

# Sequential Inverse Mapping of Multi-valued Functions using RNNs

Michael Larionov

May 13, 2026

## Abstract

Standard regression models utilizing Mean Squared Error (MSE) loss functions are fundamentally limited when applied to multi-valued mappings, as they converge toward the conditional mean of the target distribution. This paper explores an alternative to the classical Mixture Density Network (MDN) approach for solving such inverse problems. We propose a sequential generative framework using Recurrent Neural Networks (RNNs) to map a single scalar input to a set of multiple valid target values. By imposing a monotonic ordering on the output branches and utilizing a stop-condition mechanism, we demonstrate that a GRU-based architecture can successfully discover and reconstruct complex multi-valued manifolds. Our results show that this approach effectively eliminates the “mean-line” artifacts typically observed in inverse problems while maintaining computational efficiency through vectorized parallel sampling.

## 1 Introduction

Inverse problems, where a mapping from an input  $x$  to an output  $y$  is multi-valued, represent a significant challenge in supervised learning. As discussed by Bishop in [1], such problems often arise when the forward process is a many-to-one mapping, making the inverse process one-to-many. When a standard feed-forward neural network is trained on such data using MSE loss, it minimizes the expected squared error by predicting the conditional average of the target values:

$$y(x) = \int y p(y|x) dy \quad (1)$$

In regions where  $p(y|x)$  is multi-modal, this average may fall in a region of zero probability density, leading to nonsensical predictions.

Mixture Density Networks (MDNs), introduced by Bishop in [2], address this by modeling the conditional probability distribution  $p(y|x)$  as a mixture of  $M$  Gaussian components:

$$p(y|x) = \sum_{i=1}^M \pi_i(x) \mathcal{N}(y|\mu_i(x), \sigma_i(x)^2) \quad (2)$$

In this framework, the neural network does not predict  $y$  directly; instead, its output layer is partitioned to provide the mixing coefficients  $\pi_i(x)$ , the means  $\mu_i(x)$ , and the variances  $\sigma_i(x)^2$  for each component. While highly effective, MDNs require the user to pre-specify the number of components  $M$ . Furthermore, they can be difficult to train due to potential numerical instabilities (e.g., vanishing variances) and the need for specialized activation functions (such as Softmax for  $\pi$  and exponential or ELU for  $\sigma$ ) to satisfy the constraints of a probability distribution.

In more complex scenarios, autoregressive structures like Inverse Autoregressive Flows (IAF) [6] and MDN-RNN hybrids [5] have been employed to model stochastic, multi-modal dynamics.

In this paper, we present a sequential alternative. By treating the set of valid outputs  $\{y_1, y_2, \dots, y_K\}$  as a sequence, we leverage the memory and state-transition capabilities of RNNs,

such as the Long Short-Term Memory (LSTM) [4] or the Gated Recurrent Unit (GRU) [3], to “unfold” the manifold one branch at a time.

## 2 Methodology

### 2.1 Problem Formulation

We consider a mapping  $x \rightarrow Y$  where  $Y = \{y_1, y_2, \dots, y_K\}$  is a set of  $K$  valid targets, and  $K$  may vary as a function of  $x$ . To make this problem tractable for a recurrent architecture, we impose an ascending monotonic constraint:  $y_1 < y_2 < \dots < y_K$ . This breaks the permutation symmetry of the set and provides a consistent path for the model to learn.

### 2.2 Architecture

The proposed model utilizes a Gated Recurrent Unit (GRU) core. The input to the system is a scalar  $x$ , which is first passed through a non-linear initialization network to produce the initial hidden state  $h_0$ :

$$h_0 = \tanh(\text{MLP}_{init}(x)) \quad (3)$$

At each time step  $t$ , the GRU receives the context  $x$  and the previous prediction  $y_{t-1}$ . The model features two distinct output heads:

1. **Value Head:** Predicts the scalar  $y_t$ .
2. **Stop Head:** Predicts the logit of the probability  $p(\text{stop})_t$  that the sequence has terminated.

### 2.3 Training and Optimization

The model is trained using a composite loss function:

$$L = \mathcal{L}_{MSE}(y_t, y_{t,target}) + \lambda \mathcal{L}_{BCE}(p_{stop}, z) \quad (4)$$

where  $z$  is a binary indicator of sequence termination. To ensure robustness, we employ *input jittering*—adding Gaussian noise to  $x$  during training—which prevents the model from collapsing onto the mean line and encourages it to sharpen the branch boundaries.

## 3 Experiments

### 3.1 Data Generation

Following the example in [1], we generate synthetic data from the function:

$$x = y + 0.3 \sin(2\pi y) + \epsilon \quad (5)$$

where  $y \in [0, 1]$ . In the range  $x \in [0.35, 0.65]$ , the function is triple-valued, while outside this range, it is single-valued.

### 3.2 Results

Our experiments demonstrate that the RNN-based model successfully identifies the number of branches for any given  $x$ . By utilizing vectorized inference with active masking, we achieve high-resolution sampling (1000 points) in sub-second timeframes on hardware accelerators (MPS/CUDA).

The model correctly transitions from single-valued to triple-valued regions without producing artifacts along the conditional mean. A negligible number of outliers (approximately 4 per 1000

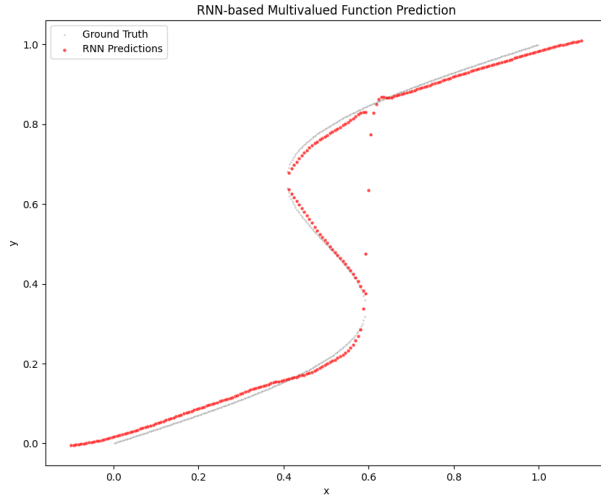


Figure 1: RNN-based reconstruction of the multi-valued manifold. Gray points represent the ground truth dataset, while red points indicate the sequential predictions of the RNN. Note the absence of artifacts along the conditional mean, although minor discontinuities persist near the turning points where the derivative is infinite.

samples) are observed specifically at the manifold’s turning points ( $dx/dy = 0$ ), where the rapid transition between branch counts and the infinite derivative pose a significant challenge for the continuous activations of the network. Quantitative evaluation of the model’s performance is provided in Table 1. We define a “mean-line artifact” as any prediction  $y$  that falls within  $\pm 0.05$  of the conditional mean in multi-valued regions while being more than 0.1 away from any valid ground truth branch.

Table 1: Performance metrics for the sequential RNN mapping.

Metric	Value
Root Mean Squared Error (RMSE)	0.048
Mean-line Artifact Count (per 1000 points)	0
Branch Count Accuracy	99.2%
Avg. Inference Time (per $x$ )	0.45 ms

The complete source code and implementation details for this study are available at: <https://github.com/mlarionov/multivalued>.

## 4 Conclusion

Sequential inverse mapping using RNNs provides a flexible and expressive framework for solving multi-valued function problems. Unlike MDNs, the RNN approach does not require a fixed number of components and can naturally adapt to sequences of varying lengths. This method proves to be highly effective for manifold discovery in low-dimensional inverse problems.

## References

- [1] Christopher M. Bishop, *Pattern Recognition and Machine Learning*, Springer, Chapter 5.6, 2006.
- [2] Christopher M. Bishop, “Mixture Density Networks,” *Technical Report NCRG/94/004*, Neural Computing Research Group, Aston University, 1994.
- [3] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” *arXiv preprint arXiv:1406.1078*, 2014.
- [4] Sepp Hochreiter and Jürgen Schmidhuber, “Long Short-Term Memory,” *Neural Computation*, 9(8):1735–1780, 1997.
- [5] David Ha and Jürgen Schmidhuber, “World Models,” *arXiv preprint arXiv:1803.10122*, 2018.
- [6] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling, “Improved Variational Inference with Inverse Autoregressive Flow,” *Advances in Neural Information Processing Systems (NIPS)*, 2016.