NONPARAMETRIC TEACHING FOR SEQUENTIAL PROPERTY LEARNERS

Anonymous authors
Paper under double-blind review

ABSTRACT

Determining the properties of sequence-structured data, *e.g.*, the sentiment of a text, fundamentally requires learning the implicit relationship that maps sequences to their corresponding properties. This learning process is often expensive for sequential property learners like Recurrent Neural Networks (RNNs). To tackle this, we introduce a paradigm called **Re**current **N**eural **T**eaching (ReNT), which reinterprets the learning process through a novel nonparametric teaching lens. Specifically, the latter provides a theoretical framework for teaching implicitly defined (*i.e.*, nonparametric) mappings via example selection. Such an implicit mapping is realized by a dense set of sequence-property pairs, with the ReNT teacher selecting a subset of them to facilitate faster convergence in RNN training. By analytically investigating the effect of sequence order on parameter-based gradient descent during training, and recasting the evolution of RNNs—driven by parameter updates—through functional gradient descent in nonparametric teaching, we reveal *for the first time* that teaching sequential property learners (*i.e.*, RNNs) is consistent with teaching order-aware nonparametric learners. These new findings readily prompt ReNT to improve the learning efficiency of the sequential property learner, achieving substantial cuts in training time for sequence-level (-32.77% to -46.39%) and element-level (-36% to -39.17%) tasks, while still preserving its generalization performance.

1 INTRODUCTION

Sequence-structured data, commonly referred to as sequences, are typically represented by ordered lists of elements, where each element is associated with a specific order or timestamp (Sutskever et al., 2014; Király & Oberhauser, 2019). Sequential properties can be classified as either sequence-level or element-level[1] (Purwins et al., 2019). For instance, the text category is a sequence-level property in text sequences (Liu et al., 2016; Kowsari et al., 2019), whereas each word transcribed at a particular time step is an element-level property in a speech-to-text sequence (Purwins et al., 2019). Inferring these sequential properties essentially requires learning the implicit mapping from sequences to these properties (Király & Oberhauser, 2019; Alley et al., 2019; Otovic et al., 2022). A visual representation of this mapping is provided in Figure 1. As a notable sequential property learner, the Recurrent Neural Network (RNN) (Elman, 1990; Jordan, 1997) has demonstrated exceptional generalizability, achieving remarkable performance in diverse domains including machine translation (Cho et al., 2014a; Liu et al., 2014), time series analysis (Hewamalage et al., 2021; Lu & Xu, 2024), and DNA sequence processing (Liu et al., 2019; AlQuraishi & Sorger, 2021; Abd-Alhalem et al., 2021).

Nevertheless, the learning process of the implicit mapping—*i.e.*, the training—can be quite costly for RNNs, especially when handling large-scale sequence tasks (Salem et al., 2019; Yadav et al., 2022; Liu et al., 2022). For instance, determining the authenticity of news—a sequence-level property—on social media involves processing millions of news articles (Nasir et al., 2021). When it comes to element-level property learning tasks, the scale can become overwhelmingly large (Sak et al., 2014; Hermanto et al., 2015; Lu & Xu, 2024). Consequently, there is an urgent need to lower training costs and enhance learning efficiency.

[1]This paper focuses on the sequence level in its discussion, unless indicated otherwise, with the element level viewed as a multi-dimensional generalization.

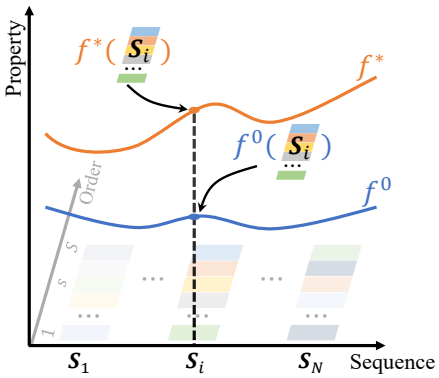


Figure 1: An intuitive illustration of the implicit mapping f^* between a sequence $\boldsymbol{S}$ and its property $f^*(\boldsymbol{S})$, where f^0 represents the mapping of the initial sequential property learner, *e.g.*, an initialized RNN.

Recent research on nonparametric teaching (Zhang et al., 2023b;a; 2024; 2025) provides a promising solution to this problem. Specifically, nonparametric teaching offers a theoretical framework for efficiently selecting examples when the target mapping is nonparametric, *i.e.*, implicitly defined. It expands on the concept of machine teaching (Zhu, 2015; Zhu et al., 2018)—involves crafting a training set (dubbed the teaching set) to aid the learner in quickly converging the target functions—but relaxes the assumption that target functions are parametric (Liu et al., 2017; 2018), thus enabling the teaching of nonparametric (viz. non-closed-form) target functions, with an emphasis on exploring function space. Unfortunately, existing studies solely focus on treating inputs as independent entities and fail to consider the sequential nature and temporal dependencies, resulting in difficulties when handling sequential data (Vinyals et al., 2016). Additionally, updating an RNN typically involves gradient descent in parameter space, which differs from the functional gradient descent approach used in nonparametric teaching within function space (Zhang et al., 2023b;a; 2024). These necessitate further investigation before applying nonparametric teaching theory to the learning of sequential properties.

To this end, we conduct a systematic examination of how sequence order influences RNN gradient-based training in both parameter and function spaces. Specifically, we analytically examine the impact of element order in an input sequence on parameter-based gradient descent within parameter space, and explicitly demonstrate that the parameter gradient retains its form when the sequence length is scaled. The order-aware update in parameter space drives the evolution of the RNN, which can be described using the dynamic recurrent neural tangent kernel (RNTK) (Alemohammad et al., 2021; Emami et al., 2021), and is then formulated into function space. We show that this dynamic RNTK converges to the order-aware canonical kernel used in functional gradient descent, indicating that the evolution of the RNN under parameter gradient descent is consistent with that under functional gradient descent. Hence, it is natural to view the learning process of sequential properties through the theoretical lens of nonparametric teaching: the target mapping is realized by a dense set of sequence-property pairs, from which the teacher selects a subset to feed to the RNN, promoting rapid convergence of the sequential property learner. Consequently, to improve RNN learning efficiency, we introduce a novel paradigm called ReNT, where the teacher uses a variation of the greedy teaching algorithm from nonparametric teaching for sequential property learning, specifically selecting sequences that exhibit the greatest discrepancy between their true property values and the RNN outputs. Lastly, we perform extensive experiments to demonstrate the effectiveness of ReNT across various scenarios, including both sequence-level and element-level tasks. Our key contributions are outlined as follows:

- We propose ReNT, a novel paradigm that frames sequential property learning within the theoretical framework of nonparametric teaching, allowing the use of greedy algorithms from this framework to significantly improve the learning efficiency of the sequential property learner, RNN.
- We analytically investigate how sequence order affects parameter-based gradient descent in parameter space, uncovering the consistency between the evolution of RNN driven by parameter updates and that under functional gradient descent in nonparametric teaching. We further show that the dynamic RNTK, derived from gradient descent on parameters, converges to the order-aware canonical kernel of functional gradient descent. These bridge nonparametric teaching theory with sequential property learning, thereby broadening the scope of nonparametric teaching in sequential property learning contexts.
- We showcase the effectiveness of ReNT through comprehensive experiments in sequential property learning, covering both sequence and element levels. Specifically, ReNT reduces training time for sequence-level (-32.77% to -46.39%) and element-level (-36% to -39.17%) tasks, all while preserving its generalization performance.

2 Related Works

Sequential property learning. With the prevalence of sequence-structured data in real-world applications, there has been a notable rise in research interest in sequences (Lipton et al., 2015; Salehinejad et al., 2017; Yu et al., 2019), particularly in learning implicit mappings from sequences to interested properties (Király & Oberhauser, 2019; Alley et al., 2019; Otovic et al., 2022) for a variety of downstream tasks. These tasks span areas such as text analysis (Liu et al., 2016; Kowsari et al., 2019), time series modeling (Hewamalage et al., 2021; Lu & Xu, 2024), and biological sequence processing (Liu et al., 2019; AlQuraishi & Sorger, 2021; Abd-Alhalem et al., 2021). Various efforts have been made to enhance learner design for better mapping learning, such as the recurrent neural network (RNN) learner (Elman, 1990; Jordan, 1997), the long short-term memory learner (Hochreiter & Schmidhuber, 1997), the gated recurrent unit learner (Cho et al., 2014b), and the state space learner (Smith et al., 2023). Efforts to boost learning efficiency have also been pursued, such as parallelized training (Chen et al., 2015; Ruíz et al., 2019), sparse training (Liu et al., 2021), and block-term tensor decomposition (Ye et al., 2018). In contrast, we tackle sequential property learning from a new angle of nonparametric teaching (Zhang et al., 2023b;a) and apply an adapted version of the greedy algorithm to improve the training efficiency of RNNs.

Nonparametric teaching. Machine teaching (Zhu, 2015; Zhu et al., 2018) is concerned with crafting a teaching set that helps the learner swiftly converge to a target model function. It can be seen as the inverse of machine learning: while machine learning focuses on learning a mapping from a given training set, machine teaching strives to design the set based on a desired mapping. Its effectiveness has been shown in a variety of fields, such as crowdsourcing (Singla et al., 2014; Zhou et al., 2018), robustness (Alfeld et al., 2017; Ma et al., 2019; Rakhsha et al., 2020), and computer vision (Wang et al., 2021; Wang & Vasconcelos, 2021). Nonparametric teaching (Zhang et al., 2023b;a) builds upon iterative machine teaching (Liu et al., 2017; 2018) by extending the parameterized family of target mappings to incorporate a broader, more general nonparametric framework. This theoretical framework has also been proven effective in improving the efficiency of multilayer perceptrons (MLPs) for learning implicit functions from signal coordinates to corresponding values (Sitzmann et al., 2020; Tancik et al., 2020; Zhang et al., 2024; Luo et al., 2024), as well as enhancing the training efficiency of graph convolutional networks for learning implicit mappings from graphs to their relevant properties (Zhang et al., 2025). Nevertheless, the neglect of the sequential nature in these studies makes it difficult to directly apply their findings to general tasks involving sequence-structured data (Sutskever et al., 2014; Vinyals et al., 2016; Liu et al., 2016; Purwins et al., 2019). This work systematically examines the impact of sequence order and highlights the alignment between the evolution of RNN driven by parameter updates and that guided by functional gradient descent in nonparametric teaching. These insights, for the first time, expand the reach of nonparametric teaching in sequential property learning and position our ReNT as a way to enhance RNN learning efficiency.

3 Background

Notation.[2] Let $\boldsymbol{S}_{1:S} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_S) \in \mathcal{S}$ be a sequence of length S, where $\boldsymbol{x}_s \in \mathbb{R}^d$ represents the d-dimensional feature vector for the element of order $s \in \mathbb{N}_S$ ($\mathbb{N}_S \coloneqq \{1, \cdots, S\}$). Each $\boldsymbol{x}_s$ is a row vector, denoted as $[x_{s,j}]_d^\top = (x_{s,1}, \cdots, x_{s,d})$, and the collection of all feature vectors is given by an $S \times d$ feature matrix, denoted $\boldsymbol{X}_{S\times d}$ (or simply $\boldsymbol{X}$). The s-th row and i-th column of this matrix, which correspond to the s-th element and i-th feature, are represented by $\boldsymbol{X}_{(s,:)}$ and $\boldsymbol{X}_{(:,i)}$, respectively. Alternatively, these can be written as $\boldsymbol{e}_s^\top \boldsymbol{X}$ and $\boldsymbol{X}\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is a basis vector with its i-th entry equal to 1 and all other entries equal to 0. The property of the sequence is denoted by $\boldsymbol{y} \in \mathcal{Y}$, where $\boldsymbol{y}$ is a scalar for sequence-level properties (*i.e.*, $\mathcal{Y} \subseteq \mathbb{R}$) and a vector for element-level properties (*i.e.*, $\mathcal{Y} \subseteq \mathbb{R}^n$). A set containing m items is written as $\{a_i\}_m$. If $\{a_i\}_m \subseteq \{a_i\}_n$, then $\{a_i\}_m$ represents a subset of $\{a_i\}_n$ with m items, where the indices are $i \in \mathbb{N}_n$. A diagonal matrix with values $a_1, \cdots, a_m$ is denoted by $\mathrm{diag}(a_1, \cdots, a_m)$, and if all m values are the same, it is simplified to $\mathrm{diag}(a; m)$.

Consider $K(\boldsymbol{S}, \boldsymbol{S}') : \mathcal{S} \times \mathcal{S} \mapsto \mathbb{R}$ as a symmetric and positive definite sequence kernel (Cancedda et al., 2003; Király & Oberhauser, 2019). It can also be expressed as $K(\boldsymbol{S}, \boldsymbol{S}') = K_{\boldsymbol{S}}(\boldsymbol{S}') = K_{\boldsymbol{S}'}(\boldsymbol{S})$, and for convenience, $K_{\boldsymbol{S}}(\cdot)$ may be abbreviated as $K_{\boldsymbol{S}}$. The reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ corresponding to $K(\boldsymbol{S}, \boldsymbol{S}')$ is defined as the closure of the linear span $\{f :$

[2] See the notation table in Appendix A.1.

$f(\cdot) = \sum_{i=1}^{r} a_i K(\boldsymbol{S}_i, \cdot), a_i \in \mathbb{R}, r \in \mathbb{N}, \boldsymbol{S}_i \in \mathcal{S}\}$, with the inner product given by $\langle f, g\rangle_{\mathcal{H}} = \sum_{ij} a_i b_j K(\boldsymbol{S}_i, \boldsymbol{S}_j)$, where $g = \sum_j b_j K_{\boldsymbol{S}_j}$ (Liu & Wang, 2016; Zhang et al., 2023b). Instead of assuming the ideal case with a closed-form solution f^*, we focus on the more realistic scenario where the realization of f^* is provided (Zhang et al., 2023b;a; 2024; 2025). To keep notation simple, we assume the function is scalar-valued, in line with the focus on the sequence level in this discussion[3]. Given the target mapping $f^* : \mathcal{S} \mapsto \mathcal{Y}$, it uniquely outputs $\boldsymbol{y}_\dagger$ for the corresponding sequence $\boldsymbol{S}_\dagger$ such that $\boldsymbol{y}_\dagger = f^*(\boldsymbol{S}_\dagger)$. Based on the Riesz–Fréchet representation theorem (Lax, 2002; Schölkopf et al., 2002; Zhang et al., 2023b), the evaluation functional is defined as follows:

Definition 1. *Let $\mathcal{H}$ be a reproducing kernel Hilbert space with a positive definite sequence kernel $K_{\boldsymbol{S}} \in \mathcal{H}$, where $\boldsymbol{S} \in \mathcal{S}$. The evaluation functional $E_{\boldsymbol{S}}(\cdot) : \mathcal{H} \mapsto \mathbb{R}$ is defined by the reproducing property as follows:*

$$E_{\boldsymbol{S}}(f) = \langle f, K_{\boldsymbol{S}}(\cdot)\rangle_{\mathcal{H}} = f(\boldsymbol{S}), f \in \mathcal{H}. \tag{1}$$

Moreover, for a functional $F : \mathcal{H} \mapsto \mathbb{R}$, the Fréchet derivative (Coleman, 2012; Liu, 2017; Zhang et al., 2023b) of F is defined as follows:

Definition 2. *(Fréchet derivative in RKHS) The Fréchet derivative of a functional $F : \mathcal{H} \mapsto \mathbb{R}$ at $f \in \mathcal{H}$, denoted by $\nabla_f F(f)$, is implicitly defined by $F(f + \epsilon g) = F(f) + \langle \nabla_f F(f), \epsilon g\rangle_{\mathcal{H}} + o(\epsilon)$ for any $g \in \mathcal{H}$ and $\epsilon \in \mathbb{R}$. This derivative is also a function in $\mathcal{H}$.*

Recurrent neural network (RNN) is designed to learn the implicit mapping between sequences and their associated properties (Elman, 1990; Jordan, 1997). Specifically, an L-layer RNN $f_\theta(\boldsymbol{S}_{1:S}) \equiv \boldsymbol{X}^{(L,S)}$ is akin to an L-layer MLP, with the key difference being the recurrent connection. For $\ell \in \mathbb{N}_{L-1}$ and $s \in \mathbb{N}_S$,

$$\boldsymbol{X}^{(\ell,s)} = \sigma\left(\boldsymbol{X}^{(\ell-1,s)}\boldsymbol{W}^{(\ell)} + \boldsymbol{X}^{(\ell,s-1)}\boldsymbol{W}_r^{(\ell)} + \boldsymbol{b}^{(\ell)}\right), \quad \boldsymbol{X}^{(L,S)} = \boldsymbol{X}^{(L-1,S)}\boldsymbol{W}^{(L)} + \boldsymbol{b}^{(L)}, \tag{2}$$

where σ represents the activation function (*e.g.*, ReLU), $\boldsymbol{W}^{(\ell)}$ is the non-recurrent weight matrix at layer ℓ, with dimensions $h_{\ell-1} \times h_\ell$, where h_ℓ denotes the width of layer ℓ and $h_0 = d$. Additionally, $\boldsymbol{W}_r^{(\ell)}$ is the recurrent weight matrix at layer ℓ, which connects $\boldsymbol{X}^{(\ell,t-1)}$ to $\boldsymbol{X}^{(\ell,t)}$; $\boldsymbol{X}^{(0,s)} \equiv \boldsymbol{X}_{(s,:)}$ represents the s-th input feature, and $\boldsymbol{X}^{(\ell,0)} = \boldsymbol{0}$ is the initial state.

Nonparametric teaching is defined as a functional minimization over a teaching set $\mathcal{D} = \{(\boldsymbol{x}^1, y^1), \dots (\boldsymbol{x}^T, y^T)\}$, where each input $\boldsymbol{x} \in \mathbb{R}^d$ signifies independent feature data without regard to sequence order (Zhang et al., 2023b). The collection of all possible teaching sets is denoted as $\mathbb{D}$:

$$\mathcal{D}^* = \arg\min_{\mathcal{D}\in\mathbb{D}} \mathcal{M}(\hat{f}, f^*) + \lambda \cdot \mathrm{card}(\mathcal{D}) \qquad \text{s.t.} \quad \hat{f} = \mathcal{A}(\mathcal{D}). \tag{3}$$

The above formulation includes three main components: $\mathcal{M}$ which quantifies the difference between $\hat{f}$ and f^* (*e.g.*, L_2 distance in RKHS $\mathcal{M}(\hat{f}^*, f^*) = \|\hat{f}^* - f^*\|_{\mathcal{H}}$); $\mathrm{card}(\cdot)$, representing the cardinality of the teaching set $\mathcal{D}$, regularized by a constant $\lambda > 0$; and $\mathcal{A}(\mathcal{D})$, which denotes the learning algorithm used by the learners, typically based on empirical risk minimization:

$$\hat{f} = \arg\min_{f\in\mathcal{H};(\boldsymbol{x},f^*(\boldsymbol{x}))\in\mathcal{D}} \mathcal{L}(f(\boldsymbol{x}), f^*(\boldsymbol{x})) \tag{4}$$

with a convex loss $\mathcal{L}$ (w.r.t. f), which is optimized using functional gradient descent[4]:

$$f^{t+1} \leftarrow f^t - \eta \underbrace{E_{\boldsymbol{x}}\left(\frac{\partial \mathcal{L}(f^*, f^t)}{\partial f^t}\right) \cdot K_{\boldsymbol{x}}}_{:=\mathcal{G}(\mathcal{L}, f^*; f^t, \boldsymbol{x}),\ \text{Functional Gradient}}, \tag{5}$$

where $t = 0, 1, \dots, T$ is the iteration index, and $\eta > 0$ represents the learning rate.

[3] In nonparametric teaching, the extension from scalar-valued functions to vector-valued ones, which relates to element-level properties, is a well-established generalization in Zhang et al., 2023a.

[4] The functional gradient is derived using the functional chain rule (Lemma 5) and the gradient of an evaluation functional (Lemma 6), both of which are presented in Appendix A.2.

4 ReNT

We start by examining the impact of the sequence order on parameter-based gradient descent. Then, by translating the evolution of RNN—driven by order-aware updates in parameter space—into function space, we show that the evolution of RNN under parameter gradient descent aligns with that under functional gradient descent. Lastly, we introduce the greedy ReNT algorithm, which efficiently selects sequences with steeper gradients to improve the learning efficiency of RNN.

4.1 Order-aware update in the parameter space

Let the column vector $\theta \in \mathbb{R}^m$ represent the weights of all layers in a flattened form, where m is the total number of parameters in the RNN, and the bias term is omitted for simplicity. Figure 2 provides an example that illustrates the workflow of this RNN. Given a training set of size N, $\{(\boldsymbol{S}_i, \boldsymbol{y}_i) | \boldsymbol{S}_i \in \mathcal{S}, \boldsymbol{y}_i \in \mathcal{Y}\}_N$, the parameters are updated using gradient descent (Ruder, 2016) as shown below:

$$\theta^{t+1} \leftarrow \theta^t - \frac{\eta}{N} \sum_{i=1}^{N} \nabla_\theta \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i). \tag{6}$$

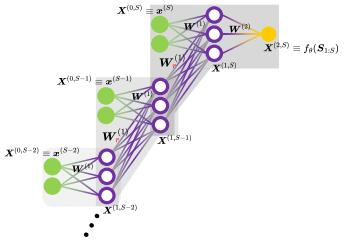


Figure 2: A workflow illustration of a three-layer RNN with a three-length input sequence.

Because the learning rate η is sufficiently small, the updates are minimal over several iterations, allowing them to be treated as a time derivative and subsequently transformed into a differential equation (Alemohammad et al., 2021; Emami et al., 2021):

$$\frac{\partial \theta^t}{\partial t} = -\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left[\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t} \right]_N. \tag{7}$$

The term $\frac{\partial f_\theta(\boldsymbol{S})}{\partial \theta}$ (with the indexes i and t omitted for simplicity), which specifies the direction for parameter updates, can be expressed more explicitly as

$$\Big[\frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(L)}}, \underbrace{\overbrace{\frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(L-1)}_{(:,1)}}, \cdots, \frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(L-1)}_{(:,h_{L-1})}}}^{\text{non-recurrent weights}}, \overbrace{\frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(L-1)}_{r(:,1)}}, \cdots, \frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(L-1)}_{r(:,h_{L-1})}}}^{\text{recurrent weights}}}_{\text{w.r.t. the } (L-1)\text{-th layer}}, \cdots, \underbrace{\overbrace{\frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(1)}_{(:,1)}}, \cdots, \frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(1)}_{(:,h_1)}}}^{\text{non-recurrent weights}}, \overbrace{\frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(1)}_{r(:,1)}}, \cdots, \frac{\partial \boldsymbol{X}^{(L,S)}}{\partial \boldsymbol{W}^{(1)}_{r(:,h_1)}}}^{\text{recurrent weights}}}_{\text{w.r.t. the first layer}} \Big].$$

Here, each term represents the derivative of output $\boldsymbol{X}^{(L,S)}$ w.r.t. weight column vectors. Unlike derivatives when treating inputs as individual features—where all derivative terms involving the recurrent weight matrix $\boldsymbol{W}_r^{(\ell)}$ are erased, and the output $f_\theta(\boldsymbol{S}_{1:S})$ depends solely on the last element of the sequence (*i.e.*, $f_\theta(\boldsymbol{S}_{1:S}) = f_\theta(\boldsymbol{S}_{S:S})$) making it independent of the preceding elements and disregarding the sequence natural or temporal dependencies—the recurrent connections dictate the aggregation of features along the sequence order, with each feature of a single element treated individually (Alemohammad et al., 2021; Emami et al., 2021). To clearly show, in an analytical and explicit way, how sequence order directs order-aware updates in the parameter space, we provide an example involving the derivative of a two-layer RNN with an input sequence of length three:

$$\frac{\partial f_\theta(\boldsymbol{S}_{1:3})}{\partial \theta} = \Big[\underbrace{\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(2)}}}_{\text{the second layer}}, \underbrace{\overbrace{\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,1)}}}^{\text{size: } 1\times h_0}, \cdots, \frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,h_1)}}}_{\text{the first layer, } h_1 \text{ terms}}, \underbrace{\overbrace{\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,1)}}}^{1\times h_1}, \cdots, \frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,h_1)}}}_{\text{the recurrent layer, } h_1 \text{ terms}} \Big], \tag{8}$$

where the term $\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(2)}}$ is

$$\sigma\left(\boldsymbol{X}^{(0,3)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right). \tag{9}$$

For $i \in \mathbb{N}_{h_1}$, the term $\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,i)}}$, which pertains to the *non-recurrent weights*, is given by

$$\underbrace{\dot{\sigma}_3 \overbrace{\boldsymbol{X}^{(0,3)}}^{\text{size: } 1\times h_0} \overbrace{\boldsymbol{W}^{(2)}_{(i,:)}}^{1\times 1}}_{\text{corresponds to } s=3} + \underbrace{\dot{\sigma}_3\dot{\sigma}_2 \overbrace{\boldsymbol{X}^{(0,2)}}^{1\times h_0} \overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{1\times h_1} \overbrace{\boldsymbol{W}^{(2)}}^{h_1\times 1}}_{\text{corresponds to } s=2} + \underbrace{\dot{\sigma}_3\dot{\sigma}_2\dot{\sigma}_1 \overbrace{\boldsymbol{X}^{(0,1)}}^{1\times h_0} \overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{1\times h_1} \overbrace{\boldsymbol{W}_r^{(1)}}^{h_1\times h_1} \overbrace{\boldsymbol{W}^{(2)}}^{h_1\times 1}}_{\text{corresponds to } s=1}, \tag{10}$$

and the term $\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}}$ which corresponds to the *recurrent weights*, is

$$\underbrace{\dot{\sigma}_3\,\sigma\left(\overbrace{\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)}+\sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}^{(1)}_r}^{\text{size: } 1\times h_1}\right)\overbrace{\boldsymbol{W}^{(2)}_{(i,:)}}^{1\times 1}}_{\text{corresponds to } s=2}+\underbrace{\dot{\sigma}_3\dot{\sigma}_2\,\sigma\left(\overbrace{\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}}^{1\times h_1}\right)\overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{1\times h_1}\overbrace{\boldsymbol{W}^{(2)}}^{h_1\times 1}}_{\text{corresponds to } s=1}, \tag{11}$$

with the scalar $\dot{\sigma}=\frac{\partial\sigma(x)}{\partial x}$, indexed by order s for specific inputs. By introducing the concatenation operation $\bigoplus$ and defining $\boldsymbol{A}^{[\kappa]}:=\bigoplus_{i=\kappa-1}^{0}\boldsymbol{A}^i=[\boldsymbol{A}^{\kappa-1}\cdots\boldsymbol{A}\,\boldsymbol{I}]$, Equation 10 and 11 can be equivalently rewritten in matrix form as follows:[5]

$$\underbrace{\boldsymbol{e}_i^\top}_{\text{size: } 1\times h_1}\underbrace{\boldsymbol{W}_r^{(1)^{[3]}}}_{h_1\times 3h_1}\underbrace{\operatorname{diag}\left(\boldsymbol{W}^{(2)};3\right)}_{3h_1\times 3}\underbrace{\left[\prod_{k=j}^{3}\dot{\sigma}_k\boldsymbol{X}^{(0,j)}\right]_3}_{3\times h_0},\quad \underbrace{\boldsymbol{e}_i^\top}_{\text{size: } 1\times h_1}\underbrace{\boldsymbol{W}_r^{(1)^{[3]}}}_{h_1\times 3h_1}\underbrace{\operatorname{diag}\left(\boldsymbol{W}^{(2)};3\right)}_{3h_1\times 3}\underbrace{\left[\prod_{k=j}^{3}\dot{\sigma}_k\boldsymbol{X}^{(1,j-1)}\right]_3}_{3\times h_0}. \tag{12}$$

The derivation is provided in Appendix A.3. When the sequence length S is reduced to 1, meaning the input is a single element, the RNN gradient for that input matches exactly with the MLP gradient for the same feature. This suggests that the order-aware, parameter-based gradient is more general than the MLP gradient, indicating that this work generalizes Zhang et al., 2024 by incorporating sequence order. Moreover, the explicit expressions in Equations 10 and 11 show that the sequence order determines the power of $\boldsymbol{W}_r^{(1)}$ in the gradients, with temporal dependencies in the input sequence accounted for by the different powers of the recurrent weights. In matrix form, as shown in Equation 12, it is evident that the RNN gradient shape does not depend on the input sequence length (*i.e.*, the number of elements), but rather on the feature dimension. In other words, the parameter gradient maintains the same form even if the input sequence length S is scaled.

4.2 The Functional Evolution of RNN

The order-aware update in the parameter space guides the functional evolution of $f_\theta\in\mathcal{H}$. The resulting variation in f_θ, reflecting how f_θ evolves in response to changes in θ, can be derived using Taylor's theorem as follows:

$$f(\theta^{t+1})-f(\theta^t)=\langle\nabla_\theta f(\theta^t),\theta^{t+1}-\theta^t\rangle+o(\theta^{t+1}-\theta^t), \tag{13}$$

where $f(\theta^\square)\equiv f_{\theta^\square}$. Similar to the transformation of parameter updates, it can be expressed in differential form (Zhang et al., 2024):

$$\frac{\partial f_{\theta^t}}{\partial t}=\underbrace{\left\langle\frac{\partial f(\theta^t)}{\partial\theta^t},\frac{\partial\theta^t}{\partial t}\right\rangle}_{(*)}+o\left(\frac{\partial\theta^t}{\partial t}\right). \tag{14}$$

By substituting the specific parameter updates, *i.e.*, Equation 7, into the first-order approximation term $(*)$ of this variation, we obtain

$$\frac{\partial f_{\theta^t}}{\partial t}=-\frac{\eta}{N}\left[\frac{\partial\mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i),\boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top\cdot\left[K_{\theta^t}(\boldsymbol{S}_i,\cdot)\right]_N+o\left(\frac{\partial\theta^t}{\partial t}\right), \tag{15}$$

where the symmetric and positive definite $K_{\theta^t}(\boldsymbol{S}_i,\cdot):=\left\langle\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial\theta^t},\frac{\partial f_{\theta^t}(\cdot)}{\partial\theta^t}\right\rangle$ (refer to the detailed derivation in Appendix A.4). The inclusion of nonlinear activation functions in $f(\theta)$ introduces nonlinearity with respect to θ, making the remainder $o(\theta^{t+1}-\theta^t)$ nonzero. In contrast, Jacot et al., 2018; Alemohammad et al., 2021; Emami et al., 2021 apply the chain rule directly, giving less focus on the convexity of $\mathcal{L}$ with respect to θ. As a result, the first-order approximation is derived as the variation, with K_θ referred to as the recurrent neural tangent kernel (RNTK). It has been proved that the RNTK remains constant during training under the assumption of an infinite RNN width (Alemohammad et al., 2021; Emami et al., 2021). However, in practical scenarios, there is no need for the RNN width to be infinite, prompting us to explore the dynamic RNTK (An example of how the RNTK is computed is shown in Figure 6 in Appendix A.4).

[5]The gradient vanishing/exploding issue (Bengio et al., 1994) is apparent in $\boldsymbol{W}_r^{(1)^{[3]}}$, but it is not the focus of this work.

Consider approaching the variation of $f_\theta \in \mathcal{H}$ from a high-level, functional viewpoint (Zhang et al., 2024). Through functional gradient descent, it can be written as:

$$\frac{\partial f_{\theta^t}}{\partial t} = -\eta \mathcal{G}(\mathcal{L}, f^*; f_{\theta^t}, \{\boldsymbol{S}_i\}_N), \tag{16}$$

where the functional gradient is expressed as:

$$\mathcal{G}(\mathcal{L}, f^*; f_{\theta^t}, \{\boldsymbol{S}_i\}_N) = \frac{1}{N}\left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top \cdot [K(\boldsymbol{S}_i, \cdot)]_N . \tag{17}$$

The asymptotic connection between the RNTK and the order-aware canonical kernel (Cancedda et al., 2003; Király & Oberhauser, 2019; Zhang et al., 2024) in the context of functional gradient is presented in Theorem 3 below, with the proof provided in Appendix B.2.

Theorem 3. *Given a convex loss $\mathcal{L}$ and a training set $\{(\boldsymbol{S}_i, \boldsymbol{y}_i) | \boldsymbol{S}_i \in \mathcal{S}, \boldsymbol{y}_i \in \mathcal{Y}\}_N$, the dynamic RNTK, which results from gradient descent on the parameters of an RNN, converges pointwise to the order-aware canonical kernel in the dual functional gradient with respect to the input sequences. Specifically, the following holds:*

$$\lim_{t\to\infty} K_{\theta^t}(\boldsymbol{S}_i, \cdot) = K(\boldsymbol{S}_i, \cdot), \forall i \in \mathbb{N}_N. \tag{18}$$

This indicates that RNTK, which includes sequence order information, acts as a dynamic alternative to the order-aware canonical kernel in functional gradient descent with sequence inputs, causing the RNN evolution through parameter gradient descent to align with the evolution in functional gradient descent (Kuk, 1995; Alemohammad et al., 2021; Geifman et al., 2020). This functional insight bridges the gap between the teaching of the sequential property learner, RNN, and that of order-aware nonparametric learners, while also making further analysis simpler (*e.g.*, a convex functional $\mathcal{L}$ retains its convexity with respect to f_θ from a functional viewpoint, but is generally nonconvex when considering θ). By utilizing the functional insight and adopting the canonical kernel (Dou & Liang, 2021) instead of RNTK (which should be considered in conjunction with the remainder), it helps in deriving the sufficient reduction regarding $\mathcal{L}$ in Proposition 4, with the proof deferred to Appendix B.3.

Proposition 4. *(Sufficient Loss Reduction) Let the convex loss $\mathcal{L}$ be Lipschitz smooth with a constant $\tau > 0$, and the order-aware canonical kernel be bounded above by $\gamma > 0$. If the learning rate η satisfies $\eta \leq 1/(2\tau\gamma)$, then a sufficient reduction in $\mathcal{L}$ is guaranteed, as shown by*

$$\frac{\partial \mathcal{L}}{\partial t} \leq -\frac{\eta\gamma}{2}\left(\frac{1}{N}\sum_{i=1}^{N}\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right)^2 . \tag{19}$$

This shows that the variation of $\mathcal{L}$ over time is bounded above by a negative value, implying it decreases by at least the magnitude of this bound as time goes on, thereby guaranteeing convergence.

4.3 ReNT algorithm

Building on the insights into how sequence order influences parameter-based gradient descent and the consistency between teaching an RNN and a nonparametric learner, we propose the ReNT algorithm. This algorithm aims to amplify the steepness of gradients in order to enhance the learning efficiency of the RNN. By treating the gradient as the sum of projections of $\frac{\partial \mathcal{L}(f_\theta, f^*)}{\partial f_\theta}$ onto the basis $\{K(\boldsymbol{S}_i, \cdot)\}_N$, the gradient norm can be increased simply by maximizing the projection coefficient $\frac{\partial \mathcal{L}(f_\theta(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_\theta(\boldsymbol{S}_i)}$, eliminating the need to compute the norm of the basis $\|K(\boldsymbol{S}_i, \cdot)\|_\mathcal{H}$ (Wright, 2015; Zhang et al., 2024). This indicates that selecting sequences that either maximize $\left|\frac{\partial \mathcal{L}(f_\theta(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_\theta(\boldsymbol{S}_i)}\right|$ or correspond to the larger components of $\frac{\partial \mathcal{L}(f_\theta, f^*)}{\partial f_\theta}$ can effectively amplify the gradient, implying that

$$\{\boldsymbol{S}_i\}_m{}^* = \underset{\{\boldsymbol{S}_i\}_m \subseteq \{\boldsymbol{S}_i\}_N}{\arg\max} \left\| \left[\frac{\partial \mathcal{L}(f_\theta(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_\theta(\boldsymbol{S}_i)}\right]_m \right\|_2 . \tag{20}$$

From a functional perspective, for a convex loss functional $\mathcal{L}$, the norm of the partial derivative of $\mathcal{L}$ with respect to f_θ, denoted as $\|\frac{\partial \mathcal{L}(f_\theta)}{\partial f_\theta}\|_\mathcal{H}$, is positively correlated with $\|f_\theta - f^*\|_\mathcal{H}$. As f_θ

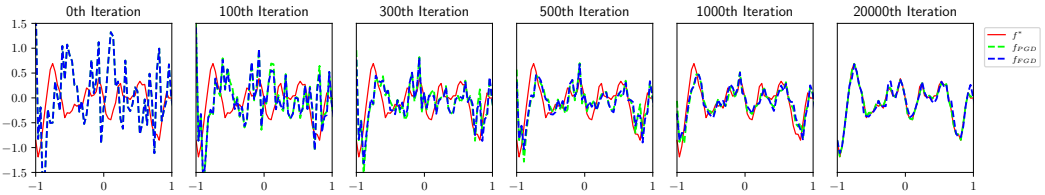


Figure 3: Training dynamics of f with PGD and FGD, where f_{PGD} closely tracks f_{FGD}, demonstrating empirical consistency in their evolution.

gradually approaches f^*, the value of $\|\frac{\partial \mathcal{L}(f_\theta)}{\partial f_\theta}\|_{\mathcal{H}}$ decreases (Boyd et al., 2004; Coleman, 2012). This relationship becomes particularly significant when $\mathcal{L}$ is strongly convex with a larger convexity constant (Kakade & Tewari, 2008; Arjevani et al., 2016). Building on these insights, the ReNT algorithm selects sequences by

$$\{\boldsymbol{S}_i\}_m{}^* = \underset{\{\boldsymbol{S}_i\}_m \subseteq \{\boldsymbol{S}_i\}_N}{\arg\max} \|[f_\theta(\boldsymbol{S}_i) - f^*(\boldsymbol{S}_i)]_m\|_2 \,. \tag{21}$$

The pseudocode, along with the element-level version, is presented in Algorithm 1.

5 Experiments and Results

We begin by using a synthetic sequence to empirically show the evolution consistency between parameter-based gradient descent (PGD) and functional gradient descent (FGD). Next, we evaluate ReNT on sequence-level tasks, then proceed to validate it on element-level tasks. The overall results on the test set are shown in Table 1, which clearly highlights the effectiveness of ReNT in sequential property learning: it reduces training time by -32.77% to -46.39% for sequence-level tasks and -36% to -39.17% for element-level tasks, all while maintaining comparable testing performance. Detailed settings and additional discussion are given in Appendix C.

Synthetic 1D sequence. For an intuitive visualization, we utilize a synthetic 1D sequence and present the training dynamics of f obtained through both PGD and FGD. Specifically, this sequence is generated using AR(2) (*i.e.*, the target mapping) as $f^*(\{x_{t-2}, x_{t-1}\}) \equiv x_t = 1.4 * x_{t-1} - 0.6 * x_{t-2} + \epsilon_t$, $t \in \mathbb{N}$, with initial $x_{-2} = 1.2$ and $x_{-1} = -0.1$ where ϵ_t is a white noise process with zero mean and constant variance 0.4^2. The function corresponding to PGD is obtained by inputting the Fourier Feature embedding (Tancik et al., 2020) of $\{x_{t-2}, x_{t-1}\}$ into the RNN trained using PGD, while the function corresponding to FGD is represented by dense points of the nonparametric function updated using FGD. As depicted in Figure 3, f^* is well fitted by both PGD and FGD. Moreover, the function obtained through PGD closely mirrors the one obtained through FGD. This observation indicates the consistency in the evolution of the function through both PGD and FGD, suggesting that teaching an RNN aligns with teaching a nonparametric target function.

ReNT		Dataset	Time (s)	Loss ↓	MAE ↓	ACC ↑
✗		Electricity	2394.97	0.0060±0.0011	0.0539±0.0042	-
		Weather	1004.08	0.0035±0.0002	0.0278±0.0001	-
		Yelp	22726.54	0.9547±0.0186	-	0.6033±0.0027
		AG News	447.11	0.4656±0.0102	-	0.8782±0.0015
		UD	519.27	0.5418±0.0453	-	0.8643±0.0028
		CoNLL	394.40	0.4302±0.0172	-	0.8934±0.0035
✓	B	Electricity	1283.98 (-46.39%)	0.0049±0.0004	**0.0515**±0.0017	-
		Weather	675.05 (-32.77%)	0.0034±0.0001	0.0277±0.0006	-
		Yelp	14455.28 (-36.40%)	0.9323±0.0039	-	**0.6043**±0.0026
		AG News	287.73 (-35.68%)	0.4095±0.0135	-	**0.8783**±0.0013
		UD	316.03 (-39.17%)	0.4637±0.0279	-	**0.8716**±0.0042
		CoNLL	252.35 (-36.00%)	0.3369±0.0245	-	0.8987±0.0071
	S	Electricity	1330.68 (-44.44%)	0.0050±0.0003	0.0517±0.0008	-
		Weather	696.70 (-30.63%)	0.0034±0.0001	**0.0270**±0.0006	-
		Yelp	15071.09(-33.68%)	0.9344±0.0096	-	0.6027±0.0035
		AG News	295.15 (-33.97%)	0.4062±0.0133	-	0.8775±0.0020
		UD	335.32 (-35.45%)	0.4744±0.0195	-	0.8628±0.0019
		CoNLL	261.89 (-33.61%)	0.3293±0.0136	-	**0.9040**±0.0046

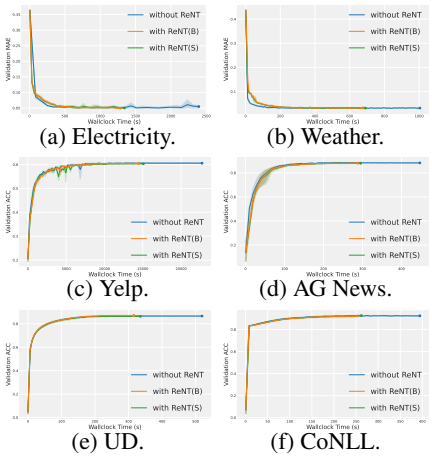
Table 1: Training time and testing results across different benchmarks. ReNT (B) and ReNT (S) demonstrate similar testing performance while significantly reducing training time compared to the "without ReNT", across sequence-level (Electricity, Weather, Yelp, AG News) and element-level (UD, CoNLL) tasks.

Given the common practice of training RNN in batches, *i.e.*, sequences are fed in batches, it is both natural and intuitive to implement ReNT at the batch level. This involves selecting batches that exhibit the largest average discrepancy between the actual properties and the corresponding RNN outputs, referred to as ReNT (B). Meanwhile, another variant, called ReNT (S), selects single sequence with the largest discrepancies within each batch in proportion, then reorganizes the selected sequences into new batches.

Sequence-level tasks. We evaluate ReNT on time series forecasting (*i.e.*, sequence-level regression) using the UCI Electricity Load Diagrams (Trindade, 2015) and Weather (for Biogeochemistry,

2020) datasets, and on text classification (*i.e.*, sequence-level classification) with the Yelp Review Full (Zhang et al., 2015) and AG News (Zhang et al., 2015) datasets. To clearly showcase the practical training efficiency of ReNT, we plot wall-clock time against evaluation metrics (MAE or ACC) for each dataset. Validation is conducted at the end of each training epoch, meaning the model is evaluated on the validation set after every training cycle.

Figures 4 (a) and (b) display the validation Mean Absolute Error (MAE) curves for the two regression tasks. As seen, the curve widths for ReNT (B) and ReNT (S) are approximately halved and reduced by about one-third, respectively, demonstrating significant reductions in total training time. Along with the results in Table 1, it is clear that both ReNT (B) and ReNT (S) outperform the "without ReNT" in terms of training loss and validation MAE, while also considerably shortening training time.



(a) Electricity. (b) Weather. (c) Yelp. (d) AG News. (e) UD. (f) CoNLL.

Figure 4: Performance on the validation set across various benchmarks.

Figures 4 (c) and (d) show the validation accuracy curves for two text classification datasets of different sizes: the large-scale Yelp Review Full and the smaller AG News. As indicated by the curves, ReNT consistently reduces total training time by more than a third, while achieving similar validation accuracy on both datasets. Table 1 further underscores the time-saving effect of ReNT, where ReNT (B) reduces training time by over 2 hours on the large-scale Yelp Review Full dataset (Zhang et al., 2015). In both datasets, ReNT (B) slightly outperforms the "without ReNT" in terms of test metrics, achieving a lower test loss. Overall, ReNT provides substantial training time savings in sequence-level classification tasks without compromising performance.

Element-level tasks. We also assess ReNT on element-level classification tasks, using two widely recognized benchmarks: Part-of-Speech tagging with the Universal Dependencies dataset (UD) (Nivre et al., 2016), and Named Entity Recognition with the CoNLL-2003 dataset (CoNLL) (Tjong Kim Sang & De Meulder, 2003).

As seen in Figures 4 (e) and (f), both ReNT (B) and ReNT (S) achieve validation performance on par with the "without ReNT" setting for element-level classification tasks, within the same wall-clock time. Furthermore, Table 1 indicates that ReNT (B) cuts total training time by over a third, while consistently maintaining test performance comparable to the "without ReNT" setting across both datasets. These results underscore the substantial improvements in training efficiency offered by ReNT in element-level tasks, with generalization ability intact.

All experimental results across a range of sequential property learning tasks show that, despite the overhead introduced by the sampling process, ReNT (B) and ReNT (S) consistently provide significant training time reductions while achieving similar generalization performance, and in some cases, even outperform the "without ReNT".

6 Concluding Remarks and Future Work

This paper introduces ReNT, a novel paradigm that improves the learning efficiency of sequential property learners (RNNs) through nonparametric teaching theory. Specifically, ReNT cuts the wallclock time required to learn the implicit mapping from sequences to properties of interest by 32.77% to 46.39%, while preserving comparable test performance, as demonstrated by extensive experiments. Moreover, ReNT establishes a theoretical link between the evolution of an RNN through parameter-based gradient descent and the evolution of a function using functional gradient descent in nonparametric teaching. This connection between nonparametric teaching theory and RNN training expands the potential applications of nonparametric teaching in sequential property learning.

In future work, it would be intriguing to investigate the practical applications of ReNT to enhance the efficiency of data-driven approaches (Touvron et al., 2021; AlQuraishi & Sorger, 2021) in sequential property learning, especially in areas like DNA sequence analysis.

Reproducibility Statement

We have made significant efforts to ensure the reproducibility of our work. The detailed notation, theoretical background, and algorithm are provided in Appendix A. All proofs of lemmas, theorems, and propositions are included in Appendix B. Appendix C contains the full experimental setup, including dataset-specific preprocessing procedures, implementation details of the algorithms, and additional discussions.

Statement on the Use of Large Language Models

We used a large language model for language polishing on the manuscript (*e.g.*, grammar and wording). The research ideas, methods, experiments, analyses, figures/tables, and conclusions were conceived and produced by the authors. The authors take full responsibility for all content.

References

Samia M Abd-Alhalem, El-Sayed M El-Rabaie, Naglaa Soliman, Salah Eldin SE Abdulrahman, Nabil A Ismail, Fathi E Abd El-samie, et al. Dna sequences classification with deep learning: a survey. *Menoufia Journal of Electronic Engineering Research*, 30(1):41–51, 2021. 1, 3

Sina Alemohammad, Zichao Wang, Randall Balestriero, and Richard Baraniuk. The recurrent neural tangent kernel. In *ICLR*, 2021. 2, 5, 6, 7, 20

Scott Alfeld, Xiaojin Zhu, and Paul Barford. Explicit defense actions against test-set attacks. In *AAAI*, 2017. 3

Ethan C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, 2019. 1, 3

Mohammed AlQuraishi and Peter K Sorger. Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms. *Nature methods*, 18(10): 1169–1180, 2021. 1, 3, 9

Yossi Arjevani, Shai Shalev-Shwartz, and Ohad Shamir. On lower and upper bounds in smooth and strongly convex optimization. *The Journal of Machine Learning Research*, 17(1):4303–4353, 2016. 8

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. 6

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009. 28

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 8

Nicola Cancedda, Eric Gaussier, Cyril Goutte, and Jean-Michel Renders. Word-sequence kernels. *Journal of machine learning research*, 3(Feb):1059–1082, 2003. 3, 7

Xie Chen, Xunying Liu, Mark JF Gales, and Philip C Woodland. Improving the training and evaluation efficiency of recurrent neural network language models. In *ICASSP*, 2015. 3

Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014a. 1

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111, 2014b. 3, 28

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996. 29

Rodney Coleman. *Calculus on normed vector spaces*. Springer Science & Business Media, 2012. 4, 8, 16

Xialiang Dou and Tengyuan Liang. Training neural networks as learning data-adaptive kernels: Provable representation and approximation benefits. *Journal of the American Statistical Association*, 116(535):1507–1520, 2021. 7

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990. 1, 3, 4

Melikasadat Emami, Mojtaba Sahraee-Ardakan, Parthe Pandit, Sundeep Rangan, and Alyson K Fletcher. Implicit bias of linear rnns. In *ICML*, 2021. 2, 5, 6, 20

Leo Feng, Frederick Tung, Mohamed Osama Ahmed, Yoshua Bengio, and Hossein Hajimirsadeghi. Were rnns all we needed? *arXiv preprint arXiv:2410.01201*, 2024. 28

Max Planck Institute for Biogeochemistry. Weather dataset. https://www.bgc-jena.mpg.de/wetter/, 2020. Accessed: 2025-05-14, License: CC-BY-4.0. 8, 26

Amnon Geifman, Abhay Yadav, Yoni Kasten, Meirav Galun, David Jacobs, and Basri Ronen. On the similarity between the laplace and neural tangent kernels. In *NeurIPS*, 2020. 7

Izrail Moiseevitch Gelfand and Richard A Silverman. *Calculus of variations*. Courier Corporation, 2000. 16

Andi Hermanto, Teguh Bharata Adji, and Noor Akhmad Setiawan. Recurrent neural network language model for english-indonesian machine translation: Experimental study. In *2015 International conference on science in information technology (ICSITech)*, pp. 132–136. IEEE, 2015. 1

Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37 (1):388–427, 2021. 1, 3

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. 3, 28

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *NeurIPS*, 2018. 6

Michael I Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pp. 471–495. Elsevier, 1997. 1, 3, 4

Sham M Kakade and Ambuj Tewari. On the generalization ability of online strongly convex programming algorithms. In *NeurIPS*, 2008. 8

Franz J Király and Harald Oberhauser. Kernels for sequentially ordered data. *Journal of Machine Learning Research*, 20(31):1–45, 2019. 1, 3, 7

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown. Text classification algorithms: A survey. *Information*, 10(4):150, 2019. 1, 3

Anthony YC Kuk. Asymptotically unbiased estimation in generalized linear models with random effects. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 57(2):395–407, 1995. 7

Peter D Lax. *Functional analysis*, volume 55. John Wiley & Sons, 2002. 4

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015. 3

Chenyang Liu, Rui Zhao, Hao Chen, Zhengxia Zou, and Zhenwei Shi. Remote sensing image change captioning with dual-branch transformers: A new method and a large scale dataset. *IEEE Transactions on Geoscience and Remote Sensing*, 60:1–20, 2022. 1

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pp. 2873–2879, 2016. 1, 3

Qian Liu, Li Fang, Guoliang Yu, Depeng Wang, Chuan-Le Xiao, and Kai Wang. Detection of dna base modifications by deep recurrent neural network on oxford nanopore sequencing data. *Nature communications*, 10(1):2449, 2019. 1, 3

Qiang Liu. Stein variational gradient descent as gradient flow. In *NeurIPS*, 2017. 4

Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *NeurIPS*, 2016. 4

Shiwei Liu, Iftitahu Ni'mah, Vlado Menkovski, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Efficient and effective training of sparse recurrent neural networks. *Neural Computing and Applications*, 33:9625–9636, 2021. 3

Shujie Liu, Nan Yang, Mu Li, and Ming Zhou. A recursive recurrent neural network for statistical machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1491–1500, 2014. 1

Weiyang Liu, Bo Dai, Ahmad Humayun, Charlene Tay, Chen Yu, Linda B Smith, James M Rehg, and Le Song. Iterative machine teaching. In *ICML*, 2017. 2, 3

Weiyang Liu, Bo Dai, Xingguo Li, Zhen Liu, James Rehg, and Le Song. Towards black-box iterative machine teaching. In *ICML*, 2018. 2, 3

Minrong Lu and Xuerong Xu. Trnn: An efficient time-series recurrent neural network for stock price prediction. *Information Sciences*, 657:119951, 2024. 1, 3

Ziyuan Luo, Boxin Shi, Haoliang Li, and Renjie Wan. Imaging interiors: An implicit solution to electromagnetic inverse scattering problems. In *ECCV*, 2024. 3

Yuzhe Ma, Xuezhou Zhang, Wen Sun, and Jerry Zhu. Policy poisoning in batch reinforcement learning and control. In *NeurIPS*, 2019. 3

Jamal Abdul Nasir, Osama Subhani Khan, and Iraklis Varlamis. Fake news detection: A hybrid cnn-rnn based deep learning approach. *International journal of information management data insights*, 1(1):100007, 2021. 1

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. Universal dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 1659–1666, 2016. 9, 26

Erik Otovic, Marko Njirjak, Daniela Kalafatovic, and Goran Mausa. Sequential properties representation scheme for recurrent neural network-based prediction of therapeutic peptides. *Journal of chemical information and modeling*, 62(12):2961–2972, 2022. 1, 3

Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Mathieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 26

Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2): 206–219, 2019. 1, 3

Amin Rakhsha, Goran Radanovic, Rati Devidze, Xiaojin Zhu, and Adish Singla. Policy teaching via environment poisoning: Training-time adversarial attacks against reinforcement learning. In *ICML*, 2020. 3

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 5

Luis G Baca Ruíz, Manuel I Capel, and MC Pegalajar. Parallel memetic algorithm for training recurrent neural networks for the energy efficiency problem. *Applied Soft Computing*, 76:356–368, 2019. 3

Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. In *Proc. Interspeech 2014*, pp. 338–342, 2014. 1

Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078*, 2017. 3

Fatima K Abu Salem, Roaa Al Feel, Shady Elbassuoni, Mohamad Jaber, and May Farah. Fa-kes: A fake news dataset around the syrian war. In *Proceedings of the international AAAI conference on web and social media*, volume 13, pp. 573–582, 2019. 1

Bernhard Schölkopf, Alexander J Smola, and Francis Bach. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2002. 4

Adish Singla, Ilija Bogunovic, Gábor Bartók, Amin Karbasi, and Andreas Krause. Near-optimally teaching the crowd to classify. In *ICML*, 2014. 3

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. In *NeurIPS*, 2020. 3

Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. In *ICLR*, 2023. 3

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. 1, 3

Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020. 3, 8

Erik F Tjong Kim Sang and Fien De Meulder. Introduction to the conll-2003 shared task: language-independent named entity recognition. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pp. 142–147, 2003. 9, 26

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 9

Artur Trindade. ElectricityLoadDiagrams20112014. UCI Machine Learning Repository, 2015. DOI: https://doi.org/10.24432/C58C86. 8, 26

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. In *ICLR*, 2016. 2, 3

Pei Wang and Nuno Vasconcelos. A machine teaching framework for scalable recognition. In *ICCV*, 2021. 3

Pei Wang, Kabir Nagrecha, and Nuno Vasconcelos. Gradient-based algorithms for machine teaching. In *CVPR*, 2021. 3

Stephen J Wright. Coordinate descent algorithms. *Mathematical programming*, 151(1):3–34, 2015. 7

Satya Prakash Yadav, Subiya Zaidi, Annu Mishra, and Vibhash Yadav. Survey on machine learning in speech emotion recognition and vision systems using a recurrent neural network (rnn). *Archives of Computational Methods in Engineering*, 29(3):1753–1770, 2022. 1

Jinmian Ye, Linnan Wang, Guangxi Li, Di Chen, Shandian Zhe, Xinqi Chu, and Zenglin Xu. Learning compact recurrent neural networks with block-term tensor decomposition. In *CVPR*, 2018. 3

Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7):1235–1270, 2019. 3

Chen Zhang, Xiaofeng Cao, Weiyang Liu, Ivor Tsang, and James Kwok. Nonparametric teaching for multiple learners. In *NeurIPS*, 2023a. 2, 3, 4, 16

Chen Zhang, Xiaofeng Cao, Weiyang Liu, Ivor Tsang, and James Kwok. Nonparametric iterative machine teaching. In *ICML*, 2023b. 2, 3, 4, 16

Chen Zhang, Steven Tin Sui Luo, Jason Chun Lok Li, Yik-Chung Wu, and Ngai Wong. Nonparametric teaching of implicit neural representations. In *ICML*, 2024. 2, 3, 4, 6, 7, 27

Chen Zhang, Weixin Bu, Zeyi Ren, Zhengwu Liu, Yik-Chung Wu, and Ngai Wong. Nonparametric teaching for graph property learners. In *ICML*, 2025. 2, 3, 4

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015. 9, 26

Yao Zhou, Arun Reddy Nelakurthi, and Jingrui He. Unlearn what you have learned: Adaptive crowd teaching with exponentially decayed memory learners. In *SIGKDD*, 2018. 3

Xiaojin Zhu. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 2015. 2, 3

Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018. 2, 3

Appendix

A Additional Discussions

A.1 Notation Overview

Notation	Description
$\boldsymbol{S}_{1:S}$	Sequence of length S consisting of ordered elements $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_S$
$[x_{s,j}]_d^\top$	d-dimensional feature vector corresponding to the s-th element with components $x_{s,j}$
$\boldsymbol{x}$	Abbreviated notation for $[x_j]_d$
$\boldsymbol{X}_{S\times d}$	Collection of all feature vectors, an $S \times d$ feature matrix
$\boldsymbol{X}_{(s,:)}$	The s-th row of $\boldsymbol{X}$ (the feature vector for the s-th element)
$\boldsymbol{X}_{(:,i)}$	The i-th column of $\boldsymbol{X}$ (The i-th feature across all elements)
$\boldsymbol{e}_i$	The i-th basis vector (1 at the i-th position, 0 elsewhere)
$\mathcal{S}$	Collection of all sequences
$\boldsymbol{y}$	Property of the sequences (scalar or vector)
$\mathcal{Y}$	Space of sequential properties ($\mathbb{R}$ or $\mathbb{R}^n$)
$\{a_i\}_m$	A set containing m items
$\mathrm{diag}(a_1, \ldots, a_m)$	Diagonal matrix with entries $a_1, \ldots, a_m$
$\mathrm{diag}(a; m)$	Diagonal matrix with m repeated values a
$\mathbb{N}_S \coloneqq \{1, \cdots, S\}$	Set of natural numbers from 1 to S
$K(\boldsymbol{S}, \boldsymbol{S}')$	A symmetric and positive definite sequence kernel
$\mathcal{H}$	Reproducing kernel Hilbert space (RKHS) defined by K
f^*	Target mapping from $\mathcal{S}$ to $\mathcal{Y}$
$\boldsymbol{y}_\dagger$	Property $f^*(\boldsymbol{S}_\dagger)$ of sequence $\boldsymbol{S}_\dagger$

Table 2: Summary of Key Notations.

A.2 Functional Gradient

Zhang et al., 2023b;a introduce the chain rule for functional gradients (Gelfand & Silverman, 2000) (see Lemma 5) and use the Fréchet derivative to compute the derivative of the evaluation functional in RKHS (Coleman, 2012) (cf. Lemma 6).

Lemma 5. *(Chain rule for functional gradients) For differentiable functions $G(F) : \mathbb{R} \mapsto \mathbb{R}$ that depend on functionals $F(f) : \mathcal{H} \mapsto \mathbb{R}$, the expression*

$$\nabla_f G(F(f)) = \frac{\partial G(F(f))}{\partial F(f)} \cdot \nabla_f F(f) \tag{22}$$

is typically referred to as the chain rule.

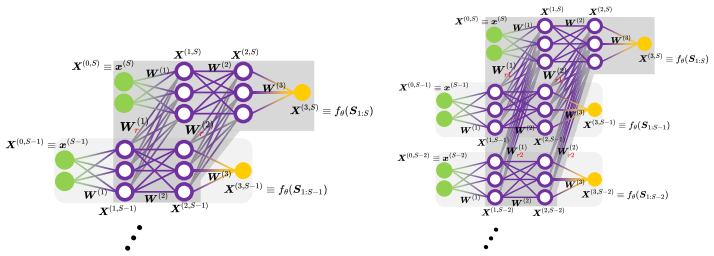
Lemma 6. *The gradient of the evaluation functional at the feature $\boldsymbol{x}$, defined as $E_{\boldsymbol{x}}(f) = f(\boldsymbol{x}) : \mathcal{H} \to \mathbb{R}$, is given by $\nabla_f E_{\boldsymbol{x}}(f) = K(\boldsymbol{x}, \cdot)$, where $K(\boldsymbol{x}, \boldsymbol{x}') : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ represents a feature-based kernel.*

A.3 The derivation of order-aware updates in the parameter space.

Before presenting the detailed derivation, we first provide visualizations of general RNNs: Figure 5a illustrates a three-layer RNN, while Figure 5b shows a three-layer RNN with a lag-two recurrent connection, the formulation of which is given in Equation 23.

$$\begin{aligned}
\boldsymbol{X}^{(\ell,s)} &= \sigma\left(\boldsymbol{X}^{(\ell-1,s)}\boldsymbol{W}^{(\ell)} + \boldsymbol{X}^{(\ell,s-1)}\boldsymbol{W}_{r1}^{(\ell)} + \boldsymbol{X}^{(\ell,s-2)}\boldsymbol{W}_{r2}^{(\ell)} + \boldsymbol{b}^{(\ell)}\right) \\
\boldsymbol{X}^{(L,S)} &= \boldsymbol{X}^{(L-1,S)}\boldsymbol{W}^{(L)} + \boldsymbol{b}^{(L)}
\end{aligned} \tag{23}$$

The gradient for these extensions can be derived using the same approach as the one demonstrated below, though with more complex, yet straightforward, notations.



(a) A three-layer RNN.

(b) A three-layer RNN with a lag-two recurrent connection.

Figure 5: Visualizations of general RNNs.

Consider the derivative of a two-layer RNN with an input sequence of length three:

$$\frac{\partial f_\theta(\boldsymbol{S}_{1:3})}{\partial \theta} = \left[\underbrace{\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(2)}}}_{\text{the second layer}}, \underbrace{\overbrace{\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,1)}}}^{\text{row vector of size } h_0}, \cdots, \frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,h_1)}}}_{\text{the first layer, } h_1 \text{ terms}}, \underbrace{\overbrace{\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,1)}}}^{\text{row vector of size } h_1}, \cdots, \frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,h_1)}}}_{\text{the recurrent layer, } h_1 \text{ terms}} \right] \tag{24}$$

By applying the chain rule, we can compute the derivative of $f_\theta(\boldsymbol{S}_{1:3})$ with respect to the second-layer weights $\boldsymbol{W}^{(2)}$, a vector of size h_1, as follows:

$$\begin{aligned} \frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(2)}} &= \frac{\partial \boldsymbol{X}^{(1,3)} \cdot \boldsymbol{W}^{(2)}}{\partial \boldsymbol{W}^{(2)}} = \boldsymbol{X}^{(1,3)} \\ &= \sigma\left(\boldsymbol{X}^{(0,3)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)} + \boldsymbol{0}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right) \end{aligned} \tag{25}$$

The derivative of $f_\theta(\boldsymbol{S}_{1:3})$ with respect to the first-layer weights, which are non-recurrent, is more intricate. For $i \in \mathbb{N}_{h_1}$,

$$\begin{aligned} &\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,i)}} \\ =\ & \frac{\partial\, \sigma\left(\boldsymbol{X}^{(0,3)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)} + \boldsymbol{0}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}^{(2)}}{\partial \boldsymbol{W}^{(1)}_{(:,i)}} \\ =\ & \frac{\partial\left(\boldsymbol{X}^{(0,3)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)}{\partial \boldsymbol{W}^{(1)}_{(:,i)}} \dot\sigma_3 \boldsymbol{W}^{(2)} \\ =\ & \left(\boldsymbol{X}^{(0,3)}\boldsymbol{e}_i^\top + \frac{\partial\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)}{\partial \boldsymbol{W}^{(1)}_{(:,i)}} \dot\sigma_2 \boldsymbol{W}_r^{(1)}\right)\dot\sigma_3 \boldsymbol{W}^{(2)} \\ =\ & \left(\boldsymbol{X}^{(0,3)}\boldsymbol{e}_i^\top + \left(\boldsymbol{X}^{(0,2)}\boldsymbol{e}_i^\top + \frac{\partial\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)}{\partial \boldsymbol{W}^{(1)}_{(:,i)}}\dot\sigma_1 \boldsymbol{W}_r^{(1)}\right)\dot\sigma_2 \boldsymbol{W}_r^{(1)}\right)\dot\sigma_3 \boldsymbol{W}^{(2)} \\ =\ & \left(\boldsymbol{X}^{(0,3)}\boldsymbol{e}_i^\top + \left(\boldsymbol{X}^{(0,2)}\boldsymbol{e}_i^\top + \boldsymbol{X}^{(0,1)}\boldsymbol{e}_i^\top \dot\sigma_1 \boldsymbol{W}_r^{(1)}\right)\dot\sigma_2 \boldsymbol{W}_r^{(1)}\right)\dot\sigma_3 \boldsymbol{W}^{(2)} \\ =\ & \left(\boldsymbol{X}^{(0,3)}\boldsymbol{e}_i^\top + \boldsymbol{X}^{(0,2)}\boldsymbol{e}_i^\top \dot\sigma_2 \boldsymbol{W}_r^{(1)} + \boldsymbol{X}^{(0,1)}\boldsymbol{e}_i^\top \dot\sigma_1 \boldsymbol{W}_r^{(1)} \dot\sigma_2 \boldsymbol{W}_r^{(1)}\right)\dot\sigma_3 \boldsymbol{W}^{(2)} \\ =\ & \underbrace{\dot\sigma_3 \boldsymbol{X}^{(0,3)}\boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^0 \boldsymbol{W}^{(2)}}_{\text{corresponds to } s=3} + \underbrace{\dot\sigma_3\dot\sigma_2 \boldsymbol{X}^{(0,2)}\boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^1 \boldsymbol{W}^{(2)}}_{\text{corresponds to } s=2} + \underbrace{\dot\sigma_3\dot\sigma_2\dot\sigma_1 \boldsymbol{X}^{(0,1)}\boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^2 \boldsymbol{W}^{(2)}}_{\text{corresponds to } s=1} \\ =\ & \dot\sigma_3 \underbrace{\boldsymbol{X}^{(0,3)}}_{\text{size: } 1\times h_0} \overbrace{\boldsymbol{W}^{(2)}_{(i,:)}}^{\text{size: } 1\times 1} + \dot\sigma_3\dot\sigma_2 \underbrace{\boldsymbol{X}^{(0,2)}}_{\text{size: } 1\times h_0} \overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{\text{size: } 1\times h_1} \underbrace{\boldsymbol{W}^{(2)}}_{\text{size: } h_1\times 1} + \dot\sigma_3\dot\sigma_2\dot\sigma_1 \underbrace{\boldsymbol{X}^{(0,1)}}_{\text{size: } 1\times h_0} \overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{\text{size: } 1\times h_1} \underbrace{\boldsymbol{W}_r^{(1)}}_{\text{size: } h_1\times h_1} \overbrace{\boldsymbol{W}^{(2)}}^{\text{size: } h_1\times 1} \end{aligned} \tag{26}$$

with the scalar $\dot{\sigma} = \frac{\partial \sigma(x)}{\partial x}$, indexed by order s for specific inputs. Colored notations are used for improved readability. With $\boldsymbol{A}^{[\kappa]} := \bigoplus_{i=\kappa-1}^{0} \boldsymbol{A}^i = [\boldsymbol{A}^{\kappa-1} \cdots \boldsymbol{A}\ \boldsymbol{I}]$, it can be rewritten in matrix form as

$$\begin{aligned}\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,i)}} &= \sum_{j=1}^{3} \left(\boldsymbol{X}^{(0,j)} \boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^{3-j} \boldsymbol{W}^{(2)} \prod_{k=j}^{3} \dot{\sigma}_k \right) \\ &= \underbrace{\boldsymbol{e}_i^\top}_{\text{size: } 1\times h_1} \underbrace{{\boldsymbol{W}_r^{(1)}}^{[3]}}_{h_1 \times 3h_1} \underbrace{\operatorname{diag}\left(\boldsymbol{W}^{(2)}; 3\right)}_{3h_1 \times 3} \underbrace{\left[\prod_{k=j}^{3} \dot{\sigma}_k \boldsymbol{X}^{(0,j)}\right]_3}_{3\times h_0} \end{aligned} \tag{27}$$

The term $\boldsymbol{e}_i^\top$ offers a clear approach to deriving the derivative for the other weights, $\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{(:,j)}}, j \neq i$. Take into account the derivative of $f_\theta(\boldsymbol{S}_{1:3})$ with respect to the first-layer recurrent weights, which is more complicated. For $i \in \mathbb{N}_{h_1}$,

$$\begin{aligned}
&\frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} \\
=\ & \frac{\partial\, \sigma\left(\boldsymbol{X}^{(0,3)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)} + \boldsymbol{0}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}^{(2)}}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} \\
=\ & \frac{\partial \left(\boldsymbol{X}^{(0,3)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} \dot{\sigma}_3 \boldsymbol{W}^{(2)} \\
=\ & \frac{\partial\, \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{W}_r^{(1)}}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} \dot{\sigma}_3 \boldsymbol{W}^{(2)} \\
=\ & \left(\frac{\partial \left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} \dot{\sigma}_2 \boldsymbol{W}_r^{(1)} \right. \\
& \left. + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right) \frac{\partial \boldsymbol{W}_r^{(1)}}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} \right) \dot{\sigma}_3 \boldsymbol{W}^{(2)} \\
=\ & \left(\sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{e}_i^\top \dot{\sigma}_2 \boldsymbol{W}_r^{(1)} + \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{e}_i^\top\right)\dot{\sigma}_3 \boldsymbol{W}^{(2)} \\
=\ & \dot{\sigma}_3 \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{e}_i^\top \boldsymbol{W}^{(2)} + \dot{\sigma}_3\dot{\sigma}_2\sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{e}_i^\top \boldsymbol{W}_r^{(1)} \boldsymbol{W}^{(2)} \\
=\ & \underbrace{\dot{\sigma}_3 \sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)\boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^0 \boldsymbol{W}^{(2)}}_{\text{corresponds to } s=2} \\
& + \underbrace{\dot{\sigma}_3\dot{\sigma}_2\sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^1 \boldsymbol{W}^{(2)}}_{\text{corresponds to } s=1} \\
=\ & \dot{\sigma}_3 \underbrace{\sigma\left(\boldsymbol{X}^{(0,2)}\boldsymbol{W}^{(1)} + \sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)\boldsymbol{W}_r^{(1)}\right)}_{\text{size: } 1\times h_1} \overbrace{\boldsymbol{W}^{(2)}_{(i,:)}}^{\text{size: } 1\times 1} + \dot{\sigma}_3\dot{\sigma}_2 \underbrace{\sigma\left(\boldsymbol{X}^{(0,1)}\boldsymbol{W}^{(1)}\right)}_{\text{size: } 1\times h_1} \overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{\text{size: } 1\times h_1} \underbrace{\boldsymbol{W}^{(2)}}_{\text{size: } h_1\times 1} \\
=\ & \dot{\sigma}_3 \underbrace{\boldsymbol{X}^{(1,2)}}_{\text{size: } 1\times h_1} \overbrace{\boldsymbol{W}^{(2)}_{(i,:)}}^{\text{size: } 1\times 1} + \dot{\sigma}_3\dot{\sigma}_2 \underbrace{\boldsymbol{X}^{(1,1)}}_{\text{size: } 1\times h_1} \overbrace{\boldsymbol{W}^{(1)}_{r(i,:)}}^{\text{size: } 1\times h_1} \underbrace{\boldsymbol{W}^{(2)}}_{\text{size: } h_1\times 1}
\end{aligned} \tag{28}$$

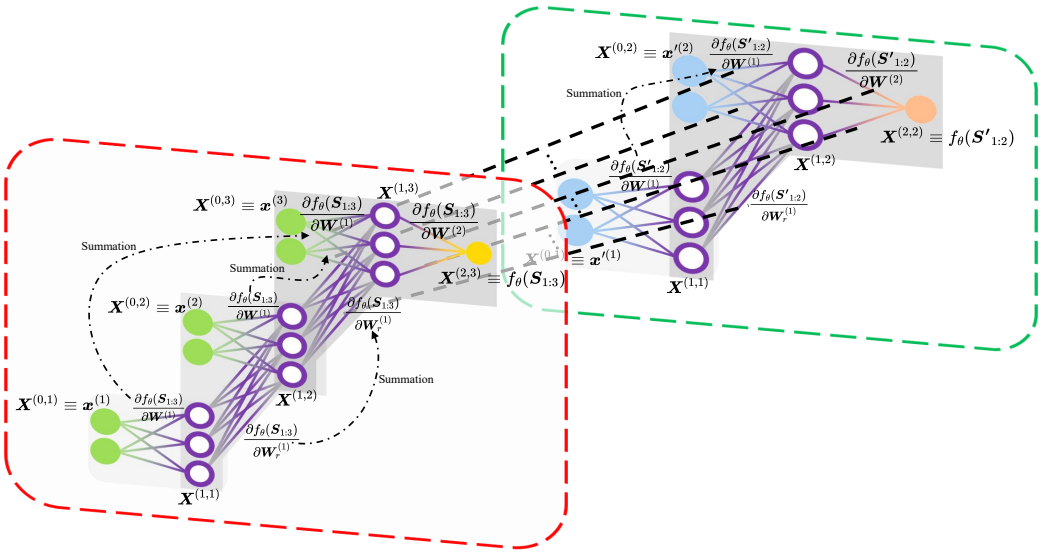


Figure 6: Graphical illustration of RNTK computation: $K_\theta(\boldsymbol{S}_{1:3}, \boldsymbol{S}'_{1:2}) = \left\langle \frac{\partial f_\theta(\boldsymbol{S})}{\partial \theta}, \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \theta} \right\rangle = \frac{\partial f_\theta(\boldsymbol{S})}{\partial \boldsymbol{W}^{(2)}_{(1)}} \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \boldsymbol{W}^{(2)}_{(1)}} + \cdots + \frac{\partial f_\theta(\boldsymbol{S})}{\partial \boldsymbol{W}^{(2)}_{(h_1)}} \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \boldsymbol{W}^{(2)}_{(h_1)}} + \frac{\partial f_\theta(\boldsymbol{S})}{\partial \boldsymbol{W}^{(1)}_{(1,1)}} \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \boldsymbol{W}^{(1)}_{(1,1)}} + \cdots + \frac{\partial f_\theta(\boldsymbol{S})}{\partial \boldsymbol{W}^{(1)}_{(d,h_1)}} \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \boldsymbol{W}^{(1)}_{(d,h_1)}} + \frac{\partial f_\theta(\boldsymbol{S})}{\partial \boldsymbol{W}^{(1)}_{r(1,1)}} \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \boldsymbol{W}^{(1)}_{r(1,1)}} + \cdots + \frac{\partial f_\theta(\boldsymbol{S})}{\partial \boldsymbol{W}^{(1)}_{r(h_1,h_1)}} \frac{\partial f_\theta(\boldsymbol{S}')}{\partial \boldsymbol{W}^{(1)}_{r(h_1,h_1)}}$.

It can be expressed in matrix form as

$$\begin{aligned} \frac{\partial \boldsymbol{X}^{(2,3)}}{\partial \boldsymbol{W}^{(1)}_{r(:,i)}} &= \sum_{j=1}^{3} \left(\boldsymbol{X}^{(1,j-1)} \boldsymbol{e}_i^\top {\boldsymbol{W}_r^{(1)}}^{3-j} \boldsymbol{W}^{(2)} \prod_{k=j}^{3} \dot{\sigma}_k \right) \qquad \#\boldsymbol{X}^{(1,0)} = \boldsymbol{0} \\ &= \underbrace{\boldsymbol{e}_i^\top}_{\text{size: } 1\times h_1} \underbrace{\boldsymbol{W}_r^{(1)[3]}}_{h_1 \times 3h_1} \underbrace{\operatorname{diag}\left(\boldsymbol{W}^{(2)}; 3\right)}_{3h_1 \times 3} \underbrace{\left[\prod_{k=j}^{3} \dot{\sigma}_k \boldsymbol{X}^{(1,j-1)} \right]_3}_{3\times h_0} \end{aligned} \tag{29}$$

The explicit expressions above reveal that the sequence order governs the power of $\boldsymbol{W}_r^{(1)}$ in the gradients, with temporal dependencies in the input sequence reflected in the varying powers of the recurrent weights. The matrix form clearly shows that the RNN gradient is not affected by the length of the input sequence (*i.e.*, the number of elements), but instead depends on the feature dimension. In other words, the parameter gradient stays the same even if the input sequence length S is scaled.

A.4 Recurrent Neural Tangent Kernel (RNTK)

By substituting the parameter evolution (Equation 7)

$$\frac{\partial \theta^t}{\partial t} = -\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left[\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t} \right]_N . \tag{30}$$

into the first-order approximation term $(*)$ of Equation 14, it gets

$$\begin{aligned}(*) &= \left\langle \frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t}, -\frac{\eta}{N}\left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top \cdot \left[\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}\right]_N \right\rangle \\ &= -\frac{\eta}{N}\left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top \cdot \left\langle \frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t}, \left[\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}\right]_N \right\rangle \\ &= -\frac{\eta}{N}\left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top \cdot \left[\left\langle \frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t}, \frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t} \right\rangle\right]_N \\ &= -\frac{\eta}{N}\left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top \cdot \left[K_{\theta^t}(\boldsymbol{S}_i, \cdot)\right]_N, \end{aligned} \tag{31}$$

which derives Equation 15 as

$$\frac{\partial f_{\theta^t}}{\partial t} = -\frac{\eta}{N}\left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right]_N^\top \cdot \left[K_{\theta^t}(\boldsymbol{S}_i, \cdot)\right]_N + o\left(\frac{\partial \theta^t}{\partial t}\right), \tag{32}$$

where the symmetric and positive definite $K_{\theta^t}(\boldsymbol{S}_i, \cdot) \coloneqq \left\langle \frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}, \frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t} \right\rangle$ is referred to as recurrent neural tangent kernel (RNTK) (Alemohammad et al., 2021; Emami et al., 2021). Figure 6 shows the process of calculating the RNTK. Simply put, examining a model behavior by focusing on the model itself, rather than on its parameters, typically involves using kernel functions.

It can be seen that the quantity $\frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t}$, which represents the partial derivative of the RNN with respect to its parameters, appears in $K_{\theta^t}(\boldsymbol{S}_i, \cdot) = \left\langle \frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}, \frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t} \right\rangle$, is determined by both the structure and the specific parameters θ^t, but does not rely on the input sequence. The other term $\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}$ depends not only on the RNN structure and specific θ^t, but also on the input sequence. If the input to $\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}$ is not specified, the RNTK reduces to its general form $K_{\theta^t}(\cdot, \cdot)$. When a specific sequence $\boldsymbol{S}_j$ is defined as the input for $\frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t}$, RNTK becomes a scalar, expressed as $K_{\theta^t}(\boldsymbol{S}_i, \boldsymbol{S}_j) = \langle \frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}, \frac{\partial f_{\theta^t}(\boldsymbol{S}_j)}{\partial \theta^t} \rangle$. These align with the kernel used in functional gradient descent. By specifying the input sequence $\boldsymbol{S}_i$, one coordinate of K_{θ^t} is fixed, directing the RNN to update along $K_{\theta^t}(\boldsymbol{S}_i, \cdot)$, with the update magnitude determined by $\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}$. This process adheres to the fundamental concept of functional gradient descent. In essence, the RNTK and the canonical kernel share a consistent mathematical structure and exhibit similar influences on the evolution of the corresponding RNN. Additionally, Theorem 3 underscores the asymptotic relationship between the RNTK and the canonical kernel within the context of functional gradient descent.

A.5 RENT ALGORITHM

Algorithm 1 ReNT Algorithm

Input: Target mapping f^* realized by a dense set of sequence-property pairs, initial RNN f_{θ^0}, the size of selected training set $m \leq N$, small constant $\epsilon > 0$ and maximal iteration number T.

Set $f_{\theta^t} \leftarrow f_{\theta^0}$, $t = 0$.

while $t \leq T$ and $\|[f_{\theta^t}(\boldsymbol{S}_i) - f^*(\boldsymbol{S}_i)]_N\|_2 \geq \epsilon$ **do**

The teacher selects m teaching sequences:

/* (**Sequence-level**) Sequences corresponding to the m largest $|f_{\theta^t}(\boldsymbol{S}_i) - f^*(\boldsymbol{S}_i)|$. */

$$\{\boldsymbol{S}_i\}_m{}^* = \underset{\{\boldsymbol{S}_i\}_m \subseteq \{\boldsymbol{S}_i\}_N}{\arg\max} \|[f_{\theta^t}(\boldsymbol{S}_i) - f^*(\boldsymbol{S}_i)]_m\|_2.$$

/* (**Element-level**) Sequences associated with the m largest $\frac{\|f_{\theta^t}(\boldsymbol{S}_i) - f^*(\boldsymbol{S}_i)\|_2}{S_i}$. */

$$\{\boldsymbol{S}_i\}_m{}^* = \underset{\{\boldsymbol{S}_i\}_m \subseteq \{\boldsymbol{S}_i\}_N}{\arg\max} \left\| \left[\frac{f_{\theta^t}(\boldsymbol{S}_i) - f^*(\boldsymbol{S}_i)}{S_i} \right]_m \right\|_{\mathcal{F}}, \text{ with Frobenius norm } \|\cdot\|_{\mathcal{F}}.$$

Provide $\{\boldsymbol{S}_i\}_m{}^*$ to the RNN learner.

The learner updates f_{θ^t} based on received $\{\boldsymbol{S}_i\}_m{}^*$:

// Parameter-based gradient descent.

$$\theta^t \leftarrow \theta^t - \frac{\eta}{m} \sum_{\boldsymbol{S}_i \in \{\boldsymbol{S}_i\}_m{}^*} \nabla_\theta \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), f^*(\boldsymbol{S}_i)).$$

Set $t \leftarrow t + 1$.

end

B Detailed Proofs

Before diving into the detailed proofs, we first introduce the gradient of an evaluation functional $E_{\boldsymbol{S}}(f)$.

Lemma 7. *The gradient of an evaluation functional $E_{\boldsymbol{S}}(f) = f(\boldsymbol{S}) : \mathcal{H} \mapsto \mathbb{R}$ is $\nabla_f E_{\boldsymbol{S}}(f) = K_{\boldsymbol{S}}$.*

B.1 Proof of Lemma 7

Let us define a function ϕ by adding a small perturbation ϵg ($\epsilon \in \mathbb{R}, g \in \mathcal{H}$) to $f \in \mathcal{H}$, so that $\phi = f + \epsilon g$. Since RKHS is closed under addition and scalar multiplication, $\phi \in \mathcal{H}$. Thus, for an evaluation functional $E_{\boldsymbol{S}}[f] = f(\boldsymbol{S}) : \mathcal{H} \mapsto \mathbb{R}$, we can evaluate ϕ at $\boldsymbol{S}$ as

$$\begin{aligned} E_{\boldsymbol{S}}[\phi] &= E_{\boldsymbol{S}}[f + \epsilon g] \\ &= E_{\boldsymbol{S}}[f] + \epsilon E_{\boldsymbol{S}}[g] + 0 \\ &= E_{\boldsymbol{S}}[f] + \epsilon \langle K(\boldsymbol{S}, \cdot), g \rangle_{\mathcal{H}} + 0 \end{aligned} \tag{33}$$

Referring to the implicit definition of the Fréchet derivative in an RKHS (see Definition 2), given by $E_{\boldsymbol{S}}[f + \epsilon g] = E_{\boldsymbol{S}}[f] + \epsilon \langle \nabla_f E_{\boldsymbol{S}}[f], g \rangle_{\mathcal{H}} + o(\epsilon)$, it follows from Equation 33 that the gradient of the evaluation functional is $\nabla_f E_{\boldsymbol{S}}[f] = K_{\boldsymbol{S}}$.

■

B.2 Proof of Theorem 3

By analyzing the evolution of an RNN through parameter changes and from a high-level viewpoint within the function space, we gain

$$\begin{aligned} & -\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot [K(\boldsymbol{S}_i, \cdot)]_N \\ = \quad & -\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left[\left\langle \frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t}, \frac{\partial f_{\theta^t}(\cdot)}{\partial \theta^t} \right\rangle \right]_N + o\left(\frac{\partial \theta^t}{\partial t} \right). \end{aligned} \tag{34}$$

After the reorganization, we obtain

$$-\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot [K(\boldsymbol{S}_i, \cdot) - K_{\theta^t}(\boldsymbol{S}_i, \cdot)]_N = o\left(\frac{\partial \theta^t}{\partial t} \right). \tag{35}$$

By incorporating the evolution of the parameters

$$\frac{\partial \theta^t}{\partial t} = -\eta \frac{\partial \mathcal{L}}{\partial \theta^t} = -\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left[\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t} \right]_N \tag{36}$$

into the remainder, we obtain

$$-\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot [K(\boldsymbol{S}_i, \cdot) - K_{\theta^t}(\boldsymbol{S}_i, \cdot)]_N = o\left(-\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left[\frac{\partial f_{\theta^t}(\boldsymbol{S}_i)}{\partial \theta^t} \right]_N \right). \tag{37}$$

When training an RNN with a convex loss $\mathcal{L}$, which is convex with respect to f_θ but not necessarily with respect to θ, the following limit holds for the vector: $\lim_{t \to \infty} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N = \mathbf{0}$. Since the right-hand side of this equation is a higher-order infinitesimal than the left, maintaining this equality leads to the conclusion that

$$\lim_{t \to \infty} [K(\boldsymbol{S}_i, \cdot) - K_{\theta^t}(\boldsymbol{S}_i, \cdot)]_N = \mathbf{0}. \tag{38}$$

This indicates that for every $\boldsymbol{S} \in \{\boldsymbol{S}_i\}_N$, GNTK converges pointwise to the canonical kernel.

■

B.3 Proof of Proposition 4

Referring back to the definition of the Fréchet derivative in Definition 2, the convexity of $\mathcal{L}$ indicates that

$$\frac{\partial \mathcal{L}}{\partial t} \leq \underbrace{\left\langle \frac{\partial \mathcal{L}}{\partial f_{\theta^{t+1}}}, \frac{f_{\theta^t}}{\partial t} \right\rangle_{\mathcal{H}}}_{\Upsilon}. \tag{39}$$

By determining the Fréchet derivative of $\frac{\partial \mathcal{L}}{\partial f_{\theta^{t+1}}}$ and the evolution of f_{θ^t}, the term on the right-hand side, Υ, can be written as

$$\begin{aligned}
\Upsilon &= \left\langle \mathcal{G}^{t+1}, -\eta \mathcal{G}^t \right\rangle_{\mathcal{H}} \\
&= -\frac{\eta}{N^2} \left\langle \left[\frac{\partial \mathcal{L}(f_{\theta^{t+1}}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^{t+1}}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left[K_{\boldsymbol{S}_i} \right]_N, \left[K_{\boldsymbol{S}_i} \right]_N^\top \cdot \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N \right\rangle_{\mathcal{H}} \\
&= -\frac{\eta}{N^2} \left[\frac{\partial \mathcal{L}(f_{\theta^{t+1}}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^{t+1}}(\boldsymbol{S}_i)} \right]_N^\top \cdot \left\langle \left[K_{\boldsymbol{S}_i} \right]_N, \left[K_{\boldsymbol{S}_i} \right]_N^\top \right\rangle_{\mathcal{H}} \cdot \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N \\
&= -\frac{\eta}{N} \left[\frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)} \right]_N^\top \bar{\boldsymbol{K}} \left[\frac{\partial \mathcal{L}(f_{\theta^{t+1}}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^{t+1}}(\boldsymbol{S}_i)} \right]_N,
\end{aligned} \tag{40}$$

where $\bar{\boldsymbol{K}} = \boldsymbol{K}/N$, and $\boldsymbol{K}$ is an $N \times N$ symmetric, positive definite matrix with elements $K(\boldsymbol{S}_i, \boldsymbol{S}_j)$ located at the i-th row and j-th column. For simplicity, we use the shorthand notation $\partial_{f_{\theta^\square}} \mathcal{L}(f_{\theta^\square}; \boldsymbol{S}_i) \coloneqq \frac{\partial \mathcal{L}(f_{\theta^\square}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^\square}(\boldsymbol{S}_i)}$. The last term in Equation 40 can then be rewritten as

$$\begin{aligned}
& -\frac{\eta}{N} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N \\
= & -\frac{\eta}{N} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N + \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right) \\
= & -\frac{\eta}{N} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \\
& -\frac{\eta}{N} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right) \\
= & -\frac{\eta}{N} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \\
& +\frac{\eta}{N} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N^\top - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top - \left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N^\top \right) \\
& \cdot \bar{\boldsymbol{K}} \cdot \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right).
\end{aligned} \tag{41}$$

The final term in Equation 41 above can be expanded as

$$\begin{aligned}
& \frac{\eta}{N} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N^\top - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top - \left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N^\top \right) \\
& \cdot \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right) \\
= & \frac{\eta}{N} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right)^\top \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right) \\
& -\frac{\eta}{N} \left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right) \\
= & \frac{\eta}{N} \left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) - \partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) - \partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \\
& -\frac{\eta}{N} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \frac{1}{2} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right)^\top \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \frac{1}{2} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right) \\
& +\frac{\eta}{4N} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N^\top \bar{\boldsymbol{K}} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N.
\end{aligned} \tag{42}$$

Since $\bar{\boldsymbol{K}}$ is positive definite, it is evident that

$$\frac{\eta}{N} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \frac{1}{2} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right)^\top \bar{\boldsymbol{K}} \left(\left[\partial_{f_{\theta^{t+1}}} \mathcal{L}(f_{\theta^{t+1}}; \boldsymbol{S}_i) \right]_N - \frac{1}{2} \left[\partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i) \right]_N \right)$$

is a non-negative term. Therefore, by combining Equations 40, 41, and 42, we get

$$\begin{aligned}\Upsilon \quad \leq \quad & -\frac{3\eta}{4N}\underbrace{\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top \bar{\boldsymbol{K}}\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N}_{\Phi} \\ & +\frac{\eta}{N}\underbrace{\left[\partial_{f_{\theta^{t+1}}}\mathcal{L}(f_{\theta^{t+1}};\boldsymbol{S}_i)-\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top \bar{\boldsymbol{K}}\left[\partial_{f_{\theta^{t+1}}}\mathcal{L}(f_{\theta^{t+1}};\boldsymbol{S}_i)-\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N}_{\Psi}.\end{aligned} \tag{43}$$

Given the definition of the evaluation functional and the assumption that $\mathcal{L}$ is Lipschitz smooth with a constant $\tau > 0$, the term Ψ in the final part of Equation 43 is upper-bounded as follows:

$$\begin{aligned}\Psi \quad &= \quad \left[\partial_{f_{\theta^{t+1}}}\mathcal{L}(f_{\theta^{t+1}};\boldsymbol{S}_i)-\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top \bar{\boldsymbol{K}}\left[\partial_{f_{\theta^{t+1}}}\mathcal{L}(f_{\theta^{t+1}};\boldsymbol{S}_i)-\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N \\ &= \quad \left[E_{\boldsymbol{S}_i}\left(\frac{\partial\mathcal{L}(f_{\theta^{t+1}})}{\partial f_{\theta^{t+1}}}-\frac{\partial\mathcal{L}(f_{\theta^t})}{\partial f_{\theta^t}}\right)\right]_N^\top \bar{\boldsymbol{K}}\left[E_{\boldsymbol{S}_i}\left(\frac{\partial\mathcal{L}(f_{\theta^{t+1}})}{\partial f_{\theta^{t+1}}}-\frac{\partial\mathcal{L}(f_{\theta^t})}{\partial f_{\theta^t}}\right)\right]_N \\ &\leq \quad \tau^2\left[E_{\boldsymbol{S}_i}\left(f_{\theta^{t+1}}-f_{\theta^t}\right)\right]_N^\top \bar{\boldsymbol{K}}\left[E_{\boldsymbol{S}_i}\left(f_{\theta^{t+1}}-f_{\theta^t}\right)\right]_N \\ &= \quad \tau^2\left\langle\left(f_{\theta^{t+1}}-f_{\theta^t}\right),[K_{\boldsymbol{S}_i}]_N^\top\right\rangle_{\mathcal{H}}\cdot\bar{\boldsymbol{K}}\cdot\left\langle[K_{\boldsymbol{S}_i}]_N,\left(f_{\theta^{t+1}}-f_{\theta^t}\right)\right\rangle_{\mathcal{H}} \\ &= \quad \eta^2\tau^2\cdot\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top\frac{\left\langle[K_{\boldsymbol{S}_i}]_N,[K_{\boldsymbol{S}_i}]_N^\top\right\rangle_{\mathcal{H}}}{N}\cdot\bar{\boldsymbol{K}}\cdot\frac{\left\langle[K_{\boldsymbol{S}_i}]_N,[K_{\boldsymbol{S}_i}]_N^\top\right\rangle_{\mathcal{H}}}{N}\cdot\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N.\end{aligned} \tag{44}$$

Given the assumption that the canonical kernel is bounded above by a constant $\gamma > 0$, we have

$$\left\langle[K_{\boldsymbol{S}_i}]_N,[K_{\boldsymbol{S}_i}]_N^\top\right\rangle_{\mathcal{H}} \leq \gamma\left\langle[1]_N,[1]_N^\top\right\rangle,$$

and

$$\bar{\boldsymbol{K}} \leq \frac{\gamma}{N}\left\langle[1]_N,[1]_N^\top\right\rangle.$$

Consequently, Φ is upper bounded by

$$\begin{aligned}\Phi \quad &\leq \quad \frac{\gamma}{N}\left\langle\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top,[1]_N\right\rangle\left\langle[1]_N^\top,\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N\right\rangle \\ &= \quad \frac{\gamma}{N}\left(\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right)^2.\end{aligned} \tag{45}$$

Additionally, the final term in Equation 44 is also bounded from above:

$$\begin{aligned}& \eta^2\tau^2\cdot\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top\frac{\left\langle[K_{\boldsymbol{S}_i}]_N,[K_{\boldsymbol{S}_i}]_N^\top\right\rangle_{\mathcal{H}}}{N}\cdot\bar{\boldsymbol{K}}\cdot\frac{\left\langle[K_{\boldsymbol{S}_i}]_N,[K_{\boldsymbol{S}_i}]_N^\top\right\rangle_{\mathcal{H}}}{N}\cdot\left[\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N \\ \leq \quad & \eta^2\tau^2\left[\frac{\gamma}{N}\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]^\top\cdot\bar{\boldsymbol{K}}\cdot\left[\frac{\gamma}{N}\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N \\ \leq \quad & \frac{\eta^2\tau^2\gamma^3}{N}\left\langle\left[\frac{1}{N}\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N^\top,[1]_N\right\rangle\left\langle[1]_N^\top,\left[\frac{1}{N}\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right]_N\right\rangle \\ = \quad & \frac{\eta^2\tau^2\gamma^3}{N}\left(\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right)^2.\end{aligned} \tag{46}$$

Therefore, by combining Equations 43, 44, 45, and 46, we obtain

$$\Upsilon \leq -\eta\gamma\left(\frac{3}{4}-\eta^2\tau^2\gamma^2\right)\left(\frac{1}{N}\sum_{i=1}^N\partial_{f_{\theta^t}}\mathcal{L}(f_{\theta^t};\boldsymbol{S}_i)\right)^2, \tag{47}$$

which means

$$\frac{\partial \mathcal{L}}{\partial t} \leq \Upsilon \leq -\eta\gamma \left(\frac{3}{4} - \eta^2\tau^2\gamma^2\right) \left(\frac{1}{N}\sum_{i=1}^{N} \partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i)\right)^2. \tag{48}$$

Thus, if $\eta \leq \frac{1}{2\tau\gamma}$, it follows that

$$\frac{\partial \mathcal{L}}{\partial t} \leq -\frac{\eta\gamma}{2} \left(\frac{1}{N}\sum_{i=1}^{N} \partial_{f_{\theta^t}} \mathcal{L}(f_{\theta^t}; \boldsymbol{S}_i)\right)^2 = -\frac{\eta\gamma}{2} \left(\frac{1}{N}\sum_{i=1}^{N} \frac{\partial \mathcal{L}(f_{\theta^t}(\boldsymbol{S}_i), \boldsymbol{y}_i)}{\partial f_{\theta^t}(\boldsymbol{S}_i)}\right)^2. \tag{49}$$

■

C EXPERIMENT DETAILS

This section provides an overview of the experiment, including the setup, additional results, and a brief analysis of sequence-level and element-level tasks on benchmark datasets. The code will be made available.

C.1 EXPERIMENTAL SETUP

Device Setup. All experiments are conducted using NVIDIA GeForce RTX 3090 (24GB) GPUs.

Datasets. We assess ReNT on a range of widely used benchmark datasets across various tasks:

- UCI Electricity Load Diagrams (Trindade, 2015) (referred to as Electricity; CC BY 4.0): A multivariate time series dataset featuring electricity consumption data from 370 clients, commonly used for forecasting tasks. It contains 26,304 hourly records across all series.
- Weather (for Biogeochemistry, 2020) (CC BY 4.0): A meteorological time series dataset that includes hourly climate data such as temperature, humidity, and wind speed. It comprises 52,696 hourly observations collected over six years.
- Yelp Review Full (Zhang et al., 2015) (referred to as Yelp): A sentiment classification dataset containing complete user reviews, with ratings ranging from 1 to 5 stars. The dataset includes 650,000 labeled samples evenly distributed across five classes.
- AG News (Zhang et al., 2015): A topic classification dataset consisting of news articles from four categories: World, Sports, Business, and Sci / Technology. It contains 120,000 samples, with 30,000 instances per class.
- Universal Dependencies (UD) (Nivre et al., 2016): A part-of-speech tagging dataset built from treebanks in several languages, designed for syntactic analysis. It includes over 15,000 annotated samples across multiple languages.
- CoNLL-2003 (Tjong Kim Sang & De Meulder, 2003) (referred to as CoNLL): A named entity recognition benchmark made up of annotated newswire data from Reuters, covering entities such as people, organizations, locations, and miscellaneous categories. The dataset contains over 20,000 annotated samples.

We use the Electricity and Weather datasets in sequence-level tasks for time series forecasting. To accommodate model capacity and sequence diversity, we select the top 40 most variant series from the Electricity dataset, meanwhile, utilize the complete Weather dataset. For the Electricity dataset, we adopt a sliding window setup, where each input sequence consists of the past 180 time steps used to predict the next 72 time steps. For the Weather dataset, we use the past 96 time steps as input to forecast the next 24 time steps. All time series inputs are standardized using a StandardScaler (Pedregosa et al., 2011), and then split into training, validation, and test sets with a ratio of 70% / 15% / 15%.

We use the Yelp and AG News datasets in sequence-level tasks for text classification. Specifically, We hold out 10% of the original training set for validation, and adopt the official test sets for evaluation.

We use the UD and CoNLL datasets in element-level tasks for part-of-speech tagging and named entity recognition, respectively. We follow the official splits provided by the dataset authors for training, validation, and testing.

The train/validation/test configurations for all benchmark datasets are summarized in Table 3.

Dataset	train	validation	test
Electricity	3063	657	657
Weather	1149	237	237
Yelp	585000	65000	50000
AG News	108000	12000	7600
UD	12543	2002	2077
CoNLL	14987	3466	3684

Table 3: Dataset splitting for the benchmark datasets.

ReNT strategy. To accommodate the specific characteristics of different datasets, ReNT adopts various sampling ratios and interval strategies accordingly.

- Sampling Ratio
  - Step – Incrementing the sampling ratio from a predefined minimum (*i.e.*, init-ratio) to 100% in equal intervals, where the ratio increases stage-by-stage based on training progress.
  - R-Step – Decrementing the sampling ratio from a predefined maximum (*i.e.*, init-ratio) to a predefined minimum in equal intervals, where the ratio gradually decreases stage-by-stage as training progresses.
- Sampling Interval
  - Incremental (Zhang et al., 2024) – Sampling is performed frequently during an initial proportion of training, after which the sampling interval gradually increases as training progresses.
  - Constant – Sampling is triggered at the beginning of training, and subsequently performed at a fixed interval throughout the rest of training.

Given the distinct modality characteristics of each dataset, we design our sampling strategies accordingly. For time series regression tasks, the input exhibits strong temporal dependencies and relatively stable structural patterns. In early training stages, the model typically lacks awareness of global trends. Therefore, we apply the Step + Incremental strategy, starting with frequent sampling of representative subsets and gradually increasing the sampling ratio. This helps the model progressively learn temporal dynamics while reducing redundant computation in later phases.

In contrast, textual classification tasks involve data with high diversity, complex syntactic structures, and strong context-dependent semantics. To enable the model to broadly capture diverse linguistic patterns in early stages, we employ the R-step + Constant strategy, *i.e.*, a high initial sampling ratio with a fixed interval, then gradually reduce the sampling ratio as training proceeds, which is more suitable for textual modalities, allowing efficient training while preserving model stability and improving generalization.

The detailed hyperparameter settings for each dataset are listed in Table 4. We also conduct ablation studies on the key sampling parameter init-ratio to evaluate its effect on training efficiency and performance. Detailed results are presented in Table 5 and Table 6, which report results based on the ReNT (B) variant. For a detailed description of the sampling strategies applied to each dataset, please refer to Section C.3.

Hyperparameter Settings. The key hyperparameter settings for all benchmark datasets are listed in Table 4.

Dataset	lr	Batch-size	RNN Layers	Hidden size	Sampling ratio	Sampling interval	Init-ratio	Epochs
Electricity	0.0001	32	2	128	Step	Incremental	0.1	30
Weather	0.00005	32	2	128	Step	Incremental	0.05	100
Yelp	0.0001	256	2	64	R-Step	Constant	0.7	100
AG News	0.0002	256	2	64	R-Step	Constant	0.7	50
UD	0.0002	32	2	64	R-Step	Constant	0.8	75
CoNLL	0.0002	32	2	64	R-Step	Constant	0.7	50

Table 4: Key hyperparameter settings for the benchmark datasets, with the "Sampling ratio", "Sampling interval", and "Init-ratio" specified for ReNT.

Dataset	Metric	0.05	0.1	0.2	0.3	0.5	full
Electricity	MAE	0.0542	0.0515	0.0526	0.053	0.0509	0.0539
	Training time (s)	1225.31	1283.98	1459.99	1502.57	2040.74	2394.97
Weather	MAE	0.0268	0.0282	0.0270	0.0315	0.0265	0.0278
	Training time (s)	419.98	560.23	624.71	709.78	829.22	1004.08

Table 5: Performance comparison under different init-ratio settings on regression datasets.

Time Consumption Details. The selection process in ReNT mainly involves model inference and a subsequent ranking step, whose cost is much lower than backpropagation, especially for high-dimensional sequence data. As shown in Table 7, on the Yelp dataset, both ReNT (B) and ReNT (S)

Dataset	Metric	0.9	0.8	0.7	0.75	0.6	full
UD	ACC	0.8659	0.8716	0.8652	0.8652	0.8623	0.8643
	Training time (s)	355.68	316.03	303.69	299.73	285.76	519.27
CoNLL	ACC	0.9045	0.9020	0.8987	0.8930	0.8988	0.8934
	Training time (s)	401.99	282.88	252.35	232.47	201.84	394.40

Table 6: Performance comparison under different init-ratio settings on classification datasets.

considerably reduce the overall training time compared with the baseline, even when including the sampling stage.

ReNT	Sampling Time (s)	Training/Validating Time (s)	Full Time (s)
✗	-	24815.66	24815.66
B	957.67	13117.38	14075.05
S	988.63	13316.91	14305.54

Table 7: Time consumption comparison with and without ReNT on the Yelp dataset.

C.2 Generalizability of ReNT on RNN Variants

The concept and analysis of ReNT hold great potential for broader applications. Although it is beyond the scope of this paper, we also validate the adaptability and generality of ReNT across different recurrent neural network architectures. We replaced the default two-layer vanilla RNN used in the main experiments with two variants: a two-layer GRU and a two-layer LSTM. Experiments were conducted on the Electricity dataset, and the results are summarized in Table 8. As shown, both ReNT (B) and ReNT (S) consistently outperform the "without ReNT" in terms of testing loss and mean absolute error (MAE). More notably, both variants achieve a substantial reduction in training time—approximately one-third—demonstrating significant efficiency gains consistent with those observed in the original RNN setting. These results demonstrate that ReNT's training efficiency enhancement is not restricted to specific RNN architectures, confirming its general applicability as expected.

ReNT	Variants	Training Time (s)	Testing Loss ↓	Testing MAE ↓
✗	GRU (Cho et al., 2014b)	1941.13	0.0041	0.0461
	LSTM (Hochreiter & Schmidhuber, 1997)	2247.06	0.0052	0.0489
B	GRU (Cho et al., 2014b)	1278.01	**0.0040**	**0.0458**
	LSTM (Hochreiter & Schmidhuber, 1997)	**1223.72**	0.0050	0.0471
S	GRU (Cho et al., 2014b)	**1242.38**	0.0041	0.0468
	LSTM (Hochreiter & Schmidhuber, 1997)	1254.27	**0.0048**	**0.0470**

Table 8: Performance comparison of ReNT on RNN variants on the Electricity dataset.

To further examine the scalability of ReNT, we conducted additional experiments on larger recurrent architectures with increased numbers of layers and hidden sizes. Although existing research on RNNs typically adopts relatively lightweight architectures with no more than four layers Feng et al. (2024), which is also consistent with most of our experimental settings, we aim to verify whether the efficiency benefits of ReNT persist in deeper and wider configurations. As summarized in Table 9, both ReNT (B) and ReNT (S) maintain competitive or improved predictive accuracy while substantially reducing training time—often by nearly half—compared with the corresponding settings without ReNT. This confirms that the advantages of ReNT extend robustly to larger RNN configurations.

C.3 Additional Discussion

After an extensive literature survey, we find that no mature methods have emerged in recent years that apply sample selection strategies—including curriculum learning (Bengio et al., 2009) and active

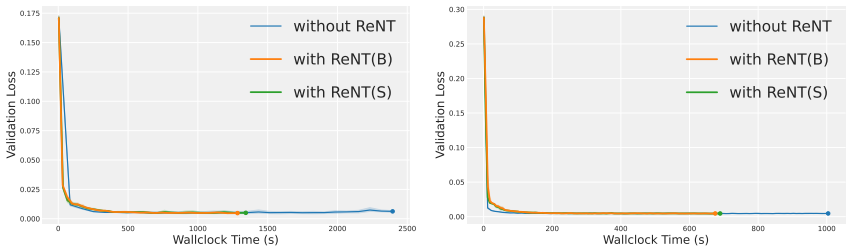
ReNT	Layers / Hidden Size	Training Time (s)	Testing Loss ↓	Testing MAE ↓
✗	6 / 128	3253.90	0.0061	0.0566
	6 / 256	3173.86	0.0088	0.0700
	8 / 128	3486.68	0.0078	0.0657
	8 / 256	4117.61	0.0074	0.0633
B	6 / 128	1816.87	0.0061	**0.0542**
	6 / 256	1696.54	**0.0055**	**0.0541**
	8 / 128	2044.74	**0.0058**	**0.0557**
	8 / 256	2209.65	0.0058	0.0544
S	6 / 128	1994.76	**0.0059**	0.0547
	6 / 256	1827.34	0.0061	0.0560
	8 / 128	1935.30	0.0062	0.0566
	8 / 256	2268.56	**0.0056**	**0.0542**

Table 9: Performance comparison of ReNT with different layer and hidden size settings.

learning (Cohn et al., 1996)—to specifically improve the training efficiency of RNNs, which may also represent a promising direction for future research.

Sequence-level tasks. In sequence-level regression tasks, we adopt the Step + Incremental sampling strategy. Specifically, during an initial short phase of training, sampling is performed frequently. After this phase, the sampling interval gradually increases. Throughout the entire training process, the sampling ratio starts from the initial value init-ratio and progressively increases until it reaches 100%. For the relatively smaller Weather dataset, we set the ratio to 0.05, while for the more challenging Electricity dataset, where feature patterns are harder to capture, we use a higher value of 0.1.

As shown in Figure 7, although frequent sampling is applied at the beginning, ReNT (B) and ReNT (S) select only a small subset of the most diverse samples in the time series. As a result, their validation loss decreases at a comparable rate to the "without ReNT" setting, while substantially shortening the overall training time.



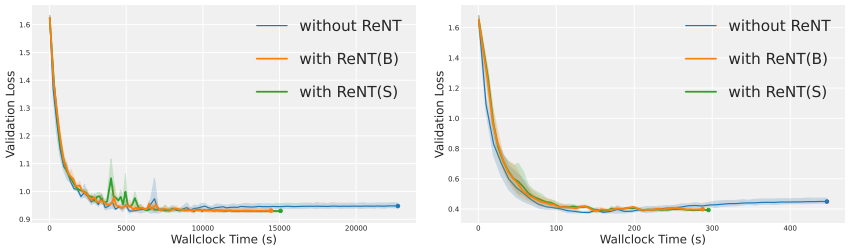
(a) Electricity (b) Weather

Figure 7: Validation loss performance on sequence-level regression tasks.

In sequence-level classification tasks, we adopt the R-Step + Constant sampling strategy. Specifically, a higher sampling ratio of 0.7 is used at the beginning of training to cover more examples, and the ratio gradually decreases to a minimum of 0.4 as training progresses. The sampling interval remains fixed throughout the entire training process. This strategy is applied consistently across both datasets used for classification.

Figure 8 presents the validation loss curves under this strategy on two datasets of different scales. ReNT (B) and ReNT (S) exhibit similar descending trends compared to the "without ReNT" setting. Due to the dynamic nature of the sampled data subsets, some fluctuations appear during the earlier stages, but the curves gradually stabilize and eventually converge to lower loss values than those of "without ReNT". Moreover, the training process terminates much earlier, further demonstrating the efficiency of ReNT under this strategy. Detailed results are reported in Table 6.

Element-level tasks. Similar to the setup for sequence-level classification, we adopt the R-Step + Constant sampling strategy for element-level classification tasks. Specifically, an initial sampling

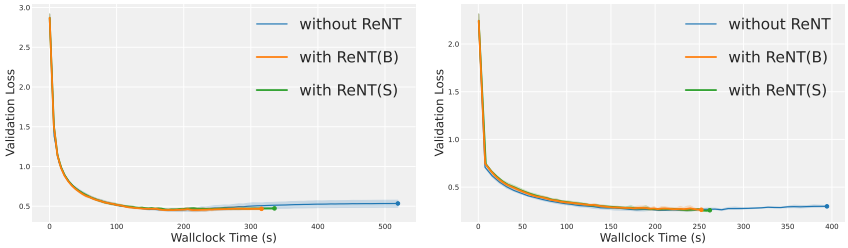


(a) Yelp (b) AG News

Figure 8: Validation loss performance on sequence-level classification tasks.

ratio of 0.8 is used for the named entity recognition task on the UD dataset, while a ratio of 0.7 is applied to the part-of-speech tagging task on the CoNLL-2003 dataset. The minimum sampling ratio in both cases is set to 0.4, and the sampling interval remains constant throughout training.

Figure 9 shows the validation loss curves under this configuration. In both tasks, ReNT (B) and ReNT (S) exhibit almost identical loss reduction trends to the "without ReNT" setting during the early training stages. Nevertheless, both methods terminate training substantially earlier and converge to lower final loss values, demonstrating clear efficiency gains with no compromise in model quality.



(a) UD (b) CoNLL

Figure 9: Validation loss performance on element-level classification tasks.