AI4Science: Striking the Best Data-Knowledge Tradeoff

Ju Sun (Computer Sci. & Eng., UMN)

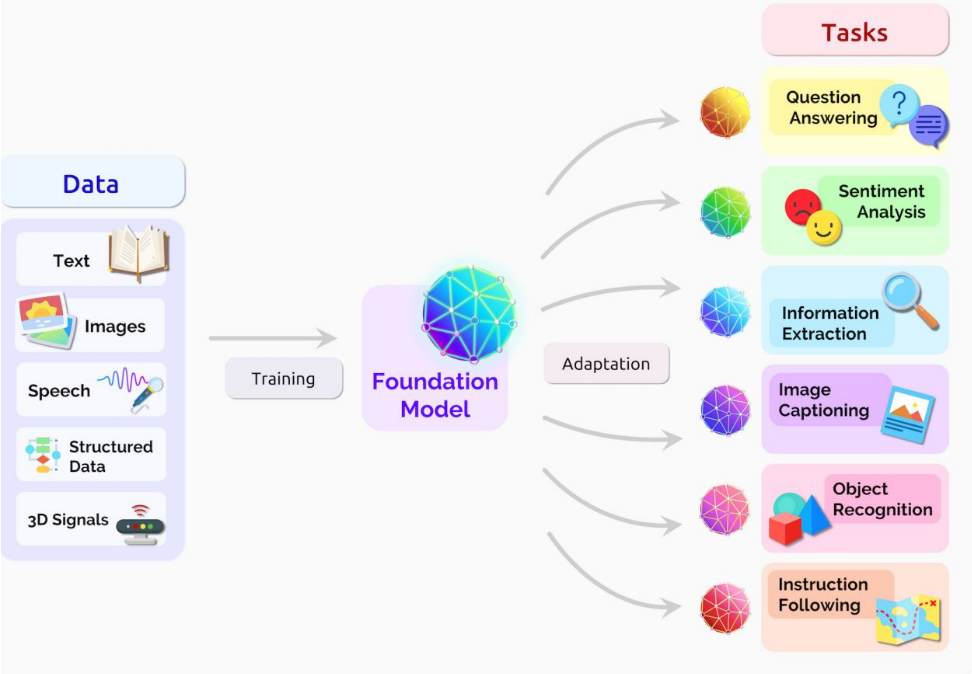
Apr 30, 2024

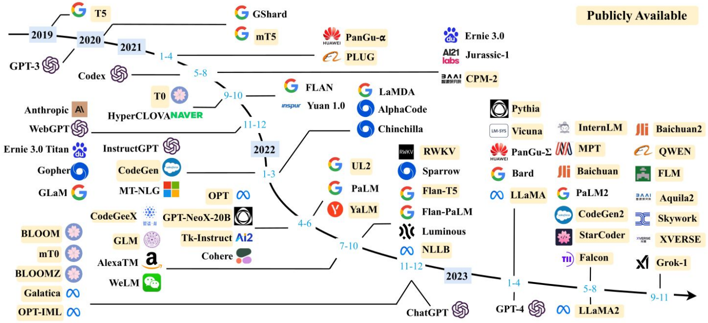
AI4Science Seminar Series, AWS

UNIVERSITY OF MINNESOTA
Driven to Discover℠

Deep learning models are data monsters

LLMs





	Model	Release Time	Size (B)	Base Model	Adaptation IT	Adaptation RLHF	Pre-train Data Scale	Latest Data Timestamp	Hardware (GPUs / TPUs)	Training Time	Evaluation ICL	Evaluation CoT
	T5 [82]	Oct-2019	11	-	-	-	1T tokens	Apr-2019	1024 TPU v3	-	✓	-
	mT5 [83]	Oct-2020	13	-	-	-	1T tokens	-	-	-	✓	-
	PanGu-α [84]	Apr-2021	13*	-	-	-	1.1TB	-	2048 Ascend 910	-	✓	-
	CPM-2 [85]	Jun-2021	198	-	-	-	2.6TB	-	-	-	-	-
	T0 [28]	Oct-2021	11	T5	✓	-	-	-	512 TPU v3	27 h	✓	-
	CodeGen [86]	Mar-2022	16	-	-	-	577B tokens	-	-	-	✓	-
	GPT-NeoX-20B [87]	Apr-2022	20	-	-	-	825GB	-	96 40G A100	-	✓	-
	Tk-Instruct [88]	Apr-2022	11	T5	✓	-	-	-	256 TPU v3	4 h	✓	-
	UL2 [89]	May-2022	20	-	-	-	1T tokens	Apr-2019	512 TPU v4	-	✓	✓
	OPT [90]	May-2022	175	-	-	-	180B tokens	-	992 80G A100	-	✓	-
	NLLB [91]	Jul-2022	54.5	-	-	-	-	-	-	-	✓	-
	CodeGeeX [92]	Sep-2022	13	-	-	-	850B tokens	-	1536 Ascend 910	60 d	✓	-
	GLM [93]	Oct-2022	130	-	-	-	400B tokens	-	768 40G A100	60 d	✓	-
	Flan-T5 [69]	Oct-2022	11	T5	✓	-	-	-	-	-	✓	✓
	BLOOM [78]	Nov-2022	176	-	-	-	366B tokens	-	384 80G A100	105 d	✓	-
	mT0 [94]	Nov-2022	13	mT5	✓	-	-	-	-	-	✓	-
	Galactica [35]	Nov-2022	120	-	-	-	106B tokens	-	-	-	✓	✓
	BLOOMZ [94]	Nov-2022	176	BLOOM	✓	-	-	-	-	-	✓	-
Publicly Available	OPT-IML [95]	Dec-2022	175	OPT	✓	-	-	-	128 40G A100	-	✓	✓
	LLaMA [57]	Feb-2023	65	-	-	-	1.4T tokens	-	2048 80G A100	21 d	✓	-
	Pythia [96]	Apr-2023	12	-	-	-	300B tokens	-	256 40G A100	-	✓	-
	CodeGen2 [97]	May-2023	16	-	-	-	400B tokens	-	-	-	✓	-
	StarCoder [98]	May-2023	15.5	-	-	-	1T tokens	-	512 40G A100	-	✓	✓
	LLaMA2 [99]	Jul-2023	70	-	✓	✓	2T tokens	-	2000 80G A100	-	✓	-
	Baichuan2 [100]	Sep-2023	13	-	✓	✓	2.6T tokens	-	1024 A800	-	✓	-
	QWEN [101]	Sep-2023	14	-	✓	✓	3T tokens	-	-	-	✓	-
	FLM [102]	Sep-2023	101	-	✓	-	311B tokens	-	192 A800	22 d	✓	-
	Skywork [103]	Oct-2023	13	-	-	-	3.2T tokens	-	512 80G A800	-	✓	-

Credit: **On the Opportunities and Risks of Foundation Models**
https://arxiv.org/abs/2108.07258

Credit: **A Survey of Large Language Models**
https://arxiv.org/abs/2303.18223

Towards Geospatial Foundation Models via Continual Pretraining

Matías Mendieta[1*] Boran Han[2] Xingjian Shi[3] Yi Zhu[3] Chen Chen[1]
[1] Center for Research in Computer Vision, University of Central Florida
[2] Amazon Web Services [3] Boson AI

Figure 2. We visualize som
datasets with Sentinel-2 (le
noticeably much lower feat
across images than that of our GeoPile pretraining dataset.

ous pretraining paradigm GFM.

Method	# Images	Epochs	ARP ↑	Time ↓	CO_2 ↓
ImageNet-22k Sup.	14M	-	0.0	-	-
Sentinel-2 [30]	1.3M	100	-5.83	155.6	22.2
GeoPile	600k	200	0.92	133.3	19.0
GeoPile†	600k	200	1.24	133.3	19.0
GeoPile†	600k	800	1.45	533.2	76.0
GFM	600k	100	3.31	93.3	13.3

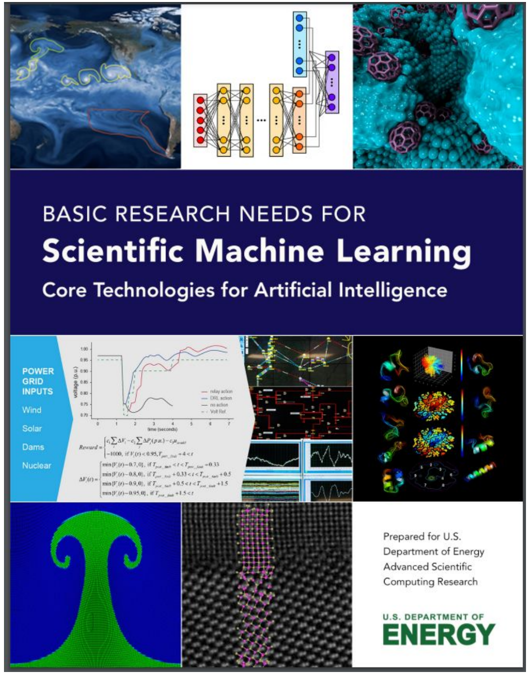
Not all fields are as lucky

Thrust B: How Should Domain Knowledge Be Incorporated into Supervised Machine Learning?

The central question for this thrust is "which knowledge should be leveraged in SciML, and how should this knowledge be included?" Any answers will naturally depend on the SciML task and computational budgets, thus mirroring standard considerations in traditional scientific computing.

Hard Constraints. One research avenue involves incorporation of domain knowledge through imposition of constraints that cannot be violated. These hard constraints could be enforced during training, replacing what typically is an unconstrained optimization problem with a constrained one. In general, such constraints could involve simulations or highly nonlinear functions of the training parameters. Therefore, there is a need to identify particular cases when constraint qualification conditions can be ensured as these conditions are necessary regularity conditions for constrained optimization [57–59]. Although incorporating constraints during training generally makes maximal use of training data, there may be additional opportunities to employ constraints at the time of prediction (e.g., by projecting predictions onto the region induced by the constraints).

Soft Constraints. A similar avenue for incorporating domain knowledge involves modifying the objective function (soft constraints) used in training. It is understood that ML loss function selection should be guided by the task and data. Therefore, opportunities exist for developing loss functions that incorporate domain knowledge and analyzing the resulting impact on solvability

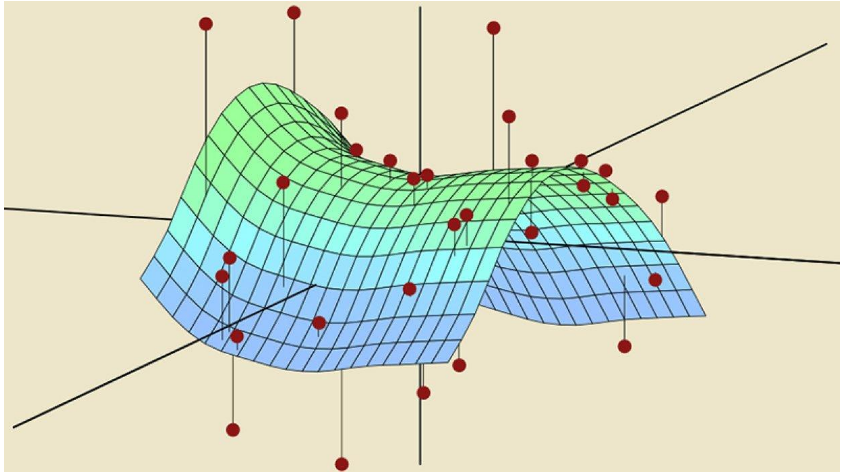


Ref https://www.osti.gov/servlets/purl/1478744

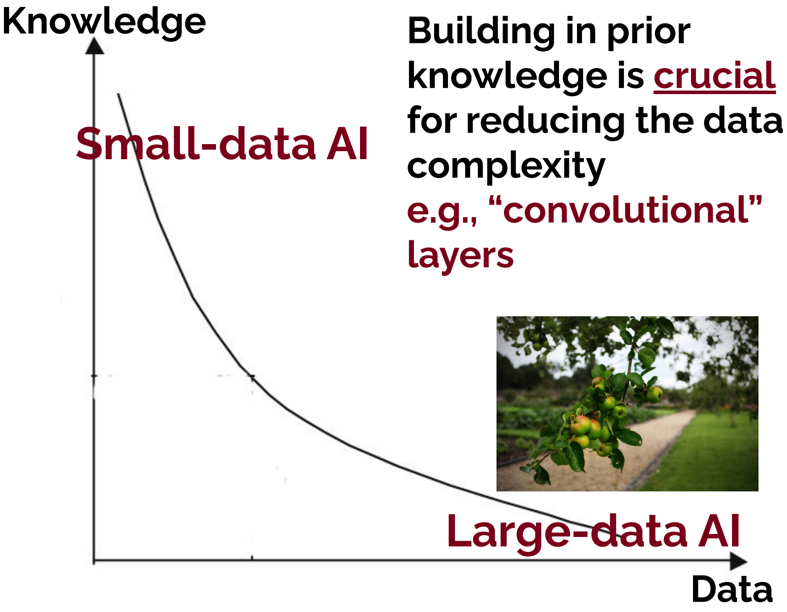
Domain-Aware Scientific Machine Learning

There's no free lunch!

Supervised learning as data fitting



Typically, #data points we need grow **exponentially** with respect to dimension (i.e., **curse of dimensionality**)



Building in prior knowledge is <u>crucial</u> for reducing the data complexity
e.g., "convolutional" layers

This talk:

several stories about data-knowledge tradeoffs

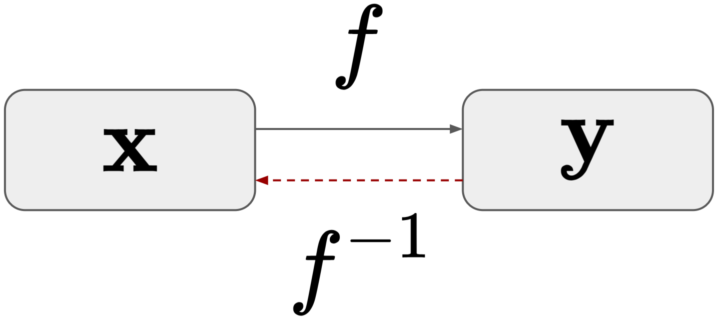
- Scientific inverse problems (SIPs)
  - Data-driven (data-rich) methods for SIPs
  - Single-instance (data-poor) methods for SIPs
- Principled computational tool for data-knowledge tradeoffs

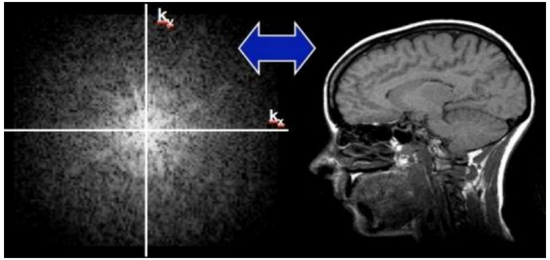
Scientific Inverse Problems

Inverse problems

$\boxed{\text{Inverse problem: given } \mathbf{y} = f(\mathbf{x}), \quad \text{recover } \mathbf{x}}$

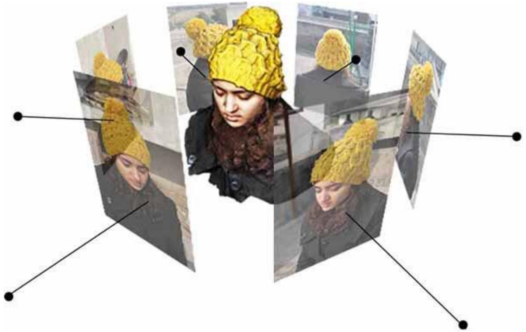
Image denoising



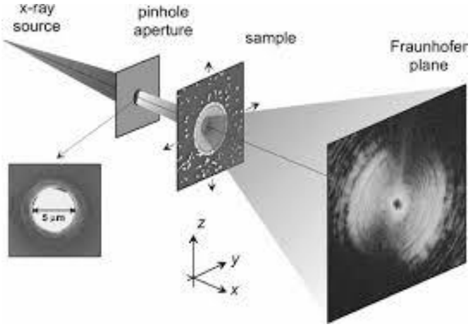


MRI reconstruction

Image super-resolution



3D reconstruction



Coherent diffraction imaging (CDI)

Traditional methods

Inverse problem: given $\mathbf{y} = f(\mathbf{x})$, recover $\mathbf{x}$

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y}, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

RegFit

Limitations:

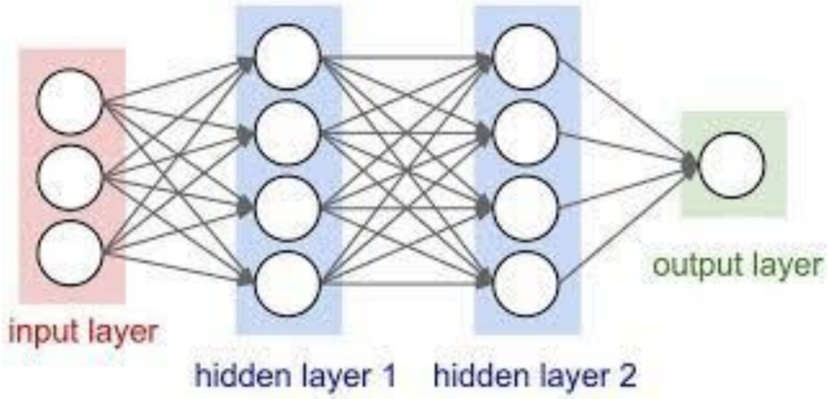
- Which ℓ ? (e.g., unknown/compound noise)
- Which R ? (e.g., structures not amenable to math description)
- Speed

DL methods for SIPs: the radical way

Inverse problem: given $\mathbf{y} = f(\mathbf{x})$, recover $\mathbf{x}$

Learn the f^{-1} with a training set $\{(\mathbf{y}_i, \mathbf{x}_i)\}$

$\mathbf{y}_i$



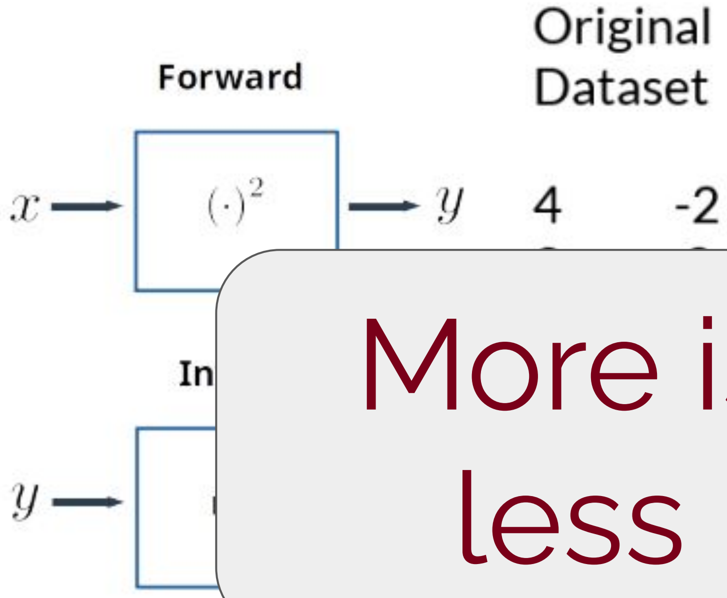
$\mathbf{x}_i$

Limitations:

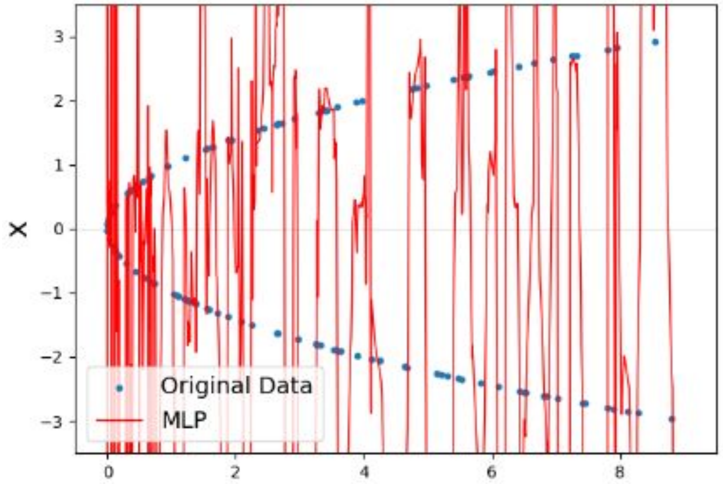
- Wasteful: not using f
- Representative data?
- Not always straightforward

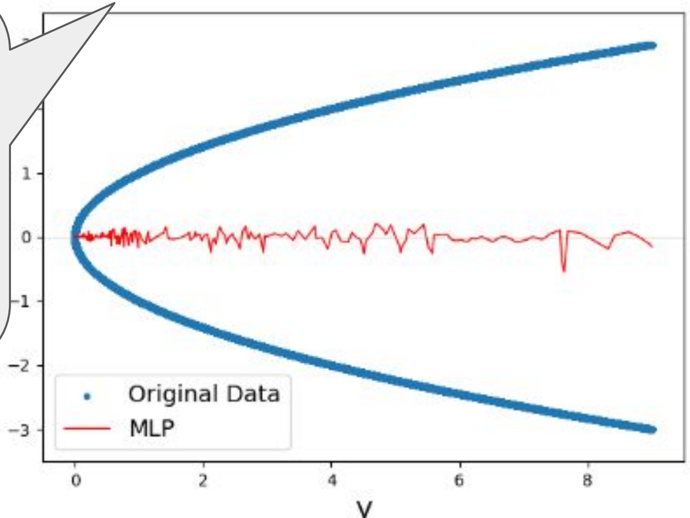
(see, e.g., Tayal et al. **Inverse Problems, Deep Learning, and Symmetry Breaking**. https://arxiv.org/abs/2003.09077)

Story I: More could be less



More is less

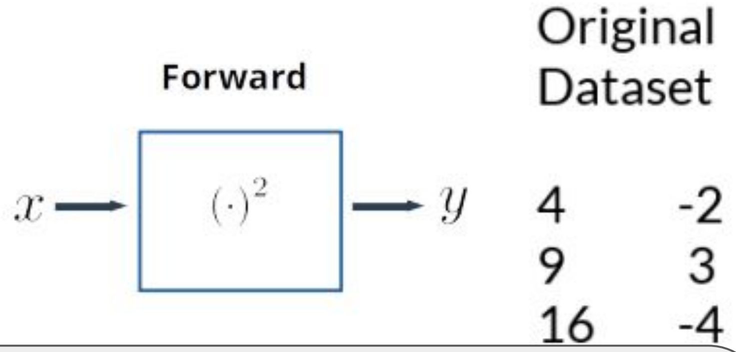


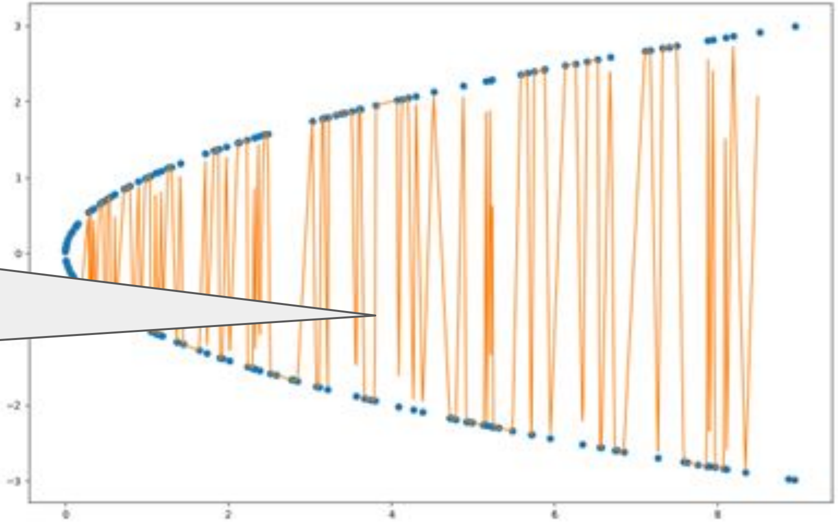


Why "more is less" here?

Forward symmetry: $\{+\sqrt{y}, -\sqrt{y}\} \leftrightarrow y$

Implies: on dense training set, very close y's can mapped to very far aways x's different by signs

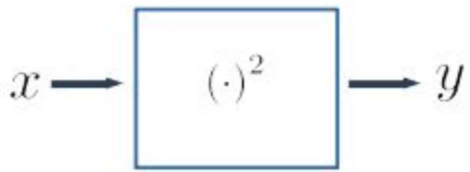




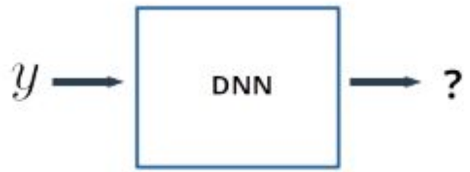
Highly oscillatory target function to learn by DNNs—difficult

Remedy: symmetry breaking

Forward

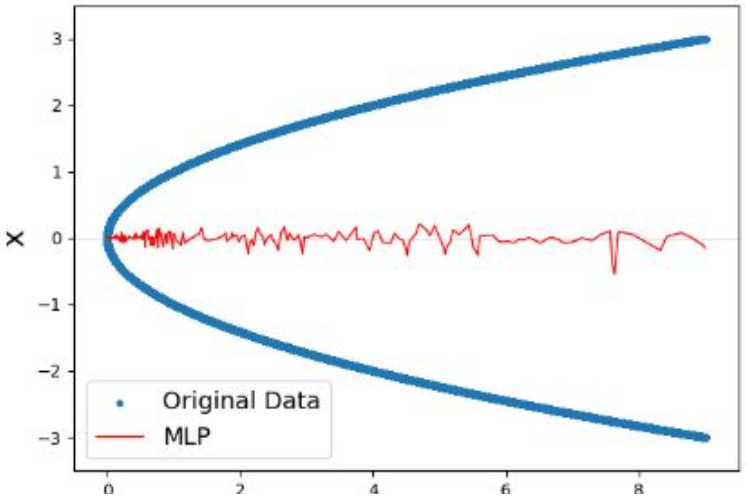


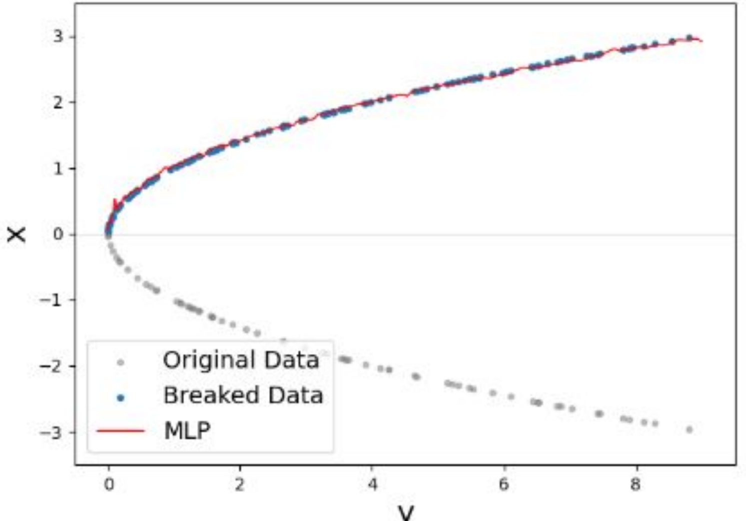
Inverse



Original Dataset

4	-2
9	3
16	-4
100	10
4	2
64	-8
36	6





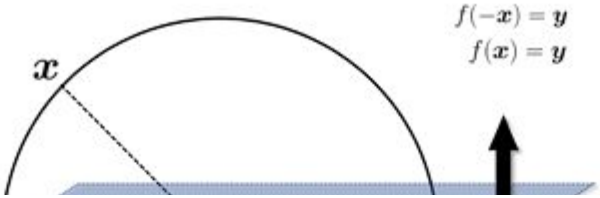
Fix all signs to be positive

A slightly more complicated example

$$\mathbf{y} = |\mathbf{A}\mathbf{x}|^2 \quad \mathbf{A} : \text{iid Gaussian}$$ **(Gaussian phase retrieval)**

Forward symmetry: global sign

$$\mathbf{y} = |\mathbf{A}\mathbf{x}|^2 = |\mathbf{A}(-\mathbf{x})|^2$$

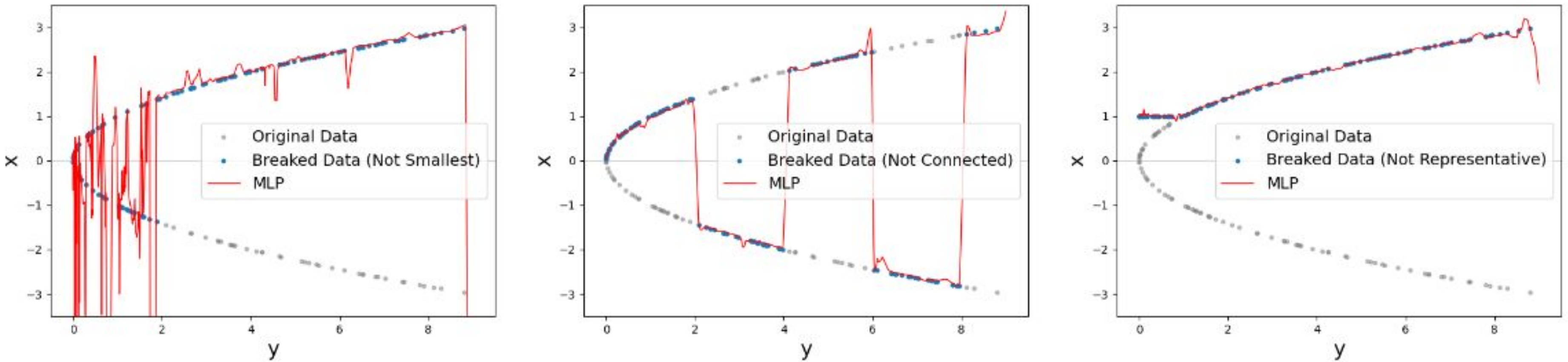


		After Symmetry Breaking		Before Symmetry Breaking	
Dim	Sample	DNN	K-NN	DNN	K-NN
5	2e4	4.08	11.82	85.37	68.26
	5e4	2.20	9.41	90.51	66.58
	1e5	1.30	7.98	96.66	66.18
	1e6	0.37	4.71	122.71	65.08

More is more **More is less**

Symmetry-breaking principle

Symmetry breaking: a preprocessing step on the training set



Finding the smallest, connected, representative set

[Submitted on 18 Mar 2024]

What is Wrong with End-to-End Learning for Phase Retrieval?

Wenjie Zhang, Yuxiang Wan, Zhong Zhuang, Ju Sun

For nonlinear inverse problems that are prevalent in imaging science, symmetries in the forward model are common. When data-driven deep learning approaches are used to solve such problems, these intrinsic symmetries can cause substantial learning difficulties. In this paper, we explain how such difficulties arise and, more importantly, how to overcome them by preprocessing the training set before any learning, i.e., symmetry breaking. We take far-field phase retrieval (FFPR), which is central to many areas of scientific imaging, as an example and show that symmetric breaking can substantially improve data-driven learning. We also formulate the mathematical principle of symmetry breaking.

DL methods for SIPs: the middle way

Inverse problem: given $\mathbf{y} = f(\mathbf{x})$, recover $\mathbf{x}$

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y}, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

RegFit

<u>**Recipe**</u>: revamp numerical methods for RegFit with **pretrained/trainable DNNs**

DL methods for SIPs: the middle way

Algorithm unrolling

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y},\, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

If R proximal friendly

$$\mathbf{x}^{k+1} = \mathcal{P}_R\left(\mathbf{x}^k - \eta \nabla^\top f(\mathbf{x}^k)\,\ell'(\mathbf{y},\, f(\mathbf{x}^k))\right)$$

<u>**Idea**</u>: make $\mathcal{P}_R$ trainable, using $\{(\mathbf{x}_i\,,\,\mathbf{y}_i)\}$

E.g.,

$$\ell(\mathbf{y},\, f(\mathbf{x})) = \|\mathbf{y} - \mathbf{A}\,\mathbf{x}\|_2^2$$

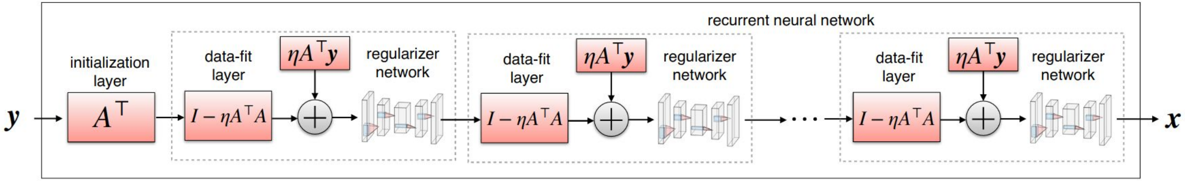


Fig credit: Deep Learning Techniques for Inverse Problems in Imaging https://arxiv.org/abs/2005.06001

DL methods for SIPs: the middle way

$$\boxed{\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y},\, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}}$$

Using $\{\mathbf{x}_i\}$ only

Plug-and-Play

$$\mathbf{x}^{k+1} = \mathcal{P}_R\left(\mathbf{x}^k - \eta \nabla^\top f(\mathbf{x}^k) \ell'(\mathbf{y},\, f(\mathbf{x}^k))\right)$$

E.g. replace $\mathcal{P}_R$ with pretrained denoiser

Deep generative models

Pretraining: $\mathbf{x}_i \approx G_\theta(\mathbf{z}_i) \ \forall i$

Deployment: $\min_{\mathbf{z}} \ \ell(\mathbf{y},\, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$

DL methods for SIPs: a survey

Deep Learning Techniques for Inverse Problems in Imaging

Gregory Ongie,[*] Ajil Jalal,[†] Christopher A. Metzler[‡]
Richard G. Baraniuk,[§] Alexandros G. Dimakis,[¶] Rebecca Willett[||]

April 2020

Abstract

Recent work in machine learning shows that deep neural networks can be used to solve a wide variety of inverse problems arising in computational imaging. We explore the central prevailing themes of this emerging area and present a taxonomy that can be used to categorize different problems and reconstruction methods. Our taxonomy is organized along two central axes: (1) whether or not a forward model is known and to what extent it is used in training and testing, and (2) whether or not the learning is supervised or unsupervised, i.e., whether or not the training relies on access to matched ground truth image and measurement pairs. We also discuss the tradeoffs associated with these different reconstruction approaches, caveats and common failure modes, plus open problems and avenues for future work.

Focuses on **linear** inverse problems, i.e., f linear

https://arxiv.org/abs/2005.06001

Limitations of middle ways:

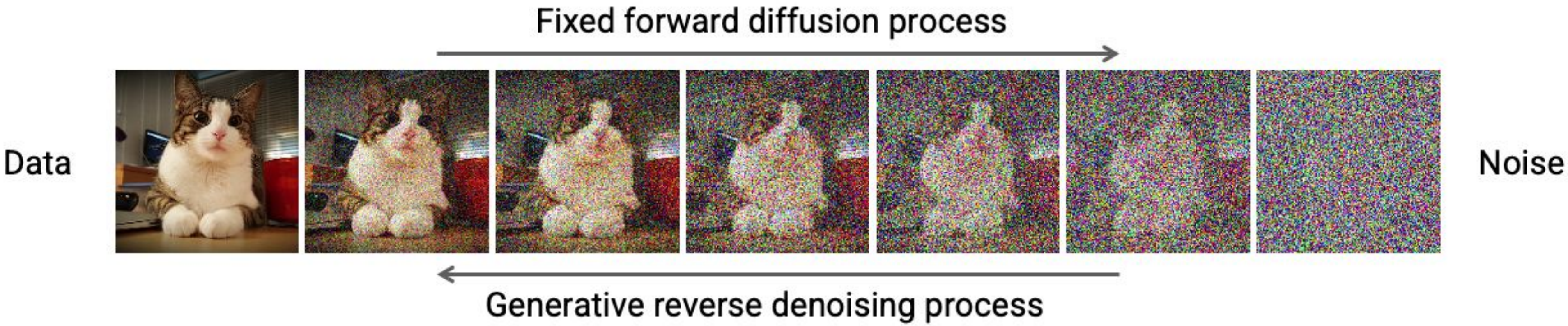
- Representative data?
- Algorithm-sensitive
- Good initialization? (e.g., Manekar et al. **Deep Learning Initialized Phase Retrieval.** https://sunju.org/pub/NIPS20-WS-DL4FPR.pdf)

Story II: Don't be too Bayesian

Deep generative models

Pretraining: $\mathbf{x}_i \approx G_\theta(\mathbf{z}_i) \ \forall i$

Deployment: $\min_{\mathbf{z}} \ \ell(\mathbf{y}, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$



How to use pretrained diffusion models for SIPs?

Bayesian thinking

Reverse SDE for DM

$$d\boldsymbol{x} = \left[-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) \right] dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}}$$

$$\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t|\boldsymbol{y}) = \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)$$

Reverse conditional SDE for SIPs

$$d\boldsymbol{x} = \left[-\frac{\beta(t)}{2}\boldsymbol{x} - \beta(t)(\nabla_{\boldsymbol{x}_t} \log p_t(\boldsymbol{x}_t) + \nabla_{x_t} \log p_t(\boldsymbol{y}|\boldsymbol{x}_t)) \right] dt + \sqrt{\beta(t)}d\bar{\boldsymbol{w}}$$

Bayesian thinking: after several approx steps

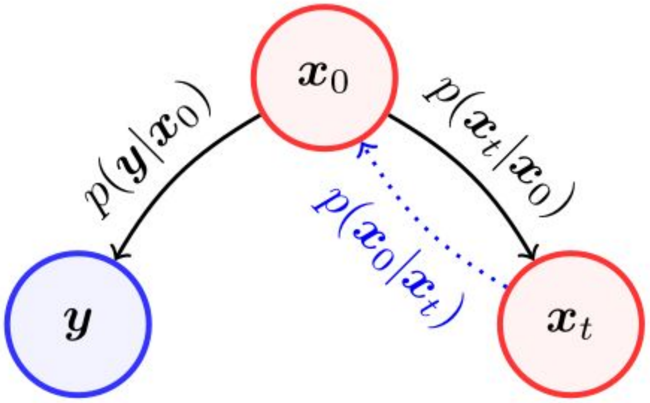


Figure 2: Probabilistic graph. Black solid line: tractable, blue dotted line: intractable in general.

Algorithm 1 DPS - Gaussian

Require: N, $\boldsymbol{y}$, $\{\zeta_i\}_{i=1}^N$, $\{\tilde{\sigma}_i\}_{i=1}^N$

$\boldsymbol{x}_N \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

for $i = N - 1$ **to** 0 **do**

$\quad \hat{\boldsymbol{s}} \leftarrow \boldsymbol{s}_\theta(\boldsymbol{x}_i, i)$

$\quad \hat{\boldsymbol{x}}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_i}}(\boldsymbol{x}_i + (1 - \bar{\alpha}_i)\hat{\boldsymbol{s}})$

$\quad \boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$

$\quad \boldsymbol{x}'_{i-1} \leftarrow \frac{\sqrt{\alpha_i}(1-\bar{\alpha}_{i-1})}{1-\bar{\alpha}_i}\boldsymbol{x}_i + \frac{\sqrt{\bar{\alpha}_{i-1}}\beta_i}{1-\bar{\alpha}_i}\hat{\boldsymbol{x}}_0 + \tilde{\sigma}_i \boldsymbol{z}$

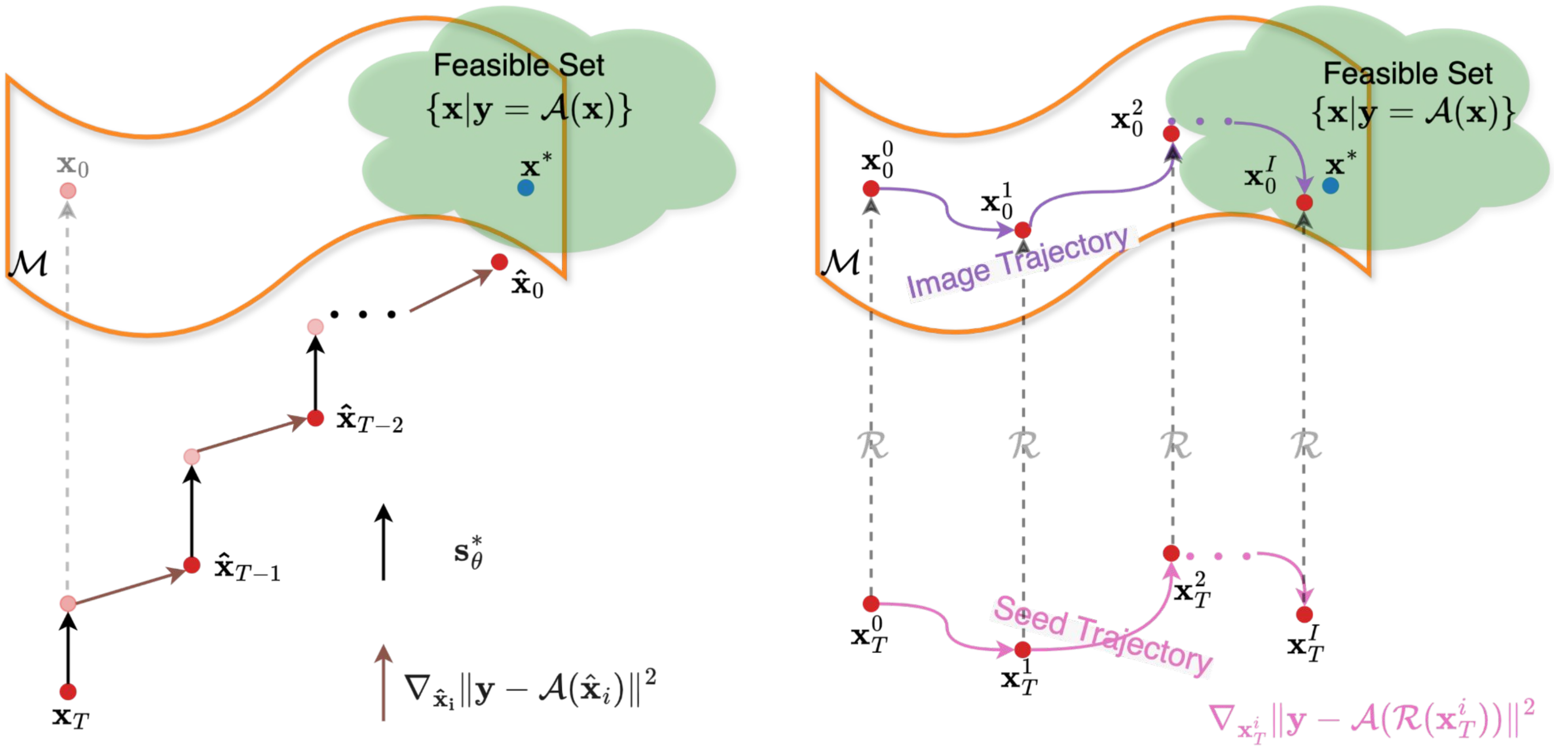
$\quad \boldsymbol{x}_{i-1} \leftarrow \boldsymbol{x}'_{i-1} - \zeta_i \nabla_{\boldsymbol{x}_i} \|\boldsymbol{y} - \mathcal{A}(\hat{\boldsymbol{x}}_0)\|_2^2$

end for

return $\hat{\mathbf{x}}_0$

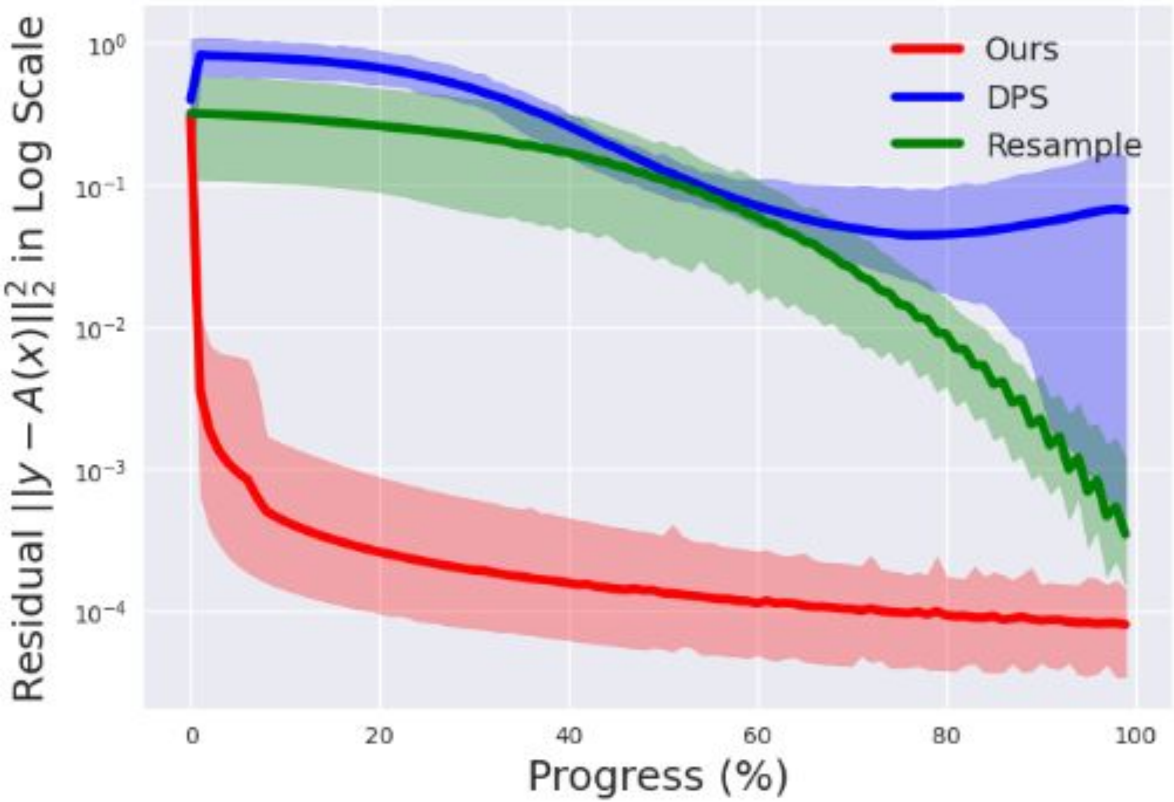
Credit: **Diffusion Posterior Sampling for General Noisy Inverse Problems** https://openreview.net/forum?id=OnD9zGAGT0k

Explained in one picture (vs. our plugin idea)



Feasibility crisis

Inverse problem: given $\mathbf{y} = f(\mathbf{x})$, recover $\mathbf{x}$



Preliminary result on linear SIPs

Table 1: **(Linear IPs)** Quantitative comparison for **super-resolution** and **inpainting** on CelebA [25] and FFHQ [34] with additional Gaussian noise ($\sigma = 0.01$). (**Bold**: best, under: second best, green: performance increase, red: performance decrease)

	Super-resolution (4×)						**Inpainting (Random** 70%)					
	CelebA (256 × 256)			**FFHQ** (256 × 256)			**CelebA** (256 × 256)			**FFHQ** (256 × 256)		
	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑
ADMM-PnP [39]	0.217	26.99	0.808	0.229	26.25	0.794	0.091	31.94	0.923	0.104	30.64	0.901
DMPS [40]	0.070	28.89	0.848	**0.076**	28.03	0.843	0.297	24.52	0.693	0.326	23.31	0.664
DDRM [11]	0.226	26.34	0.754	0.282	25.11	0.731	0.185	26.10	0.712	0.201	25.44	0.722
MCG [10]	0.725	19.88	0.323	0.786	18.20	0.271	1.283	10.16	0.049	1.276	10.37	0.050
ILVR [4]	0.322	21.63	0.603	0.360	20.73	0.570	0.447	15.82	0.484	0.483	15.10	0.450
DPS [18]	0.087	28.32	0.823	0.098	27.44	0.814	0.043	32.24	0.924	0.046	30.95	0.913
ReSample [9]	0.080	28.29	0.819	0.108	25.22	0.773	**0.039**	30.12	0.904	0.044	27.91	0.884
Ours	**0.067**	**31.25**	**0.878**	0.079	**30.25**	**0.871**	**0.039**	**34.03**	**0.936**	**0.038**	**33.01**	**0.931**
Ours vs. Best compe.	−0.003	+2.36	+0.030	+0.003	+2.22	+0.028	−0.000	+1.79	+0.012	−0.006	+2.06	+0.018

Preliminary result on nonlinear SIPs

Table 3: **(Nonlinear IP)** Quantitative comparison for **BID** on CelebA [25] and FFHQ [34] with additional Gaussian noise ($\sigma = 0.01$). (**Bold**: best, under: second best, green: performance increase, red: performance decrease)

	CelebA (256×256)						**FFHQ** (256×256)					
	Motion			**Gaussian**			**Motion**			**Gaussian**		
	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑
SelfDeblur [43]	0.568	16.59	0.417	0.579	16.55	0.423	0.628	16.33	0.408	0.604	16.22	0.410
DeBlurGANv2 [44]	0.313	20.56	0.613	0.350	24.29	0.743	0.353	19.67	0.581	0.374	23.58	0.726
Stripformer [45]	0.287	22.06	0.644	0.316	25.03	<u>0.747</u>	0.324	21.31	0.613	0.339	<u>24.34</u>	<u>0.728</u>
MPRNet [46]	0.332	20.53	0.620	0.375	22.72	0.698	0.373	19.70	0.590	0.394	22.33	0.685
Pan-DCP [47]	0.606	15.83	0.483	0.653	20.57	0.701	0.616	15.59	0.464	0.667	20.69	0.698
Pan-ℓ_0 [48]	0.631	15.16	0.470	0.654	20.49	0.675	0.642	14.43	0.443	0.669	20.34	0.671
ILVR [4]	0.398	19.23	0.520	0.338	21.20	0.588	0.445	18.33	0.484	0.375	20.45	0.555
BlindDPS [13]	<u>0.164</u>	<u>23.60</u>	<u>0.682</u>	<u>0.173</u>	<u>25.15</u>	0.721	<u>0.185</u>	<u>21.77</u>	<u>0.630</u>	<u>0.193</u>	23.83	0.693
Ours	**0.104**	**29.61**	**0.825**	**0.140**	**28.84**	**0.795**	**0.135**	**27.99**	**0.794**	**0.169**	**28.26**	**0.811**
Ours vs. Best compe.	-0.060	$+6.01$	$+0.143$	-0.033	$+3.69$	$+0.048$	-0.050	$+6.22$	$+0.164$	-0.024	$+3.92$	$+0.083$

Am I supposed/allowed to show this?

DMPlug: A Plug-in Method for Solving Inverse Problems with Diffusion Models

Anonymous Author(s)
Affiliation
Address
`email`

DL methods for SIPS: the **economic** way

Deep image prior (DIP) $\mathbf{x} \approx G_\theta(\mathbf{z})$ G_θ (and $\mathbf{z}$) trainable

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y},\, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

$$\Downarrow$$

$$\min_{\theta} \ell(\mathbf{y},\, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$$

No extra training data!

Ulyanov et al. **Deep image prior**. IJCV'20. https://arxiv.org/abs/1711.10925

Contrasting: Deep generative models

Pretraining: $\mathbf{x}_i \approx G_\theta(\mathbf{z}_i) \;\; \forall\, i$

Deployment: $\min_{\mathbf{z}} \ell(\mathbf{y},\, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$

DIP's cousin(s)

Deep image prior (DIP)

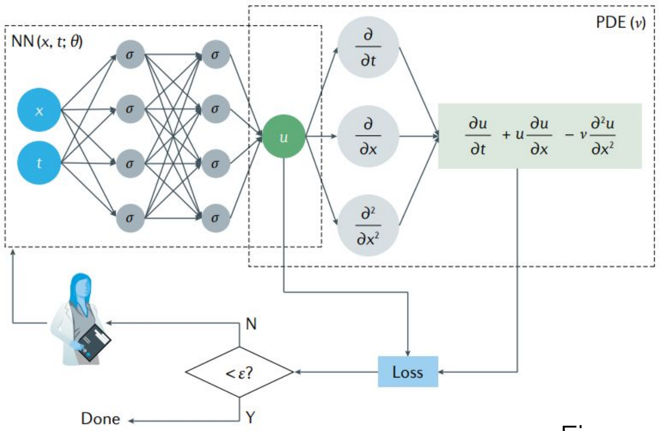
$\mathbf{x} \approx G_\theta(\mathbf{z})$ $\quad G_\theta$ (and $\mathbf{z}$) trainable

Idea: (visual) objects as continuous functions

Neural implicit representation (NIR)

$$\mathbf{x} \approx \mathcal{D} \circ \overline{\mathbf{x}} \qquad \mathcal{D} : \text{discretization} \qquad \overline{\mathbf{x}} : \text{continuous function}$$

Physics-informed neural networks (PINN)



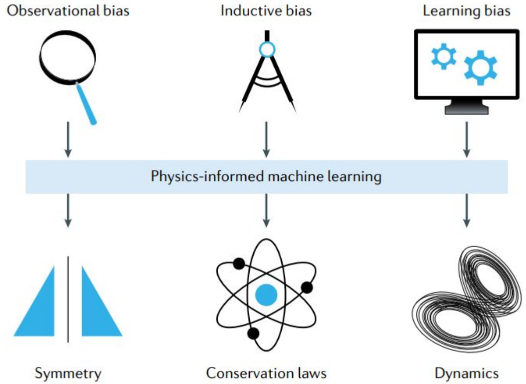
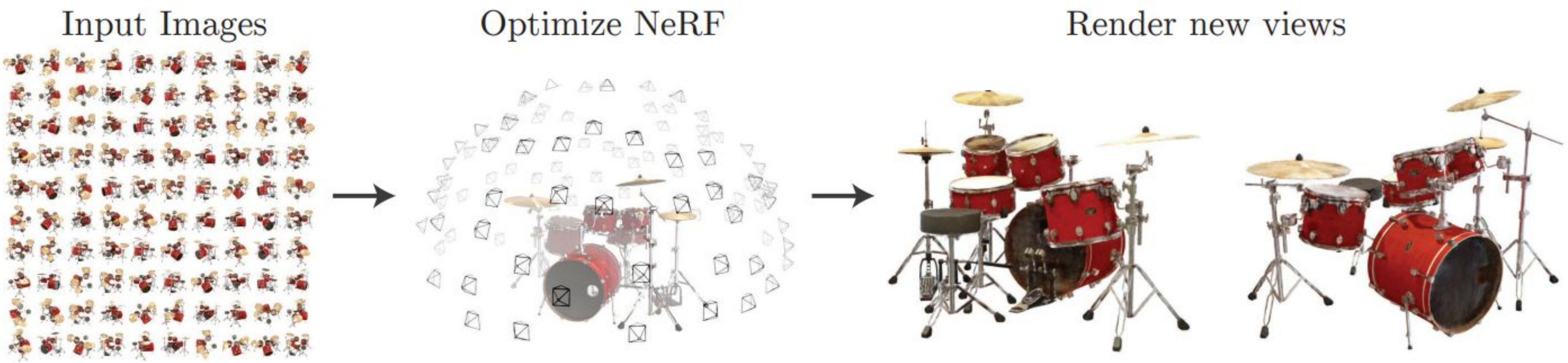
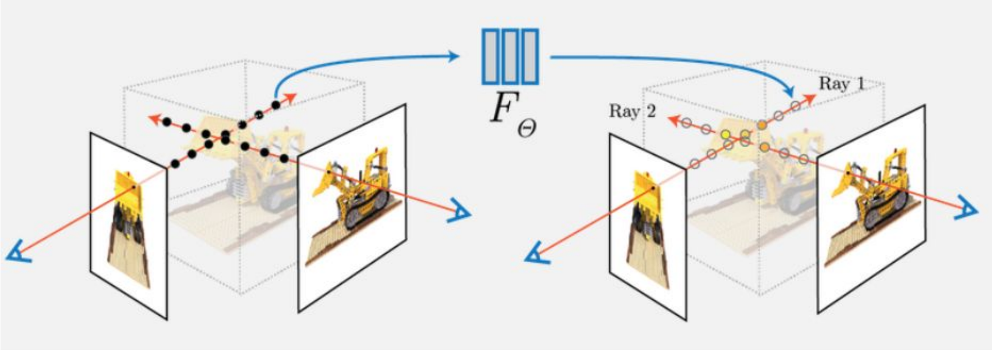
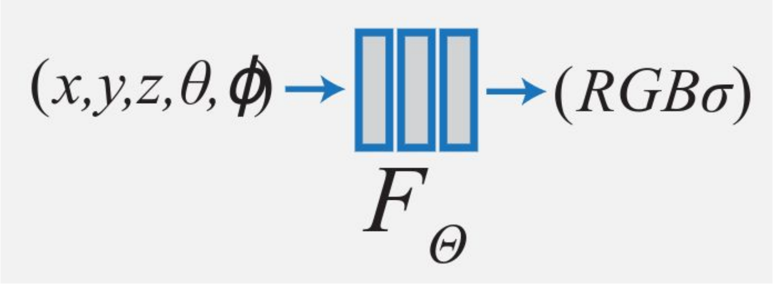


Figure credit: https://www.nature.com/articles/s42254-021-00314-5

NIR for 3D rendering and view synthesis







https://www.matthewtancik.com/nerf

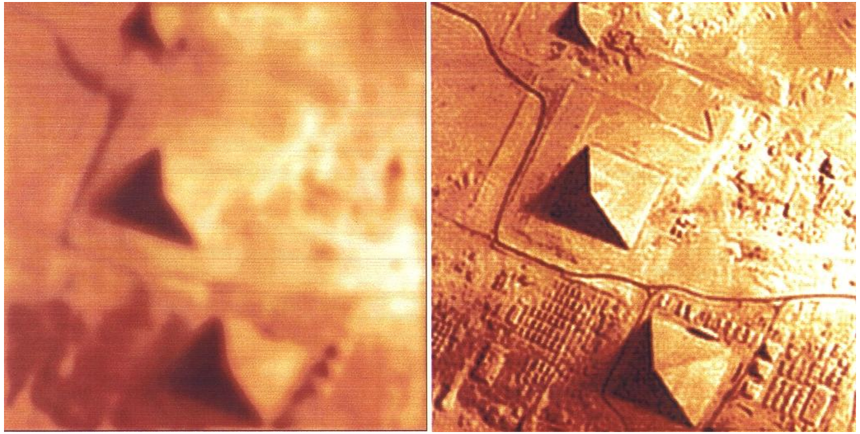
Story III: We benefit from DL even with a single data point

Blind image deblurring (BID)

$$\underbrace{\mathbf{y}}_{\text{blurry and noisy image}} = \overbrace{\mathbf{k}}^{\text{blur kernel}} * \underbrace{\mathbf{x}}_{\text{clean image}} + \overbrace{\mathbf{n}}^{\text{noise}}$$

Given $\mathbf{y}$,
recover $\mathbf{x}$ (and/or $\mathbf{k}$)

Also **Blind Deconvolution**



Landmark surveys

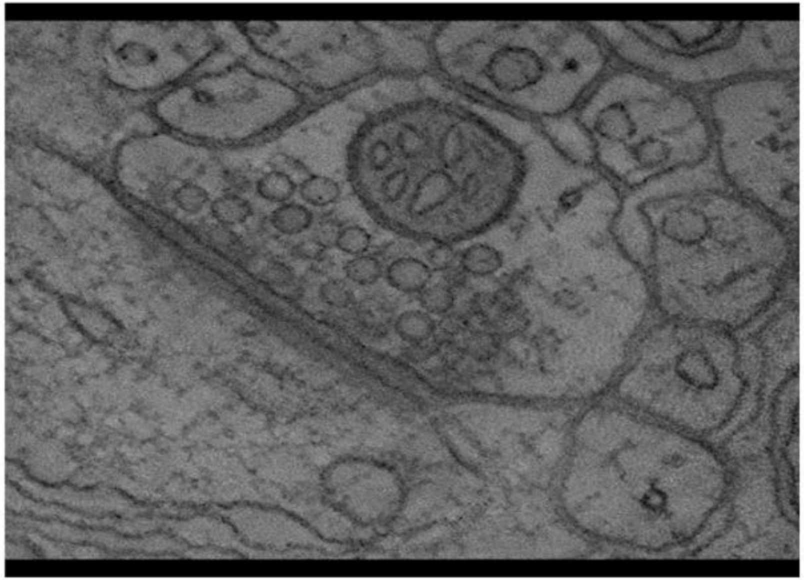
- 1996: Kundur and Hatzinakos. **Blind image deconvolution**. https://doi.org/10.1109/79.489268
- 2011: Levin et al. **Understanding blind deconvolution algorithms**. https://doi.org/10.1109/TPAMI.2011.148
- 2012: Kohler et al. **Recording and playback of camera shake: Benchmarking blind deconvolution with a real-world database**. https://doi.org/10.1007/978-3-642-33786-4_3
- 2016: Lai et al. **A comparative study for single image blind deblurring**. https://doi.org/10.1109/CVPR.2016.188
- 2021: Koh et al. **Single image deblurring with neural networks: A comparative survey** https://doi.org/10.1016/j.cviu.2020.103134
- 2022: Zhang et al. **Deep image blurring: A survey** https://doi.org/10.1007/s11263-022-01633-5

See also: **Awesome Deblurring**
https://github.com/subeeshvasu/Awesome-Deblurring

Key challenge of data-driven approach:

obtaining sufficiently expressive data (Koh et al'21. Zhang et al'22)

Untouched practical questions



Key question addressed in this paper: How do we solve blind image deblurring without knowing: (1) the size of the blur kernel, (2) the type and level of noise, and (3) whether it is blur / noise only or both ?

Double DIPs

$$\underbrace{\mathbf{y}}_{\text{blurry and noisy image}} = \overbrace{\mathbf{k}}^{\text{blur kernel}} * \underbrace{\mathbf{x}}_{\text{clean image}} + \overbrace{\mathbf{n}}^{\text{noise}}$$

$$\min_{\boldsymbol{k},\boldsymbol{x}} \underbrace{\ell(\boldsymbol{y}, \boldsymbol{k} * \boldsymbol{x})}_{\text{data fitting}} + \lambda_{\boldsymbol{k}} \underbrace{R_{\boldsymbol{k}}(\boldsymbol{k})}_{\text{regularizing } \boldsymbol{k}} + \lambda_{\boldsymbol{x}} \underbrace{R_{\boldsymbol{x}}(\boldsymbol{x})}_{\text{regularizing } \boldsymbol{x}}$$

Idea: parameterize both $\mathbf{k}$ and $\mathbf{x}$ as DIPs

- CNN + CNN (Wang et al'19, https://doi.ieeecomputersociety.org/10.1109/ICCVW.2019.00127; Tran et al'21, https://arxiv.org/abs/2104.00317)
- MLP + CNN (SelfDeblur; Ren et al'20, https://arxiv.org/abs/1908.02197)

Still problematic with

1) kernel size over-specification 2) substantial noise

A glance of our modifications

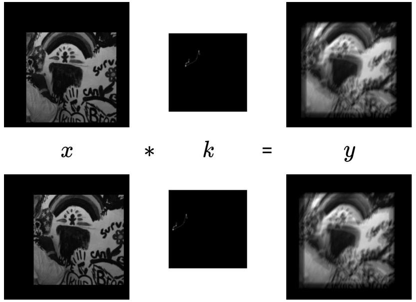
Over-specify $\mathbf{k}$

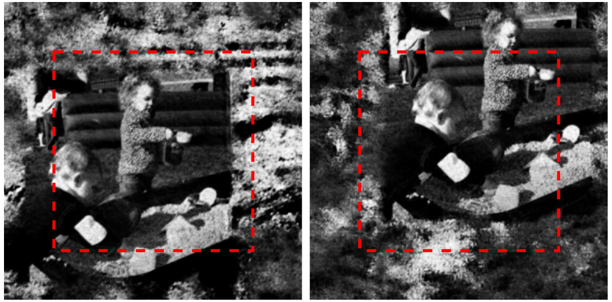
Over-specify $\mathbf{x}$

$\mathbf{k}$ **~half of the image sizes**

Ground Truth | Over-specified x | Exact-specified x

Handle bounded shift





$$\min_{\boldsymbol{\theta}_k,\boldsymbol{\theta}_x} \|\boldsymbol{y} - G_{\boldsymbol{\theta}_k}(\boldsymbol{z}_k) * G_{\boldsymbol{\theta}_x}(\boldsymbol{z}_x)\|_2^2 + \lambda \frac{\|\nabla G_{\boldsymbol{\theta}_x}(\boldsymbol{z}_x)\|_1}{\|\nabla G_{\boldsymbol{\theta}_x}(\boldsymbol{z}_x)\|_2}$$

ℓ_1/ℓ_2 vs ℓ_1

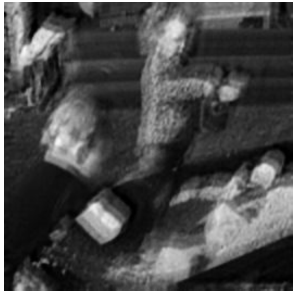
Table 1: ℓ_1/ℓ_2 vs TV for noise: mean and (std).

	Low Level		High Level	
	PSNR	λ	PSNR	λ
$\frac{L1}{L2}$	32.64 (0.69)	0.0001 (0.018)	27.74 (0.23)	0.0002 (0.0019)
TV	31.12 (0.52)	0.002 (0.07)	24.34 (0.78)	0.02 (0.10)

SelfDeblur vs our method


Clean


Blurry and noisy


SelfDeblur


Ours

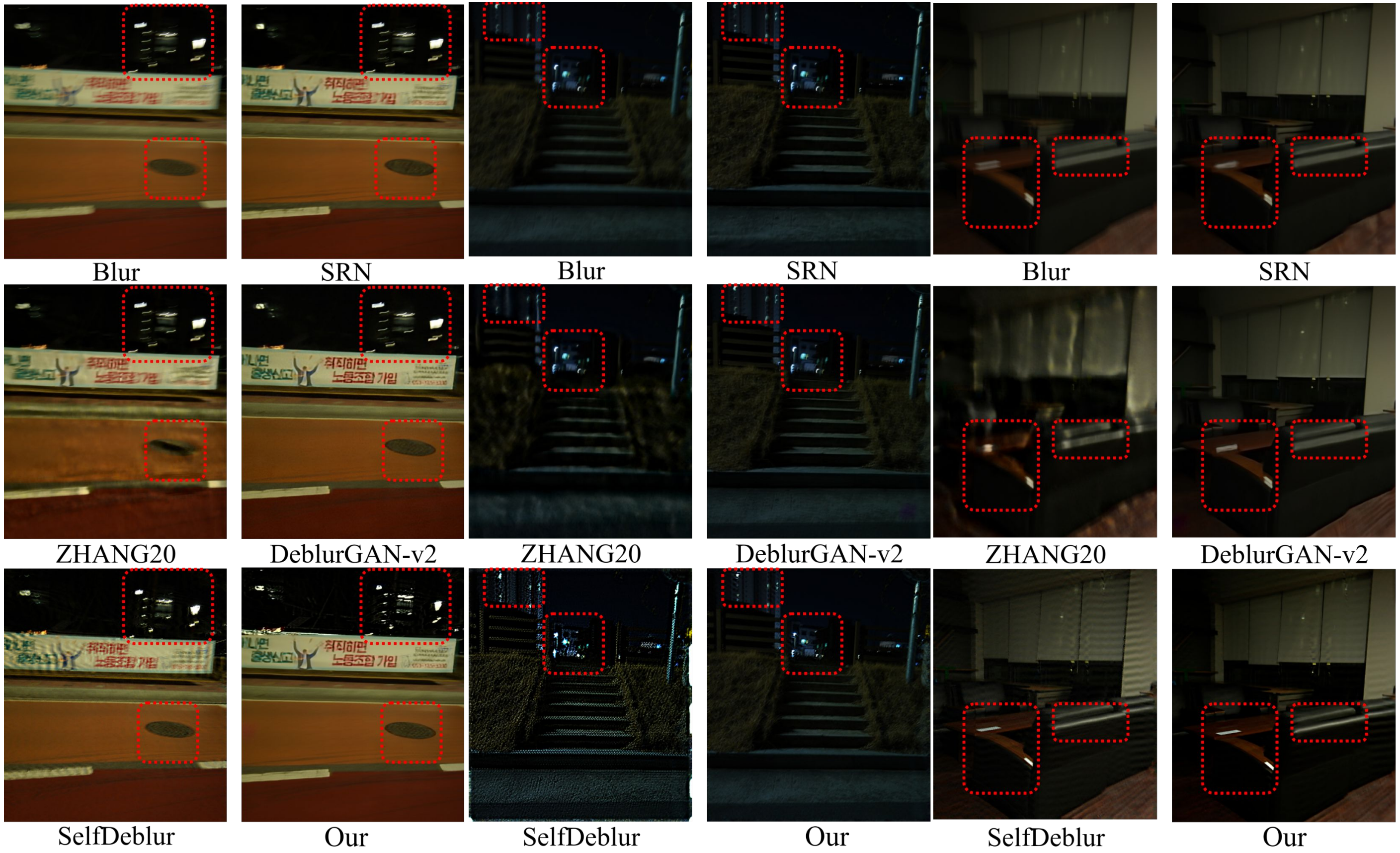

Clean


Blurry and noisy


SelfDeblur


Ours

Real world results

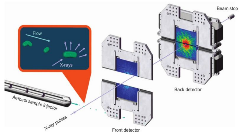


Difficult cases

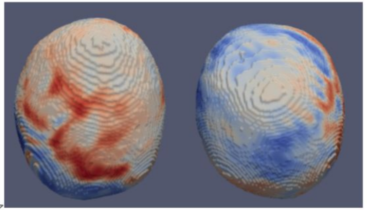
1) High depth contrast
2) High brightness contrast

Outperform SOTA data-driven methods!

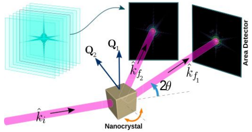
Breakthroughs in imaging



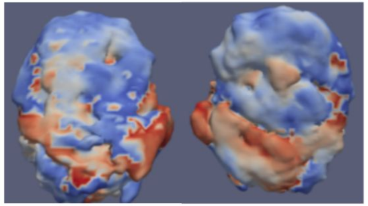
Coherent Diffraction Imaging



Our



Bragg Coherent Diffraction Imaging



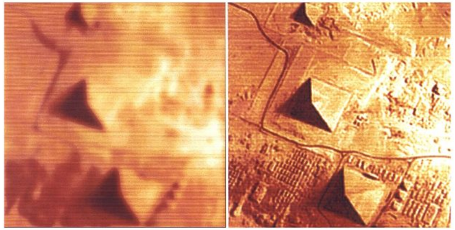
HIO+ER with Shrinkwrap

$$\underbrace{\mathbf{y}}_{\text{blurry and noisy image}} = \overbrace{\mathbf{k}}^{\text{blur kernel}} * \underbrace{\mathbf{x}}_{\text{clean image}} + \overbrace{\mathbf{n}}^{\text{noise}}$$

Mostly due to optical deficiencies (e.g., defocus) and motions

Given $\mathbf{y}$,
recover $\mathbf{x}$ (and/or $\mathbf{k}$)

Also **Blind Deconvolution**



First PR method that won in a double-blind test, and systematic evaluation, beating a 40-years old legacy

Practical Phase Retrieval Using Double Deep Image Priors

Zhong Zhuang, David Yang, Felix Hofmann, David Barmherzig, Ju Sun

First BID method that works with unknown kernel size AND substantial noise

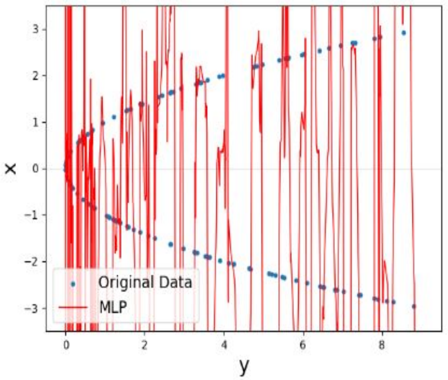
Blind Image Deblurring with Unknown Kernel Size and Substantial Noise

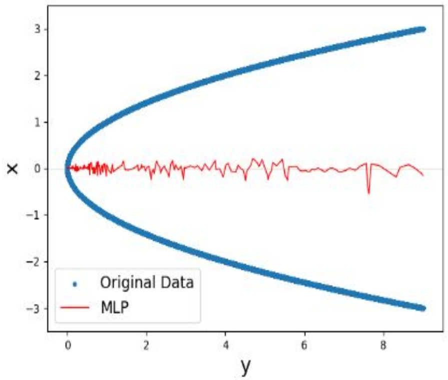
Zhong Zhuang, Taihui Li, Hengkang Wang, Ju Sun

Related papers

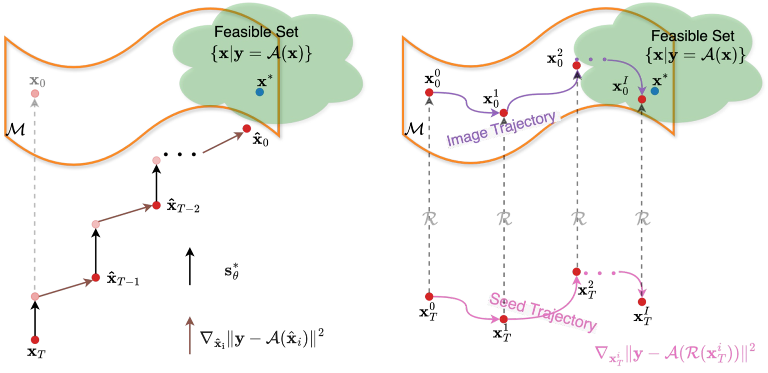
- Li et al. **Self-Validation: Early Stopping for Single-Instance Deep Generative Priors** (BMVC'21) https://arxiv.org/abs/2110.12271
- Wang et al. **Early Stopping for Deep Image Prior** https://arxiv.org/abs/2112.06074 (TMLR'23)
- Zhuang et al. **Blind Image Deblurring with Unknown Kernel Size and Substantial Noise**. https://arxiv.org/abs/2208.09483 (IJCV'24)
- Zhuang et al. **Practical Phase Retrieval Using Double Deep Image Priors**. https://arxiv.org/abs/2211.00799 (Electronic Imaging'24)
- Li et al. **Deep Random Projector: Toward Efficient Deep Image Prior**. (CVPR'23)

Data-driven methods for SIPs





Story I: More could be less



Story II: Don’t be too Bayesian

Single-instance methods for SIPs

Story III: Benefit from DL with a single data point

Deep image prior (DIP) $\mathbf{x} \approx G_\theta(\mathbf{z})$ G_θ (and $\mathbf{z}$) trainable

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y}, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{R(\mathbf{x})}_{\text{regularizer}}$$

$$\Downarrow$$

$$\min_{\theta} \ell(\mathbf{y}, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$$

No extra training data!

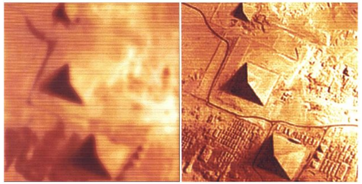
Ulyanov et al. **Deep image prior**. IJCV'20. https://arxiv.org/abs/1711.10925

$$\underbrace{\mathbf{y}}_{\text{blurry and noisy image}} = \overbrace{\mathbf{k}}^{\text{blur kernel}} * \underbrace{\mathbf{x}}_{\text{clean image}} + \overbrace{\mathbf{n}}^{\text{noise}}$$

Mostly due to optical deficiencies (e.g., defocus) and motions

Given $\mathbf{y}$,
recover $\mathbf{x}$ (and/or $\mathbf{k}$)

Also **Blind Deconvolution**



Principled data-knowledge tradeoff

Knowledge

Building in prior

Thrust B: How Should Domain Knowledge Be Incorporated into Supervised Machine Learning?

The central question for this thrust is "which knowledge should be leveraged in SciML, and how should this knowledge be included?" Any answers will naturally depend on the SciML task and computational budgets, thus mirroring standard considerations in traditional scientific computing.

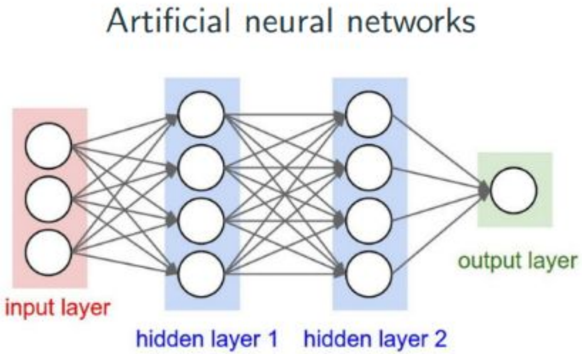
Hard Constraints. One research avenue involves incorporation of domain knowledge through imposition of constraints that cannot be violated. These hard constraints could be enforced during training, replacing what typically is an unconstrained optimization problem with a constrained one. In general, such constraints could involve simulations or highly nonlinear functions of the training parameters. Therefore, there is a need to identify particular cases when constraint qualification conditions can be ensured as these conditions are necessary regularity conditions for constrained optimization [57–59]. Although incorporating constraints during training generally makes maximal use of training data, there may be additional opportunities to employ constraints at the time of prediction (e.g., by projecting predictions onto the region induced by the constraints).

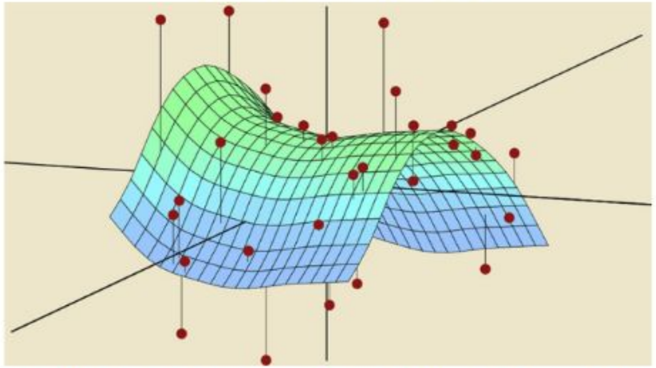
Soft Constraints. A similar avenue for incorporating domain knowledge involves modifying the objective function (soft constraints) used in training. It is understood that ML loss function selection should be guided by the task and data. Therefore, opportunities exist for developing loss functions that incorporate domain knowledge and analyzing the resulting impact on solvability

Ref https://www.osti.gov/servlets/purl/1478744

Domain-Aware Scientific Machine Learning

When DL meets constraints





used to approximate nonlinear functions

Unconstrained optimization

$$\min_{\boldsymbol{w}_i' s, \boldsymbol{b}_i' s} \frac{1}{n} \sum_{i=1}^{n} \ell\left[\boldsymbol{y}_i, \left\{\mathsf{NN}\left(\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k, b_1, \ldots, b_k\right)\right\}\left(\boldsymbol{x}_i\right)\right]$$

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x})$$

“Solved”

Constrained optimization

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s. t. } \; g(\boldsymbol{x}) \leq \mathbf{0}$$

largely “unsolved”

GAPS

Constrained optimization

$$\min_{\boldsymbol{x}} \; f(\boldsymbol{x}) \quad \text{s.t. } g(\boldsymbol{x}) \leq \mathbf{0}$$

largely "unsolved"

An imaginary chat between a PhD student working in deep learning (**DLP**) and a PhD student working in optimization (**OP**)

DLP: Man, I've solved a constrained DL problem recently
OP: Oh, that's a hard problem
DLP: Really? I actually did it
OP: How?
DLP: My problem is $\min_x \; f(x)$, s.t. $g(x) \leq 0$. I put $g(x)$ as a penalty and then call ADAM
OP: Are you sure it works?
DLP: Yes, the performance is improved and my paper is published at ICML
OP: Why don't you try augmented Lagrangian methods?
DLP: No implementation in Pytorch. Is it possible we work out some theory about my method?
OP: I think it's hard. It's not convex

DL with nontrivial constraints: many pitfalls

- **Robustness evaluation**
- Imbalanced learning
- Topology optimization

Deep Learning with Nontrivial Constraints: Methods and Applications

Chuan He[1] , **Ryan Devera**[1] , **Wenjie Zhang**[1] , **Ying Cui**[2] , **Zhaosong Lu**[3] and **Ju Sun**[1]
[1]Computer Science and Engineering, University of Minnesota
[2]Industrial Engineering and Operations Research, University of California, Berkeley
[3]Industrial and Systems Engineering, University of Minnesota
{he000233, dever120, zhan7867}@umn.edu, yingcui@berkeley.edu, {zhaosong, jusun}@umn.edu

Robustness evaluation: penalty methods for complicated d (perceptual attack)

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$

$$\text{s.t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \leq \varepsilon\ , \quad \boldsymbol{x}' \in [0,1]^n$$

$$d(\boldsymbol{x}, \boldsymbol{x}') \doteq \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2$$

$$\text{where} \quad \phi(\boldsymbol{x}) \doteq [\ \widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x})\ ]$$

perceptual distance

Projection onto the constraint is complicated

Penalty methods

$$\max_{\widetilde{\mathbf{x}}} \quad \mathcal{L}(f(\widetilde{\mathbf{x}}), y) - \lambda \max\left(0, \|\phi(\widetilde{\mathbf{x}}) - \phi(\mathbf{x})\|_2 - \epsilon\right)$$

Solve it for each fixed λ and then increase λ

Algorithm 2 Lagrangian Perceptual Attack (LPA)

```
procedure LPA(classifier network f(·), LPIPS distance d(·,·), input x, label y, bound ε)
    λ ← 0.01
    x̃ ← x + 0.01 ∗ N(0, 1)          ▷ initialize perturbations with random Gaussian noise
    for i in 1, ..., S do           ▷ we use S = 5 iterations to search for the best value of λ
        for t in 1, ..., T do                                ▷ T is the number of steps
            Δ ← ∇_x̃ [L(f(x̃), y) − λ max(0, d(x̃, x) − ε)]      ▷ take the gradient of (5)
            Δ̂ = Δ/‖Δ‖₂                                         ▷ normalize the gradient
            η = ε ∗ (0.1)^(t/T)                   ▷ the step size η decays exponentially
            m ← d(x̃, x̃ + hΔ̂)/h    ▷ m ≈ derivative of d(x̃, ·) in the direction of Δ̂; h = 0.1
            x̃ ← x̃ + (η/m)Δ̂                       ▷ take a step of size η in LPIPS distance
        end for
        if d(x̃, x) > ε then
            λ ← 10λ                   ▷ increase λ if the attack goes outside the bound
        end if
    end for
    x̃ ← PROJECT(d, x̃, x, ε)
    return x̃
end procedure
```

Ref Perceptual adversarial robustness: Defense against unseen threat models. Laidlaw, C., Singla, S., & Feizi, S. https://arxiv.org/abs/2006.12655

Problem with penalty methods

Method	cross-entropy loss		margin loss	
	Viol. (%) ↓	Att. Succ. (%) ↑	Viol. (%) ↓	Att. Succ. (%) ↑
Fast-LPA	73.8	3.54	41.6	56.8
LPA	**0.00**	80.5	**0.00**	97.0
PPGD	5.44	25.5	**0.00**	38.5
PWCF (ours)	0.62	93.6	**0.00**	100

$$\max_{\boldsymbol{x}'} \ell\left(\boldsymbol{y}, f_{\boldsymbol{\theta}}(\boldsymbol{x}')\right)$$
$$\text{s.t. } d\left(\boldsymbol{x}, \boldsymbol{x}'\right) \le \varepsilon, \quad \boldsymbol{x}' \in [0,1]^n$$

$$d(\boldsymbol{x}, \boldsymbol{x}') \doteq \|\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')\|_2$$
$$\text{where} \quad \phi(\boldsymbol{x}) \doteq [\, \widehat{g}_1(\boldsymbol{x}), \ldots, \widehat{g}_L(\boldsymbol{x}) \,]$$

LPA, Fast-LPA: penalty methods **PPGD**: Projected gradient descent

Penalty methods tend to encounter **large constraint violation** (i.e., infeasible solution, known in optimization theory) or **suboptimal solution**

PWCF, an optimizer with a principled stopping criterion on **stationarity & feasibility**

Ref Optimization and Optimizers for Adversarial Robustness. Liang, H., Liang, B., Peng, L., Cui, Y., Mitchell, T., & Sun, J. arXiv preprint arXiv:2303.13401.

Key algorithm

GRANSO

http://www.timmitchell.com/software/GRANSO/

Nonconvex, nonsmooth, constrained

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}), \quad \text{s.t. } c_i(\boldsymbol{x}) \leq 0, \ \forall\, i \in \mathcal{I}; \ \ c_i(\boldsymbol{x}) = 0, \ \forall\, i \in \mathcal{E}.$$

Penalty sequential quadratic programming (P-SQP)

$$\begin{aligned} \min_{d \in \mathbb{R}^n,\ s \in \mathbb{R}^p} \quad & \mu(f(x_k) + \nabla f(x_k)^\mathsf{T} d) + e^\mathsf{T} s + \frac{1}{2} d^\mathsf{T} H_k d \\ \text{s.t.} \quad & c(x_k) + \nabla c(x_k)^\mathsf{T} d \leq s, \quad s \geq 0, \end{aligned}$$

Ref: ***Curtis, Frank E., Tim Mitchell, and Michael L. Overton****. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.*

Algorithm highlights

Steering strategy for the penalty parameter

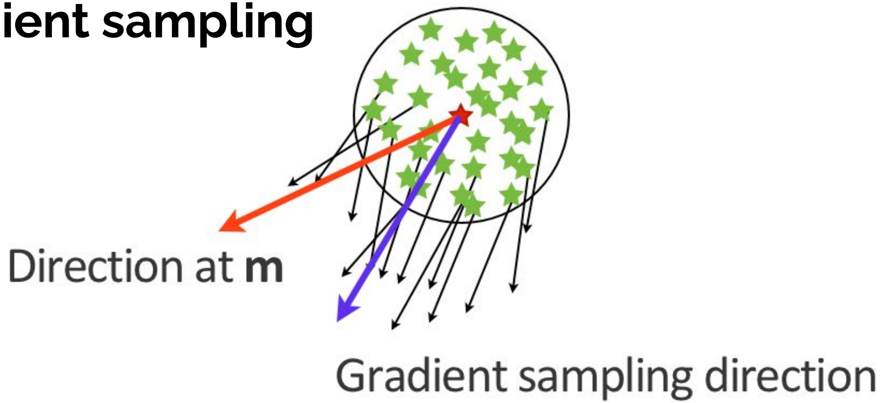
If feasibility improvement is insufficient : $l_\delta(d_k; x_k) < c_v v(x_k)$

Stationarity based on (approximate) gradient sampling

$$G_k := \begin{bmatrix} \nabla f(x^k) & \nabla f(x^{k,1}) & \cdots & \nabla f(x^{k,m}) \end{bmatrix}$$

$$\min_{\lambda \in \mathbb{R}^{m+1}} \ \frac{1}{2}\|G_k \lambda\|_2^2$$

$$\text{s.t. } \mathbb{1}^T \lambda = 1, \ \lambda \geq 0$$



Key take-away

GRANSO

- Principled stopping criterion and line search, to obtain a **solution with certificate** (stationarity & feasibility check)
- Quasi-newton style method for fast convergence, i.e., **reasonable speed and high-precision solution**

Ref Curtis, Frank E., Tim Mitchell, and Michael L. Overton. "A BFGS-SQP method for nonsmooth, nonconvex, constrained optimization and its evaluation using relative minimization profiles." Optimization Methods and Software 32.1 (2017): 148-181.

Limitations of GRANSO

```
% Gradient of inner product with respect to A
f_grad        = imag((conj(Bty)*Cx.')/(y'*x));
f_grad        = f_grad(:);

% Gradient of inner product with respect to A
ci_grad       = real((conj(Bty)*Cx.')/(y'*x));
ci_grad       = ci_grad(:);
```

analytical gradients required

```
p             = size(B,2);
m             = size(C,1);
X             = reshape(x,p,m);
```

vector variables only

Lack of Auto-Differentiation

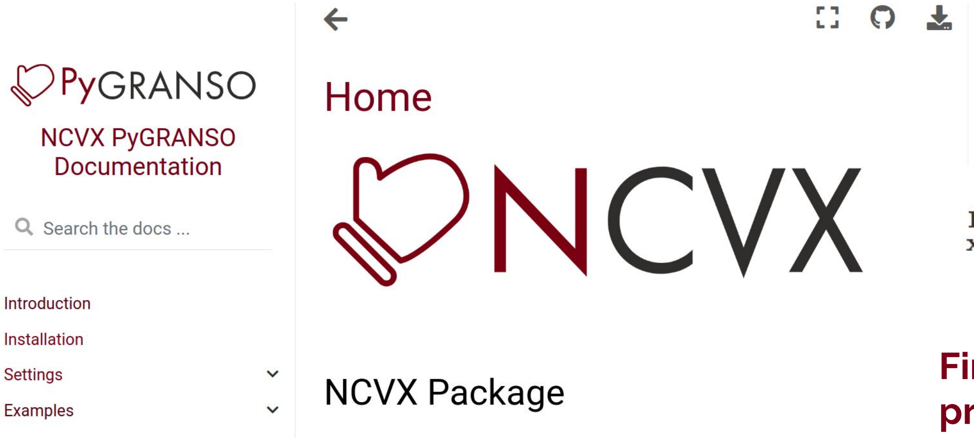
Lack of GPU Support

No native support of tensor variables

⇒ impossible to do deep learning with GRANSO

GRANSO meets PyTorch

GRANSO + PyTorch



$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \text{ s.t. } c_i(\mathbf{x}) \le 0, \forall i \in \mathcal{I};\ c_i(\mathbf{x}) = 0, \forall i \in \mathcal{E}$$

First general-purpose solver for constrained DL problems

NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning

Buyun Liang, Tim Mitchell, Ju Sun

Example 1: Support Vector Machine (SVM)

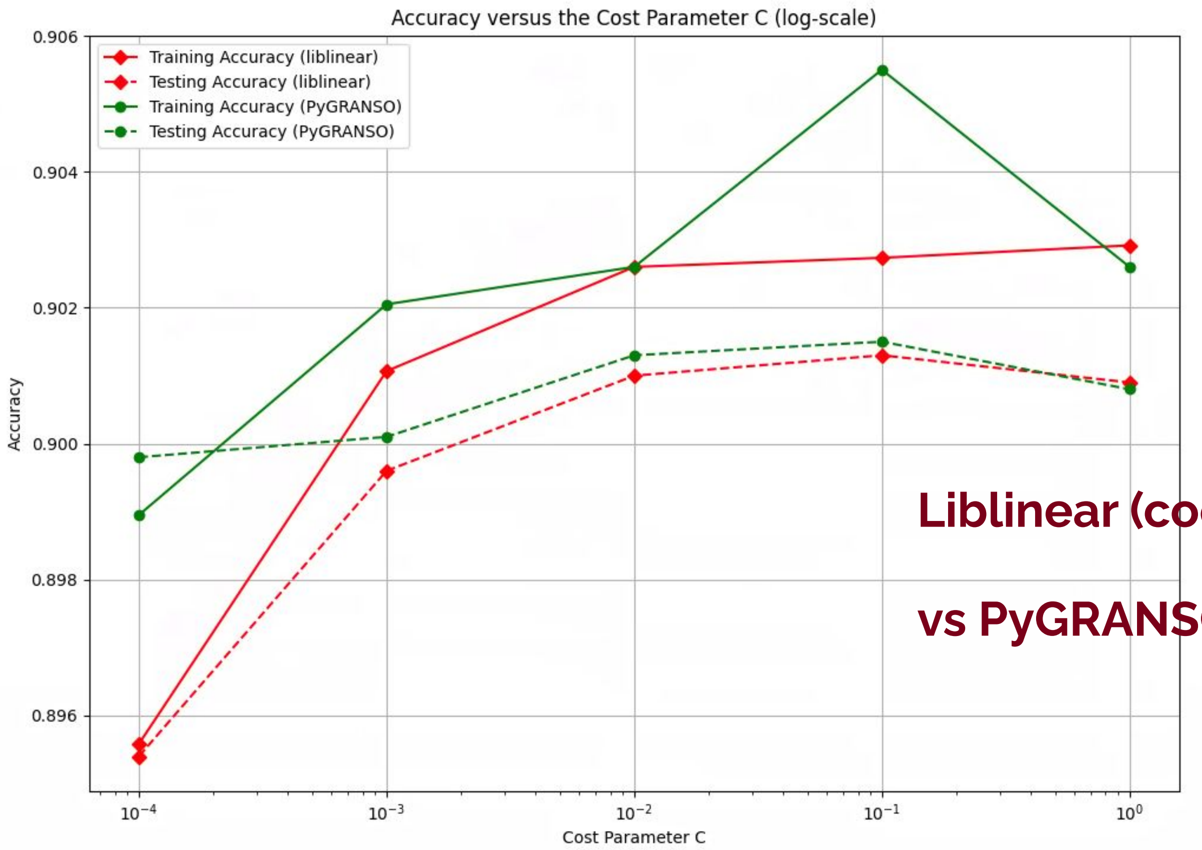
Soft-margin SVM

$$\min_{\boldsymbol{w},b,\zeta} \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{n}\zeta_i$$

$$\text{s.t.}\quad y_i\left(\boldsymbol{w}^\intercal \boldsymbol{x}_i + b\right) \geq 1 - \zeta_i,\ \zeta_i \geq 0\ \ \forall i = 1, \ldots, n$$

```python
def comb_fn(X_struct):
    # obtain optimization variables
    w = X_struct.w
    b = X_struct.b
    zeta = X_struct.zeta
    # objective function
    f = 0.5*w.T@w + C*torch.sum(zeta)
    # inequality constraints
    ci = pygransoStruct()
    ci.c1 = 1 - zeta - y*(x@w+b)
    ci.c2 = -zeta
    # equality constraint
    ce = None
    return [f,ci,ce]
# specify optimization variables
var_in = {"w": [d,1], "b": [1,1], "zeta": [n,1]}
# pygranso main algorithm
soln = pygranso(var_in,comb_fn)
```

Binary classification (odd vs even digits) on MNIST dataset



Liblinear (coordinate descent)

vs PyGRANSO

Example 2: Robustness—min formulation

$$
\begin{aligned}
\min_{\boldsymbol{x}'} \quad & d(\boldsymbol{x}, \boldsymbol{x}') \\
\text{s.t.} \quad & \max_{\ell \neq c} f_{\boldsymbol{\theta}}^{\ell}(\boldsymbol{x}') \geq f_{\boldsymbol{\theta}}^{c}(\boldsymbol{x}') \\
& \boldsymbol{x}' \in [0,1]^n
\end{aligned}
$$

```python
def comb_fn(X_struct):
    # obtain optimization variables
    x_prime = X_struct.x_prime
    # objective function
    f = d(x,x_prime)
    # inequality constraints
    ci = pygransoStruct()
    f_theta_all = f_theta(x_prime)
    fy = f_theta_all[:,y] # true class output
    # output execpt true class
    fi = torch.hstack((f_theta_all[:,:y],f_theta_all[:,y+1:]))
    ci.c1 = fy - torch.max(fi)
    ci.c2 = -x_prime
    ci.c3 = x_prime-1
    # equality constraint
    ce = None
    return [f,ci,ce]
# specify optimization variable (tensor)
var_in = {"x_prime": list(x.shape)}
# pygranso main algorithm
soln = pygranso(var_in,comb_fn)
```

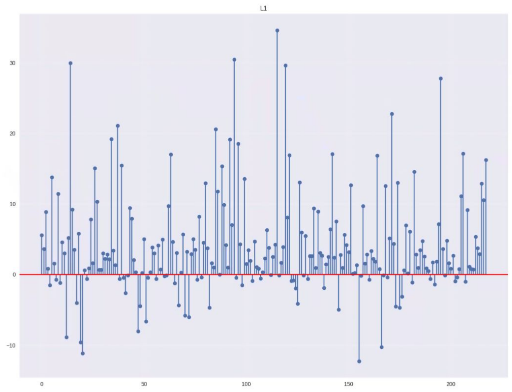
CIFAR10 dataset

Compared with FAB [iterative constraint linearization + projected gradient]

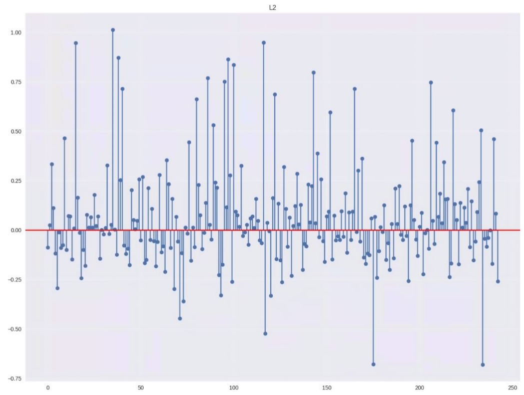
https://github.com/fra31/auto-attack

$$
\begin{aligned}
\min_{\boldsymbol{x}'} \quad & d(\boldsymbol{x}, \boldsymbol{x}') \\
\text{s.t.} \quad & \max_{\ell \neq c} f_{\boldsymbol{\theta}}^{\ell}(\boldsymbol{x}') \geq f_{\boldsymbol{\theta}}^{c}(\boldsymbol{x}') \\
& \boldsymbol{x}' \in [0,1]^n
\end{aligned}
$$

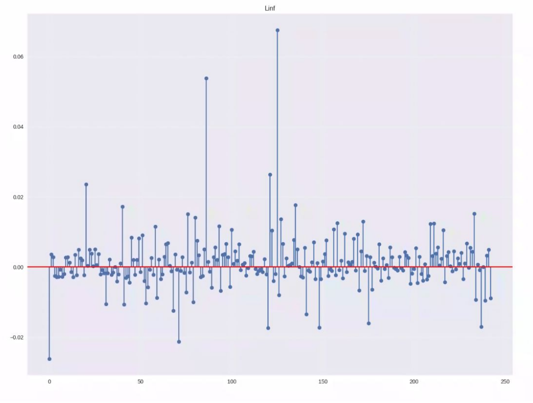
X-axis: image index; Y-axis: PyGRANSO radius - FAB radius



L1 attack



L2 attack



Linf attack

https://ncvx.org/

Many others

NCVX PyGRANSO Documentation

Search the docs ...

- Introduction
- Installation
- Settings
- Examples
  - Rosenbrock
  - Eigenvalue Optimization
  - Dictionary Learning
  - Nonlinear Feasibility Problem
  - Sphere Manifold
  - Trace Optimization
  - Robust PCA
  - Generalized LASSO
  - Logistic Regression
  - LeNet5
  - Perceptual Attack
  - Orthogonal RNN
- Highlights

Home

NCVX

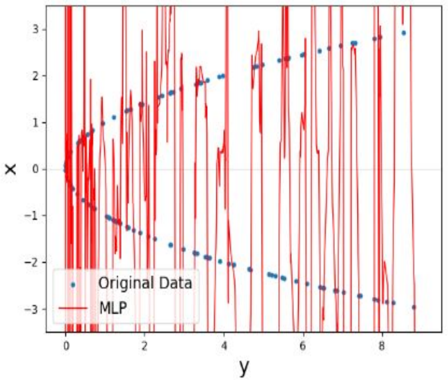
NCVX Package

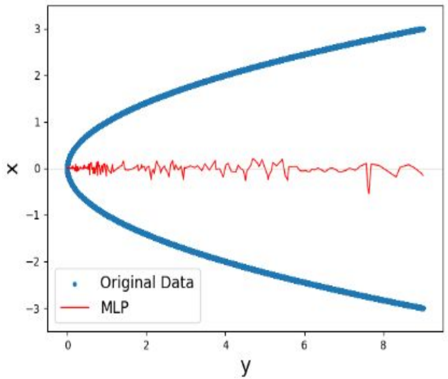
NCVX (NonConVeX) is a user-friendly and scalable python software package targeting general nonsmooth NCVX problems with nonsmooth constraints. **NCVX** is being developed by GLOVEX at the Department of Computer Science & Engineering, University of Minnesota, Twin Cities.

The initial release of **NCVX** contains the solver **PyGRANSO**, a PyTorch-enabled port of GRANSO incorporating auto-differentiation, GPU acceleration, tensor input, and support for new QP solvers. As a highlight, **PyGRANSO** can solve general constrained deep learning problems, the first of its kind.

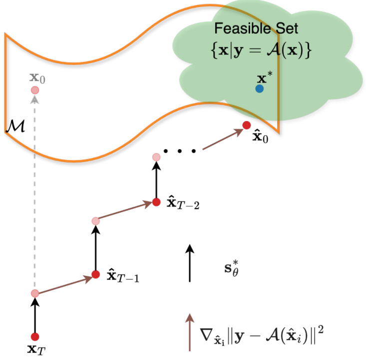
PyGRANSO

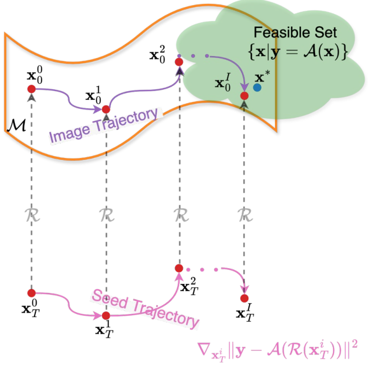
Data-driven methods for SIPs





Story I: More could be less





Story II: Don't be too Bayesian

Single-instance methods for SIPs

Story III: Benefit from DL with a single data point

Deep image prior (DIP) $\mathbf{x} \approx G_\theta(\mathbf{z})$ G_θ (and $\mathbf{z}$) trainable

$$\min_{\mathbf{x}} \underbrace{\ell(\mathbf{y},\, f(\mathbf{x}))}_{\text{data fitting}} + \lambda \underbrace{\mathrm{R}(\mathbf{x})}_{\text{regularizer}}$$

$$\Downarrow$$

$$\min_{\theta} \ell(\mathbf{y},\, f \circ G_\theta(\mathbf{z})) + \lambda R \circ G_\theta(\mathbf{z})$$

No extra training data!

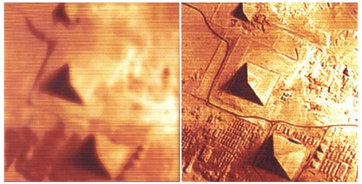
Ulyanov et al. **Deep image prior**. IJCV'20. https://arxiv.org/abs/1711.10925

$$\underbrace{\mathbf{y}}_{\text{blurry and noisy image}} = \overbrace{\mathbf{k}}^{\text{blur kernel}} * \underbrace{\mathbf{x}}_{\text{clean image}} + \overbrace{\mathbf{n}}^{\text{noise}}$$

Mostly due to optical deficiencies (e.g., defocus) and motions

Given $\mathbf{y}$,
recover $\mathbf{x}$ (and/or $\mathbf{k}$)

Also **Blind Deconvolution**



A (the?) tool for DL with nontrivial constraints

GRANSO + PyTorch

PyGRANSO

NCVX PyGRANSO Documentation

Search the docs ...

Introduction

Installation

Settings

Examples

Home

NCVX

NCVX Package

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}), \text{ s.t. } c_i(\mathbf{x}) \le 0, \forall i \in \mathcal{I};\; c_i(\mathbf{x}) = 0, \forall i \in \mathcal{E}$$

First general-purpose solver for constrained DL problems

NCVX: A General-Purpose Optimization Solver for Constrained Machine and Deep Learning

Buyun Liang, Tim Mitchell, Ju Sun