

Article

Amortized Parameter Inference for the Arbitrary-Order Hidden Markov Model

Sixiang Zhang * and Liming Cai

School of Computing, University of Georgia, Athens, GA 30602, USA; liming@uga.edu

* Correspondence: sixiang.zhang@uga.edu

Abstract

The arbitrary-order hidden Markov model (α -HMM) is a nontrivial generalization of the standard HMM, designed to model stochastic processes with higher-order dependences among arbitrarily distant random events. The α -HMM admits an efficient Viterbi-style optimal decoding algorithm, making it feasible to discover higher-order dependences among data objects in observed sequential data. Because the α -HMM exceeds the expressive power of standard HMMs, fixed k th-order HMMs, and stochastic context-free grammars, effective probabilistic parameter estimation approaches are required to translate this theoretical expressiveness of the α -HMM into practical utility. This paper introduces a principled methodology for effective estimation of probabilistic parameters of the α -HMM from observed data. In large-scale sequential datasets, higher-order dependencies can vary widely across instances, so a single global parameter set may be inadequate. Instead, an amortized parameter inference approach is proposed for the α -HMM, in which an input-conditioned parameter estimator is learned from data and used to infer instance-specific parameters for each input instance to the decoding algorithm. Specifically, the neural parameter estimator is trained using a composite learning objective that is partially enabled by the optimal decoding algorithm. The effectiveness of the proposed parameter estimation method is demonstrated through empirical results of the application of the α -HMM in biomolecular structure modeling and prediction.

Keywords: arbitrary-order hidden Markov model; probabilistic parameter estimation; amortized parameter learning; optimal decoding algorithm; biomolecule structure prediction

MSC: 62M05



Academic Editors: Natasa Kontrec,
Jelena Vujakovic and Hranislav
Milosevic

Received: 10 February 2026

Revised: 24 March 2026

Accepted: 10 April 2026

Published: 14 April 2026

Copyright: © 2026 by the authors.
Licensee MDPI, Basel, Switzerland.
This article is an open access article
distributed under the terms and
conditions of the [Creative Commons
Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

The hidden Markov model (HMM) is a probabilistic graphical model well-suited for stochastic processes and has achieved broad success in applications involving inference over time-series and sequential data [1–3]. While efficient inference via the linear-time Viterbi’s algorithm enables tractable decoding of latent state sequences from observed data [4], the practical effectiveness of the HMM depends critically on the availability of reliable methods for parameter estimation. In particular, well-established learning techniques, in particular, maximum likelihood estimation for fully observed training data and the expectation–maximization-based Baum–Welch algorithm for data with latent or missing structures, make it possible for HMM parameters to be robustly learned from data [5,6]. It is this combination, and especially the maturity of parameter estima-

tion methodologies, that has translated the HMM's theoretical formulation into a widely adopted modeling framework.

While the standard HMM is effective at modeling local dependencies, applications of sequential data may require inference of richer structural information that goes beyond simple stochastic processes. Such structures may arise in diverse settings, such as syntactic and semantic dependencies in linguistic sequences [7,8], bio-residue interactions that determine biomolecular structure and function [9,10], and latent patterns in time-series data such as weather forecasting and market prediction [11–14]. While such higher-order dependencies between random events have been modeled with extensions to the standard HMM [15,16], they rarely achieve the same level of practical effectiveness as the standard HMM. Due to the increased model complexity, parameter estimation becomes substantially harder than in the standard HMM; the richer structural dependencies dramatically increase the number of parameters and complicate latent inference. In particular, learning in k th-order HMMs suffers from exponential parameter growth and data sparsity [17,18], while stochastic context-free grammar (SCFG) [19] training is computationally expensive and statistically ineffective [20,21].

More recently, the arbitrary-order hidden Markov model (α -HMM) [22] has been introduced as a generalization of the standard HMM with the expressiveness to overcome the shortcomings of previous extensions to the HMM, e.g., the k th-order HMM and SCFG. The α -HMM is equipped with a mechanism to express dependencies between events in arbitrary distances with a well-defined joint probability distribution over an observable sequence of data and hidden events that generate the data. Such dependencies may be nested, parallel, and crossing patterns, some of which are beyond what SCFG is capable of characterizing. One significant utility of the α -HMM, as with the standard HMM, is its ability to uncover higher-order dependencies among latent events through decoding. Though the task of decoding with the full strength of the α -HMM can be computationally intractable, it has been shown that prediction of an optimal set of dependencies induced by exclusive pairs of hidden events can be accomplished much more efficiently, in $\mathcal{O}(n^3)$ time complexity, on observed sequences of n data items [22].

To translate the theoretical potential of the α -HMM into practical utility, its decoding algorithm must be paired with an effective method for probabilistic parameter estimation. However, because the α -HMM is strictly more expressive than k th-order HMMs and stochastic context-free grammars, parameter estimation methods developed for those models are no longer adequate. Moreover, with large-scale datasets, higher-order dependency structures can vary substantially across instances, making a single global parameter set insufficient. To address these challenges, we propose an amortized learning method [23,24] that enables scalable, instance-specific parameter inference and fully exploits the modeling capacity of the α -HMM. Rather than estimating a single set of global parameters shared by all sequences, we learn an input-conditioned parameter estimator from data which generates instance-specific α -HMM parameters for each input sequence of data whose dependence structures are to be predicted.

Specifically, the amortized parameter estimator is trained to infer parameters by optimizing a composite objective that integrates global and local structural constraints on the latent-event sequence. One term encourages the reference hidden-event path to receive a high score under the inferred parameters. A second term applies supervision to the reference hidden-state sequence, helping stabilize locally consistent state-transition behavior. A third term introduces a structured margin that separates the reference path from competing paths produced by Viterbi-style decoding under the same parameters. A fourth, task-specific term further guides the model toward the desired structured output. Together, these terms favor parameter settings that support correct global decoding,

locally consistent latent states, and task-relevant structural predictions. Evaluated on RNA secondary structure prediction, the proposed framework achieves strong empirical performance competitive with state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 reviews the hidden Markov model and introduces some fundamentals of machine learning. Section 3 gives a thorough description of the α -HMM's optimal decoding algorithm, and its expressiveness. Section 4 presents the amortized parameter inference framework in detail. Section 5 demonstrates the performance of the proposed work through comparisons with the-state-of-art methods on RNA secondary structure prediction. We conclude in Section 6 with some discussions.

2. Fundamentals

In this section, we review some fundamentals of the hidden Markov model (HMM) and in machine learning that are necessary for further reading of the α -HMM's decoding algorithm, and the parameter estimation method to be introduced in later sections. We refer the reader to [2,25] for more detailed information.

2.1. Hidden Markov Model

A *stochastic process* is a family of random variables $\{X_k : k \in \mathbb{N}\}$, where $\mathbb{N}$ is the natural number set and random variables X_k are defined over the same probability space.

Definition 1. A *discrete hidden Markov model (HMM)* [5] is a *bivariate stochastic process* $\{(X_k, Y_k) : k \in \mathbb{N}\}$, where X_k draws values from a finite set S of states and Y_k draws values from a finite set Σ of symbols, such that, for every finite $n \geq 1$, the joint probability $P(\{(X_k, Y_k) : k \leq n\})$ is defined as

$$P(\{(X_k, Y_k) : k \leq n\}) = P(X_1) \prod_{k=1}^n P(Y_k | X_k) \prod_{k=1}^{n-1} P(X_{k+1} | X_k) \quad (1)$$

Note that Equation (1) is derived based on the local Markov property as well as the chain rule of probability theory. Graphically, an HMM is represented as a probabilistic finite state machine $(S, \Sigma, \epsilon, \tau)$ in which each state $r \in S$ can *emit* a symbol $a \in \Sigma$ with probability $\epsilon_r(a)$ of distribution $\sum_{a \in \Sigma} \epsilon_r(a) = 1$. Every state r can *transit* to another state s with probability $\tau_r(s)$ of distribution $\sum_{s \in S} \tau_r(s) = 1$. Specifically, for any symbol $a \in \Sigma$ and any state $r \in S$, the conditional probability $P(Y_k = a | X_k = r)$ in (1) is defined as $\epsilon_r(a)$, and conditional probability $P(X_{k+1} = s | X_k = r)$ is defined as $\tau_r(s)$ for any state $s \in S$. The distributions ϵ and τ are the *parameters* that completely determine the HMM model.

A typical task associated with the HMM is the *decoding* of hidden states $\{X_k^* : k \leq n\}$ from an observed finite sequences of symbols $\{Y_k : k \leq n\}$ with the following optimization

$$\{X_k^* : k \leq n\} = \arg \max_{\{X_k : k \leq n\}} P(\{Y_k : k \leq n\} | \{X_k : k \leq n\})$$

which can be computed with the $O(n)$ -time dynamic programming *Viterbi* algorithm [4] by taking the advantage probability factorization in Equation (1).

To deploy the decoding algorithm for practical applications, the parameter $\theta = (\epsilon, \tau)$ of an HMM model needs to be known. While the accurate values of the the parameters may not always be known, estimation of parameter values from training data can be achieved. Depending on availability of data, there are broadly two different approaches for parameter estimation associated with the HMM. When training data contain both observed sequences of symbols and the hidden states that have generated them, distributions ϵ and τ can be computed with the maximum likelihood method through counting of all the

emissions and transitions labeled in the training data and assigning the frequencies to the probabilistic parameters.

A difficult situation arises when the states emitting observable symbols are not known or not included in the training data. To deal with this scenario of missing data, typically, the expectation maximization (EM) method is used to compute expected frequencies of event occurrences. Technically, EM is an iterative process consisting of alternate steps to compute expectation and to maximize it. Given the current parameter $\theta^{(t)}$, the *expected log-likelihood* $P(\mathbf{X}, \mathbf{Y}|\theta)$ of θ over all possible values of hidden variables in $\mathbf{X}$ (drawn from the training data) is computed and maximized for new parameters as

$$\begin{aligned}\theta^{(t+1)} &= \arg \max_{\theta} E_{\mathbf{X} \sim P(\cdot|\mathbf{Y}, \theta^{(t)})} [\log P(\mathbf{X}, \mathbf{Y}|\theta)] \\ &= \arg \max_{\theta} \sum_{\mathbf{X}} \log P(\mathbf{X}, \mathbf{Y}|\theta) P(\mathbf{X}|\mathbf{Y}, \theta^{(t)})\end{aligned}$$

For the HMM, the iterative process is efficiently realized by Baum–Welch algorithm [6] that coordinates two processes *forward* and *backward*, computing the total probability $\sum_{\mathbf{X}} P(\mathbf{X}, \mathbf{Y})$. The algorithm guarantees converging to a local optima.

2.2. Deep-Learning Preliminaries

Deep learning is a state-of-the-art technique to approximate functions of complex relationships between inputs and outputs via neural networks [26]. A neural network consists of stacked layers of computational units, each computing the weighted sum of its inputs followed by a nonlinear transformation. In practice, earlier layers often capture simple or local patterns of abstract, while deeper layers combine them to encode more global and structured information.

A neural network is learned through training on available data by settling those parameters associated with the computation units in the network. They are determined by minimizing an empirical objective function that quantifies the discrepancy between the network's predicted outputs and observed data. Arising from probabilistic modeling, the most common choice of objective, among others, is the negative log-likelihood of the data under a parameterized conditional distribution. Due to the typical non-convexity and high-dimensionality of the optimization problem, the network parameters are computed with iterative gradient-based methods via backpropagation that is reasonably efficient to achieve gradients [25].

The following are some basic functions which will also be used in our proposed α -HMM neural parameter estimator to be introduced in Section 4.

- **Unit-level functions**

sigmoid: The sigmoid function transforms a real number (e.g., output of a neural gate) to a probability in interval $(0, 1)$ for every $t \in \mathbb{R}$,

$$\sigma(t) = \frac{1}{1 + \exp(-t)} \in (0, 1)$$

softplus: The softplus function maps a real number to a nonnegative value for every $t \in \mathbb{R}$,

$$\text{softplus}(t) = \log(1 + \exp(t)) \in \mathbb{R}_{\geq 0}$$

- **Layer-level functions**

MLP [27]: An MLP is multilayer perceptron that defines mapping: $\mathbb{R}^d \rightarrow \mathbb{R}^m$ from an input layer to the output layer with one or more hidden layers in between, where neurons in two consecutive layers are fully connected.

softmax: The softmax function maps a vector $u \in \mathbb{R}^K$ to a probability vector $p \in [0, 1]^K$, such that, for $k = 1, \dots, K$,

$$p_k = \text{softmax}(u_k) = \frac{\exp(u_k)}{\sum_{j=1}^K \exp(u_j)}.$$

satisfying $\sum_{k=1}^K p_k = 1$. It is typically used as a normalization operator and applied at the output layer only.

- **Functions on sequences with embeddings**

Transformer encoder [28]: A Transformer encoder is a mapping function: $T_\theta : \mathbb{R}^{L \times d} \rightarrow \mathbb{R}^{L \times d}$ from an input length- L sequence of vectors to a sequence of contextualized representations of the same length. Each input vector (of d entries) is an embedding (semantics) of the corresponding input position, while each output vector is a weighted combination of all input positions (computed using self-attention).

Attention pooling: An attention pooling function, $\mathbb{R}^{L \times d} \rightarrow \mathbb{R}^d$, computes a weighted summary of a sequence of vectors, where the weights are learned to emphasize the most relevant elements. It assigns higher weight to positions that are more informative for the task, represented with aggregated values that focus on what matters most.

3. Arbitrary-Order Hidden Markov Model

3.1. α -HMM

The HMM is extended to encompass a framework for modeling higher-order dependencies that go beyond linear relationships between consecutive events in stochastic processes. Technically, the dependency between two states can be specified via a distribution of the joint probability of data emissions by the two states [22].

Definition 2. An influence is an ordered pair of states $\langle r, s \rangle$, $r, s \in S$ with probability distribution $\eta_{r,s}^a$ such that for every fixed $a \in \Sigma$,

$$\sum_{b \in \Sigma} \eta_{r,s}^a(b) = 1 \quad (2)$$

Let $\{(X_k, Y_k | k \in \mathbb{N})\}$ be a stochastic process where X_i and X_j , $i < j$, are two variables. Then X_i is a *parent* of X_j if there is an influence $\langle r, s \rangle$ such that $X_j = s$, $X_i = r$, and i is the largest integer.

In addition, we denote with $\pi(X_j) = \{X_i : i < j, X_i \text{ is a parent of } X_j\}$ the set of parents of X_j . Also let $\delta(X_i) = \{X_k : X_i \in \pi(X_k)\}$ represent the set of *decedents* of X_i .

Definition 3. An arbitrary-order hidden Markov model (α -HMM) is a bivariate stochastic process $\{(X_k, Y_k) : k \in \mathbb{N}\}$, where X_k draws values from a finite set S of states and Y_k draws values from a finite set Σ of symbols, such that, for finite $n \geq 1$, the joint probability $P(\{(X_k, Y_k) : k \leq n\})$ is defined as

$$P(\{(X_k, Y_k) : k \leq n\}) = P(X_1) \prod_{k=1}^{n-1} P(X_{k+1} | X_k) \prod_{k=1}^n P(Y_k | X_k, Y_i, X_i \in \pi(X_k)) \quad (3)$$

We point out that Equation (3) does not specify how the last conditional probability term $P(Y_k | X_k, Y_i, X_i \in \pi(X_k))$ is computed. This resembles the Bayesian network setting, where different computation methods may be adopted. Since variables $X_i \in \pi(X_k)$ may be interdependent, one viable method is the Noisy-OR [29]. While it is a heuristic method, normalization with the partition function can be applied to ensure a well-defined probability distribution, as used in clique factorization in Markov random fields [30,31], assuming

that influences on X_k from its parent variables X_i are independent. In the present work, however, we focus on the α_1 -HMM case. Accordingly,

$$P(Y_k | X_k, Y_i, X_i \in \pi(X_k)) = \prod_{X_i \in \pi(X_k)} P(Y_k | X_k, X_i, Y_i) \quad (4)$$

which is the assumption we adopt for this work.

By Definition 3, the conditional probability in (4) is defined as,

$$P(Y_k = b | X_k = s, X_i = r, Y_i = a) = \eta_{r,s}^a(b)$$

Definition 4. For fixed integer $h \geq 0$, an α -HMM is called an α_h -HMM if for every state variable X_k in the stochastic process, both $|\pi(X_k)| \leq h$ and $|\delta(X_k)| \leq h$.

In an α_h -HMM, no state variable can influence or be influenced by more than h variables. Thus, the α_0 -HMM is exactly the standard HMM, where symbol Y_k is completely determined by the its emission state X_k . The parameter h defines a family of α_h -HMM models with different expressive levels. In this paper, we focus on the α_1 -HMM case, which is the specific model used in our decoding algorithm and experiments.

3.2. Expressiveness of the α -HMM

The extension to the HMM with the new notion of influence has endowed modeling capability to the α -HMM. First, due to Definition 2 (for influence from X_i to X_j), where $i, i < j$, needs to be the largest, it becomes possible to model, with the α_1 -HMM, the influence of nested as well as parallel patterns that underline the essentials of SCFG. In addition, the α_1 -HMM can also model influences of crossing patterns, exceeding SCFG in expressiveness [19]. Figure 1 (1) shows an example of the α_1 -HMM's ability to model nested, parallel, and crossing patterns of influences.

Moreover, the α_1 -HMM has the modeling power of the k th-order HMMs, for any fixed $k \geq 1$. Figure 1 (2) shows an example of an α_1 -HMM that is a canonical simulation of second-order HMMs. It is not difficult to see that similar examples of the α_1 -HMM can be constructed to simulate the work of k th-order HMMs, for any fixed $k \geq 3$.

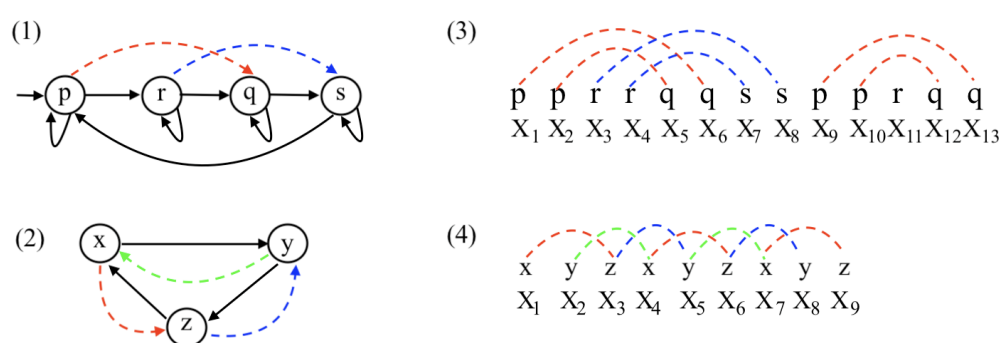


Figure 1. α_1 -HMM examples (1) and (2), where directed solid (black) arcs are transitions and directed dotted (colored) arcs are (directional) influences; emissions not shown. (3) A stochastic process showing nested, parallel, and crossing patterns of influences modeled by the α_1 -HMM in (1). (4) A stochastic process showing the effect of the α_1 -HMM in (2) simulating a second-order HMM.

3.3. Optimal Decoding with α_1 -HMM

One of the central tasks of modeling with the α -HMM is to decode the hidden states $\mathbf{X}$ that have generated an observed sequence $\mathbf{Y}$ of symbols over the alphabet Σ , such that the decoded hidden variables achieve the maximum likelihood $\mathbf{X}^* = \arg \max_{\mathbf{X}} P(\mathbf{Y} | \mathbf{X})$.

Definition 5. For an α -HMM of state set S and alphabet set Σ , the decoding problem is, given observation $\mathbf{a} = a_1 \dots a_n \in \Sigma^n$, to find a sequence of states $\rho^* \in S^n$, such that

$$\rho^* = \arg \max_{\rho \in S^n} P(\mathbf{X} = \rho, \mathbf{Y} = \mathbf{a}) \quad (5)$$

We call ρ^* the predicted optimal path for observation $\mathbf{a}$. To solve Equation (5), we resort to Formulas (3) and (4), which make it possible to develop a dynamic programming algorithm for the optimal decoding.

Definition 6. Let $1 \leq k \leq n$, $r \in S$, and $0 \leq i < k$. Define $p_{k,r,i}$ as the maximum probability of a state-sequence $\rho = s_1 \dots s_k$ in which s_j emits symbol a_j , $j = 1, \dots, k$, $s_k = r$, $\langle s_i, s_k \rangle$ is an influence, and i is the largest.

By (3) the joint probability can be factorized; we derive a recurrence for $p_{k,r,i}$ as, in an observed sequence of symbols $\mathbf{a} = a_1 \dots a_n$,

$$p_{k,r,i} = \eta_{s_i,r}^{a_i}(a_k) \cdot \max_{t \in S, 1 \leq h < k-1} \{\tau_t(r) \cdot p_{k-1,t,h}\} \quad (6)$$

The optimally decoded hidden state-sequence from the observed sequence $\mathbf{a}$ is

$$\rho^* = \arg \max_{r,i} \{p_{n,r,i}\}$$

It is clear that a table of dimensions $n \times n \times |S|$ can be used to store values $p_{k,r,i}$, which can be computed by a dynamic programming algorithm with the worst case time complexity of $O(n^3|S|^2)$, where $|S|$ is the number of states in the α -HMM and is a constant.

4. Amortized Parameter Inference for α_1 -HMM

Higher-order dependency structures modeled by the α -HMM can vary substantially across instances, especially in the present large dataset, making a single global parameter set insufficient. To address these challenges, we propose an amortized learning approach that enables scalable, instance-specific parameter inference and fully exploits the modeling capacity of the α -HMM. Rather than estimating a single set of global parameters shared by all sequences, we learn an input-conditioned neural parameter estimator f_θ from data which generates instance-specific α -HMM parameters for each input sequence of data whose dependence structures are to be predicted.

The input to an α -HMM is a variable-length sequence $\mathbf{a} = a_1 \dots a_n$ whose symbols are drawn from a fixed vocabulary (alphabet) Σ . The vocabulary size $|\Sigma|$ and the number of states $|S|$ determine the dimensionality of key model components: emission parameters are defined over Σ (distributions of size $|\Sigma|$ per emitting state), and both pairwise influence and transition parameters form square tables in space $\mathbb{R}^{|S| \times |S|}$. Specifically, every influence is defined over an ordered pair of states $\langle r, s \rangle$, encoding a left-to-right influence that accounts for dependence between their symbol emissions.

For amortized learning, we first map each input sequence into a vectorized pre-encoder representation before feeding it into the estimator f_θ . This vectorization can be implemented either by one-hot encoding, $\mathbf{x}_k = \mathbf{e}(a_k) \in \{0, 1\}^{|\Sigma|}$, or by a learned embedding table that maps each symbol to a d -dimensional vector $\mathbf{x}_k \in \mathbb{R}^d$, $1 \leq k \leq n$, where d is a user-specified hyperparameter. To preserve order information, we optionally add a positional embedding $\mathbf{p}_k$ so that the same symbol occurring at different positions can receive different representations. Positional embeddings can be implemented either as fixed sinusoidal features or as learned d -dimensional vectors. This yields a pre-encoder representation

sequence $(\tilde{\mathbf{x}}_1, \dots, \tilde{\mathbf{x}}_n)$, where $\tilde{\mathbf{x}}_k = \mathbf{x}_k + \mathbf{p}_k$. Overall, the above steps define a deterministic encoding from the input sequence to its pre-encoder representation.

However, both one-hot vectors and token embeddings are context-independent: the representation of a symbol depends only on its identity (and position, if positional encodings are used), but not on the surrounding symbols in the sequence. As a result, the same symbol occurring at the same position receives the same representation across different sequences, regardless of context. To incorporate contextual information, we apply a lightweight *Transformer encoder*. Its *self-attention* mechanism allows each position to aggregate information from the entire sequence, yielding context-dependent representations that are subsequently used by the parameter estimator. We denote the resulting contextual representation sequence by $\mathbf{H} = [\mathbf{h}_1, \dots, \mathbf{h}_n]^\top \in \mathbb{R}^{n \times d}$, where $\mathbf{h}_k \in \mathbb{R}^d$ is the contextual embedding at position k .

As discussed in Section 3, any specific α -HMM is parameterized by the parameters $\theta = \{\epsilon, \tau, \eta\}$ for emission, transition, and influence parameters, respectively. We perform amortized inference with an input-conditioned estimator f_θ that maps an input sequence $\mathbf{a}$ to instance-specific parameters:

$$f_\theta(\mathbf{a}) = (\epsilon(\mathbf{a}), \tau(\mathbf{a}), \eta(\mathbf{a})).$$

We point out that designing transitions τ and influences η is particularly important because they govern the complexity of the latent states and the overall modeling capacity. An overly simple model structure may lack the specificity in expressiveness required by the task, while an overly complex one can make inference and training unstable or data-inefficient. In the simplest setting, the total number of unconstrained outputs can be viewed as the sum of three parts: $|\Sigma|$ for emissions (of the involved state) per position, and $|\Sigma|^2$ for every pairwise influence, and an additional number of parameters for transitions (depending on the chosen transition structure and $|S|$, the number of states).

To map from the encoder space $\mathbf{H}$, we first apply an attention-based *pooling layer* to obtain a fixed-length summary vector $\bar{\mathbf{h}}$, which highlights informative positions in variable-length sequences. A *multi-layer perceptron* (MLP) then maps $\bar{\mathbf{h}}$ to three groups of unconstrained logits,

$$\text{MLP}_\theta(\bar{\mathbf{h}}) = (\mathbf{o}^{(\epsilon)}, \mathbf{o}^{(\tau)}, \mathbf{o}^{(\eta)}).$$

where $\mathbf{o}$ is the output of MLP.

Since ϵ and η represent (conditional) probability distributions, we apply *softmax* along the appropriate dimensions to project unconstrained logits into the corresponding probability space. For τ , whose transition structure is user-specified and may vary across settings, we recommend a *sigmoid*-based constrained construction so that the resulting transition probabilities respect the prescribed structure and each row sums to one.

We train the amortized estimator f_θ end-to-end (including the encoder) on a dataset

$$\mathcal{D} = \{(\mathbf{a}^{(i)}, \mathbf{y}^{(i)}, \boldsymbol{\rho}^{(i)})\}_{i=1}^N,$$

where $\mathbf{a}^{(i)} = a_1^{(i)} \dots a_{n_i}^{(i)}$ is the observed sequence, $\mathbf{y}^{(i)}$ is the task annotation, and $\boldsymbol{\rho}^{(i)} = \rho_1^{(i)} \dots \rho_{n_i}^{(i)}$ is a reference hidden-state sequence consistent with $\mathbf{y}^{(i)}$ under the task constraints.

We optimize f_θ with a composite objective consisting of four terms: (i) a path-level loss $\mathcal{L}_{\text{path}}$ that encourages the reference hidden-event path ρ to receive a high score under the inferred parameters; (ii) a state-sequence loss $\mathcal{L}_{\text{state}}$ that applies supervision to the reference hidden-state sequence through its local state transitions, thereby helping stabilize the learning of locally consistent state-transition behavior; (iii) a structured adversarial loss

$\mathcal{L}_{\text{adv}}$ that enforces separation between the reference path ρ and a competing decoded path ρ' under the current predicted parameters; and (iv) a task-specific structured loss $\mathcal{L}_{\text{task}}$ that further guides the model toward the desired structured output.

We combine these losses with learnable uncertainty-style weights:

$$\mathcal{L} = \sum_{c \in \{\text{path}, \text{state}, \text{adv}, \text{task}\}} \exp(-s_c) \mathcal{L}_c + s_c,$$

where $\{s_c\}$ are scalar parameters learned jointly with θ . Concrete task-specific instantiations of these four terms are given in the following section.

5. Performance Evaluation

We evaluate the proposed amortized parameter inference learning method for the α -HMM on RNA secondary structure prediction, a well-studied problem in computational biology with a rich history of modeling approaches. Existing approaches to RNA secondary structure prediction often involve a trade-off between interpretability and predictive accuracy. In this section, we show that RNA secondary structure prediction can be formulated in a simple, accurate, and interpretable manner with the α_1 -HMM, thereby demonstrating the effectiveness of the proposed parameter inference method.

5.1. α_1 -HMM for RNA Secondary Structure

A *ribonucleic acid* (RNA) consists of *nucleotides* adenine (A), guanine (G), cytosine (C), and uracil (U) that are chained together by covalent bonds to form the *primary sequence*. An RNA sequence can fold back itself via *canonical* Watson–Crick *base pairs* A–U and C–G, and the wobble base pair G–U via hydrogen bonds between the nucleotides. Consecutive, stacked base pairs, called *stems*, connected by unpaired single strands, called *loops*, constitute the *secondary structure* acting as a scaffold for its three-dimensional structure (Figure 2). Therefore, determination or prediction of RNA secondary structure is important to understanding the three-dimensional structures and thus functions of RNAs.

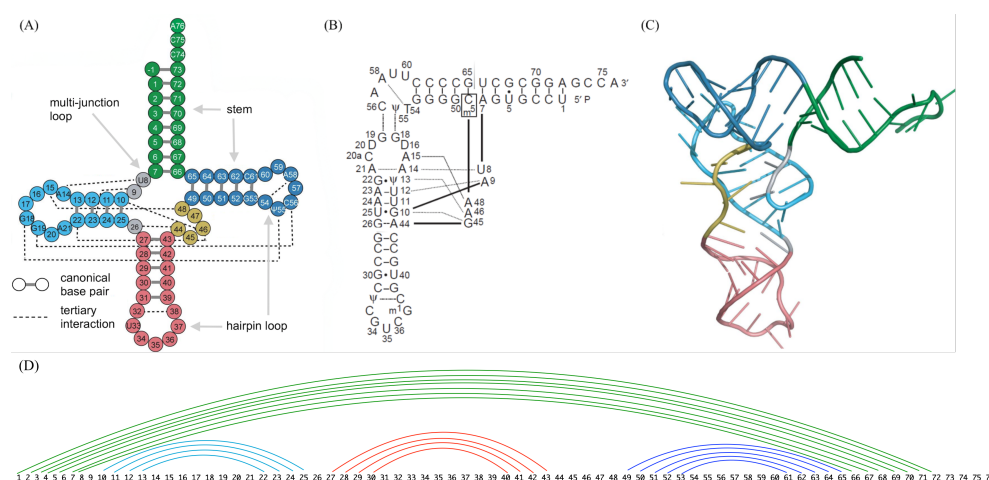


Figure 2. tRNA structure. (A) The typical secondary structure of tRNA, where stems, stacked canonical base pairs, are connected by non-pairing single-strand loops. (B) Details of additional tertiary interactions, twisting the molecule, scaffolded by the secondary structure, into a 3D shape rendered in (C). (D) An arc diagram to show the tRNA secondary structure, where nested and parallel patterns of relationships are induced by stacked canonical base pairs.

The problem of RNA secondary structure prediction is computationally discovering the most plausible secondary structure given only a primary sequence. In essence, this requires determining which (remote) pairs of nucleotides form exclusively canonical

base pairs. Such a phenomenon can be well-captured by the expressive power of the α_1 -HMM. Specifically, with the α_1 -HMM, the nucleotides of an RNA molecule are modeled by emissions of states, the primary sequence of nucleotides connected by covalent bonds is modeled by the sequence of state transitions, and most importantly, the base pairs are modeled by influences between states and their emissions of nucleotides.

Figure 3B shows a general α_1 -HMM graph for modeling RNA secondary structures. The model can capture all patterns of higher-level stem relationships in the RNA secondary structure, including nested, parallel, as in tRNA (Figure 2), and crossing (pseudoknot) as in Figure 3A. In addition, every stem, a stack of base pairs, is modeled with three influences η_1 , η_2 , and η_3 from state L to states P , Q , and R , respectively, where the inner most base pair is influenced by η_1 , the second inner most by η_2 , and the third and beyond by η_3 .

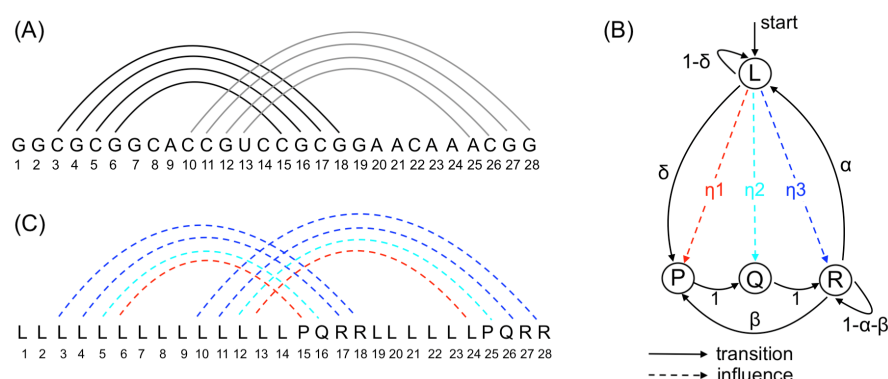


Figure 3. Illustration of the α -HMM modeling of RNA secondary structure. (A) The secondary structure of viral RNA (PDB id: 1L2X) with stacked base pairs in crossing patterns, a pseudoknot. (B) A general 4-state α_1 -HMM with influences to capture base pairs of nested, parallel, and crossing patterns in RNA secondary structures. (C) A potential sequence of states decoded with the α_1 -HMM given only the primary sequence, implicating the expected secondary structure.

It is clear to see that hidden states in the α_1 -HMM emitting an observable nucleotide sequence contain all the needed information for the secondary structure of the sequence. Thus, the structure prediction task is translated to a problem of decoding with the α_1 -HMM.

5.2. RNA-Specific Parameterization of the α_1 -HMM Estimator

To decode hidden states with the α_1 -HMM on an input RNA sequence $\mathbf{a} = a_1 \dots a_n$ of length n , the transition, emission, and influence parameters (τ, ϵ, η) are predicted by the learned parameter estimator f_θ , such that

$$f_{\theta}(\mathbf{a}) = (\tau, \epsilon, \eta).$$

From Figure 3, we formulate the following matrices:

$$\tau = \begin{pmatrix} 1-\Delta & \Delta & 0 & 0 \\ p_{\text{leave}} & 0 & 1-p_{\text{leave}} & 0 \\ q_{\text{leave}} & 0 & 0 & 1-q_{\text{leave}} \\ \alpha & \beta & 0 & 1-\alpha-\beta \end{pmatrix}, \quad \epsilon = (\epsilon_A \quad \epsilon_C \quad \epsilon_G \quad \epsilon_U), \quad \eta = \begin{pmatrix} \eta_{AA} & \eta_{AC} & \eta_{AG} & \eta_{AU} \\ \eta_{CA} & \eta_{CC} & \eta_{CG} & \eta_{CU} \\ \eta_{GA} & \eta_{GC} & \eta_{GG} & \eta_{GU} \\ \eta_{UA} & \eta_{UC} & \eta_{UG} & \eta_{UU} \end{pmatrix}. \quad (7)$$

Here, τ is the transition probability matrix over states L, P, Q, R . In particular, $\tau_L(P) = \Delta$ and $\tau_L(L) = 1 - \Delta$, while the R -row of τ is parameterized by $\alpha, \beta \geq 0$ with $\alpha + \beta \leq 1$. In the RNA implementation used in this work, we further introduce two leave probabilities, p_{leave} and q_{leave} , to control transitions from states P and Q , respectively.

Estimating (τ, ϵ, η) by f_θ for the α_1 -HMM on each input RNA sequence $\mathbf{a}$ requires five scalars for $(\Delta, \alpha, \beta, p_{\text{leave}}, q_{\text{leave}})$, four logits for ϵ , and $4 \times 4 = 16$ logits for η , for a total of $5 + 4 + 16 = 25$ sequence-specific parameters. Other quantities used during decoding, such

as the start distribution and several state-dependent costs, are global trainable parameters shared across sequences.

To infer these 25 parameters, we first pass the RNA sequence through a neural encoder, which maps each nucleotide (A, C, G, U) to a learnable 640-dimensional embedding vector. The resulting embedding sequence is then processed by a three-layer Transformer encoder with rotary positional encoding, a hidden dimension of 640, 10 attention heads, a feed-forward dimension of $2 \times 640 = 1280$ with GELU activations, and a dropout rate of 0.05. The contextualized representation at each sequence position is then passed through a token-level trunk comprising a linear projection from 640 to 320 dimensions, a GELU activation, a layer-normalization layer, and one residual feed-forward block with hidden dimensions of 320 and a dropout rate of 0.05. These token features are then aggregated by an attention-pooling layer to obtain a sequence-level representation, which is mapped to 25 sequence-specific parameters by a layer-normalization layer followed by a linear projection. For the tRNA experiments, we use a simplified instantiation of the model in which only Δ , α , and β are predicted as sequence-specific transition parameters, while p_{leave} and q_{leave} are not inferred and are instead fixed to zero. Consequently, the tRNA variant predicts $3 + 4 + 16 = 23$ sequence-specific parameters rather than 25. The tRNA encoder is also smaller, using a two-layer Transformer with hidden dimensions of 128, feed-forward dimensions of 192, 4 attention heads, learned absolute positional embeddings, and a dropout rate of 0.1. In both model variants, the three influence roles are parameter-tied through a single shared 4×4 nucleotide-pair matrix η .

5.3. Training Objective for RNA Secondary Structure Prediction

We train the model using AdamW [32] with a learning rate of 10^{-4} and a weight decay of 10^{-2} . Following the general loss formulation introduced in Section 4, we instantiate the RNA secondary structure objective as

$$\mathcal{L} = \exp(-s_{\text{path}}) \mathcal{L}_{\text{path}} + s_{\text{path}} + \exp(-s_{\text{state}}) \mathcal{L}_{\text{state}} + s_{\text{state}} \\ + \exp(-s_{\text{adv}}) \mathcal{L}_{\text{adv}} + s_{\text{adv}} + \exp(-s_{\text{task}}) \mathcal{L}_{\text{task}} + s_{\text{task}},$$

where s_{path} , s_{state} , s_{adv} , and s_{task} are learnable scalar weights.

Let $\mathbf{a}$ denote an input RNA sequence, $\boldsymbol{\rho}$ the reference hidden-state sequence, and $\mathbf{y}$ the reference dot-bracket secondary structure. Let $S_{\theta}(\mathbf{a}, \boldsymbol{\rho}, \mathbf{y})$ denote the score of the reference state path under the sequence-specific parameters predicted by $f_{\theta}(\mathbf{a})$. Then the path-level loss is defined as

$$\mathcal{L}_{\text{path}} = \text{softplus}(m_{\text{path}} - S_{\theta}(\mathbf{a}, \boldsymbol{\rho}, \mathbf{y})),$$

which encourages the reference path to receive a sufficiently high score.

The structured adversarial loss is defined as

$$\mathcal{L}_{\text{adv}} = \frac{1}{K} \sum_{k=1}^K \text{softplus}(S_{\theta}(\mathbf{a}, \boldsymbol{\rho}'_k, \mathbf{y}'_k) - S_{\theta}(\mathbf{a}, \boldsymbol{\rho}, \mathbf{y}) + m_{\text{adv}}),$$

where $\{(\boldsymbol{\rho}'_k, \mathbf{y}'_k)\}_{k=1}^K$ are decoded hard negative candidates and $m_{\text{path}}, m_{\text{adv}} \geq 0$ are margin constants.

The state-sequence loss is defined as the negative log-likelihood of the reference hidden-state chain under the predicted start and transition distributions:

$$\mathcal{L}_{\text{state}} = -\log P_{\theta}(\rho_1) - \sum_{i=1}^{n-1} \log P_{\theta}(\rho_{i+1} | \rho_i).$$

In the RNA secondary structure setting, the task-specific term $\mathcal{L}_{\text{task}}$ is instantiated as a pairwise base-pair supervision loss:

$$\mathcal{L}_{\text{task}} = \frac{1}{|\mathcal{P}^+|} \sum_{(i,j) \in \mathcal{P}^+} \text{softplus}(-s_{\theta}(i,j)) + \frac{1}{|\mathcal{P}^-|} \sum_{(i,j) \in \mathcal{P}^-} \text{softplus}(s_{\theta}(i,j)),$$

where $\mathcal{P}^+$ and $\mathcal{P}^-$ denote positive and sampled negative base-pair candidates, respectively.

Decoding is discrete and non-differentiable in our implementation. In particular, the Viterbi-style dynamic program is used only to generate hard negative candidates for $\mathcal{L}_{\text{adv}}$, and gradients are not propagated through the discrete backtracking step. Thus, training does not rely on REINFORCE-style gradient estimation; instead, the amortized estimator is optimized through the differentiable surrogate losses above.

5.4. Datasets and Evaluation Settings

We evaluate the α_1 -HMM on RNA secondary structure prediction under four complementary settings designed to assess performance within the tRNA family, on temporally held-out tRNA sequences, on broader RNA families within the training distribution, and on out-of-domain RNA families.

First, we evaluate the model on the curated T-psi-C dataset [33], which contains annotated tRNA sequences. On this dataset, we conduct two evaluations: (i) an **in-domain** evaluation using **10-fold cross-validation** on tRNA sequences released before 2024; and (ii) a **temporally held-out** evaluation in which the model is trained on all pre-2024 sequences and evaluated on 14 tRNA sequences released in 2024.

Second, to assess generalization beyond the relative tRNA family, we train the broader-family model on RNAStrAlign and evaluate it on both an in-domain test split and the out-of-domain ArchiveII benchmark. RNAStrAlign is a widely used RNA secondary structure dataset containing 37,149 sequences from eight RNA families: 16S rRNA, tRNA, 5S rRNA, group I intron (grp1), SRP, tmRNA, RNase P, and telomerase RNA. ArchiveII contains 3975 sequences from ten RNA families, including two additional families, 23S rRNAs and group II introns (grp2), beyond those represented in RNAStrAlign.

In the broader-family experiments, we follow the general data-processing protocol adopted for learning-based RNA secondary structure prediction benchmarks and exclude pseudoknotted structures and sequences longer than 600 nucleotides, since these are beyond the scope of the present work. After preprocessing, the resulting dataset sizes are 13,950 sequences for training, 2073 for validation, 1439 for the in-domain RNAStrAlign test split, and 1119 for the out-of-domain ArchiveII test set.

5.5. Baselines and Evaluation Protocol

We compare against CONTRAFOLD [34], ViennaRNA (RNAfold MFE and centroid) [35], LINEARFOLD [36], SPOT-RNA [37], RNA-FM [38], E2efold [39], MXfold2 [40], and KnotFold [41].

For the pre-2024 T-psi-C benchmark, all methods are evaluated under the same 10-fold cross-validation protocol, and we report mean $\pm$ std across folds (Table 1). For broader-family evaluation, we additionally report results on ArchiveII (Table 2).

5.6. Results on tRNA Benchmarks

As summarized in Table 1, our neural α_1 -HMM achieves strong pair-level accuracy on tRNA, with particularly strong precision and F1 compared with the baselines considered here. These results indicate that the proposed amortized parameter inference framework is highly effective in the homogeneous tRNA setting.

Negative controls. To discourage memorization of common tRNA secondary-structure patterns, we construct two negative sets: (i) **random negatives**, where for each tRNA length L we generate a sequence by sampling each nucleotide uniformly from $\{A, C, G, U\}$; and (ii) **shuffled negatives**, where each tRNA sequence is randomly permuted, preserving the mononucleotide composition but disrupting the biological structure. These negatives are used only for evaluation. Our method yields paired-nucleotide ratios (PNRs) of 0.223 ± 0.005 on shuffled negatives and 0.219 ± 0.006 on random negatives, indicating that the model does not simply predict high pairing rates on arbitrary sequences.

Temporal evaluation. To assess temporal generalization, we evaluate the model on 14 tRNA sequences released in 2024, all of which are excluded from training and validation. α_1 -HMM achieves precision = 1.000, recall = 1.000, and F1 = 1.000 on this held-out set. Although this result is encouraging, it should be interpreted cautiously because the temporal test set is small and drawn from a structurally homogeneous RNA family. We visualize one such prediction in Figure 4, showing high-quality structure recovery on entry tdbR00000239 compared with several baseline methods.

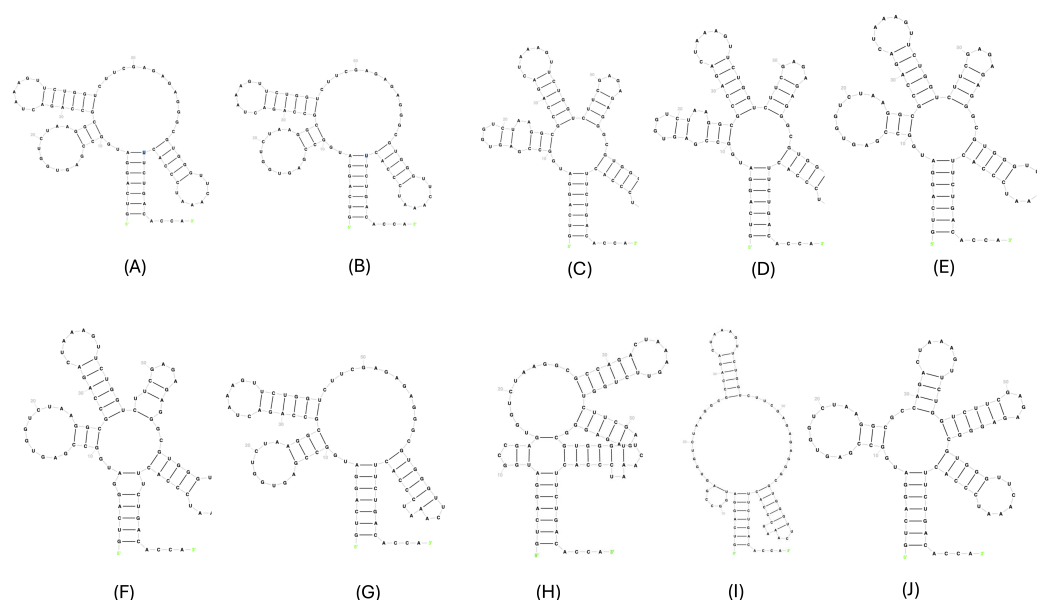


Figure 4. Qualitative comparison of RNA secondary structure predictions on the same sequence. (A) Ground truth (GT). (B) Our prediction, which is the only one identical to the GT. (C) CONTRAfold. (D) RNAfold (ViennaRNA). (E) ViennaRNA (centroid). (F) E2Efold. (G) KnotFold. (H) RNA-FM. (I) SPOT-RNA. (J) MXfold2. All secondary-structure visualizations were generated using R2DT [42]. The sequence is from the T-Psi-C Database 2.0 (entry tdbR00000239), which is part of the temporal test set.

Table 1. Performance comparison across methods on the tRNA benchmark. Pair-level precision (P), recall (R), and F1 are reported as mean $\pm$ std across 10 folds on the pre-2024 T-psi-C dataset.

Method	P	R	F1
ViennaRNA-MFE (2011) [35]	0.668 $\pm$ 0.044	0.720 $\pm$ 0.046	0.692 $\pm$ 0.045
ViennaRNA-Centroid (2011) [35]	0.711 $\pm$ 0.049	0.672 $\pm$ 0.041	0.682 $\pm$ 0.043
LinearFold (2019) [36]	0.668 $\pm$ 0.044	0.719 $\pm$ 0.047	0.691 $\pm$ 0.045
CONTRAfold (2006) [34]	0.695 $\pm$ 0.015	0.765 $\pm$ 0.016	0.724 $\pm$ 0.015
SPOT-RNA (2019) [37]	0.830 $\pm$ 0.009	0.988 $\pm$ 0.008	0.901 $\pm$ 0.009
RNA-FM (2022) [38]	0.739 $\pm$ 0.007	0.962 $\pm$ 0.009	0.835 $\pm$ 0.007
E2Efold (2020) [39]	0.950 $\pm$ 0.022	0.940 $\pm$ 0.023	0.944 $\pm$ 0.022
MXfold2 (2021) [40]	0.951 $\pm$ 0.011	0.953 $\pm$ 0.012	0.951 $\pm$ 0.011
KnotFold (2024) [41]	0.909 $\pm$ 0.007	0.993 $\pm$ 0.006	0.948 $\pm$ 0.006
Our method	0.986 $\pm$ 0.008	0.989 $\pm$ 0.008	0.987 $\pm$ 0.008

5.7. Results on Broader RNA Families

We further evaluate the proposed method beyond the relatively homogeneous tRNA family using the broader-family RNAstralign and ArchiveII benchmarks. On the in-domain RNAstralign test split, our method achieves precision = 0.9820, recall = 0.9759 and F1 = 0.9789. On the out-of-domain ArchiveII benchmark, Table 2 shows that our method achieves the strongest F1 among the compared methods while also maintaining very high precision. Together, these results suggest that the proposed amortized α_1 -HMM framework remains competitive not only within the broader training distribution but also on structurally diverse RNA families outside it. Figure 5 shows a long 16S rRNA example (484 nt) with high base-pair accuracy (F1 = 0.9682), indicating that our model captures long-range base-pair interactions.

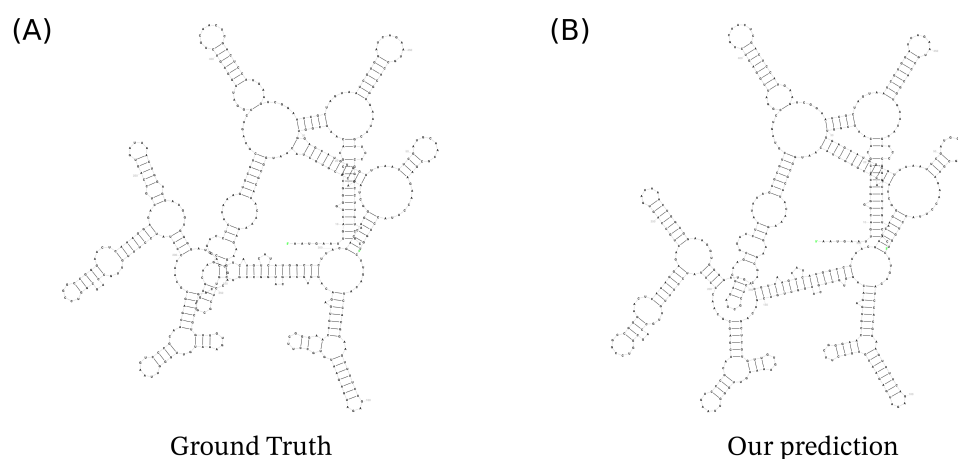


Figure 5. Qualitative visualization on a 16S rRNA example. (A) Ground truth. (B) Structure predicted by our method. Despite the sequence length (484 nt), our method achieved high base-pair accuracy (F1 = 0.9682).

Table 2. Baseline performance on ArchiveII. We report pair-level precision (P), recall (R), and F1.

Method	P	R	F1
ViennaRNA-MFE	0.5880	0.6690	0.6260
ViennaRNA-Centroid	0.6430	0.6440	0.6430
LinearFold	0.5860	0.6660	0.6230
CONTRAFold	0.6158	0.6895	0.6506
MXfold2	0.7630	0.8200	0.7900
SPOT-RNA	0.7340	0.8260	0.7770
KnotFold	0.7830	0.8720	0.8250
RNA-FM	0.6710	0.8120	0.7350
E2efold	0.7380	0.5810	0.6500
Our method	0.9013	0.8084	0.8523

6. Conclusions

We proposed a novel amortized inference pipeline for sequence modeling, centered on the degree-1 arbitrary-order hidden Markov model (α_1 -HMM). By learning input-conditioned parameters via a neural estimator, our method enables instance-specific probabilistic modeling in a single forward pass, avoiding costly per-sequence iterative parameter estimation while preserving the interpretability of structured latent-variable models.

In the context of RNA secondary structure prediction, our framework demonstrates high accuracy and robustness, even under challenging negative controls, while offering explainable predictions. Although the current model only captures first-degree influence

(α_1), future work could explore higher-order dependencies (α_k for $k > 1$) to model richer long-range interactions. Importantly, amortized inference makes such extensions practical by providing efficient instance-conditioned parameterization; complementary directions include constrained or approximate decoding strategies to further improve scalability beyond the $O(N^3)$ worst-case complexity of exact exclusive-pair decoding.

Author Contributions: Conceptualization, S.Z. and L.C.; methodology, S.Z.; software, S.Z.; writing—review and editing, S.Z. and L.C.; supervision, L.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The datasets used in this study are publicly available. The T-Psi-C Database is available at T-Psi-C Database. The RNAStralign and ArchiveII datasets used in this work are available via the MultiMolecule Hugging Face dataset releases at RNAStralign and ArchiveII, respectively.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Rabiner, L.; Juang, B. An Introduction to Hidden Markov Models. *IEEE ASSP Mag.* **1986**, *3*, 4–16. [\[CrossRef\]](#)
2. Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. IEEE* **1989**, *77*, 257–286. [\[CrossRef\]](#)
3. Ephraim, Y.; Merhav, N. Hidden Markov processes. *IEEE Trans. Inf. Theory* **2002**, *48*, 1518–1569. [\[CrossRef\]](#)
4. Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **1967**, *13*, 260–269. [\[CrossRef\]](#)
5. Baum, L.E.; Petrie, T. Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Ann. Math. Stat.* **1966**, *37*, 1554–1563. [\[CrossRef\]](#)
6. Baum, L.E.; Petrie, T.; Soules, G.; Weiss, N. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Ann. Math. Stat.* **1970**, *41*, 164–171. [\[CrossRef\]](#)
7. Sainburg, T.; Mai, A.; Gentner, T.Q. Long-range sequential dependencies precede complex syntactic production in language acquisition. *Proc. R. Soc. B Biol. Sci.* **2022**, *289*, 20212657. [\[CrossRef\]](#)
8. Collins, M. Three generative, lexicalised models for statistical parsing. In Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, Madrid, Spain, 7–12 July 1997; pp. 16–23. [\[CrossRef\]](#)
9. Tinoco, I.J.; Bustamante, C. How RNA folds. *J. Mol. Biol.* **1999**, *293*, 271–281. [\[PubMed\]](#)
10. Akutsu, T.; Miyano, S. On the approximation of protein threading. *Theor. Comput. Sci.* **1999**, *210*, 261–275. [\[CrossRef\]](#)
11. Sezer, O.B.; Gudelek, M.U.; Ozbayoglu, A.M. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Appl. Soft Comput.* **2020**, *90*, 106181. [\[CrossRef\]](#)
12. Duarte, F.S.; Rios, R.A.; Hruschka, E.R.; de Mello, R.F. Decomposing time series into deterministic and stochastic influences: A survey. *Digit. Signal Process.* **2019**, *95*, 102582. [\[CrossRef\]](#)
13. Allen, J.F.; Ferguson, G. Actions and events in interval temporal logic. *J. Log. Comput.* **1994**, *4*, 531–579. [\[CrossRef\]](#)
14. Mitchell, T.R.; Thompson, L.; Peterson, E.; Cronk, R. Temporal adjustments in the evaluation of events: The “rosy view”. *J. Exp. Soc. Psychol.* **1997**, *33*, 421–448. [\[CrossRef\]](#)
15. Koski, T. *Hidden Markov Models for Bioinformatics Volume 2*; Springer Science and Business Media: New York, NY, USA, 2001.
16. Zaki, M.J.; Carothers, C.D.; Szymanski, B.K. VOGUE: A variable order hidden markov model with duration based on frequent sequence mining. *ACM Trans. Knowl. Discov. Data (TKDD)* **2010**, *4*, 1–31. [\[CrossRef\]](#)
17. Xiao, J.H.; Liu, B.Q.; Wang, X.L. Principles of non-stationary hidden markov model and its applications to sequence labeling task. In Proceedings of the Second International Joint Conference on Natural Language Processing, Jeju Island, Republic of Korea, 11–13 October 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 827–837. [\[CrossRef\]](#)
18. Begleiter, R.; El-Yaniv, R.; Yona, G. On prediction using variable order Markov models. *J. Artif. Int. Res.* **2004**, *22*, 385–421. [\[CrossRef\]](#)
19. Lari, K.; Young, S. The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Comput. Speech Lang.* **1990**, *4*, 35–56. [\[CrossRef\]](#)
20. Lari, K.; Young, S. Applications of stochastic context-free grammars using the Inside-Outside algorithm. *Comput. Speech Lang.* **1991**, *5*, 237–257. [\[CrossRef\]](#)

21. Johnson, M. PCFG Models of Linguistic Tree Representations. *Comput. Linguist.* **1998**, *24*, 613–632.
22. Dastjerdi, F.R.; Robinson, D.A.; Cai, L. α -HMM and optimal decoding higher-order structures on sequential data. *J. Comput. Math. Data Sci.* **2022**, *5*, 100065. [[CrossRef](#)]
23. Margossian, C.C.; Blei, D.M. Amortized Variational Inference: When and Why? In Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence, Barcelona, Spain, 15–19 July 2024; Volume 244, pp. 2434–2449.
24. Zammit-Mangion, A.; Sainsbury-Dale, M.; Huser, R. Neural Methods for Amortized Inference. *Annu. Rev. Stat. Its Appl.* **2025**, *12*, 311–335. [[CrossRef](#)]
25. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016. Available online: <http://www.deeplearningbook.org> (accessed on 24 March 2026).
26. Hornik, K.; Stinchcombe, M.; White, H. Multilayer feedforward networks are universal approximators. *Neural Netw.* **1989**, *2*, 359–366. [[CrossRef](#)]
27. Rumelhart, D.E.; Hinton, G.E.; Williams, R.J. Learning representations by back-propagating errors. *Nature* **1986**, *323*, 533–536. [[CrossRef](#)]
28. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; pp. 6000–6010.
29. Srinivas, S. A generalization of the Noisy-Or model. In Proceedings of the Conference on Uncertainty in Artificial Intelligence, Washington, DC, USA, 9–11 July 1993.
30. Grimmett, G.R. A theorem about random fields. *Bull. Lond. Math. Soc.* **1973**, *5*, 81–84. [[CrossRef](#)]
31. Clifford, R. Markov random fields in statistics. In *Disorder in Physical Systems: A Volume in Honour of John M. Hammersley*; Grimmett, G.R., Welsh, D.J.A., Eds.; Oxford University Press: Oxford, UK, 1990.
32. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, Toulon, France, 24–26 April 2017.
33. Sajek, M.P.; Woźniak, T.; Sprinzl, M.; Jaruzelska, J.; Barciszewski, J. T-psi-C: User friendly database of tRNA sequences and structures. *Nucleic Acids Res.* **2020**, *48*, D256–D260. [[CrossRef](#)]
34. Do, C.B.; Woods, D.A.; Batzoglu, S. CONTRAfold. *Bioinformatics* **2006**, *22*, e90–e98. [[CrossRef](#)]
35. Lorenz, R.; Bernhart, S.H.; Höner zu Siederdissen, C.; Tafer, H.; Flamm, C.; Stadler, P.F.; Hofacker, I.L. ViennaRNA Package 2.0. *Algorithms Mol. Biol.* **2011**, *6*, 26. [[CrossRef](#)]
36. Huang, L.; Zhang, H.; Deng, D.; Zhao, K.; Liu, K.; Hendrix, D.A.; Mathews, D.H. LinearFold: Linear-time approximate RNA folding by 5'-to-3' dynamic programming and beam search. *Bioinformatics* **2019**, *35*, i295–i304. [[CrossRef](#)]
37. Singh, J.; Hanson, J.; Paliwal, K.; Zhou, Y. RNA secondary structure prediction using an ensemble of two-dimensional deep neural networks and transfer learning. *Nat. Commun.* **2019**, *10*, 5407. [[CrossRef](#)] [[PubMed](#)]
38. Chen, J.; Hu, Z.; Sun, S.; Tan, Q.; Wang, Y.; Yu, Q.; Zong, L.; Hong, L.; Xiao, J.; Shen, T.; et al. Interpretable RNA Foundation Model from Unannotated Data for Highly Accurate RNA Structure and Function Predictions. *arXiv* **2022**, arXiv:2204.00300. [[CrossRef](#)]
39. Chen, X.; Li, Y.; Umarov, R.; Gao, X.; Song, L. RNA Secondary Structure Prediction By Learning Unrolled Algorithms. *arXiv* **2020**, arXiv:2002.05810. [[CrossRef](#)]
40. Sato, K.; Akiyama, M.; Sakakibara, Y. RNA secondary structure prediction using deep learning with thermodynamic integration. *Nat. Commun.* **2021**, *12*, 941. [[CrossRef](#)] [[PubMed](#)]
41. Gong, T.; Ju, F.; Bu, D. Accurate prediction of RNA secondary structure including pseudoknots through solving minimum-cost flow with learned potentials. *Commun. Biol.* **2024**, *7*, 297. [[CrossRef](#)] [[PubMed](#)]
42. McCann, H.; Meade, C.; Williams, L.; Petrov, A.; Johnson, P.; Simon, A.; Hoksza, D.; Nawrocki, E.; Chan, P.; Lowe, T.; et al. R2DT: A comprehensive platform for visualizing RNA secondary structure. *Nucleic Acids Res.* **2025**, *53*, gkaf032. [[CrossRef](#)] [[PubMed](#)]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.