Deceptron: Learned Local Inverses for Fast and Stable Physics Inversion[*]

Aaditya L. Kachhadiya
Independent Researcher
Surat, India
kachhadiyaaaditya@gmail.com

Abstract

Inverse problems in the physical sciences are often ill-conditioned in input space, making progress step-size sensitive. We propose the *Deceptron*, a lightweight bidirectional module that learns a local inverse of a differentiable forward surrogate. Training combines a supervised fit, forward–reverse consistency, a lightweight spectral penalty, a soft bias tie, and a *Jacobian Composition Penalty* (JCP) that encourages $J_g\big(f(x)\big) J_f(x) \approx I$ via JVP/VJP probes. At solve time, D-IPG (Deceptron Inverse-Preconditioned Gradient) takes a descent step in output space, pulls it back through g, and projects under the same backtracking and stopping rules as baselines. On Heat-1D initial-condition recovery and a Damped Oscillator inverse problem, D-IPG reaches a fixed normalized tolerance with $\sim 20\times$ fewer iterations on Heat and ~ 2–$3\times$ fewer on Oscillator than projected gradient, competitive in iterations and cost with Gauss–Newton. Diagnostics show JCP reduces a measured composition error and tracks iteration gains. We also preview a single-scale 2D instantiation, *DeceptronNet (v0)*, that learns few-step corrections under a strict fairness protocol and exhibits notably fast convergence.

1 Introduction

Recovering unknown inputs or parameters from indirect, noisy measurements is central to PDE inversion, system identification, and imaging. A common approach minimizes a data misfit in input space with projections enforcing physics constraints. Such objectives are often ill-conditioned; gradients are poorly scaled and many iterations are required. We propose the **Deceptron**, a simple alternative: learn a local inverse of a differentiable forward surrogate and use it to precondition inverse updates. A small bidirectional module parameterizes a forward map f and a reverse map g; training stabilizes g as a local inverse via a JVP/VJP-based penalty on $J_g(f(x)) J_f(x) \approx I$. At inference, D-IPG takes a residual step in output space, pulls it back through g, then projects with the same Armijo backtracking used by baselines [1]. This preserves a standard projected loop and changes only the update direction. Beyond this single-module preconditioning, we outline a broader agenda: a *DeceptronNet* family tailored to inverse problems. As a first step, we introduce a single-scale 2D unrolled variant (v0) that maps nominal residual features to image-space corrections in a few learned steps. We compare Deceptron primarily against Gauss–Newton (GN) and gradient descent in the x-space (X-GD). Here, X-GD denotes plain gradient descent updates directly on x, while D-IPG refers to updates carried out in the data space y and then pulled back through g.

Related efforts include physics-informed training [2, 3], learned unrolling and proximal priors [4–6]. Classical inverse methods like Gauss–Newton/LM [7, 8] guide our comparison; for Hessian–vector products we follow Pearlmutter [9].

---

[*]The naming "Deceptron" reflects its contrast with the forward-mapping perceptron.
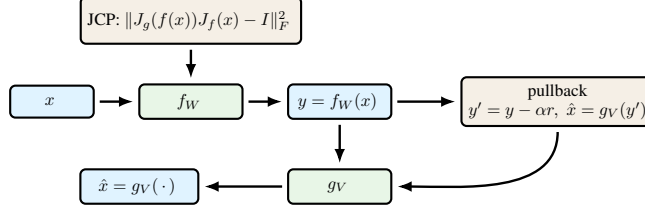
Figure 1: Deceptron: forward f_W and reverse g (instantiated as learned g_V by default, or $g_{W^\top}$ if tied) with JCP; inference pulls output-space residuals back through g.

2   Method

Let $x \in \mathbb{R}^{d_{\text{in}}}$ and $y \in \mathbb{R}^{d_{\text{out}}}$. The Deceptron defines

$$f_W(x) = \sigma(Wx + b), \qquad g_V(y) = \tilde{\sigma}(Vy + c),$$

where W, b, V, c are learned parameters and $\sigma, \tilde{\sigma}$ are lightweight activation functions (e.g., leaky). The matrices V and $W^\top$ are not tied so that g can act as a local inverse even when W is non-orthogonal. Stabilization terms include $\|W^\top W - I\|_F^2$, a soft bias tie $\|b + c\|_2^2$, and optionally $\|VW - I\|_F^2$.

Given pairs (x, y^*) from a differentiable surrogate, the loss is

$$\begin{aligned}
\mathcal{L} = {}& \lambda_{\text{task}} \|f_W(x) - y^*\|^2 + \lambda_{\text{rec}} \|g_V(f_W(x)) - x\|^2 + \lambda_{\text{cyc}} \|f_W(g_V(\tilde{y})) - \tilde{y}\|^2 \\
& + \beta_{\text{spec}} \|W^\top W - I\|_F^2 + \lambda_{\text{tie}} \|b + c\|_2^2 + \lambda_{\text{comp}} \|VW - I\|_F \\
& + \lambda_{\text{JCP}} \, \mathbb{E}_\xi \|J_g(f_W(x)) J_f(x) \xi - \xi\|^2.
\end{aligned} \tag{1}$$

Here $\tilde{y}$ denotes measurement-space samples such as $f_W(x)$ or noised variants. With probes $\mathbb{E}[\xi \xi^\top] = I$ (Rademacher or Gaussian, one to four per batch), the identity $\mathbb{E}\|(A - I)\xi\|^2 = \|A - I\|_F^2$ ensures that the JCP term estimates $\|J_g(f(x)) J_f(x) - I\|_F^2$ using only a few JVP/VJP products.

We minimize $\Phi(x) = \frac{1}{2}\|f_W(x) - y^*\|^2$ in normalized output space. At iteration t, let $y_t = f_W(x_t)$ and $r_t = y_t - y^*$. The update is

$$y_{t+1}^{\text{prop}} = y_t - \alpha r_t, \qquad x_{t+1}^{\text{prop}} = g_V(y_{t+1}^{\text{prop}}), \qquad x_{t+1} = \Pi_{\mathcal{C}}\big((1 - \rho)x_t + \rho \, x_{t+1}^{\text{prop}}\big),$$

accepted under the shared Armijo and projection rule. To first order, $g(y_t - \alpha r_t) \approx x_t - \alpha \, J_g(f(x_t)) r_t$. If $J_f(x_t)$ has full column rank and $J_g(f(x_t)) \approx J_f(x_t)^+ = (J^\top J)^{-1} J^\top$, then D-IPG matches Gauss–Newton up to the scalar step size α; as $\|J_g(f(x)) J_f(x) - I\| \to 0$, the updates converge to Gauss–Newton scaled by α [7, 8]. Limitations include locality and surrogate fidelity (see Appendix D).

We monitor the runtime Jacobian composition error $\text{RJCP}(x) = \mathbb{E}_\xi \|J_g(f(x)) J_f(x) \xi - \xi\|^2$, an unbiased estimator of $\|J_g(f(x)) J_f(x) - I\|_F^2$ obtained via Hutchinson's identity. $\text{RJCP}(x) = 0$ if and only if $J_g(f(x)) J_f(x) = I$, meaning g is a local left inverse at x. Lower values of RJCP empirically correlate with fewer iterations (Fig. 3b).

All algorithms are listed in the Appendix.[2]

3   Experiments

We evaluate the Deceptron inverse-preconditioned gradient (D-IPG) on two standard inverse problems: Heat-1D initial-condition recovery and Damped Oscillator parameter and initial-condition estimation. Outputs are z-scored, and all algorithms share the same normalized loss space ($\varepsilon = 0.30$). This keeps the line search and stopping policy comparable across methods even when the raw scales of y differ by task. Final RMSE values in Tables 1 to 3 are reported in unnormalized units to reflect physical error.

All solvers follow an identical fairness protocol. The same projector $\Pi_{\mathcal{C}}$ and Armijo rule with $c = 10^{-4}$ (up to eight halvings) are used for all methods [1]. Relaxation is fixed to $\rho = 0.4$, and the

---

[2]See pseudocode in Algorithms 1–4.

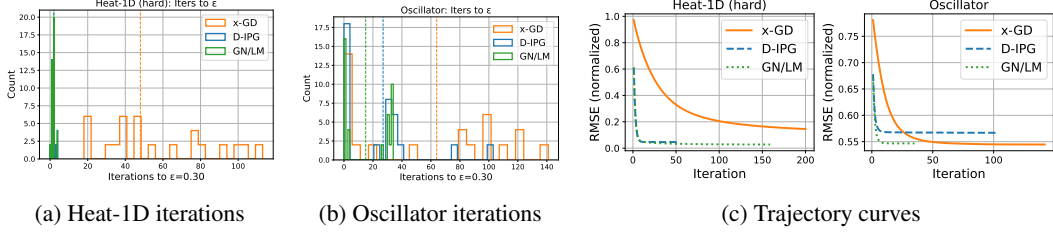| (a) Heat-1D iterations	(b) Oscillator iterations	(c) Trajectory curves

Figure 2: Iteration distributions and convergence trajectories across problems. The right panel includes both Heat-1D and Oscillator RMSE curves (mean normalized RMSE).

same initial step size 1.0 is used for each optimizer (X-GD η, D-IPG α, GN/LM α). The maximum iteration count is 200. Heat-1D begins from a zero initial condition, while the oscillator starts from a mid-range parameter vector. Backtracking evaluations of f are shared to isolate only the effect of update direction. No proximal or smoothing heuristics are used, and all runs are deterministic with fixed seeds.

Figure 2 compares iteration counts and normalized trajectories under the shared policy. On Heat-1D, D-IPG and GN/LM concentrate at very low iteration counts while X-GD is widely spread, indicating sensitivity to poor conditioning in x-space. The trajectory panel shows that all methods eventually reduce the normalized residuals, but the preconditioned directions of D-IPG and the second-order curvature of GN/LM reach the tolerance in a few steps. On the oscillator problem, GN/LM has a slight iteration edge over D-IPG, which is consistent with its access to explicit Hessian information, yet the per-iteration cost of GN/LM is higher due to inner linear solves. The separation between methods therefore reflects both direction quality and compute per step.

Table 1: Iterations-to-ε (mean±std), final RMSE, and acceptance rate (acc).

	x-GD			D-IPG			GN/LM		
Setting	it	RMSE	acc	it	RMSE	acc	it	RMSE	acc
Heat-1D (hard)	58.2±28.9	0.045	1.00	2.8±1.0	0.010	0.58	2.8±0.9	0.009	0.97
Oscillator	58.2±52.1	0.356	1.00	24.6±27.2	0.368	0.64	17.3±15.7	0.353	0.69

Table 1 summarizes averages over trials (unnormalized final RMSE). The acceptance rate is lower for D-IPG on Heat-1D because its proposals are larger; Armijo reduces the step a few times before acceptance, which is expected under stronger preconditioning and does not indicate instability. The final RMSE values in original units are comparable between D-IPG and GN/LM and significantly better than X-GD on Heat-1D. On the oscillator problem, D-IPG has more variability in iteration counts, but still outperforms X-GD and approaches GN/LM.

Table 2: Heat-1D (ε=0.30). Median [IQR] iters, success, ms/iter, and mean time-to-ε.

Method	iters [IQR]	Success	ms/iter	Time (s)
x-GD	49.0 [38.2, 80.0]	1.00	0.43	0.026
D-IPG	3.0 [2.0, 3.0]	1.00	0.51	0.001
GN/LM	3.0 [2.0, 3.0]	1.00	3.82	0.011

Table 3: Oscillator (ε=0.30). Median [IQR] iters, success, ms/iter, and mean time-to-ε.

Method	iters [IQR]	Success	ms/iter	Time (s)
x-GD	65.0 [1.0, 104.5]	0.50	0.45	0.004
D-IPG	28.0 [1.0, 34.0]	0.45	1.28	0.001
GN/LM	16.5 [1.0, 33.2]	0.50	4.22	0.007

The median summaries make the tradeoff clear. D-IPG matches GN/LM in iteration counts on Heat-1D but has much lighter iterations, resulting in shorter mean time-to-tolerance. On the oscillator, GN/LM reduces iteration counts further but pays a higher cost per step, whereas D-IPG retains a good balance between direction quality and compute. This is consistent with a Gauss–Newton–like

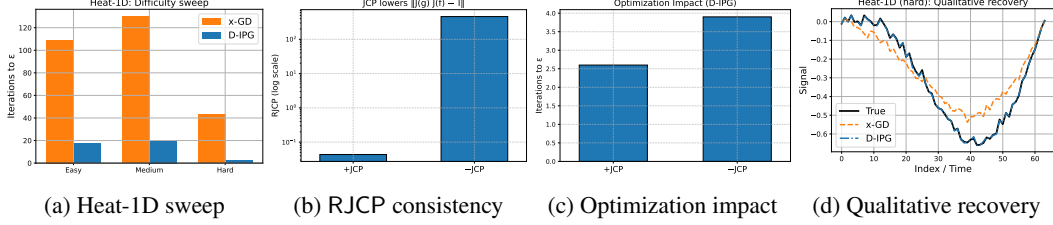| (a) Heat-1D sweep | (b) RJCP consistency | (c) Optimization impact | (d) Qualitative recovery |

Figure 3: Scaling and ablations. D-IPG remains stable under increasing Heat-1D difficulty, JCP lowers composition error and iteration count, and final reconstructions confirm accuracy.

direction from D-IPG when $J_g(f)J_f$ is close to identity, but without solving linear systems every iteration.

We next study how convergence changes as the inverse problem becomes more difficult. In the Heat-1D difficulty sweep of Figure 3, both X-GD and D-IPG require the most iterations at the medium setting, reflecting increased curvature and noise. Across all regimes, however, D-IPG consistently converges in far fewer steps and shows the largest relative gain at the hard setting, roughly an order of magnitude fewer iterations than X-GD. This indicates that once the learned inverse stabilizes, D-IPG retains efficiency even as the forward problem becomes more nonlinear, whereas X-GD remains sensitive to scale and conditioning. The same figure also reports two JCP consistency tests and a qualitative recovery. Enabling JCP reduces the composition residual RJCP $= \mathbb{E}_\xi \|J_g(f)J_f\,\xi - \xi\|^2$ by several orders of magnitude, confirming near-inverse behavior of the learned maps and yielding fewer iterations under identical Armijo and projection rules. Together, these results show that lowering composition error directly translates into faster convergence.

Quantitatively, enabling JCP reduces RJCP by several orders of magnitude and shortens convergence. With JCP active on Heat-1D, the method reaches tolerance in about 2.6 iterations with final RMSE 0.007; disabling JCP raises the composition residual to 457.7 and requires roughly 3.8 iterations with higher error. Tying $V = W^\top$ removes degrees of freedom in the reverse map and degrades conditioning, while removing reconstruction and cycle terms does not harm convergence, indicating that the preconditioning effect is driven by the local inverse property rather than the auxiliary reconstruction losses.

We also track RJCP(x) during training as a runtime diagnostic. It decreases steadily as the reverse map stabilizes, aligning with validation error and indicating that improved composition $J_g(f)J_f$ yields better-scaled updates and faster convergence.
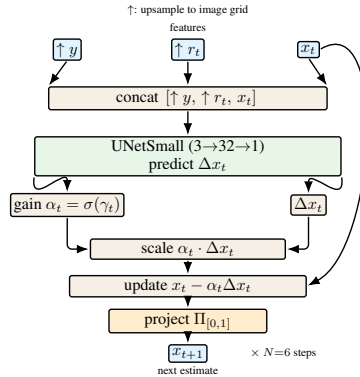
4 Scalability and Discussion

Figure 4: DeceptronNet v0. A compact unrolled corrector using measurement and residual features.

DeceptronNet v0 is a lightweight unrolled corrector that refines images in a few steps using measurement and residual features. At step t, inputs $F_t = [\uparrow y, \uparrow r_t, x_t]$ with $r_t = A_{\text{nom}}(x_t) - y$ pass through a compact U-Net (UNetSmall, $3{\to}32{\to}1$) predicting Δx_t. A learnable gain $\alpha_t = \sigma(\gamma_t) \in (0, 1)$

4

scales the update, and the iterate advances as $x_{t+1} = \Pi_{[0,1]}(x_t - \alpha_t \Delta x_t)$. Depth is fixed at $N{=}6$, initialized with $x_0 ={\uparrow} y$.

Training minimizes image error plus measurement consistency under A_{true} (blur, mild nonlinearity, downsample, Poisson-like noise), without additional regularizers. For fairness, all methods share initialization, clamping, residual-based stopping ($0.3r_0$), iteration budget (80), and Armijo backtracking for the baseline solvers.

Table 4: 2D PSF results (mean over test set). Evaluated on A_{true} with the same residual-based stopping rule. DNet reaches the target tolerance in a fixed, small number of learned steps, while LM and x-GD require many backtracked iterations.

Method	Mean iterations to stop	Mean image RMSE
LM (true model)	69.25	0.0883
x-GD (true model)	80.00	0.1271
DNet v0 (unrolled $N{=}6$)	6.00	0.0640

DNet v0 converges rapidly under the same fairness conditions, reaching the error threshold in a small, fixed number of learned steps (Table 4). This single-scale prototype demonstrates that amortized curvature and bounded updates can yield predictable convergence with minimal computation. While limited to simulated degradations, it forms the foundation for upcoming multi-scale and real-data variants. Its design emphasizes three aspects: (i) amortized curvature from residual features, (ii) stability through bounded gains and skip paths, and (iii) predictable compute from fixed iteration depth. Extending this framework to multi-scale operators, more realistic noise, and broader physical models is a natural next step.

(a) Hard real-image inverse task (Kodak24).

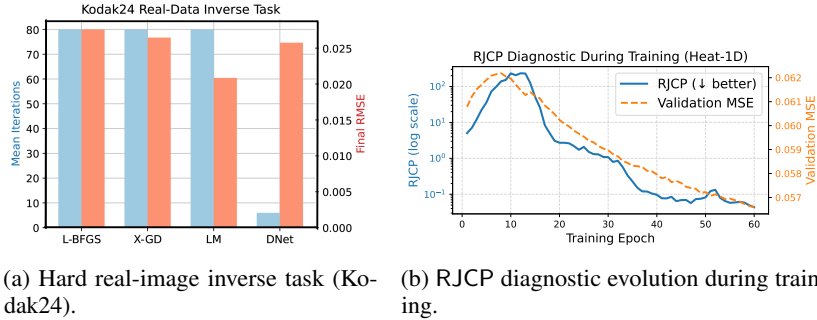(b) RJCP diagnostic evolution during training.

Figure 5: Scalability and diagnostics. DNet remains stable under harder real-data settings (Kodak24), while D-IPG shows decreasing RJCP throughout training, indicating improved local invertibility.

On the Kodak24 inverse task, all baselines use the same projection and Armijo backtracking. L-BFGS, LM, and x-GD converge slowly, while DNet reaches the residual threshold within six updates with competitive RMSE. Despite operating under noisy, downsampled conditions, the learned corrector maintains stability and reproducibility, showing that amortized updates generalize beyond synthetic settings. For the Deceptron (D-IPG), the RJCP diagnostic further reveals the mechanism behind this robustness. As training progresses, RJCP decreases steadily alongside validation error, confirming that the JCP indeed encourages the reverse map to act as a local left inverse rather than serving as a generic regularizer. This diagnostic correlates strongly with convergence speed and can be monitored at inference time to detect when surrogates fall outside their valid regime.

Our two prototypes, Deceptron and DeceptronNet (v0), highlight complementary philosophies. D-IPG enforces a learned local inverse through the Jacobian composition penalty, providing principled conditioning and interpretability, while DNet amortizes curvature through residual features, achieving fast, fixed-depth correction. Together they mark a first step toward lightweight, learned correctors for physical inverse problems. Faster, better-conditioned solvers can reduce compute cost and enable larger parameter sweeps in scientific pipelines such as imaging or system identification. However, misuse of learned surrogates outside their validity domain can yield overconfident reconstructions; we recommend explicit reporting of surrogate ranges and RJCP metrics in future learned-inverse work.

Code Availability

All code, configuration files, and reproducibility notebooks for the Deceptron and DeceptronNet experiments are publicly available at:
`https://github.com/aadityakachhadiya/deceptron-ml4ps2025`

Acknowledgements

Funding Statement

LLM Disclosure

All core ideas, formulations, experiments, and writing structure were conceived and implemented by the author. Large Language Models (e.g., ChatGPT) were used for limited assistance in code debugging, LaTeX formatting, and minor language polishing. No model generated original research ideas or results.

References

[1] L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pac. J. Math.*, 16(1):1–3, 1966.

[2] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.*, 378:686–707, 2019.

[3] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang. Physics-informed machine learning. *Nat. Rev. Phys.*, 3(6):422–440, 2021.

[4] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010.

[5] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg. Plug-and-play priors for model based reconstruction. In *IEEE Global Conf. on Signal and Information Processing (GlobalSIP)*, pages 945–948, 2013.

[6] Y. Romano, M. Elad, and P. Milanfar. The little engine that could: Regularization by denoising (red). *SIAM J. Imaging Sci.*, 10(4):1804–1844, 2017.

[7] K. Levenberg. A method for the solution of certain problems in least squares. *Q. Appl. Math.*, 2 (2):164–168, 1944.

[8] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM J. Appl. Math.*, 11(2):431–441, 1963.

[9] B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Comput.*, 6(1):147–160, 1994.

A Extra Results and Supporting Algorithms

This appendix compiles the supporting algorithms, diagnostic plots, and theoretical derivations complementing the main text. While not essential for reproducing the core experiments, these details clarify implementation, interpretability, and the underlying optimization behavior of the proposed methods.

A.1 Optimization Algorithms

All solvers share the same fairness protocol: identical projection $\Pi_\mathcal{C}$, Armijo parameter $c=10^{-4}$ (up to eight halvings), relaxation $\rho=0.4$, and identical initialization and stopping rules. The following pseudocode specifies each update exactly as used in experiments.

Algorithm 1 D-IPG (Deceptron Inverse-Preconditioned Gradient) with shared Armijo and relaxation

---

1: **Inputs:** surrogate f_W, reverse g_V, projector $\Pi_\mathcal{C}$, target y^*, init x_0, step α_0, relaxation ρ, Armijo c, halvings H, tolerance ε
2: **for** $t = 0, 1, \ldots$ **do**
3:      $y_t \leftarrow f_W(x_t)$; $r_t \leftarrow y_t - y^*$; $\Phi_t \leftarrow \frac{1}{2}\|r_t\|^2$
4:      compute $\nabla\Phi(x_t)$ by reverse-mode AD
5:      $\alpha \leftarrow \alpha_0$; accepted $\leftarrow$ **false**
6:      **for** $h = 0, \ldots, H$ **do**
7:          $y_{\text{prop}} \leftarrow y_t - \alpha\, r_t$; $x_{\text{prop}} \leftarrow g_V(y_{\text{prop}})$
8:          $p \leftarrow x_{\text{prop}} - x_t$
9:          $x_{\text{trial}} \leftarrow \Pi_\mathcal{C}\big((1-\rho)x_t + \rho(x_t+p)\big)$
10:          $\Phi_{\text{trial}} \leftarrow \frac{1}{2}\|f_W(x_{\text{trial}}) - y^*\|^2$; $g^\top p \leftarrow \nabla\Phi(x_t)^\top p$
11:          **if** $\Phi_{\text{trial}} \leq \Phi_t + c\,\rho\,g^\top p$ **then**
12:             $x_{t+1} \leftarrow x_{\text{trial}}$; accepted $\leftarrow$ **true**
13:             **break**
14:          **else**
15:             $\alpha \leftarrow \alpha/2$
16:          **end if**
17:      **end for**
18:      **if not** accepted **then**
19:          **break**
20:      **end if**
21:      stop if normalized residual $\leq \varepsilon$
22: **end for**

Algorithm 2 x-GD (projected gradient) with shared Armijo and relaxation

---

1: **Inputs:** f_W, $\Pi_\mathcal{C}$, y^*, x_0, step η_0, ρ, c, H, ε
2: **for** $t = 0, 1, \ldots$ **do**
3:      compute $\nabla\Phi(x_t)$ for $\Phi(x) = \frac{1}{2}\|f_W(x) - y^*\|^2$; set $\eta \leftarrow \eta_0$
4:      **for** $h = 0, \ldots, H$ **do**
5:          $p \leftarrow -\eta\,\nabla\Phi(x_t)$
6:          $x_{\text{trial}} \leftarrow \Pi_\mathcal{C}\big((1-\rho)x_t + \rho(x_t+p)\big)$
7:          **if** $\Phi(x_{\text{trial}}) \leq \Phi(x_t) + c\,\rho\,\nabla\Phi(x_t)^\top p$ **then**
8:             $x_{t+1} \leftarrow x_{\text{trial}}$
9:             **break**
10:          **else**
11:             $\eta \leftarrow \eta/2$
12:          **end if**
13:      **end for**
14:      stop if normalized residual $\leq \varepsilon$
15: **end for**

Algorithm 3 GN/LM (damped Gauss–Newton with CG) under shared projector, relaxation, and Armijo

---

1: **Inputs:** f_W, $\Pi_\mathcal{C}$, y^*, x_0, step α_0, ρ, c, H, damping λ, CG iters K
2: **for** $t = 0, 1, \ldots$ **do**
3:     $r \leftarrow f_W(x_t) - y^*$; $b \leftarrow J^\top r$ via VJP; define $v \mapsto (J^\top J + \lambda I)v$ via JVP+VJP
4:     solve $(J^\top J + \lambda I)\Delta x = -b$ by K CG iterations
5:     $\alpha \leftarrow \alpha_0$
6:     **for** $h = 0, \ldots, H$ **do**
7:         $x_{\text{trial}} \leftarrow \Pi_\mathcal{C}\big((1-\rho)x_t + \rho(x_t + \alpha\Delta x)\big)$
8:         **if** $\Phi(x_{\text{trial}}) \leq \Phi(x_t) + c\,\rho\,\nabla\Phi(x_t)^\top(\alpha\Delta x)$ **then**
9:             $x_{t+1} \leftarrow x_{\text{trial}}$
10:           **break**
11:         **else**
12:             $\alpha \leftarrow \alpha/2$
13:         **end if**
14:     **end for**
15: **end for**

Algorithm 4 DeceptronNet v0 (single-scale; unrolled $N=6$)

1: **Inputs:** measurement y, nominal A_{nom}, current x_t, gain $\alpha_t = \sigma(\gamma_t)$
2: $r_t \leftarrow A_{\text{nom}}(x_t) - y$; $F_t \leftarrow [\uparrow y, \uparrow r_t, x_t]$
3: $\Delta x_t \leftarrow \texttt{UNetSmall}(F_t)$
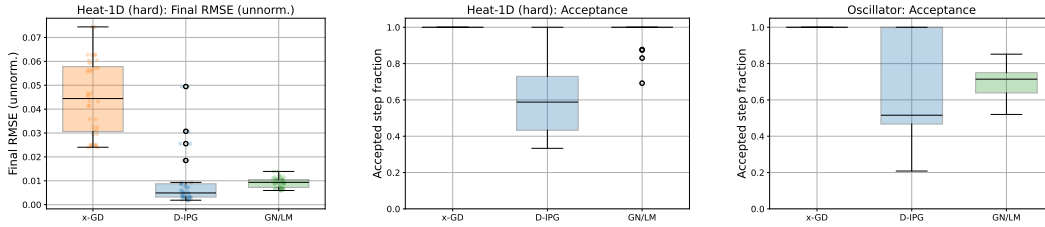4: $x_{t+1} \leftarrow \text{clip}_{[0,1]}(x_t - \alpha_t\,\Delta x_t)$

B Extra Results

Figure 6: Additional box plots. Left: Heat-1D RMSE (unnormalized). Middle/Right: acceptance ratios for Heat-1D and Oscillator. Note that acc< 1 reflects larger proposed moves under the shared Armijo rule and does *not* indicate instability.

Table 5: Ablation study under X-GD vs D-IPG. The JCP term enhances local conditioning and convergence speed without affecting runtime cost.

Ablation	Method	Iters (mean±std)	Final RMSE	Acceptance
$-$JCP	X-GD	56.3±37.0	0.0436	1.000
	D-IPG	3.25±1.18	0.0159	0.606
$V = W^\top$	X-GD	106.9±41.3	0.0444	1.000
	D-IPG	16.2±8.55	0.0894	0.061
$-$rec/$-$cycle	X-GD	44.8±21.9	0.0361	1.000
	D-IPG	2.60±0.92	0.0086	0.595
$-$comp	X-GD	44.7±21.9	0.0365	1.000
	D-IPG	2.60±0.92	0.0070	0.570

Table 6: Core per-iteration operations (excluding shared backtracking f-evaluations). The Jacobian Composition Penalty (JCP) acts only at training time and adds no runtime cost.

Method	Operations per iteration
x-GD	one reverse-mode gradient $\nabla_x \Phi$
D-IPG	one reverse-mode grad (for Armijo) $+ f + g$
GN/LM	solve $(J^\top J + \lambda I)\Delta x = -J^\top r$ by CG; each CG $\approx$ 1 JVP + 1 VJP

Table 7: Kodak24 inverse task. Quantitative RMSE, iteration count, and mean wall-time per sample. The σ-Mismatch test (train $\sigma=4.0$, eval $\sigma=3.0$) probes robustness under surrogate–data noise shift. DNet maintains low error and fixed-step convergence.

Setting	Method	RMSE	Iters	Time (s)
	L-BFGS	0.0276	80	0.173
	X-GD	0.0265	80	0.0054
Normal ($\sigma=3.0$)	LM	0.0209	80	0.0058
	DNet	0.0258	6	0.0067
	L-BFGS	0.0604	100	0.181
	X-GD	0.0589	100	0.0063
Hard ($\sigma=4.0$)	LM	0.0550	100	0.0064
	DNet	0.0575	6	0.0046
	L-BFGS	0.0548	100	0.180
	X-GD	0.0535	100	0.0063
σ-Mismatch (train 4.0, eval 3.0)	LM	0.0488	100	0.0062
	DNet	0.0525	6	0.0045

Here σ denotes the surrogate's assumed observation-noise level during training and evaluation. A higher σ corresponds to a noisier forward model used for generating synthetic measurements.

B.1 Relation to Gauss–Newton

The classical Gauss–Newton (GN) method solves least-squares inverse problems using $d_{\mathrm{GN}} = -(J^\top J + \lambda I)^{-1}J^\top r$, where J is the Jacobian of the forward model. Our D-IPG update, $d_{\mathrm{D\text{-}IPG}} = -\alpha B^{-1}J^\top r$, replaces this curvature matrix with a learned preconditioner B. When B approximates $J^\top J + \lambda I$, the two directions become nearly identical, and convergence behavior matches that of GN up to a small scaling factor. The Jacobian Composition Penalty (JCP) encourages this alignment by enforcing $J_g(f(x))J_f(x) \approx I$, and the runtime diagnostic RJCP measures how close this condition holds. Lower RJCP therefore indicates stronger curvature alignment and faster, more stable optimization.

Deviation from GN (local, range-restricted). Let $J = J_f(x_t)$ denote the Jacobian of the forward map at the current iterate. Assume J has full column rank in a neighborhood of the solution. We write the learned reverse Jacobian as $J_g(f(x_t)) = J^+ + E_g$, where $J^+ = (J^\top J)^{-1}J^\top$ is the Moore–Penrose pseudoinverse and E_g represents the residual error of the learned local inverse. For the component of the residual r that lies in $\mathrm{range}(J)$ (i.e., near a solution where $r \approx Ju$ for some u), the D-IPG and Gauss–Newton steps are approximately $\Delta x_{\mathrm{dipg}} = -\alpha\, J_g(f(x_t))\, r$ and $\Delta x_{\mathrm{GN}} = -J^+ r$, respectively. Their difference satisfies

$$\|\Delta x_{\mathrm{dipg}} - \Delta x_{\mathrm{GN}}\| \;\le\; \alpha\, \frac{\|J_g J - I\|_2}{\sigma_{\min}(J)}\, \|r\|$$

where $\|\cdot\|_2$ denotes the spectral norm and $\sigma_{\min}(J)$ is the smallest singular value of J. Thus, as the JCP target $\|J_g J - I\|_2 \to 0$, the D-IPG direction converges to the Gauss–Newton direction up to the scalar step size α. The Jacobian Composition Penalty (JCP) enforces this alignment during training but incurs no additional runtime cost during inference.

C Jacobian Composition Penalty: Diagnostics

Recall the runtime Jacobian composition error (RJCP) from the main text,

$$\mathrm{RJCP}(x) = \mathbb{E}_\xi \| J_g(f(x)) J_f(x) \xi - \xi \|^2,$$

an unbiased estimator of $\| J_g(f(x)) J_f(x) - I \|_F^2$ via Hutchinson's identity. RJCP measures how well the learned reverse map g acts as a local left inverse of f: $\mathrm{RJCP}(x) = 0$ if and only if $J_g(f(x)) J_f(x) = I$. Lower values indicate near-unit scaling and low cross-coupling, corresponding to well-conditioned updates, while larger values signal mis-scaling or axis mixing.

Computation is efficient: with k probes ξ_j, compute $v_j = \mathrm{JVP}_{f_W}(x; \xi_j)$, then $u_j = \mathrm{JVP}_g(y; v_j)$ at $y = f_W(x)$, and accumulate $\| u_j - \xi_j \|^2$. Averaging over $j = 1..k$ yields $\mathrm{RJCP}(x)$. This requires only JVP/VJP products, no explicit Jacobians, and costs $\mathcal{O}(k)$ forward/adjoint passes. In practice, $k = 2$–4 probes suffice.

During training, RJCP also appears as a weighted penalty (JCP) to shape g toward acting as a left inverse. Reductions in RJCP across epochs correlate with fewer iterations to reach tolerance, while plateaus at high values typically reflect an unstable surrogate, an overly strong or early JCP weight, or severe non-identifiability. At evaluation, we log RJCP as a scalar diagnostic and observe that monotone decreases track improved iteration counts.

Interpretation is straightforward: low RJCP corresponds to well-scaled, stable steps; moderate values to partial conditioning; and high values to weak or unstable preconditioning. RJCP is diagnostic only and does not imply global invertibility, but it is informative within the surrogate's validity region. In highly non-identifiable regimes (multiple x yielding similar y), RJCP cannot resolve global ambiguity but still reflects local conditioning.

Implementation is lightweight: normalize outputs for the objective and line search, warm up the JCP weight after the forward fit stabilizes, use $k = 2$–4 Rademacher probes per batch, apply only JVP/VJP products, keep the spectral term on $W^\top W$ modest, avoid tying $V = W^\top$, and apply Armijo/backtracking consistently with baselines. If RJCP remains high, delay JCP warm-up, improve the surrogate near initialization, modestly tune α or λ_{JCP}, increase probes slightly, or switch to projected RJCP in under-determined regimes.

D Limitations of Deceptron

Deceptron relies on a reasonably accurate surrogate model, locality of linearization, and identifiable structure; outside these regimes its benefits diminish. If the surrogate is poorly fitted or strongly rank-deficient, corrective updates become unstable and RJCP remains persistently high. In highly non-identifiable problems, where many solutions map to similar measurements, the method cannot resolve global ambiguity and only improves conditioning locally.

The approach further assumes that the constraint projector preserves most of the proposed update; if projections dominate, effective progress is lost. Performance is also sensitive to the scheduling of step-size gains α_t and to the timing and weighting of the JCP term, which may require manual tuning for stability. The deliberately lightweight network improves efficiency but limits expressivity, making the method less competitive in problems that require strong nonlinearities, long-range dependencies, or nonlocal priors. In such settings, richer architectures may achieve higher fidelity at the cost of speed.

Finally, Deceptron is intended as a corrective accelerator rather than a full solver replacement, with the largest gains when the surrogate provides a useful local model. This motivates the **DeceptronNet variant for 2D (and higher-dimensional)** tasks, where a slightly richer architecture and multi-scale design help overcome some of these limitations. While still lightweight, DeceptronNet demonstrates improved stability on real datasets such as Kodak24 and better captures spatial structure and nonlocal correlations, extending the practical range of our approach beyond the single-scale version.