Learned wave-based imaging – variational inference at scale

Felix J. Herrmann, June 29, 2021

Delft University of Technology

Released to public domain under Creative Commons license type BY (https://creativecommons.org/licenses/by/4.0)
Copyright (c) 2021, Felix J. Herrmann (Georgia Tech)
Learned wave-based imaging – variational inference at scale

Ali Siahkoohi, Rafael Orozco, Gabrio Rizutti*, Philipp Witte**, and Mathias Louboutin

* now at Utrecht University
** now at Microsoft Research

Released to public domain under Creative Commons license type BY (https://creativecommons.org/licenses/by/4.0)
Copyright (c) 2021, Felix J. Herrmann (Georgia Tech)
Research program
computational imaging @SLIM

Interested in

► solving imaging & monitoring problems that involve physics (PDEs, waves)
► tackling challenges that come w/ 3D
► seismic & medical imaging

Use techniques from

► high performance computing
► randomized linear algebra & stochastic optimization
► machine learning & Bayesian inference

https://slim.gatech.edu
Seismic inverse problems

Estimate unknown subsurface properties from seismic (surface) data:

- image geological structures/discontinuities
- estimate physical rock properties (wavespeed, density, etc.)
- turn around times order 1 year
Monitoring CO2 plumes

Simulations 2 phase flow of CO2

- monitor w/ time-lapse seismic imaging
- optimize sequestration & lower cost & risk


Converted to wave speeds
Joint recovery model
Medical inverse problems

Estimate unknown tissue properties from (omnidirectional) ultrasound data:

- image healthy & malignant tissue
- estimate physical tissue properties (wavespeed, density, etc.)
- turn around times seconds to minutes

3D wave simulation
Monitoring thermal ablation
Forward problem

The *forward* problem:

\[ F[m] \cdot q = P_r A[m]^{-1} P_s^T q \]

Discretized acoustic wave equation:

\[ A[m] = m \otimes \frac{\partial^2}{\partial t^2} - \nabla^2 \]

Solve via finite-difference time-stepping:

\[ u^{n+1} = [2 + \frac{\Delta t^2}{m} \otimes L]u^n - u^{n-1} + \frac{\Delta t^2}{m} \otimes P_s^T q^{n+1} \]
Inverse problem

The inverse problem:

\[
\min_m \Phi(m) = \frac{1}{2} \sum_{i=1}^{n_s} \|d_i - F[m]q_i\|_2^2
\]

Sensitivities w.r.t. model parameters:

\[
\frac{\partial F[m]q}{\partial m}
\]

\[
J = -P_r A[m]^{-1} \text{diag} \left( \frac{\partial A[m]}{\partial m} A[m]^{-1} P_s^T q \right)
\]

Gradient of objective function via backpropagation:

\[
g = \sum_{i=1}^{n_s} J^T (F[m]q_i - d_i)
\]
Challenges

risk mitigation

Increased calls for systematic assessment of uncertainty

- propagate errors in observed data to tasks on the image (e.g. automatic horizon tracking)
- Bayesian Variational Inference—i.e., *distribution* learning

Hurdles:

- Markov Chain Monte Carlo methods for Uncertainty Quantification are too expensive
- tackle complexity numerical PDE solves w/ abstractions & automatic code generation
- data-driven Variational Inference calls for integration physics & machine learning
- access to training data
The Julia Devito Inversion framework (JUDI)

Julia
- linear operators, data containers, IO
- parallel modeling function
- parallelization: distribute sources, data
- serial modeling function
- interface to Devito (Python)

Python
- Devito: symbolic definition of PDE
- automatic code generation and JIT compilation

C
- solve PDE w/ OpenMP parallelism

\[ g = J'(Pr \cdot A_{\text{inv}} \cdot P_s \cdot q - d_{\text{obs}}) \]

# Symbolic wavefield
\[ u = \text{TimeFunction}(\text{name}='u', \text{space}_\text{order}=16) \]

# Acoustic wave equation
\[ \text{pde} = \text{model.m} \ast u.d\text{t}^2 - u.\text{laplace} \]
Serverless seismic imaging on Azure

“Small” 3D Imaging case study w/ Devito DSL + JUDI

- Data set: 1,500 source locations (~2.1 TB data)
- Model: 10 x 10 x 3.325 km (270 million unknowns)
- PDE: tilted transversely isotropic (TTI) wave equation, 3,500 time steps
- Cost: < 10,000$ on 100 E64/E64s instances (2 VMs per gradient with MPI)
- Peak performance: 140 TFLOP/s
- cost single image is comparable to training a large NN

Example 2: Deep Learning

JUDI operators W/ deep learning packages (e.g. Flux.jl):

```
network(x) = W2*(J*(W1*x + b1)) + b2
# Or:
network(x) = conv2((F(conv1(x)))
loss(x, y) = Flux.mse(network(x), y)
```

Once again, abstractions pay off:

- No need to backpropagate through solvers using automatic differentiation (AD)
- Backpropagation through JUDI operators

```
@adjoint *(J::judiJacobian, x) = *(J, x), Δ -> (nothing, transpose(J) * Δ)
```
Code Repositories – slimgroup

General purpose examples
- **JUDI.jl** – the Julia Devito Inversion framework
- **SetIntersectionProjection.jl** – projections onto intersections & sums of sets
- **InvertibleNetworks.jl** – building blocks for invertible neural networks
- **JOLI.jl** – serial and distributed linear operators in Julia

Specialized examples
- **FastApproximateInference.jl** – variational inference for inverse problems
- **XConv** – Julia/Python code for memory efficient CNNs
- **TimeProbeSeismic.jl** – low memory WE based inversion
Problem setup

Find $x$ such that:

$$y = F(x) + \epsilon$$

observed data $y$

(non-Gaussian) noise $\epsilon$

expensive (nonlinear) forward operator $F$
Stochastic gradient Langevin dynamics

\[ w_{k+1} = w_k - \alpha_k M_k \nabla_w \left( \frac{n_s}{2\sigma^2} \|d_i - J[m_0, q_i]g(z, w_k)\|^2_2 + \frac{\lambda^2}{2} \|w_k\|^2_2 \right) + \xi_k, \]

\[ \xi_k \sim \mathcal{N}(0, \alpha_k M_k) \]

sampling the posterior distribution

requires at least 10k iterations


Conditional mean estimate

Pointwise standard deviation, normalized by the envelope of conditional mean

Uncertainty in horizon tracking due to uncertainties in imaging


Xinming Wu and Sergey Fomel. xinwucwp/mhe. 2018. URL: https://github.com/xinwucwp/mhe.

Inverse problems and Bayesian inference

Represent the solution as a distribution over the model space

i.e., posterior distribution

Bayes’ rule

\[ p(\mathbf{x} | \mathbf{y}) \propto p(\mathbf{y} | \mathbf{x}) \, p(\mathbf{x}) \]

posterior distribution \( p(\mathbf{x} | \mathbf{y}) \)
likelihood function \( p(\mathbf{y} | \mathbf{x}) \)
prior distribution \( p(\mathbf{x}) \)

---

Maximum likelihood estimator

\[
\arg \max_x p(y \mid x) = \arg \min_x \frac{1}{2\sigma^2} \| y - F(x) \|_2^2
\]

no high-dimensional integration/sampling

no prior/regularization

---

Noisy data

Noise-free (left) and noisy (right) linearized data—SNR: $-8.75 \text{ dB}$
Maximum likelihood

Estimation with no regularization/prior

SNR 8.25 dB
True reflectivity

\[ \delta m \text{— "true" reflectivity model obtain from Parihaka dataset} \]
Maximum a posteriori estimate

$$\arg\max_x \ p(x \mid y) = \arg\min_x \left[ -\log p(y \mid x) - \log p(x) \right]$$

$$= \arg\min_x \left[ \frac{1}{2\sigma^2} \|y - F(x)\|_2^2 - \log p(x) \right]$$

no high-dimensional integration/sampling

uses prior information

requires choosing prior

Maximum a posteriori estimate (MAP)

MAP estimate—SNR: 8.77 dB
True reflectivity

\[ \delta m \] — "true" reflectivity model obtained from Parihaka dataset
Conditional mean estimate

\[ \mathbb{E}[x \mid y] = \int x\ p(x \mid y)\ \mathrm{d}x \]

uses prior information

high-dimensional integration/sampling

requires choosing prior

---

Conditional mean estimate—SNR: 9.66 dB
True reflectivity

$\delta m$—"true" reflectivity model obtain from Parihaka dataset
Sampling the posterior

Markov chain Monte Carlo (MCMC) sampling guarantees of producing (asymptotically) exact samples

high-dimensional integration/sampling

costs associated with the forward operator

requires choosing a prior distribution


Variational Inference

Approximate distribution $p$ by minimizing the Kullback-Leibner (KL) divergence

$$\mathbb{D}_{KL}(p \mid\mid p_\theta) = \int p(x) \log \frac{p(x)}{p_\theta(x)} \, dx$$

$$= \mathbb{E}_{x \sim p(x)} \left[ -\log p_\theta(x) + \log p(x) \right] ,$$

parametric distribution $p_\theta$

---

Variational Inference

Objective of variational inference is to solve

$$\theta^* = \arg\min_{\theta} D_{KL}(p \parallel p_\theta)$$

$p_\theta$ designed to be easily sampled
Sampling the posterior

Variational inference

- approximate the posterior with a parametric and easy-to-sample distribution
- sampling is turned into an optimization problem
- known to scale better than MCMC in high-dimensional problems
- requires choosing a prior distribution

---


Generative models

probabilistic models to characterize an unknown distribution
Generative models

\[ p_\theta(\mathbf{x}) \approx p(\mathbf{x}), \quad \mathbf{x} \in \mathcal{X} \]

high dimensional unknown density \( p(\mathbf{x}) \)

generative model \( p_\theta(\mathbf{x}) \), parameterized by \( \theta \)

\( \theta \) estimated via available samples \( \{\mathbf{x}_i\}_{i=1}^n \sim p(\mathbf{x}) \)

Inference with NFs

Sampling from $p(x)$

$$G_{\theta}^{-1}(z) \sim p(x), \quad z \sim N(0, I),$$

and fast density estimation via the change-of-variable formula,

$$p_{\theta}(x) = p_{z}(G_{\theta}(x)) \left| \det \nabla_{x} G_{\theta}(x) \right|$$

Training NFs

\[ \theta^* = \arg \min_{\theta} \mathbb{E}_{x \sim p(x)} \left[ -\log p_z(G_{\theta}(x)) - \log \left| \det \nabla_x G_{\theta}(x^{(i)}) \right| \right] \]

\[ \approx \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{1}{2} \| G_{\theta}(x_i) \|_2^2 - \log \left| \det \nabla_x G_{\theta}(x_i) \right| \right]. \]

expectation estimated via available samples \( \{x_i\}_{i=1}^{N} \sim p(x) \)
Posterior inference with NFs

sampling the posterior distribution directly via NFs
Purely data-driven approach

\[
\min_\theta \frac{1}{n} \sum_{i=1}^{n} \left[ \frac{1}{2} \| G_\theta(y_i, x_i) \|_2^2 - \log | \det \nabla_{y,x} G_\theta(y_i, x_i) | \right]
\]

\[
G_\theta(y, x) = \begin{bmatrix} G_{\theta_y}(y) \\ G_{\theta_x}(y, x) \end{bmatrix}, \quad \theta = \begin{bmatrix} \theta_y \\ \theta_x \end{bmatrix}
\]

conditional NF \( G_\theta : \mathcal{Y} \times \mathcal{X} \rightarrow \mathcal{Z}_y \times \mathcal{Z}_x \)

expectation estimated via training pairs \( \{y_i, x_i\}_{i=1}^{n} \sim p_{y,x}(y,x) \)
Inference with the data-driven approach

\[ \mathbf{z}_y = G_{\theta_y}(\mathbf{y}) \]

Inference with the data-driven approach

\[ G_{\theta_x}^{-1}(G_{\theta_y}(y), z_x) \sim p(x \mid y), \quad z_x \sim N(0, I) \]

Seismic imaging example
Training dataset for the data-driven approach

Training pairs:

\[ \{y_i, x_i\}_{i=1}^n \sim p(y, x) \]

where

\[ (y_i, x_i) = (J^T(J\delta m_i + \epsilon_i), \delta m_i) \]

---

Some data pairs from the training set

Low-fidelity reverse-time migrated image, $\mathbf{y}$
Low-fidelity reverse-time migrated image, $y$
Bayesian inference w/ the data-driven approach
Low-fidelity reverse-time migrated image, y
Samples from the posterior

\[ G^{-1}_{θx}(G_{θy}(y), z) \sim p(x \mid y), \quad z \sim N(0, I) \]
Conditional mean estimate

\[ \mathbb{E}[x \mid y] = \int x p(x \mid y) \, dx \]
Conditional mean, $\delta m_{CM}$
Pointwise standard deviation

Depth (km)

Horizontal distance (km)

- $3 \times 10^2$
- $2 \times 10^2$
- $10^2$
- $6 \times 10^1$
Medical imaging example
Photoacoustic imaging

Junjie Yao and Lihong Wang, WUSTL
Image reproduced from © 2007 User:Bme591wikiproject, Schematic illustration of photoacoustic imaging
Photoacoustic operator $A$

Solve initial value PDE for acoustic pressure $u$

\[ m \frac{\partial}{\partial t^2} u(x, t) - \Delta u(x, t) = 0 \]

\[ u(x, 0) = p(x) \]

\[ \frac{\partial}{\partial t} u(x, 0) = 0 \]

Restrict field $u$, to limited view, subsampled receivers
Training dataset for the data-driven approach

Training pairs

\[ \{y_i, x_i\}_{i=1}^n \sim p(y, x) \]

where

\[ (y_i, x_i) = (A^\top (Ap_i + \epsilon_i), p_i) \]

---

Some data pairs from the training set
True photoacoustic pressure, $p$
True photoacoustic pressure, $p$

X Position [mm]

Z Position [mm]

Photoacoustic Source Intensity

Receivers
Adjoint solution, $\mathbf{p}_{\text{adj}}$

Photoacoustic Source Intensity

Z Position [mm]

X Position [mm]
Bayesian inference with the data-driven approach
Conditional mean, $p_{cm}$

- X Position [mm]
- Z Position [mm]

Photoacoustic Source Intensity

Values range from 0.00 to 1.00
Purely data-driven approach

- learns the prior distribution from training data
- samples from the posterior virtually for free in test time
- not specific to one observation $y$
- needs supervised pairs of model and data
- heavily relies on the training data to generalize
- not directly tied to physics/data
Purely physics-based approach

\[
\arg\min_{\phi} \mathbb{D}_{KL}(p_\phi \mid \mid p(\cdot \mid y)) = \arg\min_{\phi} \mathbb{E}_{x \sim p_\phi(x)} \left[ -\log p(x \mid y) + \log p_\phi(x) \right]
\]

\[
= \arg\min_{\phi} \mathbb{E}_{z \sim \mathcal{N}(0, I)} \left[ \frac{1}{2\sigma^2} \| F(T_\phi(z)) - y \|_2^2 - \log p(T_\phi(z)) - \log \left| \det \nabla_z T_\phi(z) \right| \right]
\]

\[\text{NF } T_\phi : \mathcal{Z} \rightarrow \mathcal{X}\]

prior distribution \( p(x) \)

specific to one observation, \( y \)


Purely physics-based approach

tied to the physics and data
no training data needed

requires choosing a prior distribution \( p(x) \)
repeated evaluations of \( F \) and \( \nabla F^T \)
specific to one observation, \( y \)
Multi-fidelity preconditioned scheme

exploit the information in the pretrained conditional NF (data-driven approach)

tie the results to physics and data
Multi-fidelity preconditioned scheme

Use the density encoded by the pretrained conditional NF as a (conditional) prior

\[ p_{\text{prior}}(x) := p_z(G_{\theta_x}(y, x)) \left| \det \nabla_x G_{\theta_x}(y, x) \right| \]

allows for using a data-driven prior

removes the bias introduced by hand-crafted priors

can be used as a prior density in inverse problems

Preconditioned MAP estimation

\[
\min_z \frac{1}{2\sigma^2} \|F(T_{\theta_x}(z; y)) - y\|^2_2 + \frac{1}{2} \|z\|^2_2
\]

initializing \( z = 0 \) acts as an implicit prior

since \( T \) is invertible, in principle it can represent any unknown \( x \in \mathcal{X} \)

can be used if \( y \) is out-of-distribution

final estimate is tied to the physics/data

Seismic imaging example
Out of distribution

Low-fidelity reverse-time migrated image, $\mathbf{y}$
Maximum a posteriori estimate $\lambda = 0$
Maximum a posteriori estimate $\lambda = 1$
Maximum Likelihood Estimate
## Signal-to-noise ratio comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>SNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTM</td>
<td>−54.83</td>
</tr>
<tr>
<td>MLE</td>
<td>7.40</td>
</tr>
<tr>
<td>Preconditioned MAP</td>
<td>11.29</td>
</tr>
<tr>
<td>Data-driven CM</td>
<td>4.34</td>
</tr>
</tbody>
</table>

SNR comparison
Preconditioned physics-based approach

\[
\min_{\theta_x} \mathbb{E}_{\mathbf{z} \sim \mathcal{N}(0, I)} \left[ \frac{1}{2\sigma^2} \| F(T_{\theta_x}(\mathbf{z}; \mathbf{y})) - \mathbf{y} \|^2_2 - \log p_{\text{prior}}(T_{\theta_x}(\mathbf{z}; \mathbf{y})) - \log \left| \det \nabla_{\mathbf{z}} T_{\theta_x}(\mathbf{z}; \mathbf{y}) \right| \right]
\]

transfer learning \( T_{\theta_x}(\mathbf{z}; \mathbf{y}) := G_{\theta_x}^{-1}(G_{\theta_y}(\mathbf{y}), \mathbf{z}) \)

- corrects for errors due to out-of-distribution data
- less evaluations of \( F \) and \( \nabla F^T \)

learned (conditional) prior density, \( p_{\text{prior}}(\mathbf{x}) := p_{\mathbf{z}}(G_{\theta_x}(\mathbf{y}, \mathbf{x})) \left| \det \nabla_{\mathbf{x}} G_{\theta_x}(\mathbf{y}, \mathbf{x}) \right| \)

fast to adapt to new observation

Comparison of number of PDE solves in the seismic imaging example

<table>
<thead>
<tr>
<th>Method</th>
<th>Number of PDE solves</th>
<th>$F$</th>
<th>$\nabla F^T$</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTM</td>
<td></td>
<td>102</td>
<td>102</td>
</tr>
<tr>
<td>MLE</td>
<td></td>
<td>204</td>
<td>204</td>
</tr>
<tr>
<td>Preconditioned MAP</td>
<td></td>
<td>510</td>
<td>510</td>
</tr>
<tr>
<td>Preconditioned VI approach</td>
<td></td>
<td>2,000</td>
<td>2,000</td>
</tr>
<tr>
<td>MCMC w/ deep prior</td>
<td></td>
<td>10,000</td>
<td>10,000</td>
</tr>
</tbody>
</table>
Related work

Purely physics-based approach NF-based Bayesian inference

show orders of magnitude speed up compared to traditional MCMC methods

need to train the NF from scratch for a new observation $y$

use handcrafted priors, often negatively bias the inversion


Data-driven approaches for directly sampling from the posterior distribution

fast Bayesian inference given new observation \( y \)

sample the posterior virtually for free

not directly tied to the physics/data

not reliable when applied to out-of-distribution data

not trivial to scale GAN-based approaches to large-scale problems

---


Related work

Injective network for inverse problems and uncertainty quantification

use an injective map to map data to a low-dimensional space
can be used as a prior (projection operator) in inverse problems

heavily relies on availability of training data to generalize
unclear how the learned projection operator behaves when applied to out-of-distribution data

for new observation $y$, Bayesian inference requires training an additional NF, involving the forward operator

Contributions

Take full advantage of existing training data to provide

- low-fidelity but fast conditional mean estimate
- a first assessment of the image’s reliability
- preconditioned physics-based high-fidelity MAP estimate via the learned prior
- preconditioned, scalable, and physics-based Bayesian inference
Conclusions

Obtaining UQ information is rendered impractical when the forward operators are expensive to evaluate

the problem is high dimensional

There are strong indications that

Bayesian inference with normalizing flows can lead to orders of magnitude computational improvements compared to MCMC methods

preconditioning with a pretrained conditional normalizing flow can lead to another order of magnitude speed up

Future work

· characterize uncertainties due to modeling errors
Validated scaling to size $N \times N \times 4 \times 1$ for $N = 2^{16}$ on Azure’s HBv2 instance (AMD EPYC)

$2^{32}$ pixels equals $64 \times 8K$ images ($4 \times 1024^3$)

Runtime 54 seconds 120 threads

480 Gb memory not enough for PyTorch...
GPU

- sizes
  \[ N \times N \times (1 - 512) \times 128 \]
  for \( N = 2^5 - 2^{10} \)
- linear scaling in PyTorch
- up to \( 3 \times \) speedup on GPUs
- not good for small sizes & small # of channels

https://github.com/slimgroup/XConv/blob/master/benchmark/perf_pyxconv.py
Code

https://github.com/slimgroup/InvertibleNetworks.jl

https://github.com/slimgroup/FastApproximateInference.jl

https://github.com/slimgroup/Software.SEG2021
Acknowledgment

This research was carried out with the support of Georgia Research Alliance and partners of the ML4Seismic Center
Affine coupling layer

a basic ”layer” with an analytic inverse

· an invertible layer with input $u$

· orthogonal Householder reflections $Q_\ell$

· arbitrary neural networks $s_\ell$, $t_\ell$

· output of $\ell$th layer $T_\ell(u)$
Affine coupling layer—forward

\[
    T_\ell(u) = \begin{bmatrix}
        T_\ell^1(\tilde{u}_1) \\
        T_\ell^2(\tilde{u}_1, \tilde{u}_2)
    \end{bmatrix} = \begin{bmatrix}
        \tilde{u}_1 \\
        \tilde{u}_2 \odot \sigma(s_\ell(\tilde{u}_1)) + t_\ell(\tilde{u}_1)
    \end{bmatrix},
\]

with \( \tilde{u} = \begin{bmatrix} \tilde{u}_1 \\ \tilde{u}_2 \end{bmatrix} = Q_\ell u \)

sigmoid function \( \sigma(\cdot) \)

elementwise multiplication \( \odot \)

Affine coupling layer—inverse

\[
\begin{bmatrix}
\tilde{u}_1 \\
\tilde{u}_2
\end{bmatrix} = 
\begin{bmatrix}
T_1^1 (\tilde{u}_1) \\
\left(T_2^2 (\tilde{u}_1, \tilde{u}_2) - t_\ell (T_1^1 (\tilde{u}_1)) \right) \odot \sigma (s_\ell (T_1^1 (\tilde{u}_1)))
\end{bmatrix},
\]

\[
u = Q_{\ell}^{-1} \begin{bmatrix}
\tilde{u}_1 \\
\tilde{u}_2
\end{bmatrix}
\]

orthogonal learnable matrix \(Q_{\ell}^{-1} = Q_{\ell}^T\)

elementwise division \(\odot\)

Determinant of the Jacobian

\[
\nabla_u T_\ell(u) = \nabla_u \begin{bmatrix}
T_\ell^1(\tilde{u}_1) \\
T_\ell^2(\tilde{u}_1, \tilde{u}_2)
\end{bmatrix} = \begin{bmatrix}
I & 0 \\
\nabla_{u_1} T_\ell^2(\tilde{u}_1, \tilde{u}_2) & \text{diag} (\sigma(s_\ell(\tilde{u}_1)))
\end{bmatrix}
\]

\[\Rightarrow \log | \det \nabla_u T_\ell(u) | = \text{sum} (\log \sigma(s_\ell(\tilde{u}_1)))\]
Challenges of affine coupling layers

lower representation power due

- sparse triangular Jacobian

requires many layer compositions and orthogonal transforms

- to capture all pixel-wise dependencies

may become numerically non-invertible

- e.g. due to very small Jacobian singular values


Hierarchical coupling layers

dense triangular Jacobian

hierarchical architecture
  • recursive affine coupling layers
  • until input can not be split any further