Seismic Imaging with Uncertainty Quantification
Sampling from the Posterior with Generative Networks

Ali Siahkoohi¹, Philipp A. Witte², Mathias Louboutin¹, Felix J. Herrmann¹, and Gabrio Rizzuti¹

¹ School of Computational Science and Engineering, Georgia Tech
² Microsoft Azure
Seismic imaging

Challenges:

- Computational cost (cannot afford 10s of data passes)
- Very large scale (multiple TB of data, ~1 billion unknown parameters)
- State variables: ~10k time steps x no. of model parameters (10 TB per source location)
- Evaluate anisotropic/elastic forward/adjoint modeling operators
- Mathematical challenges (ill-conditioning, inconsistency, noise, etc.)
Seismic imaging with neural networks?

- data-driven or combined model-data driven ("loop unrolling")?
- implicit regularization ("deep prior")?
- uncertainty quantification?
Agenda

- Speeding up imaging via loop unrolling
- Uncertainty quantification with networks and MCMC (deep priors and task UQ)
- UQ: sampling from the posterior w/ invertible networks
Least-squares imaging & loop unrolling
Sparsity-promoting least-squares RTM

\[
\begin{align*}
\text{minimize} & \quad \lambda \| C \delta m \|_1 + \frac{1}{2} \| C \delta m \|_2^2 \\
\text{subject to:} & \quad \sum_{i=1}^{n_s} \sum_{j=1}^{n_f} \| M_l^{-1} J M_r^{-1} \delta m - M_l^{-1} \tilde{d}_{i,j} \| \leq \sigma
\end{align*}
\]
Least-squares imaging & loop unrolling

Strategies:

- Sparsity-promoting LS-RTM + on-the-fly Fourier transforms
- Save cost by working with subsets of random shots/frequencies (can afford 1 or 2 data passes)
- Automatic code generation using Devito
- Data + model driven imaging via loop unrolling
Loop unrolling (example w/ gradient descent)

1. function $\Lambda_\theta(J, d)$
2. $x_0 = 0$, $s = 0$
3. for $j = 1, \ldots, n$
4. $g = J^T(Jx_{j-1} - d)$
5. $u = [x_{j-1}, g, s]$
6. $u = \text{ReLU}(u * w_1 + b_1)$
7. $u = \text{ReLU}(u * w_2 + b_2)$
8. $u = u * w_3 + b_3$
9. $s = \text{ReLU}(u[:,:,1:\text{end}-1,:,\cdot]),\cdot]$  
10. $\Delta x = u[:,:,\text{end},:,\cdot]$  
11. $x_j = x_{j-1} - \Delta x$
12. end
13. return $x_n$
14. end

Loop unrolled gradient descent
Loop unrolling w/ invertible networks

Invertible version of loop-unrolled network:
- Loop unrolling: memory scales w/ network depth
- Instead: architecture based on invertible nets

```
1. function \( G(J, d) \)
2. \( x = 0 \)
3. for \( j = 1, ..., n \)
4. \( x = Qx \)
5. \( x'_1 = x_1 \)
4. \( g = J^T (Jx'_1[1] - d) \)
5. \( s', t = NN([g; x'_1[2:end]]) \)
6. \( s = \sigma(s') \)
6. \( x'_2 = x_2 \odot s + t \)
3. \( x = Q^T x' \)
12. end
13. return \( x \)
14. end
```
Discussion and conclusions

- Loop unrolling has the potential to speed up imaging, when data and ground truth are jointly available
- What about UQ?
Seismic imaging w/ deep priors and uncertainty quantification with tasks (automatic horizon tracking)

Why uncertainty quantification?

Uncertainties in **seismic imaging**:

- Noisy/partial observations
- Ill-conditioning
- Confidence in prior model

Goals:

- Determine how likely a certain result is
- Analyze results “equally” likely
- How the uncertainty reflects on subsequent **tasks**? (e.g. horizon tracking)
Ingredient 1/2: MCMC methods

E.g., Metropolis-adjusted Langevin algorithm

\[ m_{n+1} = m_n + \frac{\varepsilon_n}{2} \nabla m_n \log \pi + \eta_n, \quad \eta_n \sim N(0, \varepsilon_n I) \]

\[ \alpha = \min(1, \frac{\pi(m_{n+1})q(m_n|m_{n+1})}{\pi(m_n)q(m_{n+1}|m_n)}) \]

\[ \alpha \geq u \sim U[0,1] \? \quad \text{Accept } m_{n+1} \]

\[ \pi : \text{ target distribution} \]

\[ q : \text{ transition density} \]
Ingredient 2/2: Generative networks

\[ G : \mathcal{Z} \rightarrow \mathcal{M} \]

\[ \mathbf{z} \sim N(0, I), \quad \mathbf{m} = G(\mathbf{z}) \sim p_{post}(\mathbf{m}|\mathbf{d}) \]

Potential advantages:

- Inductive bias (implicit “deep” prior)
- Problem-dependent bias via loop-unrolling
- Encode distribution via push-forward action

\(\mathcal{Z}\) : latent space
\(\mathcal{M}\) : model space
Seismic imaging w/ deep priors

- Implicit regularization via network

\[
\min_\theta \quad - \log p_{\text{post}}(\theta | \delta d) \\
= - \log p_{\text{like}}(\delta d | \theta) - \log p_{\text{prior}}(\theta) + \ldots \\
= \frac{1}{2\sigma^2} \sum_{i=1}^{N_s} \| \delta d_i - J(m_0, q_i)G_\theta(z) \|^2 + \frac{\lambda^2}{2} \| \theta \|^2 + \ldots
\]

- $\delta d$: data residual
- $J(m, q)$: linearized forward operator
- $G_\theta: Z \to M$ : neural network
- $\lambda$: weighting
Sampling the posterior distribution $\theta$

- Preconditioned Stochastic Gradient Langevin Dynamics

$$\theta_{k+1} = \theta_k - \frac{\epsilon}{2} M_k \nabla_{\theta_k} L + \eta_k, \quad \eta_k \sim N(0, \epsilon M_k)$$

$$L = \frac{N_s}{n} \sum_{i=1}^{n} \frac{1}{2\sigma^2} ||\delta d_i - J(m_0, q_i) G_{\theta}(z)||^2 + \frac{\lambda^2}{2} ||\theta||^2$$

stochastic approximation

$M_k$  adaptive diagonal preconditioning

$\epsilon$  steplength
Conditional estimators: from $\theta$ to $\mathbf{m}$

\[ \mathbb{E}(\theta|\delta \mathbf{d}) = \int p_{\text{post}}(\theta|\delta \mathbf{d}) \, \theta \, d\theta \quad \rightarrow \quad \mathbb{E}_{(G)}(\delta \mathbf{m}|\delta \mathbf{d}) = \int p_{\text{post}}(\theta|\delta \mathbf{d}) \, G_\theta(\mathbf{z}) \, d\theta \]

- Conditional mean

\[ \mathbb{E}_{(G)}(\delta \mathbf{m}|\delta \mathbf{d}) \approx \mathbf{\hat{m}} = \frac{1}{n} \sum_{i=1}^{n} G_{\theta_i}(\mathbf{z}) \]

- Conditional pointwise variance

\[ \mathbb{E}_{(G)}((\delta \mathbf{m} - \mathbf{\hat{m}})^2|\delta \mathbf{d}) \approx \frac{1}{n} \sum_{i=1}^{n} (G_{\theta_i}(\mathbf{z}) - \mathbf{\hat{m}})^2 \]
True model: Overthrust model
Using no prior
Conditional mean

\[ \delta \hat{m} - \text{mean of } g(z, \hat{w}_j) \text{'s}, \quad \hat{w}_j \sim p_{post}(w | \{d_i, q_i\}_{i=1}^N, z) \]
UQ: conditional pointwise standard deviation

Point-wise standard deviation of $g(z, \hat{w}_j)'s$

Depth (km)

Horizontal distance (km)
UQ: pointwise histograms
Horizon tracking

\[ H_{\text{track}} : \mathcal{M} \rightarrow \mathcal{H} \]

Pushing model uncertainty to horizons

\[ \theta \sim p_{\text{post}}(\theta | \delta d) \quad \Rightarrow \quad \delta m \sim \tilde{p}_{\text{post}}(\delta m | \delta d) \quad \Rightarrow \quad H \sim \tilde{p}_{\text{post}}(H | \delta d) \]

Neural net parameters \quad Physical models \quad Horizons

\[ G(z) : \Theta \rightarrow M \quad \quad H_{\text{track}} : M \rightarrow H \]
MLE

Maximum likelihood estimate (MLE)
Conditional mean

\[ \hat{\delta m} - \text{mean of } g(z, \hat{w}_j)'s, \hat{w}_j \sim p_{post}(w|\{\delta d_i\}_{i=1}^N) \]
UQ: conditional pointwise standard deviation

Point-wise standard deviation of $g(z, \hat{w})$'s

Depth (km)

Horizontal distance (km)
Horizon tracking on conditional mean
Horizon tracking and UQ

Mean and 99% confidence intervals of $\mathcal{H}(g(\mathbf{z}, \widehat{\mathbf{w}}))$'s

Depth (km)

Horizontal distance (km)
Horizon tracking and UQ

Mean and 99% confidence intervals of $\mathcal{H}(g(z, \hat{w}_j))$'s

Depth (km)

Horizontal distance (km)
Horizon tracking and UQ

control point: 1.225 km

control point: 3.0 km
Discussion and conclusions

- Networks provide a beneficial inductive bias for inverse problems and their UQ (comprising tasks)
- MCMC methods tend to be expensive...
Invertible neural networks and uncertainty quantification

Normalizing flows

\[ x \sim p_X(x) \quad \text{Training} \]

\[ z \sim p_Z(z) \quad \text{Generation} \]

\[ G_\theta : \mathcal{X} \rightarrow \mathcal{Z} \]
Normalizing flows

- **Invertible** $\rightarrow$ constant memory complexity

- Generate samples from target distribution

\[ z \sim p_Z(z), \quad x = G_{\theta}^{-1}(z) \]

when trained via MLE (no adversarial approach as GANS)

\[
\log p_X(x) \approx \log p_{X,\theta}(x) \quad \text{“pull-back”}
\]

\[
= \log p_Z(G_{\theta}(x)) + \log |\det J_{G_{\theta}}(x)|
\]

\[
= \frac{1}{2} ||G_{\theta}(x)||^2 + \log |\det J_{G_{\theta}}(x)|
\]
Given data pertaining to a single subsurface model,

\[
\min_{\theta} \quad \text{KL}(p_{\theta}(m) \| p_{\text{post}}(m|d)) \\
= \mathbb{E}_{z \sim p_Z(z)} \left[ \frac{1}{2\sigma^2} \left\| d - F(G_{\theta}(z)) \right\|^2 - \log p_{\text{prior}}(G_{\theta}(z)) - \log |\det J_{G_{\theta}}(z)| \right]
\]

\[F : \mathcal{M} \rightarrow \mathcal{D} \quad \text{forward modeling operator}\]
\[G_{\theta} : \mathcal{Z} \rightarrow \mathcal{M}\]
Supervised imaging and UQ

Given joint pairs \((\mathbf{m}, \mathbf{d})\),

\[
\min_{\theta} \quad \text{KL}(p_{\text{joint}}(\mathbf{m}, \mathbf{d}) \| p_{\theta}(\mathbf{m}, \mathbf{d}))
\]

\[
= \mathbb{E}_{\mathbf{m}, \mathbf{d} \sim p_{\text{joint}}(\mathbf{m}, \mathbf{d})} \frac{1}{2} \| G_\theta(\mathbf{m}, \mathbf{d}) \|^2 - \log | \det J_{G_\theta}(\mathbf{m}, \mathbf{d}) |
\]

+ special \textbf{conditional structure} on \(G\)
Supervised imaging and UQ

Special **conditional structure** on $G$

$$\begin{aligned}
(z_m, z_d) &= G_\theta(m, d) \\
&= (G^m_\theta(m, d), G^d_\theta(d))
\end{aligned}$$

Posterior sampling w/ conditional nets:

$$z_d := G^d_\theta(d), z_m \sim p_Z(z_m),$$

$$(m, d) = G^{-1}_\theta(z_m, z_d) \Leftrightarrow m \sim p_{\text{post}}(m|d)$$
Supervised UQ (pseudo-imaging example)
Supervised UQ (pseudo-imaging example)
Supervised UQ (pseudo-imaging example)
Discussion and conclusions

- Invertible nets offer a new way to think about UQ (w/ potential advantages over classical MCMC methods)
- Imaging for large scale problems will have to rely on memory savvy networks
- Training might be expensive...
Implementation details

Wave-equation operators via Devito:

- https://github.com/devitocodes/devito

Integrating Devito into PyTorch:

- https://github.com/slimgroup/Devito4PyTorch

Automatic horizon tracking:

- https://github.com/xinwucwp/mhe
Software

**InvertibleNetworks.jl**: a scalable Julia package for invertible nets

- Networks w/o need to store forward states during training
- Manually implemented derivatives that make use of invertibility
- Unit testing (adjoints, gradients, invertibility)
- [https://github.com/slimgroup/InvertibleNetworks.jl](https://github.com/slimgroup/InvertibleNetworks.jl) (MIT license)
Software

Class structure:

```c
struct CouplingLayerGlow <: NeuralNetLayer
  C::Conv1x1
  RB::ResidualBlock
  logdet::Bool
  forward::Function
  inverse::Function
  backward::Function
end
```

Usage:

```c
# Create Layer
LayerGlow = CouplingLayerGlow(nx, ny, nchannel, nhidden, batchsize)

# Operations
Y = LayerGlow.forward(X)
X_ = LayerGlow.inverse(Y)
ΔX, X_ = LayerGlow.backward(ΔY, Y)
```
Loop unrolled seismic imaging

Training loop unrolled network:
- 1000 pairs of seismic data + true image
- Single epoch w/ ADAM
Loop unrolling w/ invertible networks

Invertible version of loop-unrolled network:

- Loop unrolling: memory scales w/ network depth
- Instead: architecture based on invertible nets
- Invertible + tractable logdet

---

1. function $G(J, d)$
2. $x = 0$
3. for $j = 1, ..., n$
4. $x = Qx$
5. $x'_1 = x_1$
6. $g = J^T(Jx'_1[1] - d)$
7. $s', t = NN([g, x'_1[2:end]])$
8. $s = \sigma(s')$
9. $x'_2 = x'_2 \odot s + t$
10. $x = Q^Tx'$
11. end
12. return $x$
13. end

---

*Figures: True image, Gradient descent (SNR -0.27), Invertible loop unrolling (SNR 7.07)*
True model: Parihaka dataset
Confidence intervals empirical verification