A Bilevel Optimization Approach for Task-Specific Joint Learning Pipeline

20EC39017, Electronics and Electrical Communication Department, IIT Kharagpur

In the domain of inverse problems, traditional sequential pipelines for parameter estimation often introduce approximation errors by treating reconstruction and decision-making as disconnected steps. This project addresses the challenge of integrating task-aware strategies into the reconstruction process by utilizing a bilevel optimization framework that simultaneously optimizes image reconstruction and task performance. While existing approaches like joint learning have attempted to address these limitations, they need more mathematical flexibility to separate and explicitly optimize reconstruction and task-specific objectives. The proposed method uses the Hyperparameter Optimization with Approximate Gradient algorithm, utilizing implicit differentiation to develop a nested optimization structure with an inner level focused on image reconstruction using smoothed total variation regularization and an outer level optimizing task-specific metrics. Experimental validation on Cityscapes and Stanford Dogs datasets demonstrates significant performance improvements in segmentation and classification tasks across varying noise intensities. The research contributes a flexible, generalizable approach that not only mitigates traditional pipeline approximation errors but also provides a promising framework for future applications in complex inverse problem pipelines.

Inverse problems | Image reconstruction | Bilevel framework | Joint Learning

Introduction

Inverse problems are fundamental in many scientific fields, involving estimating model parameters to match observed data. Traditionally, these problems are addressed using a sequential pipeline, where reconstruction and decision-making are treated as separate steps. However, this approach introduces approximations at each stage, often neglecting the final task during reconstruction and ignoring measured data during feature extraction. Integrating task-aware strategies into the reconstruction process can mitigate these issues by aligning intermediate steps with the end objective.

The paper *Task Adapted Reconstruction for Inverse Problems* (Adler et al. (1)) introduces a joint learning paradigm to tackle these challenges. It focuses on estimating model parameters from noisy, indirect measurements while integrating reconstruction and task-specific decision-making into a single end-to-end process. The framework balances reconstruction fidelity with task performance using a shared loss function. Using neural networks ensures scalability and adaptability across various tasks and inverse problems, enhancing task accuracy, reducing computational overhead, and incorporating inherent regularisation for robust and interpretable results.

While effective, the joint learning approach lacks the mathematical flexibility of a bilevel framework (Crockett and Fessler (2)), which explicitly separates the reconstruction and task performance objectives into a nested optimization structure. The bilevel setup addresses several challenges inherent in joint learning by decoupling reconstruction (lower level) and task-specific optimization (upper level). This separation allows reconstruction parameters to be optimized for fidelity while accounting for their downstream impact. Furthermore, the framework supports the inclusion of domain-specific regularizers at the lower level and directly optimizes task-related metrics at the upper level. It also provides greater control over hyperparameter tuning and systematically handles non-convex objectives, offering enhanced flexibility and robustness for complex inverse problems.

Notation. The upper-level loss function, denoted as $\ell(\gamma) \mapsto \mathbb{R}$ or $\ell(\gamma; \mathbf{x}^*) \mapsto \mathbb{R}$, is used as a fitness measure of γ. While ℓ is primarily a function of γ, it is often useful to express it with two inputs, where typically $\mathbf{x} = \hat{\mathbf{x}}(\gamma)$ represents the solution to the inner optimization problem (reconstructed image). The lower-level cost function, $\Phi(\mathbf{x}; \gamma) \mapsto \mathbb{R}$, is used for reconstructing an image, and is typically parameterized by γ. Regularization is represented by $R_\gamma(\mathbf{x}) \mapsto \mathbb{R}$, which incorporates prior information about the likely characteristics of an image. We use ∇ to denote the gradient of a real-valued function. If the function has multiple input arguments, ∇_i represents the gradient with respect to the argument i. Similarly, ∇^2 denotes the Hessian, and $\nabla^2_{i,j}$ represents the second-order differential with respect to variables i and j. In this context, the task labels are represented by s (e.g., segmentation masks or class labels), and the ground truth image/data is represented by $\mathbf{x}^*$. The noisy measurement of the image is denoted by $\mathbf{y}$. The solution to the inner optimization problem is $\hat{\mathbf{x}}(\gamma)$, and it is distinct from a generic $\mathbf{x}$. The task-specific loss is represented as $\ell(\gamma; \hat{x}(\gamma))$, and the reconstruction loss (lower-level loss) is represented by $\Phi(\mathbf{x}; \gamma)$, with the regularization term $R_\gamma(\mathbf{x})$ governing the smoothness or prior structure of the solution.

Bilevel Optimization Framework. Many optimization problems, especially in hyperparameter tuning and inverse problems, aim to solve a bilevel optimization problem. The outer optimization seeks to find the optimal parameter γ^* that minimizes an objective function $\ell(\gamma; \hat{x}(\gamma))$, expressed as:

$$\hat{\gamma} = \arg\min_{\gamma \in \mathbb{F}^R} \ell(\gamma; \hat{x}(\gamma))$$

where $\hat{x}(\gamma)$ is the solution to an inner optimization problem. The inner problem minimizes another objective $\Phi(x;\gamma)$ with respect to x, for a given γ, defined as:

$$\hat{x}(\gamma) = \arg\min_{x \in \mathbb{F}^N} \Phi(x;\gamma)$$

Here, $\mathbb{F}^R$ represents the space of the parameters γ, and $\mathbb{F}^N$ represents the space of the variables x. This nested structure ensures that the outer optimization leverages the solution of the inner problem to achieve its objective.

Our problem setting. The dataset comprises images, their corresponding noisy measurements, and corresponding task labels. We aim to develop an integrated pipeline that combines image reconstruction and task execution within a bilevel optimization framework. To implement this, we employ the Hyperparameter Optimization with Approximate Gradient algorithm (Pedregosa (3)). This approach could be enhanced in the future by adopting a more robust algorithm to achieve improved performance(Salehi et al. (4)).

Hyperparameter Optimization with Approximate Gradient algorithm (HOAG)

When the lower-level optimization problem has a closed-form solution, $\hat{x}$, one can substitute that solution into the upper-level loss function. In this case, the bilevel problem is equivalent to a single-level problem and one can use classic single-level optimization methods to minimize the upper-level loss. But our project focuses on the more typical bilevel problems that lack a closed-form solution for $\hat{x}$. Using the chain rule, the gradient of the upper-level loss function with respect to the hyperparameters is

$$\nabla \ell(\gamma) = \nabla_\gamma \ell(\gamma; \hat{x}(\gamma)) + (\nabla_\gamma \hat{x}(\gamma))' \nabla_x \ell(\gamma; \hat{x}(\gamma))$$

To calculate $\nabla_\gamma \hat{x}(\gamma)$ we assume the gradient of Φ at the minimizer is zero. We use the implicit function theorem (IFT) perspective to arrive at the final expression.

Implicit Differentiation.

1. **Stationarity Condition** For smooth and convex $\Phi(\mathbf{x};\gamma)$, the solution $\hat{\mathbf{x}}(\gamma)$ satisfies the stationarity condition:
$$\nabla_{\mathbf{x}} \Phi(\hat{\mathbf{x}}(\gamma);\gamma) = 0$$
This implicitly defines $\hat{\mathbf{x}}(\gamma)$ as a function of γ.

2. **Derivative of $\hat{\mathbf{x}}(\gamma)$ w.r.t. γ** Differentiating the stationarity condition with respect to γ:
$$\nabla^2_{\mathbf{x},\gamma} \Phi + \nabla^2_{\mathbf{x},\mathbf{x}} \Phi \cdot \frac{\partial \hat{\mathbf{x}}}{\partial \gamma} = 0$$
If $\nabla^2_{\mathbf{x},\mathbf{x}} \Phi$ (the Hessian w.r.t. $\mathbf{x}$) is invertible:
$$\frac{\partial \hat{\mathbf{x}}}{\partial \gamma} = -\left(\nabla^2_{\mathbf{x},\mathbf{x}} \Phi\right)^{-1} \nabla^2_{\mathbf{x},\gamma} \Phi$$

3. **Gradient of $\ell(\gamma)$** Using the chain rule, the gradient of the upper-level loss $\ell(\gamma; \hat{\mathbf{x}}(\gamma))$ is given by:
$$\nabla \ell = \nabla_\gamma \ell + \left(\frac{\partial \hat{\mathbf{x}}}{\partial \gamma}\right)^\top \nabla_{\mathbf{x}} \ell$$
Substituting $\frac{\partial \hat{\mathbf{x}}}{\partial \gamma}$:
$$\nabla \ell = \nabla_\gamma \ell - \left(\nabla^2_{\mathbf{x},\gamma} \Phi\right)^\top \left(\nabla^2_{\mathbf{x},\mathbf{x}} \Phi\right)^{-1} \nabla_{\mathbf{x}} \ell$$

Implementation. To solve the bilevel optimization problem using implicit differentiation:

1. **Solve the Inner Problem:** Approximate the solution $\hat{\mathbf{x}}(\gamma)$ by minimizing the inner objective $\Phi(\mathbf{x};\gamma)$ with respect to $\mathbf{x}$. The optimization should achieve ϵ-optimality, where the gradient norm or objective value changes fall below a predefined tolerance. Gradient-based methods, such as Adam or gradient descent or second order quasi-Newton methods like L-BFGS, can be used to balance computational efficiency and accuracy.

2. **Compute Gradients:** Calculate the necessary gradients for implicit differentiation:
   - $\nabla_{\mathbf{x}} \ell$: The gradient of the outer objective ℓ with respect to $\mathbf{x}$, computed via chain rule.
   - $\nabla^2_{\mathbf{x},\gamma} \Phi$: The mixed partial derivative of the inner loss with respect to $\mathbf{x}$ and γ, evaluated as a vector-Jacobian product.
   - $\nabla^2_{\mathbf{x},\mathbf{x}} \Phi$: The Hessian of the inner objective with respect to $\mathbf{x}$, accessed efficiently using Hessian-vector products through PyTorch's `autograd.grad`. A Hessian-vector product is simply the product of the Hessian matrix with a vector. The key is that we often don't need the full Hessian matrix itself. We only need its action on a vector. We take advantage of this to reduce the expensive computation required to calculate full hessians.

3. **Solve Linear System:** Solve the linear system $\nabla^2_{\mathbf{x},\mathbf{x}} \Phi \cdot \mathbf{z} = \nabla_{\mathbf{x}} \ell$ to compute $\mathbf{z}$. The conjugate gradient (CG) method is recommended for efficiency, as it avoids explicit Hessian formation:
   - Define $H(p)$, a function to compute $\nabla^2_{\mathbf{x},\mathbf{x}} \Phi \cdot p$, using Hessian-vector products.
   - Use CG (`conjugate_gradient`) to iteratively solve the system, stopping when the residual norm is below a specified tolerance. CG ensures scalability for high-dimensional problems by leveraging implicit Hessian computation and sparsity.

4. **Update γ:** Compute the gradient
$$\nabla \ell = \nabla_\gamma \ell - \left(\nabla^2_{\mathbf{x},\gamma} \Phi\right)^\top \left(\nabla^2_{\mathbf{x},\mathbf{x}} \Phi\right)^{-1} \nabla_{\mathbf{x}} \ell$$

procedure HOAG($\{\epsilon^{(u)}, u = 1, 2, \ldots\}$, $\boldsymbol{\gamma}^{(0)}$, $\boldsymbol{x}^{(0)}$, $\boldsymbol{y}$)
  for u **do**=0,1,... ▷ Upper-level iteration counter
    $t = 0$ ▷ Lower-level iteration counter
    while $\|\hat{\boldsymbol{x}}(\boldsymbol{\gamma}^{(u)}) - \boldsymbol{x}^{(t)}(\boldsymbol{\gamma}^{(u)})\| \geq \epsilon^{(u)}$ **do**
      $\boldsymbol{x}^{(t+1)} = \Psi(\boldsymbol{x}^{(t)}; \boldsymbol{\gamma}^{(u)})$ ▷ Lower-level optimization step
      $t = t + 1$
    end while
    Compute gradient $\nabla_{\boldsymbol{x}} \ell(\boldsymbol{\gamma}^{(u)}; \boldsymbol{x}^{(t)})$ and
    Jacobian $\nabla_{\boldsymbol{x}\boldsymbol{\gamma}} \Phi(\boldsymbol{x}^{(t)}; \boldsymbol{\gamma}^{(u)})$
    Using CG, find $\boldsymbol{q}$ such that
    $\|\nabla_{\boldsymbol{x}\boldsymbol{x}} \Phi(\boldsymbol{x}^{(t)}; \boldsymbol{\gamma}^{(u)})\boldsymbol{q} - \nabla_{\boldsymbol{x}} \ell(\boldsymbol{\gamma}^{(u)}; \boldsymbol{x}^{(t)})\| \leq \epsilon^{(u)}$
    $\boldsymbol{g} = \nabla_{\boldsymbol{\gamma}} \ell(\boldsymbol{\gamma}^{(u)}; \boldsymbol{x}^{(t)}) - \left(\nabla_{\boldsymbol{x}\boldsymbol{\gamma}} \Phi(\boldsymbol{x}^{(t)}; \boldsymbol{\gamma}^{(u)})\right)' \boldsymbol{q}$
    $\boldsymbol{\gamma}^{(u+1)} = \boldsymbol{\gamma}^{(u)} - \frac{1}{L}\boldsymbol{g}$ ▷ L is a Lipschitz constant of $\nabla \ell(\boldsymbol{\gamma})$
  end for
  return $\boldsymbol{\gamma}^{(u+1)}$
end procedure

Fig. 1. Hyperparameter optimization with approximate gradient (Crockett and Fessler (2))

Where, the first term captures the indirect effect of γ on the outer loss through the inner solution while the second term represents the direct sensitivity of the inner objective to γ. Update γ using a projected gradient descent step:

$$\gamma \leftarrow \text{Project}(\gamma - \eta \nabla_\gamma \ell)$$

where η is the learning rate, and `Project` ensures γ remains within feasible bounds (e.g., non-negativity or upper limits).

Bilevel Joint learning pipeline

Let T^* be a pretrained task operator trained on the clean dataset. We consider a dataset (x^*, y, s), where x^* represents the ground truth data, y denotes the noisy measurements, and s represents the task labels. Let R_γ be a regularizer parameterized by γ. In our case, we use the smoothed total variation (Salehi et al. (4)) denoising regularizer. The choice of regularizer is problem-specific, and since we are addressing a denoising problem, the smoothed total variation regularizer is chosen for its ability to promote smoothness in the reconstructed data.

Inner Level Problem:

$$\hat{x}(\gamma) = \arg\min_y \|y - x^*\|_2^2 + R_\gamma(y)$$

This equation represents a denoising problem.

Outer Level Problem:

$$\hat{\gamma} = \arg\min_\gamma \ell(T^*(\hat{x}(\gamma)), s)$$

Where $\ell(\cdot, \cdot)$ is the loss function between the task operator output and the task labels. This pipeline can be optimized further by finetuning the pretrained task operator using the outer loop loss.

Experiments

Image denoising and segmentation using Cityscapes Dataset . We utilize the Cityscapes dataset ((5)) for segmentation tasks, applying preprocessing steps such as resizing, normalization, and data augmentation to prepare the data for training and validation. To implement the bilevel optimization framework, we designed several helper functions. We utilize the conjugate gradient method to efficiently solve systems involving the Hessian matrix, avoiding explicit computation and thereby reducing memory and computational overhead. Additionally, we calculate Hessian-vector products using PyTorch's `autograd`, which leverages automatic differentiation to approximate Hessian operations effectively, even for high-dimensional parameter spaces. The inner optimization incorporates regularization to enhance image reconstruction. We have used smoothed total variation (TV), which penalizes abrupt changes in pixel intensity to enforce spatial smoothness. The parameters of these regularizers are treated as hyperparameters that are optimized during the outer loop. Our implementation of the Hyperparameter Optimization via Approximate Gradients algorithm uses a bilevel optimization structure consisting of two levels:

Inner Optimization. The **inner optimization** reconstructs noisy images by minimizing a combination of reconstruction error and the smoothed total variation (TV) regularization. The objective is formulated as:

$$\hat{x}(\gamma) = \arg\min_y \left\{ \|y - x^*\|_2^2 + \text{Smoothed TV}(y, \gamma) \right\}$$

Where Smoothed TV(y) represents the smoothed total variation regularizer, which is defined as:

$$\text{Smoothed TV}(y, \gamma) = \exp(\gamma_0) \cdot \frac{1}{N} \sum_{i,j} \sqrt{\text{TV}(y)^2 + \exp(\gamma_1)^2}$$

where,

$$\text{TV}(y) = \sum_{i,j} \sqrt{(y_{i+1,j} - y_{i,j})^2 + (y_{i,j+1} - y_{i,j})^2}$$

Here, $y_{i,j}$ denotes the pixel intensity at position (i, j) in the image, and the TV regularizer penalizes abrupt changes in pixel intensities, promoting smooth transitions and preserving edges in the image. The Total Variation (TV) regularizer is a popular choice for image denoising and other inverse problems because it promotes piecewise smoothness. It penalizes sharp changes in intensity while preserving important edge information. However, the standard TV regularizer is non-differentiable, which can be problematic for optimization algorithms that rely on gradients. The smoothed TV regularizer addresses this non-differentiability. It approximates the absolute value function (used in standard TV) with a smooth function, typically something like a Huber function or a hyperbola. This makes it suitable for gradient-based optimization.

Outer Optimization. The **outer optimization** refines the hyperparameters (γ) by minimizing a composite segmentation loss. The objective is given by:

$$\hat{\gamma} = \min_\gamma \left\{ \mathcal{L}_{\text{segmentation}}(T^*(\hat{x}(\gamma)), \text{s}) \right\}$$

Where the segmentation loss $\mathcal{L}_{\text{segmentation}}$ is defined as:

$$\mathcal{L}_{\text{segmentation}} = L_{\text{CE}} + L_{\text{Dice}} + L_{\text{IoU}} + L_{\text{Log-Cosh-Dice}}$$

Where L_{CE} denotes Cross-entropy loss for pixel-wise classification accuracy, L_{Dice} denotes Dice loss to measure the overlap between predicted and ground truth masks, L_{IoU} denotes Intersection-over-union loss to improve segmentation quality, $L_{\text{Log-Cosh-Dice}}$ denotes Log-cosh Dice loss for robustness to large deviations. The outer optimization adjusts the hyperparameters to optimize for better task-specific performance, balancing reconstruction quality and segmentation accuracy.

HOAG with Smoothed TV and Fine-Tuning. In an extended implementation, we combine the smoothed TV regularizer with fine-tuning of specific layers in the segmentation model. After initializing the segmentation model with pretrained weights, we allow fine-tuning of the final two upsampling layers and the output layer, while freezing the remaining layers. This approach integrates the benefits of the smoothed TV regularizer for reconstruction and the flexibility of fine-tuning for task-specific adaptation. The Adam optimizer is used to train these layers during the outer optimization, ensuring improved performance in challenging segmentation scenarios. By alternating between inner reconstruction and outer fine-tuning steps, this configuration leverages both reconstruction-based priors and task-specific learning for superior results. In our experiments, we utilized a pretrained U-Net model as the task operator and generated a blurred dataset by introducing Gaussian noise with varying standard deviations. The training set comprised 100 samples of size 64x64, while the test set consisted of 500 samples. The inner optimization was solved for two tolerance values: 10^{-5} and 10^{-3}. The results obtained are presented in the table 1.

	Tolerance = 10^{-5}						Tolerance = 10^{-3}	
	$\sigma = 0.1$		$\sigma = 0.4$		$\sigma = 0.8$		$\sigma = 0.8$	
	dice	IOU	dice	IOU	dice	IOU	dice	IOU
T* on clean images	0.8539	0.7520	0.8539	0.7520	0.8539	0.7520	0.8539	0.7520
T* on noisy images	0.6504	0.4896	0.2662	0.1546	0.1764	0.0971	0.1754	0.0965
HOAG with smoothed TV	0.6786	0.5179	**0.8538**	**0.7518**	**0.8457**	**0.7398**	0.2079	0.1165
HOAG with Finetuning	**0.7239**	**0.5746**	0.8379	0.7292	0.8343	0.7239	**0.6213**	**0.4571**

Table 1. Segmentation using Cityscapes

Image denoising and classification using Stanford Dogs Dataset. We utilize the Stanford Dogs dataset (6) for the classification task, applying preprocessing steps such as resizing, normalization, and data augmentation to prepare the data for training and validation. This dataset has over 20,580 images with 120 breeds of dogs. We have used an off-the-shelf Pytorch resnet50 model as a baseline.

Inner Optimization. The **inner optimization** reconstructs noisy images by minimizing a combination of reconstruction error and the smoothed total variation (TV) regularization. The objective is formulated as:

$$\hat{x}(\gamma) = \arg\min_{y} \left\{ \|y - x^*\|_2^2 + \text{Smoothed TV}(y, \gamma) \right\}$$

Where Smoothed TV(x) represents the smoothed total variation regularizer.

	Tolerance = 10^{-5}			Tolerance = 10^{-3}		
	$\sigma = 0.1$	$\sigma = 0.4$	$\sigma = 0.8$	$\sigma = 0.1$	$\sigma = 0.4$	$\sigma = 0.8$
T* on clean images	48.8	48.8	48.8	48.8	48.8	48.8
T* on noisy images	16.6	5.2	2.4	15.2	4.2	2.4
HOAG with smoothed tv	20.2	**46**	**46.4**	13.3	19.5	34.6
HOAG with finetuning	**37.0**	39.6	39.8	23.2	27.2	**36.2**

Table 2. Classification using Stanford Dogs

Outer Optimization. The **outer optimization** refines the hyperparameters (γ) by minimizing categorical crossentropy loss. The objective is given by:

$$\hat{\gamma} = \min_{\gamma} \left\{ \mathcal{L}_{\text{crossentropy}}(T^*(\hat{x}(\gamma)), \text{s}) \right\}$$

The training set comprised 100 samples of size 64x64, while the test set consisted of 500 samples. The inner optimization was solved for two tolerance values: 10^{-5} and 10^{-3}. The results obtained are presented in the table 2.

Observations

From the results obtained, we observe that the pretrained model performs well on clean images but is highly sensitive to noise, with performance degrading significantly as noise levels increase. Bilevel pipeline mitigate this issue effectively, with Smoothed TV excelling in scenarios with moderate to high noise, achieving results comparable to clean data. Fine-tuning, on the other hand, performs better at lower noise levels and shows substantial improvement under higher tolerance conditions. Overall, Smoothed TV offers consistent robustness across noise levels, while Fine-tuning proves advantageous in specific cases, particularly at higher tolerances or when noise is less severe.

Future Work

Based on the preliminary results, this approach demonstrates significant potential for joint learning. Our ultimate goal is to extend this methodology to medical image reconstruction and associated task-specific pipelines. Future improvements could include the incorporation of neural networks as regularizers, offering greater flexibility and representational power. Additionally, exploring more robust and efficient bilevel optimization algorithms could enhance the computational performance of the framework. A long-term objective is to develop a generalizable pipeline where only the task-specific losses need to be defined, enabling seamless adaptation to diverse applications.

Bibliography

1. Jonas Adler, Sebastian Lunz, Olivier Verdier, Carola-Bibiane Schönlieb, and Ozan Öktem. Task adapted reconstruction for inverse problems. *Inverse Problems*, 38(7):075006, May 2022. ISSN 1361-6420. doi: 10.1088/1361-6420/ac28ec.
2. Caroline Crockett and Jeffrey A. Fessler. Bilevel methods for image reconstruction. *Foundations and Trends® in Signal Processing*, 15(2–3):121–289, 2022. ISSN 1932-8354. doi: 10.1561/2000000111.
3. Fabian Pedregosa. Hyperparameter optimization with approximate gradient, 2022.
4. Mohammad Sadegh Salehi, Subhadip Mukherjee, Lindon Roberts, and Matthias J. Ehrhardt. An adaptively inexact first-order method for bilevel optimization with application to hyperparameter learning, 2024.
5. Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

6. Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.