



ML-REFLECTOMETRY WORKSHOP

Introduction & ML supported reflectometry analysis

October 21st 2026 | Stefan Häusler, Mykhailo Levytskyi | JCNS

Organization

09:30 – 12:30 Conventional reflectometry analysis

12:30 – 14:00 Lunch

14:00 – 15:30 Foundations of machine learning and neural networks

15:30 – 17:00 Advanced analysis methods for reflectivity data

17:00 – 17:30 Q&A

Outline

Foundations of machine learning and neural networks

Advanced analysis methods for reflectivity data

Machine learning in reflectometry

Physics-based fitting and MCMC already support accurate parameter inference.

Conventional inference

Each new measurement requires inference using the physical forward model.

Exploring uncertainty can require many model evaluations.

Amortization: share the initial cost across many measurements.

Simulated training data do not enforce physics in every prediction. Physical checks and refinement remain important.

Learning from simulations

Generate examples and train a network upfront.

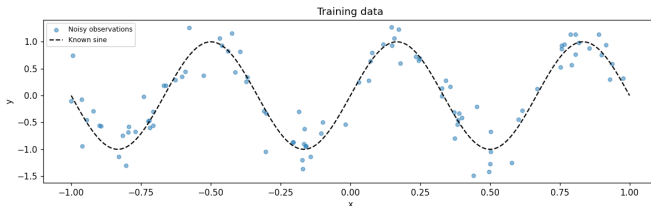
Reuse the trained network for fast approximate predictions on new curves.

Supervised learning

Examples pair an input with an observed or simulated target:

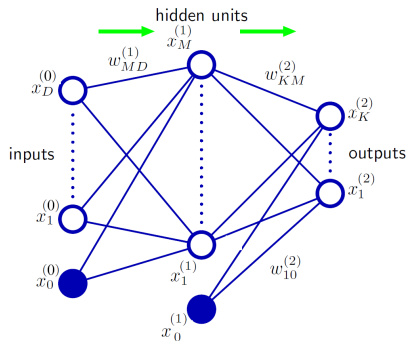
$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N, \quad \hat{y} = f_{\mathbf{w}}(\mathbf{x}).$$

	Input	Target
Exercise 1	Coordinate x	Noisy observed y
Exercise 2	Reflectivity curve	Layer SLD ρ



Training adjusts $\mathbf{w}$ using examples. Prediction applies the learned mapping to a new input.

A neuron and a neural network



Inputs, hidden units, outputs

One hidden unit in Exercise 1:

$$h_j(x) = g(w_j x + b_j).$$

Nonlinear activation:

$$g(z) = \text{sigmoid}(z) = \frac{1}{1 + e^{-z}}.$$

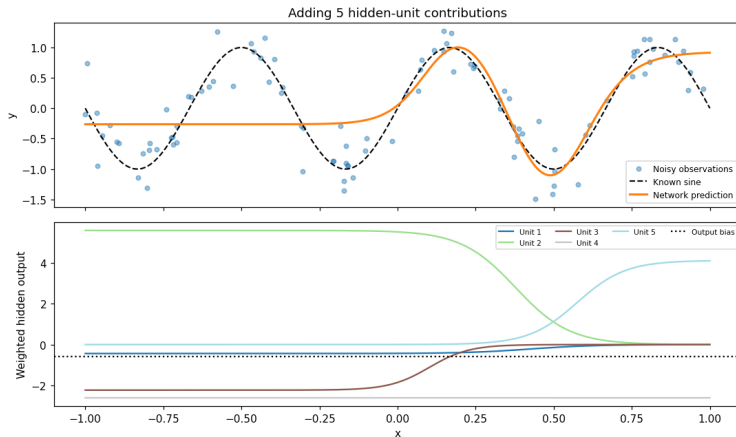
Linear output:

$$\hat{y} = b_{\text{out}} + \sum_{j=1}^M v_j h_j(x).$$

w collects all trainable weights and biases.

Without a nonlinear activation, stacked affine layers still give an affine function.

Function approximation



Add the contributions

Weighted sigmoids plus the output bias form the prediction.

More units provide more contributions to arrange.

Expressive power gives no recipe for finding useful weights.

Representing a solution, finding it, and generalizing are separate questions.

Learning objective and likelihood

Squared discrepancies to observed targets:

$$\mathcal{L}_{\text{train}}(\mathbf{w}) = \frac{1}{N} \sum_{i=1}^N (f_{\mathbf{w}}(\mathbf{x}_i) - y_i)^2.$$

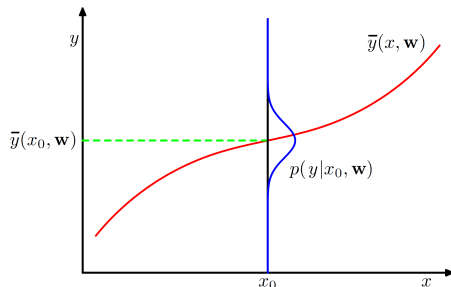
A probabilistic interpretation:

$$q_{\mathbf{w}}(y \mid \mathbf{x}) = \mathcal{N}(y; f_{\mathbf{w}}(\mathbf{x}), \sigma_0^2).$$

For fixed Gaussian variance,

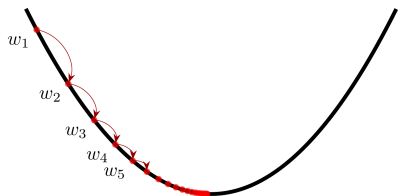
$$-\frac{1}{N} \sum_i \log q_{\mathbf{w}}(y_i \mid \mathbf{x}_i) = \frac{\mathcal{L}_{\text{train}}}{2\sigma_0^2} + C.$$

- Minimizing MSE maximizes the likelihood (ML) for this fixed-variance Gaussian model.
- Maximizing the expected log-likelihood minimizes the KL divergence between the true and model distributions.
- The MSE-optimal prediction is the conditional mean $\mathbb{E}[y \mid \mathbf{x}]$, also for non-Gaussian targets (for finite variance).



A conditional distribution at one input

Gradient descent and backpropagation



Schematic loss as a function of a weight

Gradient-descent principle

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla_{\mathbf{w}} \mathcal{L}_{\text{train}}(\mathbf{w}_t).$$

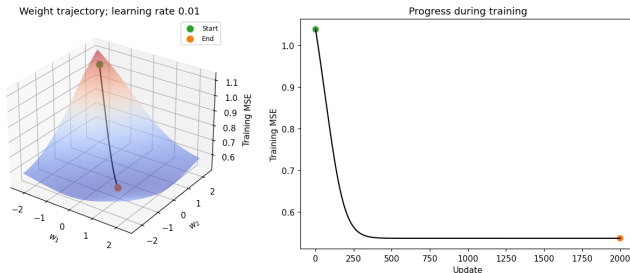
- 1 Predict and calculate the loss.
- 2 Calculate gradients efficiently (Backpropagation).
- 3 Update weights using an optimizer.

The learning rate η controls the adjustment scale. Too small can be slow, too large can overshoot or stall.

The notebooks use **Adam**, which adapts gradient-based updates. One epoch processes all training examples.

Exercise 1: how a network learns

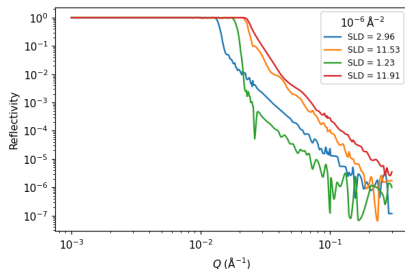
- 1 Compare hidden-unit contributions for 3, 5, and 10 units.
- 2 Change initialization while keeping the observations fixed.
- 3 Watch two weights move and compare learning rates.



Two adjustable weights make the learning process visible.

Open `Ex1_neural_networks.ipynb` **and work through both parts.**

Statistical inverse learning



Simulated curves with known layer SLD

Physical forward model

$$\theta \mapsto R(Q)$$

SLD marginal posterior

$$q_w(\rho \mid \mathbf{R}) \approx p_{\text{sim}}(\rho \mid \mathbf{R}).$$

Thickness and roughness also vary. The curve is the only input.

A point prediction summarizes possible targets:

$$\hat{\rho}(\mathbf{R}) \text{ targets } \mathbb{E}_{\text{sim}}[\rho \mid \mathbf{R}].$$

Simulation sampling supplies implicit prior information. An average of alternatives need not identify a compatible structure.

Training, validation, and testing

Training

1,901 curves

Fit network weights by minimizing $\mathcal{L}_{\text{train}}$.

Validation

634 curves

Compare settings and select a checkpoint using $\mathcal{L}_{\text{val}}$.

Test

634 curves

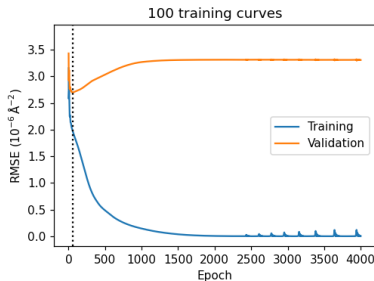
Evaluate the selected model using $\mathcal{L}_{\text{test}}$.

Test loss estimates expected loss on new examples from the same population.

Selection uses validation data, including the stopping epoch.

Ex2 uses a 60/20/20 split. Fit scaling on training data and apply it to all partitions.

Overfitting



Training error keeps falling while validation error rises.

Select the checkpoint at the validation minimum. The final epoch can give worse predictions on new curves.

Dotted lines mark selected checkpoints.

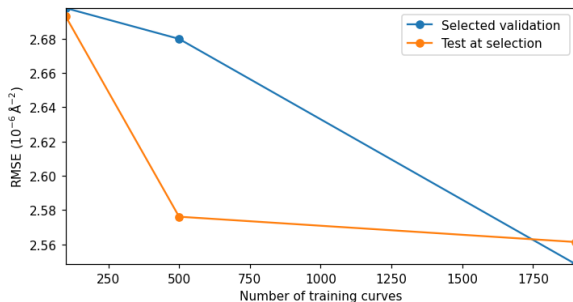
Exercise 2: predicting layer SLD

- 1 Train a point predictor and compare it with the training-mean baseline.
- 2 Inspect generalization as training duration and sample count change.
- 3 Continue until Gaussian prediction of mean and variance.

Common checkpoint: explain why the selected model can differ from the final epoch.

Open `Ex2_reflectometry.ipynb`.

Overfitting and training-set size



At the selected checkpoint

Compare validation and test error as training size changes.

More representative data can help generalization.

More data helped in this saved run. Improvement need not be monotonic. The notebook also supplies a training-mean baseline.

A Gaussian prediction: mean and variance

Learn an input-dependent centre and spread

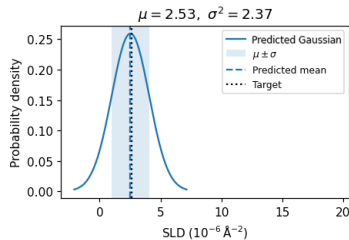
$$q_{\mathbf{w}}(\rho \mid \mathbf{R}) = \mathcal{N}\left(\rho; \mu_{\mathbf{w}}(\mathbf{R}), \sigma_{\mathbf{w}}^2(\mathbf{R})\right).$$

For example, Gaussian negative log-likelihood is

$$\ell_{\text{NLL}} = \frac{1}{2} \left[\log \sigma^2 + \frac{(\rho - \mu)^2}{\sigma^2} \right] + C.$$

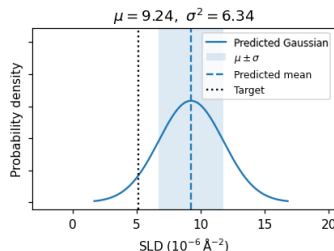
Confident misses incur a large residual penalty. Very wide predictions incur a logarithmic penalty.

The targets are conditional mean and variance. One Gaussian cannot represent separated solutions.



The predicted spread is intended to estimate **aleatoric uncertainty**. It may also absorb model error.

Predicted uncertainty and coverage



The target is about 1.64σ from the mean.

A standard deviation describes spread, not an error bound.

Global coverage checks collective frequencies within the simulation population. It does not validate each individual uncertainty estimate.

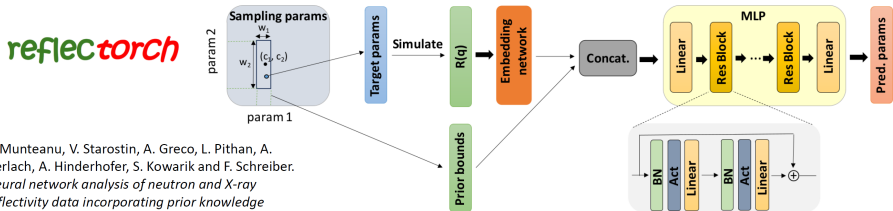
Coverage across 634 test curves

Interval	Gaussian	Observed
$\mu \pm \sigma$	68.3%	69.1%
$\mu \pm 1.96\sigma$	95.0%	95.6%

Compare coverage **and interval width**. Wide intervals can cover many targets while giving little information.

NN for neutron reflectivity

Previous implementation



- **Deep neural networks:** rapid generation of single candidate solutions.
- **Prior amortization:** the same trained network supports different prior choices
- **Anaklasis:** downstream parameter optimization via conventional iterative fitting

Learning the joint parameter posterior

SLD, thickness, and roughness can be correlated.

One SLD value	Conditional mean
Gaussian SLD density	Conditional mean and variance
Joint density	Correlations and separated solution families

PANPE: Prior-Amortized Neural Posterior Estimation

A neural network approximates the joint posterior of the physical parameters, using the measurement $\mathbf{R}$ and a chosen prior φ .

- θ : physical parameters to infer, such as SLD, thickness, and interface roughness.
- φ : parameters defining the chosen prior, such as lower and upper bounds on θ .

$$q_{\mathbf{w}}(\theta \mid \mathbf{R}, \varphi) \approx p(\theta \mid \mathbf{R}, \varphi).$$

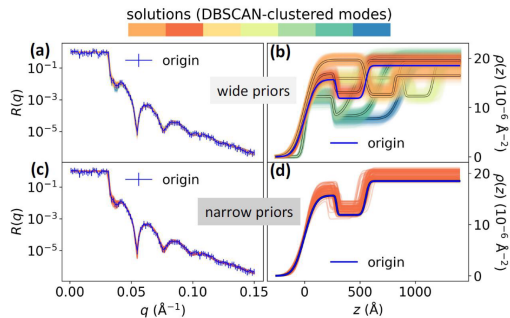
Prior amortization: the same trained network supports different prior choices.

Outline

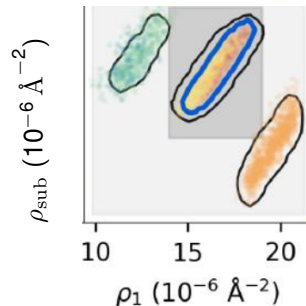
Foundations of machine learning and neural networks

Advanced analysis methods for reflectivity data

Correlations and alternative solutions



Similar curves, different depth profiles.



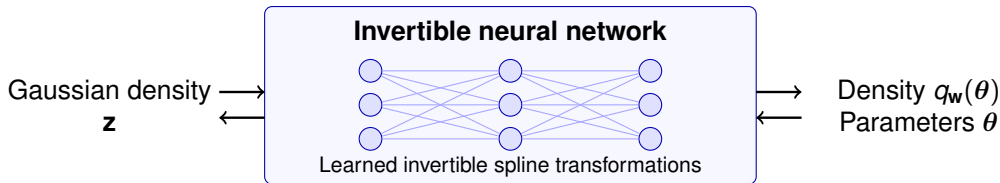
Layer SLD and substrate SLD vary together.

We need their joint distribution, including correlations.

Published PANPE-IS example: Starostin et al., Science Advances (2025).

The flow is an invertible neural network

Familiar: learned weights, nonlinear activations, backpropagation.



Sampling: $\mathbf{z} \rightarrow \theta$. Training evaluates $\theta \rightarrow \mathbf{z}$.

New: reversible layers with learned nonlinear spline functions.

Train: give high probability density to known simulated structures.

$$\ell = -\log q_{\mathbf{w}}(\theta) = \underbrace{\frac{1}{2}\|\mathbf{z}\|^2}_{\text{Gaussian score}} - \underbrace{\log \left| \det \frac{\partial \mathbf{z}}{\partial \theta} \right|}_{\text{stretch/compression correction}} + C, \quad \mathbf{z} = f_{\mathbf{w}}^{-1}(\theta)$$

A normalizing flow turns simple noise into structured parameter samples.

Start with a simple latent sample:

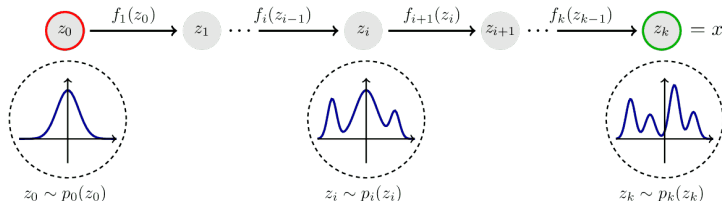
$$\mathbf{z} \sim p_0(\mathbf{z})$$

An invertible sequence of transformations maps it to a parameter vector:

$$\theta = f_\psi(\mathbf{z}).$$

Each draw gives one possible structure.

Invertibility lets the model evaluate sample densities as well as generate samples.



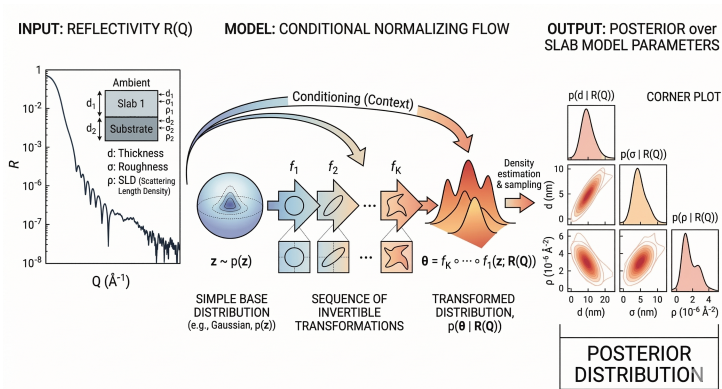
Conditioning the flow on a reflectivity curve gives a parameter posterior.

The measured curve supplies context for the flow:

$$\theta = f_{\psi}(\mathbf{z}; c(\mathbf{R})).$$

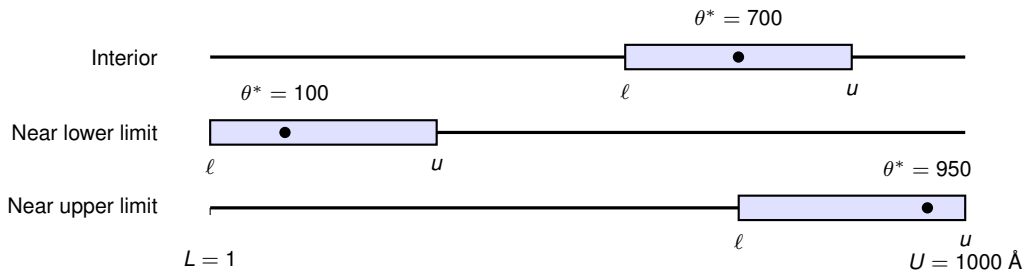
$$q_{\mathbf{w}}(\theta \mid \mathbf{R}, \varphi) \approx p(\theta \mid \mathbf{R}, \varphi)$$

PANPE adds the chosen prior settings φ to the conditioning information. One draw is a joint structure; many draws show correlations and alternative solutions.



A 30% prior is a sliding window within the model's allowed range.

For thickness, the model allows 1–1000 Å. Blue bands show a window equal to 30% of this span; black points mark the reference value.



At a boundary, shift the window to keep it inside the physical range.

Notebook 01 moves from one experiment to its inference diagnostics.

- 1 Select an experiment and load its measured curve and reference structure.
- 2 Preprocess the curve and build constraint-based prior bounds.
- 3 Draw posterior samples with the one-layer NF model and select the MAP sample.
- 4 Inspect the reflectivity fit, SLD profile, posterior summaries, and constraint MAPE.

Notebook 01: test how prior choice and the experiment affect inference.

- 1 Run one experiment with prior widths of 10%, 30%, and 60%. Compare the bounds, MAP estimate, posterior spread, reflectivity fit, and SLD profile. Which results change, and which remain stable?
- 2 Choose another experiment with the same layer count and keep the settings fixed. Compare its parameter estimates, fit, and SLD profile with the first experiment.

Wider bounds change the assumptions supplied to inference; they do not add information to the measured curve.

Notebook 02 takes an ORSO measurement through two-layer posterior sampling.

- 1 Load the measured curve, uncertainties, and available resolution values.
- 2 Set physical and nuisance-parameter prior bounds for the two-layer model.
- 3 Draw posterior samples, summarize parameters, and select the MAP sample.
- 4 Compare the MAP curve and SLD, then view all finite sampled SLD profiles.

The ORSO example has no numeric ground-truth SLD profile; the sampled profiles show ambiguity and sampling stability.

Notebook 02: check whether more samples change the posterior picture.

For the same ORSO curve and prior bounds, compare 500 and 2,000 NF samples. Rerun from setup each time to reset the random seed.

- Compare MAP parameters, posterior means, and standard deviations.
- Overlay the sampled SLD profiles. Do new profile families appear, or do more samples mainly fill in the same families?

ORSO provides no numeric ground-truth SLD. This checks sampling stability for this run, not the accuracy of the inferred structures.

Thank you for your attention!