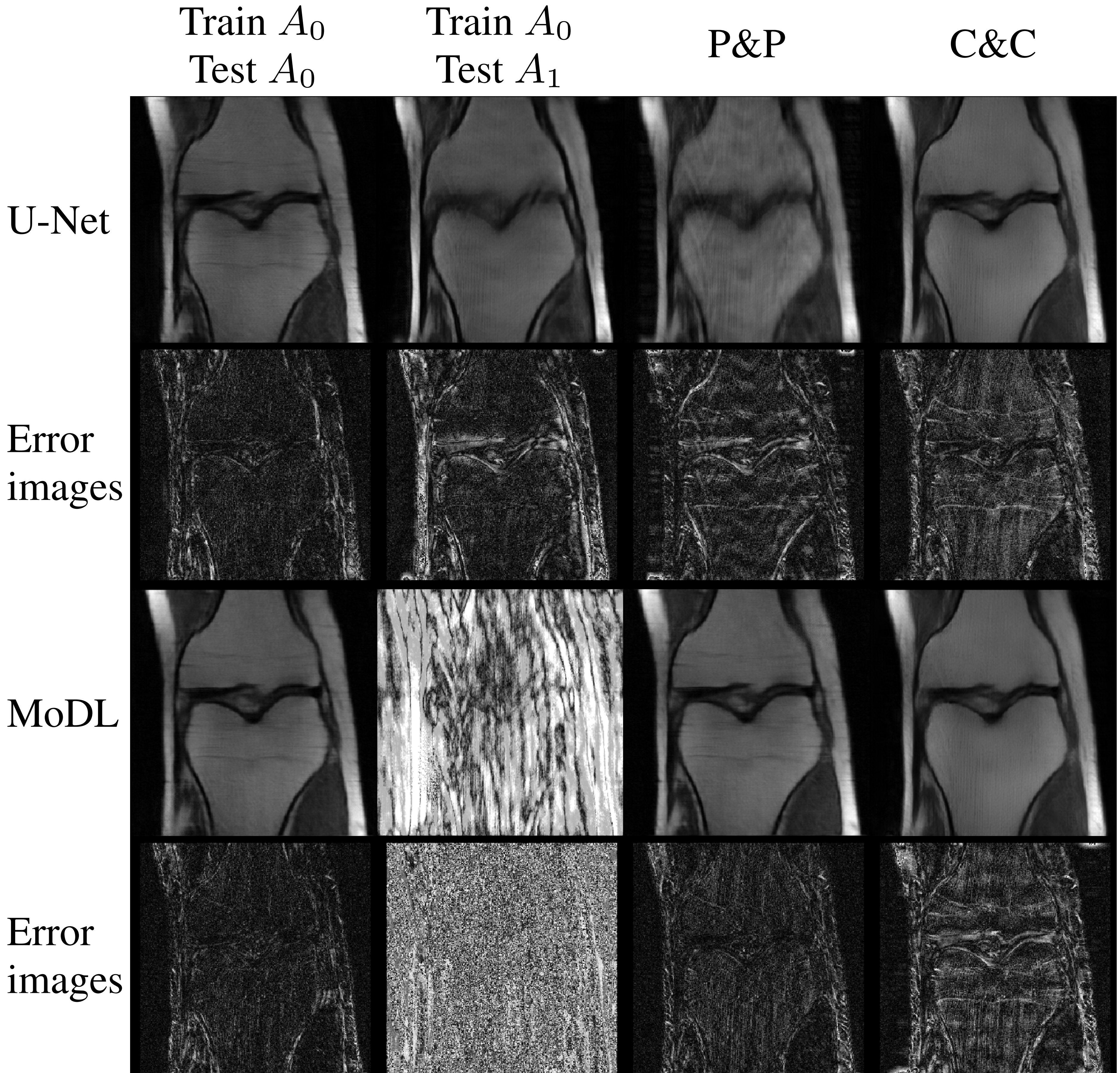
Ronneberger, Fischer, and Bro, 2015
Jure Zbontar, Florian Knoll, and others 2019

MRI Example


Ground Truth
IFFT Reconstruction
TV-Regularized Reconstruction
Error Images

MRI Example

Ground Truth



P&P with unknown A_1

$$\hat{\delta} = \underset{\delta}{\text{arg min}} \sum_{i=1}^{n_1} \|y_i^{(1)} - A_1 \hat{x}_i(\delta)\|_2^2 + \lambda \|\delta\|_2^2$$

$$\text{subject to } \hat{x}_i(\delta) = f(y_i^{(1)}; \phi_0 + \delta, A_1)$$

$$(\hat{\delta}, \hat{A}) = \underset{\delta, A}{\text{arg min}} \sum_{i=1}^{n_1} \|y_i^{(1)} - A \hat{x}_i(\delta)\|_2^2 + \lambda \|\delta\|_2^2$$

$$\text{subject to } \hat{x}_i(\delta) = f(y_i^{(1)}; \phi_0 + \delta, A)$$

C&C with unknown A_1

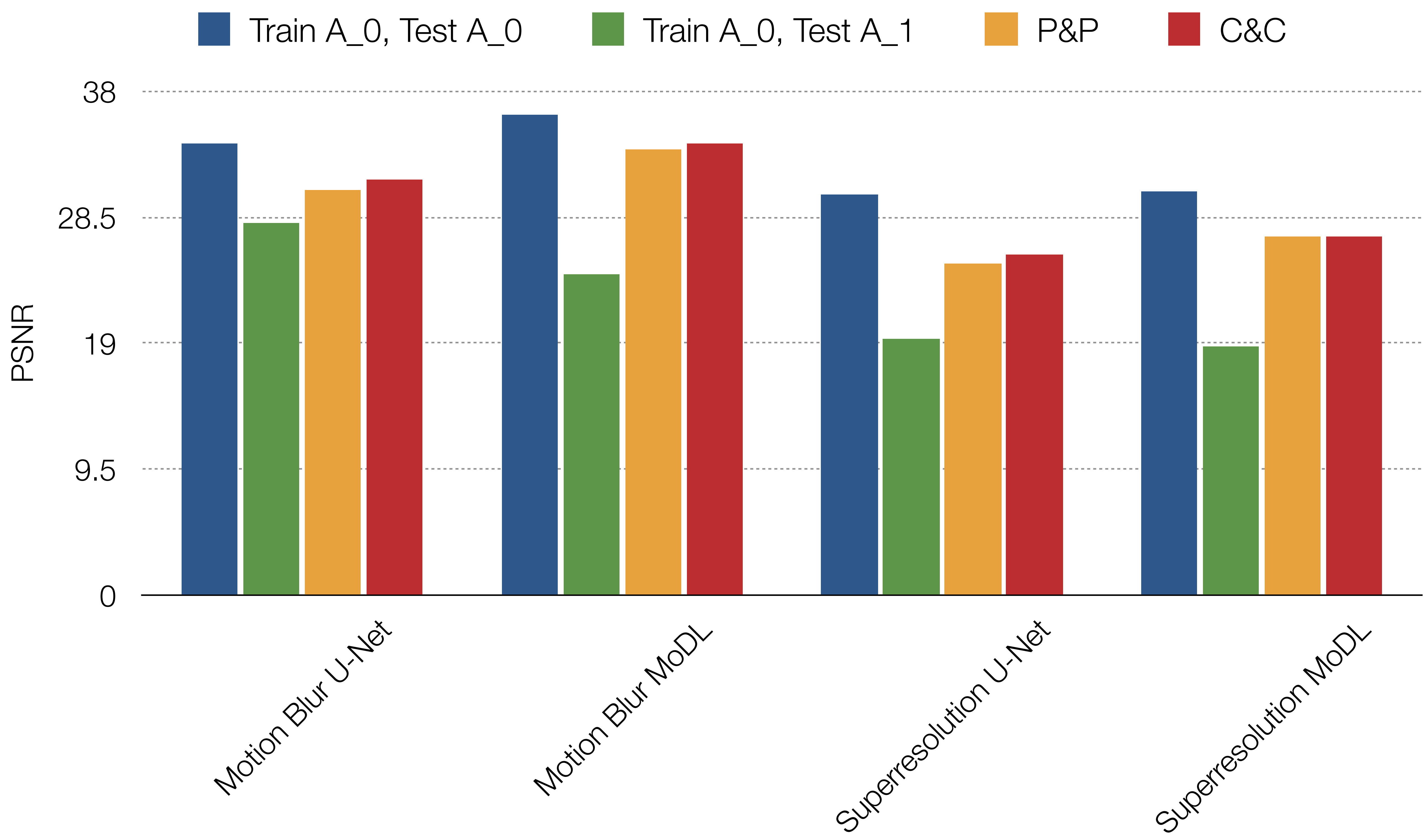
$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n_1} \|y_i^{(1)} - A_1 \hat{x}_i(\theta)\|_2^2 + \lambda \|h(y_i^{(1)};\theta)\|_2^2$$

$$\text{subject to } \hat{x}_i(\theta) = f(A_0 A_1^{\dagger} y_i^{(1)}; \phi_0, A_0) + h(y_i^{(1)};\theta)$$

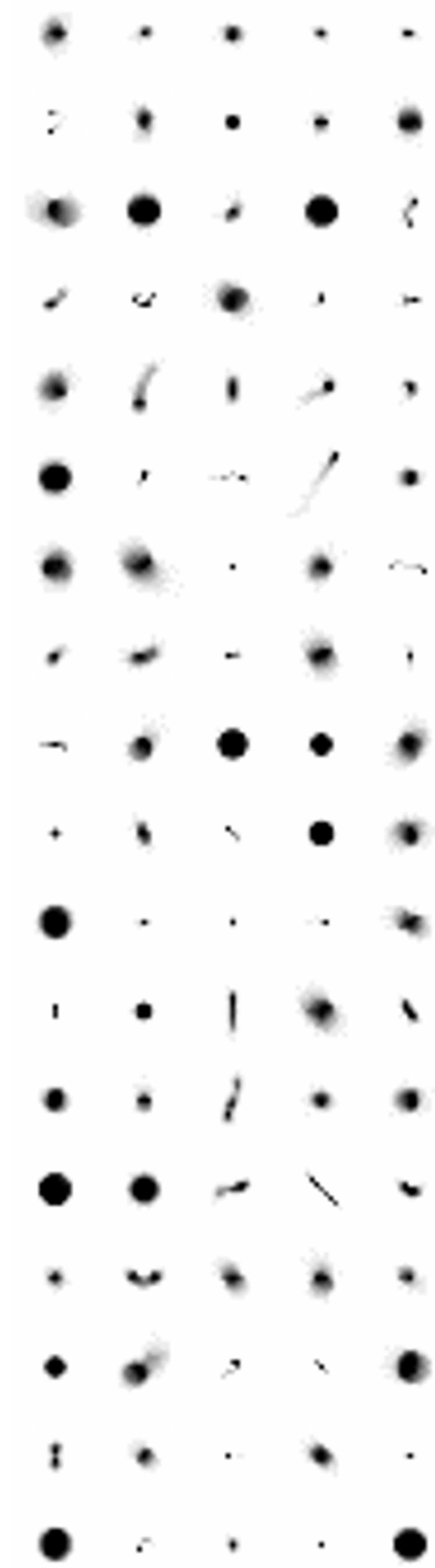
$$(\hat{\theta}, \hat{A}) = \arg\min_{\theta, A} \sum_{i=1}^{n_1} \|y_i^{(1)} - A \hat{x}_i(\theta)\|_2^2 + \lambda \|h(y_i^{(1)};\theta)\|_2^2$$

$$\text{subject to } \hat{x}_i(\theta) = f(A_0 A^{\dagger} y_i^{(1)}; \phi_0, A_0) + h(y_i^{(1)};\theta)$$

Performance with unknown A_1



A data augmentation approach

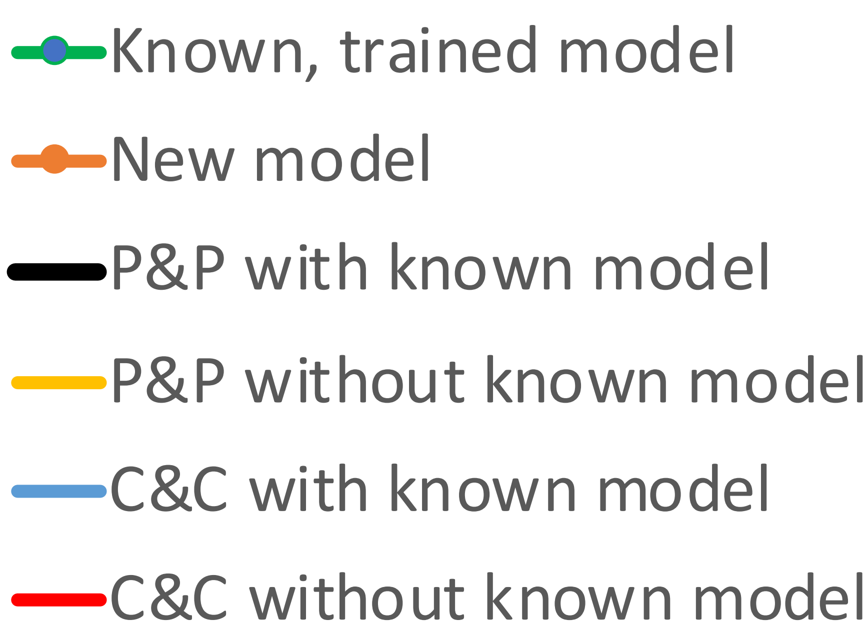


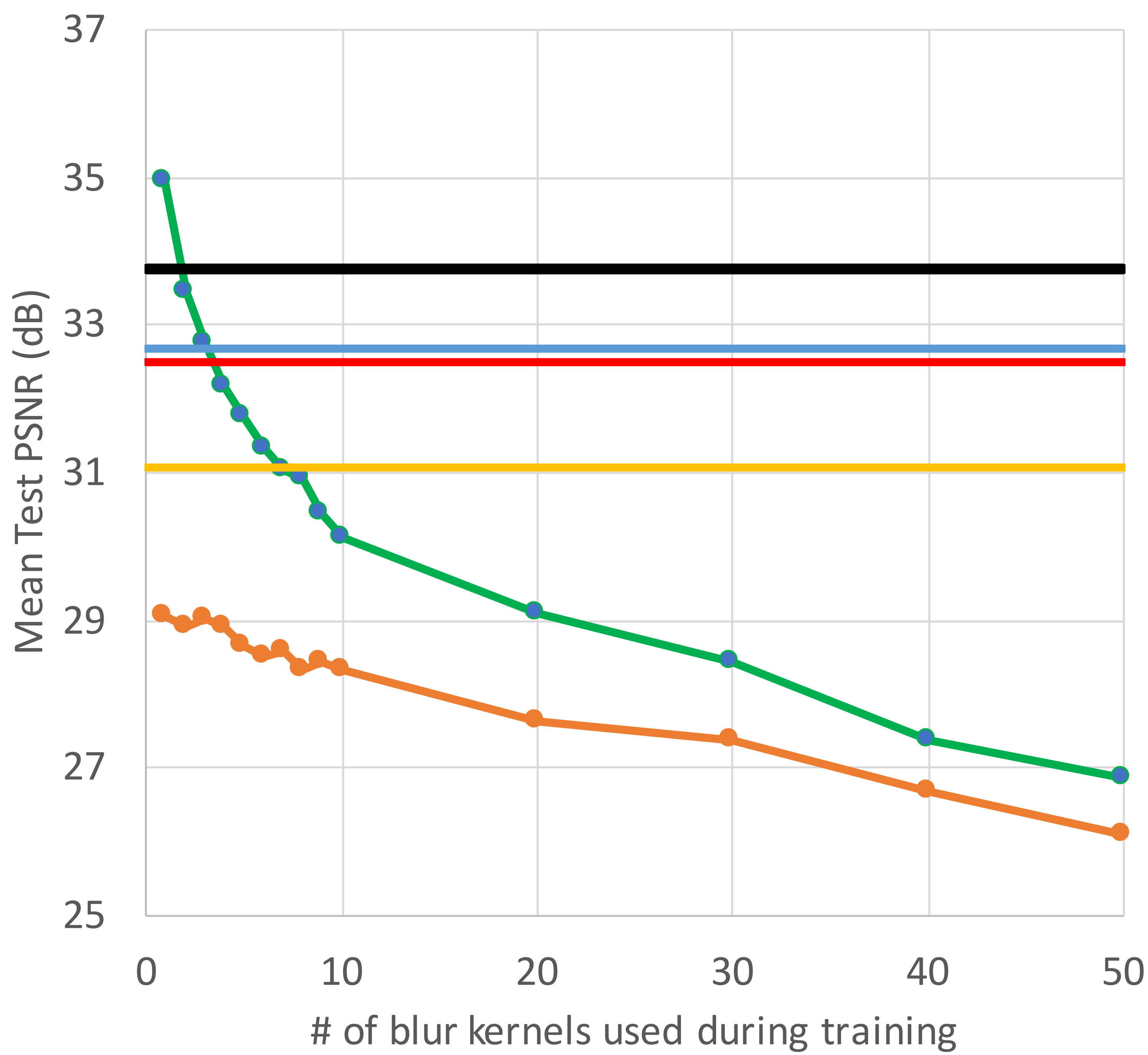
- Imagine we want to use a network for image deblurring
- We don’t know exactly what the blur kernel will be at test time
- So we perform “data augmentation” — train with multiple (x_i, y_i, A_i) samples, with each A_i corresponding to different kernels
- Does that work just as well?

Levin, 2006
Hradiš, Kotera, Zemcık, and Šroubek, 2015
Bahat, Efrat, and Irani, 2017

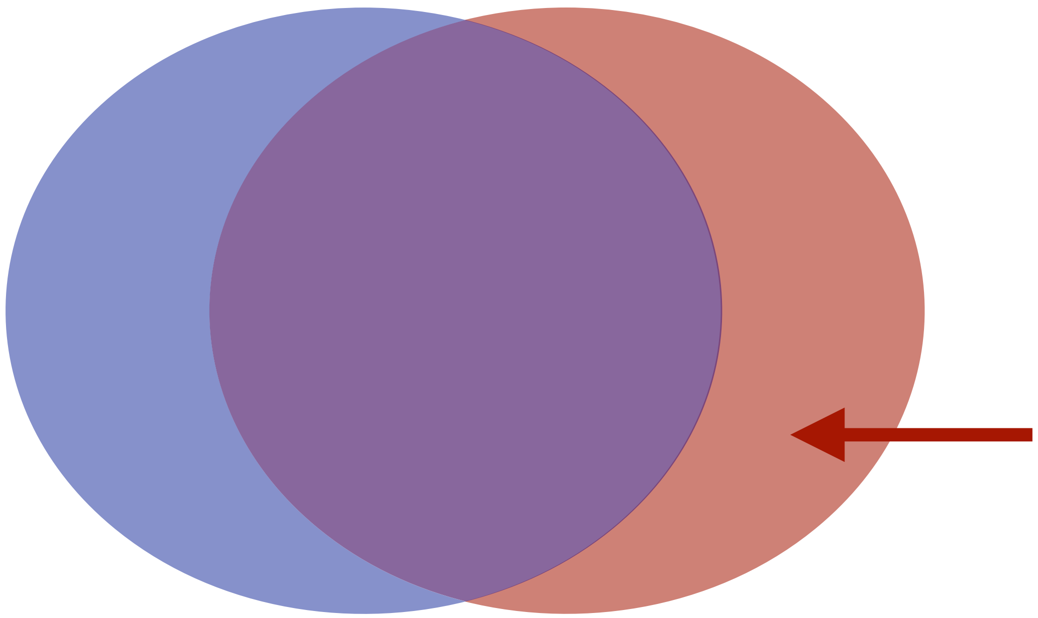
Naïvely learning to deblur with a single network and multiple blur kernels sacrifices performance on all blurs.

In green, the test-time accuracy of a network trained to deblur multiple blurs, and tested on a known kernel. In orange, the same network, but tested on a new blur that was not used during training. In black, our proposed P&P approach with a known model, and in yellow the same with a learned forward model. Blue and red show the performance of our C&C approach with and without a known forward model.



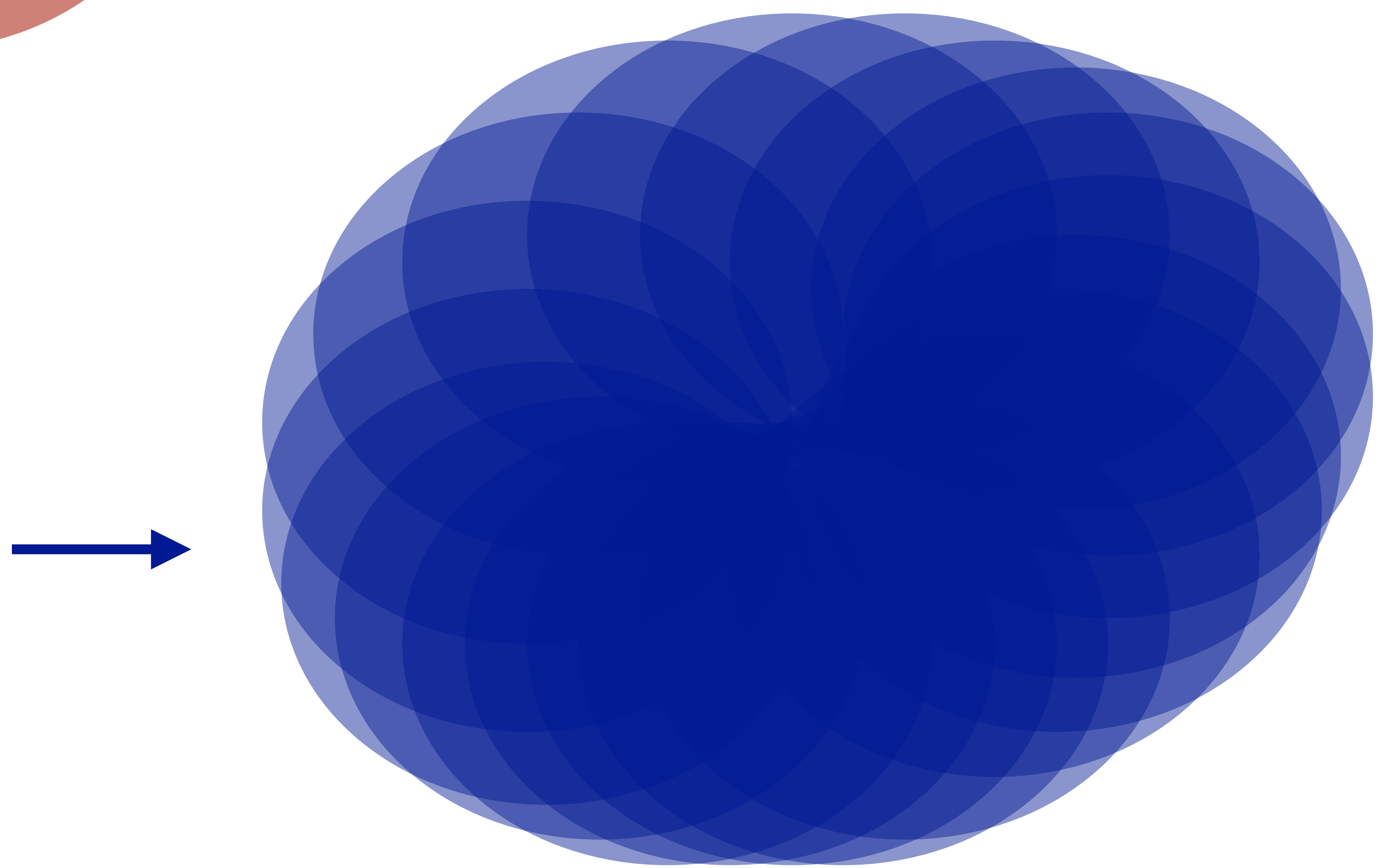


A null space perspective

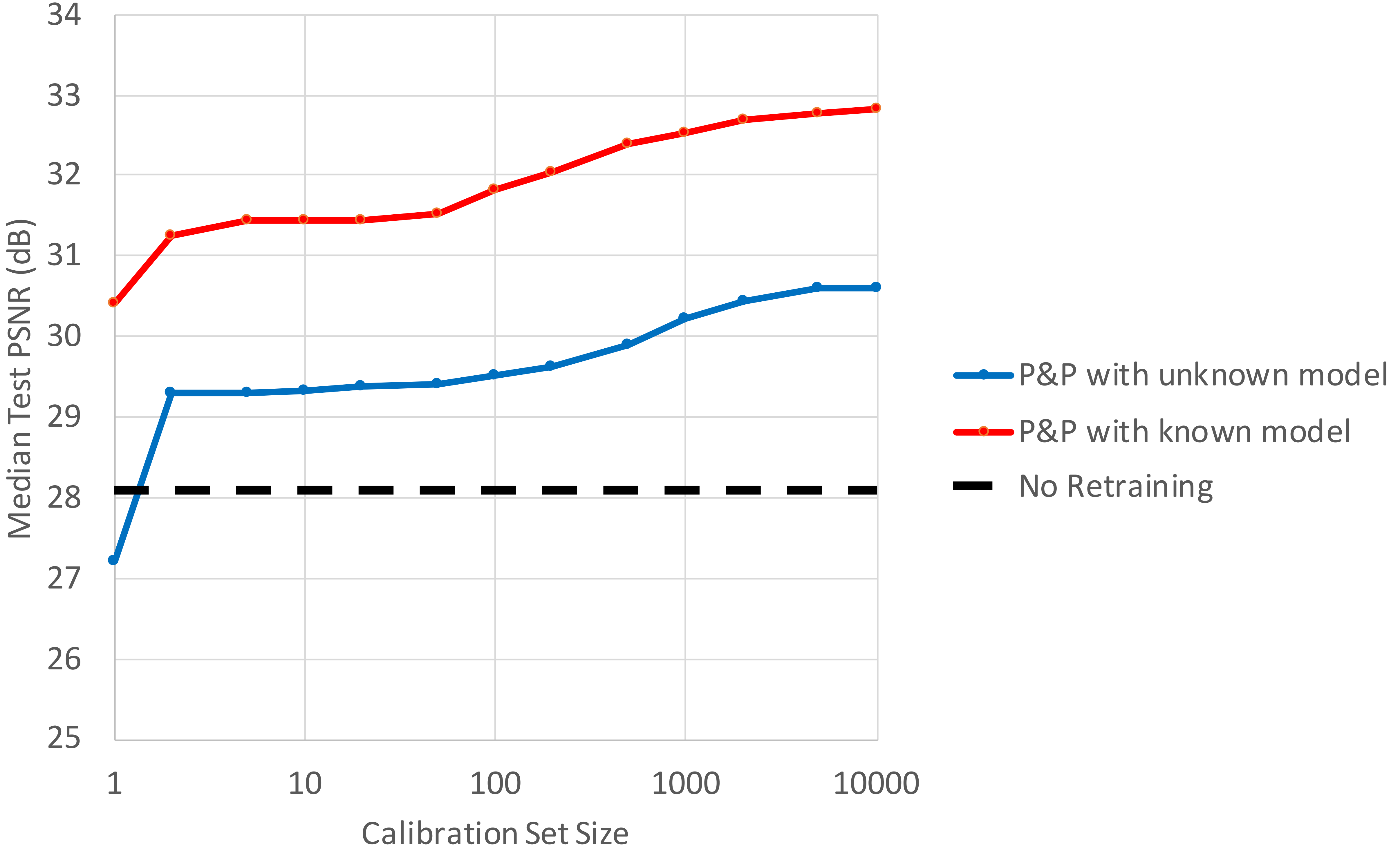


Model adaptation learns to fill this in

Data augmentation has to learn to fill in the union of many null spaces



Role of calibration data



Thank you!

- Model adaptation can dramatically improve reconstruction quality under real-world challenge of model drift
- Off-the-shelf methods of data augmentation or using GANs do not exploit known problem structure or calibration data, hurting performance
- Calibration data is easy to acquire without sharing large quantities of training data.