Meta-learning for Solving Inverse Problems

Qingqing Zhao
Stanford University
cyanzhao@stanford.edu

Abstract

Many image processing problems can be formulated as inverse problems, such as deconvolution, compressive MRI, etc. These problems are ill-posed hence it is almost impossible to recover the ground truth signals without any prior knowledge of the signals. The general framework of solving inverse problems involving minimize both data-fidelity term and a data prior term that promotes some desired properties of the signals. Our idea builds upon these classical algorithms, where we want to use meta-learning techniques, to learn a good initialization for the selected classical algorithm, so that it can converge to better solutions.

1. Introduction

Inverse problem manifests when one wants to reconstruct the unknown signal x from a noisy observation y, where the observation y is obtained from the unknown signal by a forward process $\mathcal{A}(\cdot)$ 1. Many image processing problems can be formulated as inverse problems, including inpainting, deblurring, compressed sensing and many more. These forward processes are mostly non-invertible which make inverse problem ill-posed - given one measurement and without any prior information of the signal, there are numerous possible solutions that can fit the measurement perfectly [6]. This makes inverse problems important and challenging.

$$y = \mathcal{A}(x) + \epsilon \tag{1}$$

There have been many methods developed for solving inverse problems and they can be categorized into optimization-based approaches, machine learning-based approaches and hybrid approaches.

Optimization-based approaches solve inverse problems by minimizing an objective function that consists of a data fidelity term, which measures how well the predicted signal fits the observation, and a hand-crafted prior term, which measures how likely the signal is a true signal. The prior term promotes images with some desired properties like gradient smoothness, sparseness, or self-similarity, etc. Such methods make full use of the image formation model $\mathcal{A}(\cdot)$ while does not include any learning components - that it does not leverage the available datasets.

Machine learning-based approaches leverage recent development in deep neural networks and available datasets, solving inverse problems by learning an image-to-image mapping directly [14][6]. However, those approaches do not incorporate the available knowledge of the physics model, while also lack interpretability.

There are also hybrid approaches, like unrolled optimization [1], where a classical solver with priors parameterized by a CNN is unrolled for a fixed number of steps, and trained end-to-end with ground truth signals. Such hybrid approaches leverage both model information and the dataset information. However, not all classical solvers can be unrolled in this way, making this approach lack of generality.

Our method falls into the hybrid approach category, where we want to leverage the knowledge from both the image formation model as well as the available dataset. To do so, we propose to use meta-learning techniques to learn a good initialization for the classical solver, so that it can converge to good solutions. Our approach is very general that we can plug in any differentiable classical solver.

2. Related Works

2.1. Neural Implicit Representation

Implicit representations of signals using neural network has emerged as a powerful paradigm, an alternative to grid-like representations. Supervised on ground truth signals, the neural network takes coordinate information as input and output the corresponding signal value at that point: $\phi_\theta(\text{coord}) = \text{Signal}(\text{coord})$. Recent works have demonstrated that MLPs with sinusoidal activation functions (SIREN) are capable of representing complex natural signals with high granularity[10]. In our work, we repre-

sent the initialization of the classical solver using SIREN parameterized by θ and we denote the signal sampled from this network as ϕ_θ.

2.2. Meta-learning

Meta-learning, also called learning-to-learn, refers to the process of improving a learning algorithm over many learning epochs [12]. An outer-loop algorithm is used to update an inner-loop algorithm, so that it becomes better (measured by some metrics). In our work, we mainly consider optimization-based meta-learning, more specifically, Model-Agnostic Meta Learning (MAML) [3]. In MAML, an outer-loop gradient-based learning is used to find a weight initialization of the neural network considered, so that it can adapt to different tasks more efficiently. MAML has also been used with neural implicit representations, where we can learn a weight initialization for neural representation, so that it can fit to different signals rapidly - in few gradient steps [9, 11].

Our approach uses MAML to learn a good initialization for a classical solver, where this initialization is represented by a SIREN neural network.

3. Methods

Our method has two parts. First, we introduce the inner-loop optimization procedure, which is also what we do during the inference time. Then we indicate the necessity of incorporate outer-loop optimization, and introduce the overall training scheme. We then provide two possible interpretations for the methods.

3.1. Model

Assume that we are given a dataset of ground truth signals x from a particular distribution $\mathcal{T}$. The goal is to recover x from the noisy measurement y obtained from a non-invertible forward image formation process,

$$y = \mathcal{A}(x) + \epsilon, \tag{2}$$

where ϵ represents the additive noise. In the following discussion, we assume additive white Gaussian noise.

3.2. Inner-loop Optimization

In the inner loop, we consider an initialization for the classical iterative solver represented by a SIREN network parameterized by θ, and ϕ_θ denotes the initialized signal sampled from the SIREN. When adapting to a given inverse problem with observation y_i, the SIREN's parameter θ becomes θ_i. The updated parameter θ_i is computed using gradient descent updates, i.e.,

$$\theta_i = \theta - \alpha \nabla_\theta L(\theta, y_i). \tag{3}$$

The inner-loop loss function is

$$L(\theta, y_i) = \|\mathcal{A}(f(\phi_\theta)) - y_i\|_2^2, \tag{4}$$

where $f(\cdot)$ represent a fixed number of updates done by the classical solver and α is the step size. Note that here the loss function is essentially the data fidelity term. We want to update the θ, so that with the updated θ_i, ϕ_{θ_i} will be a better initialization, that the final output from the classical solver, $f(\phi_{\theta_i})$, will fit the measurements better (hence lower data fidelity loss).

Algorithm 1: inner-loop Optimization

Require: measurement y_i, SIREN parameter θ, forward model $\mathcal{A}(\cdot)$, learning rate α, number of updates m, classical solver operator $f(\cdot)$

$\theta_i^0 = \theta$
for $k \leftarrow 0$ **to** $m-1$ **do**
  x = $f(\phi_{\theta_i^k})$
  $L(\theta_i^k, y_i) = \|\mathcal{A}(x) - y_i\|_2^2$
  $\theta_i^{k+1} \leftarrow \theta_i^k - \alpha \nabla_{\theta_i^k} L(\theta_i^k, y_i)$
end
x = $f(\phi_{\theta_i^m})$

3.3. Outer-loop Optimization

Note that for the inverse problems we are interested in, they are mostly ill-posted which does not possess a unique solution. Minimizing the data fidelity only is not sufficient. Some prior information of signal x is needed. To be able to find a good initialization so that we can not only find the signals that fit the measurements well, but also converge to solutions that are close to the ground truth signal, we wrap the inner-loop optimization by a meta-learning framework. We introduce the gradient-based outer-loop optimizer that updates the initial SIREN weight, so that the final solution from the inner-loop optimization gets closer to the ground truth signal.

Formally, the initial SIREN parameters θ_0 is trained by minimizing the distance between the inner-loop solution and the ground truth signal. More concretely, the optimization problem we are trying to solve is:

$$\theta^* = \text{argmin}_{\theta_0} \mathbb{E}_{x \sim \mathcal{T}} \mathcal{L}(f(\phi_{\theta_m(\theta_0, y)}) - x) \tag{5}$$

Note that here, the meta-parameter that we are trying to optimize is θ_0, whereas the outer loss is computed using updated parameter θ_m obtained through inner-loop optimization (as in Algorithm 1).

Our approach is very general - we can plug any differentiable classical solver into the inner loop, including gradient descent, conjugate gradient, and many others. We

Algorithm 2: Meta-learning for Inverse Problem

Require: Distribution $\mathcal{T}$ over the ground truth signals, forward model $\mathcal{A}(\cdot)$, inner-loop learning rate α, number of inner-updates m, outer learning rate β

Initialize meta-parameter θ
while *not done* **do**
  sample a batch of signals $x_i \sim \mathcal{T}$
  for *all* x_i **do**
    $y_i = \mathcal{A}(x_i) + \epsilon$
    $\theta_i^0 = \theta$
    for $k \leftarrow 0$ **to** $m-1$ **do**
      $x = f(\phi_{\theta_i^k})$
      $L_{in}(\theta_i^k, y_i) = \|\mathcal{A}(x) - y_i\|_2^2$
      $\theta_i^{k+1} \leftarrow \theta_i^k - \alpha \nabla_{\theta_i^k} L_{in}(\theta_i^k, y_i)$
    end
    $x = f(\phi_{\theta_i^m})$
    $L_{outer} \leftarrow L_{outer} + \|x - x_i\|_2^2$
  end
  $\theta \leftarrow \theta - \beta \nabla_\theta L_{outer}$
end

can also define different outer loss functions to measure the distance between the inner-loop outputted solutions, and ground truth signals. Possible choices of outer loss include classical metrics like L1 loss, L2 loss as well as neural network based metrics like perceptual loss [15].

3.3.1 Interpretation

The interpretation for the proposed method remains an open and interesting question. One possible interpretation straightly follows the meta-learning idea: we have an inner-loop optimization scheme for solving inverse problems, and we improve this inner-loop solver through a gradient-based outer-loop optimizer supervised on grounds truth signals. The inner-loop optimizer is updated by updating θ_0 over the course of learning. The hope is that we can find a θ_0, so that through inner-loop updates using data fidelity loss only, it evolves in the direction that it becomes a much better initialization for the classical solver, so that the final output from the inner-loop optimizer is close to grounds truth signals. Another way to understand the proposed approach is from the manifold learning perspective. In general a prior is used to constrain the solution to a low-dimensional manifold that the dataset lies on. We encode our prior into the θ_0 through outer-loop optimization and hope to find a θ_0 that would remain close to the data manifold after few inner-loop updates.

4. Experiments and Analysis

We test our method on two low-level image processing tasks, inpainting and deblurring. We train our method on simulated data. The dataset used is CelebA[5], and we use 150000 images from the training set for training, and 500 images from the validation set for evaluation. The performance of the method is evaluated in terms of PSNR, SSIM, MSE and LPIPS[15]. The SIREN uses 5-layers with a hidden layer size of 256. For outer-loop optimization, we use ADAM with a learning rate of 1e-5.

4.1. Inpainting

4.1.1 Missing Center Region

We choose gradient descent as the classical solver for the inpainting task. Note that for inpainting task, the classical solver 6 is not doing anything but to fill in the surrounding pixels. We use an inner gradient step size of 0.9 during the training so that the inner loss is not 0. We use a batch size of 5, and learn a per-step learning rate for inner-loop optimization, i.e. α_i for $i \in [1, m]$, where α_i is initialized at 0.01.

$$x^{k+1} = x^k - \alpha A^T(Ax - y) \tag{6}$$

We zero-out the center 1/3 of the image of resolution 128×128 as the degraded image (see 1). We test the performance our method with different number of inner-loop updates, $m = 2$, $m = 3$, $m = 5$, $m = 7$. We also train UNet[8] as a baseline for comparison. Sample outputs can be found in figure 1, and numerical evaluation can be found in table 1.

We observe that our approach learns to fill in the missing pixels with meaningful contents - that it learns the feature of the missing part like the skin color, eyes and the nose. An increase in performance is observed when the number of inner-loop optimization steps increases1. However, our approach does not beat baseline UNet in all metrics evaluated.

4.1.2 Random Missing Pixels

We randomly zero-out $10 - 1000$ pixels of the image of size 32×32 as the degraded image (see 2). In this test, we do not include any classical solver in the loop, so that the SIREN directly represents the solution. We perform inner-loop updates of SIREN for $m = 5$ times. A per-parameter (i), per-step (j) learning rate β_{ij} is learned, as proposed by Li et al [4]. We also train UNet [8] as a baseline for comparison. First, we observe that our method learn the given pixels very well. It also fills in the missing pixel well when the number of observable pixels are large. As the number of observable pixels become smaller, our method produce

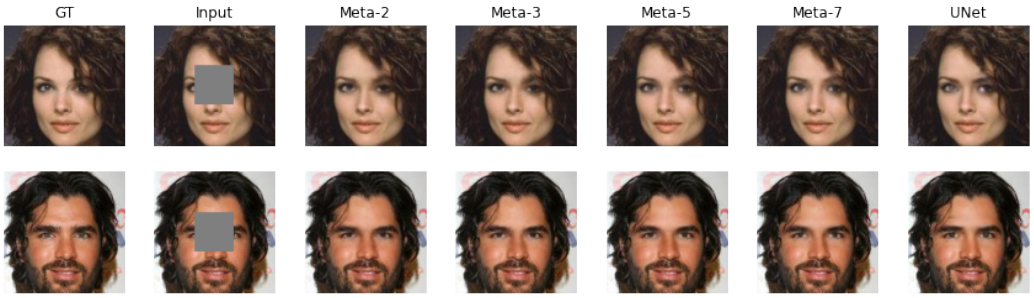

Figure 1: Sampled inpainting results

	MSE↓	PSNR↑	SSIM↑	LPIPS↓
Meta-2	8.0e-4	32.2	0.961	0.0263
Meta-3	7.8e-4	32.3	0.962	0.0251
Meta-5	7.5e-4	32.6	**0.965**	**0.0244**
Meta-7	**7.3e-4**	**32.7**	**0.965**	0.0247
UNet	**6.8e-4**	**32.9**	**0.967**	**0.0226**

Table 1: Image inpainting results for images with the center $1/9$ region zeroed out. Meta-n represent using our approach with n inner-loop optimization steps. The bold numbers represent the best result from our approach, and bold with underline represent the best result among all experiments. LPIPS is calculated using "net=alex".

	Meta		UNet	
	PSNR↑	LPIPS↓	PSNR↑	LPIPS↓
10	12.5	2.0e-1	**15.1**	**1.2e-1**
500	27.6	5.6e-3	**28.8**	**4.2e-3**
1000	38.4	3.0e-4	**39.9**	**2.8e-4**
Half	17.4	**3.8e-2**	**17.6**	4.5e-2

Table 2: Image impainting results for images with randomly missing pixels.

a purplish color which is quite puzzling. Our method does not beat the baseline UNet in this experiment.

4.2. Deconvolution

For deconvolution, we use Gaussian kernel and additive white Gaussian noise of noise level 1.5%. We use RED with fixed point method as described in [7] as the classical solver. We use DnCNN denoiser [14] as the regularizer in RED. The DnCNN denoiser is trained with noise level 5% on CelebA dataset. We perform experiments on images of size 64×64 and train with the batch size of 3. The number of RED update is 3 and the number of SIREN update is 3. A per-parameter (i), per-step (j) learning rate β_{ij} is learned [4]. We experiment with different outer loss

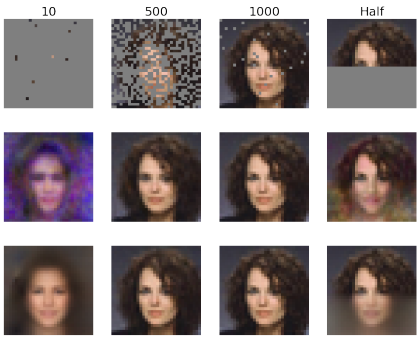

Figure 2: Sample Inpainting results: the first row represent input images, the second rows represents output images from the proposed method, the third row represents output image from UNet. LPIPS is calculated using "net=alex".

functions, L1, L2 and LPIPS to study the effect of the outer loss function. We called the proposed approach with RED as inner-loop solver as Meta-RED. We also evaluate RED [7] and half quadratic splitting (HQS) based iterative solver [13] as baselines for comparisons. For HQS, we use the implementation as in [13], with a DnCNN denoiser.

For the first set of experiments, we fixed the DnCNN denoiser, and only optimize the SIREN weight θ_0. The numerical evaluations can be found in Table 3. First, we observe that with meta framework, we achieve the best performances in term of the perceptual metric, LPIPS, compare to all baseline cases. From the sample images, we clearly see that our approach can better recover features like eyes and the nose. We also observe that our approach outperforms RED or HQS with 12 iterations and also achieves similar performance with RED running till convergence. However, our method does not beat HQS running till convergence.

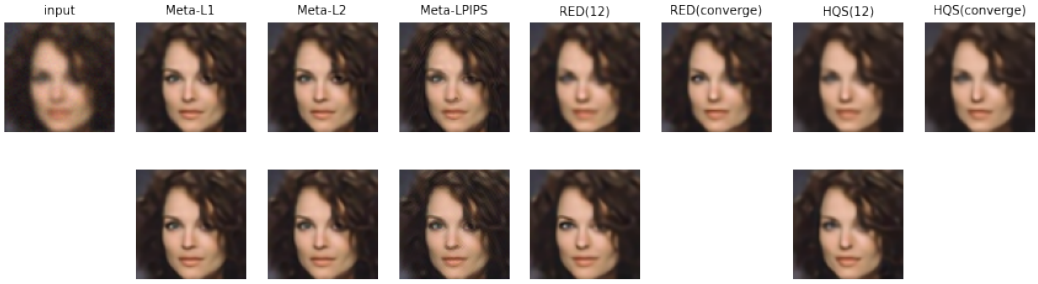

Figure 3: Sample deconvolution results: first row represent results when we use pre-trained fixed DnCNN denoiser that is fixed in any training process 3; second rows represent results when we fine tune DnCNN denoiser during training 4

	Meta-RED			RED		HQS	
	L1	L2	LPIPS	12 steps	convergence	12 steps	convergence
PSNR↑	29.3	**29.4**	28.3	27.7	29.3	28.4	**30.1**
SSIM↑	0.917	**0.918**	0.897	0.898	0.913	0.909	**0.921**
LPIPS↓	0.101	0.100	**0.094**	0.137	0.125	0.136	0.130
MSE↓	1.26e-3	**1.23e-3**	1.60e-3	1.83e-3	1.25e-3	1.55e-3	**1.06e-3**

Table 3: Deconvolution results with fixed DnCNN denoiser. Here Meta-RED refers to the proposed approach with RED with pre-trained DnCNN denoiser as the classical solver. RED refers to classical RED with pre-trained DnCNN denoiser. HQS refers to half quardrate split with per-trained DnCNN denoiser. Here, LPIPS is calculated using "net=vgg".

	Meta-RED			RED(12)	HQS(12)
	L1	L2	LPIPS	L2	L2
PSNR↑	**30.3**	**30.3**	29.1	30.1	**31.0**
SSIM↑	**0.929**	0.928	0.91	0.928	**0.938**
LPIPS↓	0.097	0.095	**0.073**	0.100	0.090
MSE↓	**1.0e-3**	**1.0e-3**	1.3e-3	1.1e-3	**8.5e-4**

Table 4: Deconvolution results with fine-tuned DnCNN denoiser. Here Meta-RED refers to the proposed approach with RED with fine-tuned DnCNN denoiser as the classical solver. For RED and HQS, we unroll both solver for 12 steps and fine tune the DnCNN denoiser in RED/HQS with L2 loss w.r.t ground truth signals. Here, LPIPS is calculated using "net=vgg".

For the second set of experiments, we fine tune the DnCNN denoiser during the training. As baselines, we unroll the iterative solver, RED and HQS, for 12 steps and train it end-to-end with L2 loss with ground truth signals - to fine tune the DnCNN denoiser. 12 steps is used here as this is the effective number of solver updates in the Meta-RED, $3\times(3+1)$. The numerical evaluations can be found in 4. By fine tuning the denoiser, we observe a boost in performance for both proposed method, and the unrolled method. Again, Meta-RED slight outperforms RED. However, unrolled HQS still achieves the best performances in PSNR, SSIM and MSE. Though Meta-RED trained with LPIPS as outer loss produce better LPIPS score, from the sample images in Table 3, we observe some artifacts - checkerboard patterns, that are not visually pleasing.

5. Discussions

Our method does learn a prior of the dataset that it produces meaningful results for both inpainting task and deconvolution task. Especially in the deconvolution task, it produces better performance compare to the classical solver with an equivalent number of steps. However, we do not obverse significant improvement from this extra step of learning. It also does not beat baselines like UNet. One possible reason is due to that SIREN is highly over-parameterized - it can overfit almost any 2D images including noises. However, in general, prior is used to constrain the solution to a low-dimensional manifold that the dataset lies on. While we constrained the evolution of SIREN via meta-learning and a fixed number of inner-loop updates, this may not be sufficient. It would be interesting to reduce the degree of freedom in the inner-loop optimization by using smaller SIREN, or fixed SIREN that is modulated by feature-wise-transformation [2]. Besides, it would also be interesting to investigate better ways to incorporate prior into the inner-loop optimization.

6. Conclusion

We successfully applied meta-learning techniques to learn a good initialization for the classical solvers for solving inverse problems. Our approach is very general that it can work with any differentiable classical solver or any inverse problems. We experimentally demonstrate that our method produces better performances compared to the classical solvers with an equivalent number of iterations. However, the method does not beat baselines like UNet (for inpainting task) or HQS (for deconvolution task). It is left for future work to study the effect of over-parameterization of SIREN, as well as to explore better ways of incorporating prior into the inner-loop optimization.

References

[1] S. Diamond, V. Sitzmann, F. Heide, and G. Wetzstein. Unrolled optimization with deep priors, 2018.

[2] V. Dumoulin, E. Perez, N. Schucher, F. Strub, H. d. Vries, A. Courville, and Y. Bengio. Feature-wise transformations. *Distill*, 2018. https://distill.pub/2018/feature-wise-transformations.

[3] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[4] Z. Li, F. Zhou, F. Chen, and H. Li. Meta-sgd: Learning to learn quickly for few-shot learning, 2017.

[5] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[6] G. Ongie, A. Jalal, C. A. Metzler, R. G. Baraniuk, A. G. Dimakis, and R. Willett. Deep learning techniques for inverse problems in imaging. *IEEE Journal on Selected Areas in Information Theory*, 1(1):39–56, 2020.

[7] Y. Romano, M. Elad, and P. Milanfar. The little engine that could: Regularization by denoising (red), 2017.

[8] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[9] V. Sitzmann, E. R. Chan, R. Tucker, N. Snavely, and G. Wetzstein. Metasdf: Meta-learning signed distance functions. In *Proc. NeurIPS*, 2020.

[10] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein. Implicit neural representations with periodic activation functions. In *arXiv*, 2020.

[11] M. Tancik, B. Mildenhall, T. Wang, D. Schmidt, P. P. Srinivasan, J. T. Barron, and R. Ng. Learned initializations for optimizing coordinate-based neural representations, 2020.

[12] J. Vanschoren. Meta-learning: A survey, 2018.

[13] K. Zhang, Y. Li, W. Zuo, L. Zhang, L. Van Gool, and R. Timofte. Plug-and-play image restoration with deep denoiser prior. *arXiv preprint*, 2020.

[14] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang. Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. *IEEE Transactions on Image Processing*, 26(7):3142–3155, 2017.

[15] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.