A Properties of Path Signatures

First, we recall that the path is uniquely determined by its signature, which motivates its use as a feature map.

Proposition A.1. *Given a path* $\widehat{X} : [0,T] \to \mathbb{R}^d$, *then the map* $P : [0,T] \to \mathbb{R}^{1+d}$ *where* $P(t) = (t, \widehat{X}(t))$ *is uniquely determined by it's signature* $S(P)_{0,T}$.

The proof can be found in Hambly and Lyons [33].

For Rough Transformers, several features of path signatures are important. First, linear functionals on path signatures possess universal approximation properties for continuous functionals.

Theorem A.2. *Fix* $T > 0$, *and let* $K \subset C_b^1([0,T]; \mathbb{R}^d)$. *Let* $f : K \to \mathbb{R}$ *be continuous with respect to the sup-norm topology on* $C_b^1([0,T]; \mathbb{R}^d)$. *Then for any* $\epsilon > 0$, *there exists a linear functional* ℓ *such that*

$$|f(\overline{X}) - \langle \ell, S(\overline{X})_{0,T} \rangle| \le \epsilon, \tag{13}$$

for any $\widehat{X} \in K$, *where* $\overline{X}$ *denotes the time-added augmentation of* $\widehat{X}$.

For a proof of A.2, see Arribas [2]. Even though Theorem A.2 guarantees that *linear* functionals are sufficient for universal approximation, linear models are not always sufficient in practice. This motivates the development of nonlinear models built upon the path signature which efficiently extract path behavior.

The second feature is that the terms of the path signature decay factorially, as described by the following proposition.

Proposition A.3. *Given* $\widehat{X} \in C_b^1([0,T]; \mathbb{R}^d)$, *for any* $s,t \in [0,T]$, *we have that for any* $I \in \mathcal{I}_d^n$

$$|S(\widehat{X})_{0,T}^I| = O\left(1/n!\right). \tag{14}$$

For a proof of Proposition A.3, see [51]. Hence, the number of terms in the signature grows exponentially in the level of the signature, but the tail of the signature is well-behaved, so only a few levels in a truncated signature are necessary to adequately approximate continuous functionals.

A.1 Signatures of Piecewise Linear Paths.

In the Rough Transformer, we use linear interpolation of input time-series to get a continuous-time representation of the data. As mentioned in Section 3, the signature computation in this case is particularly simple.

Suppose $\widehat{X}_k : [t_k, t_{k+1}] \to \mathbb{R}^d$ is a linear interpolation between two points $X_k, X_{k+1} \in \mathbb{R}^d$. That is,

$$\widehat{X}_k(t) = X_k + \frac{t - t_k}{t_{k+1} - t_k}\left(X_{k+1} - X_k\right). \tag{15}$$

Then the signature of $\widehat{X}_k$ is given explicitly by

$$S(\widehat{X}_k)_{t_k, t_{k+1}} = \left(1, X_{k+1} - X_k, \frac{1}{2}(X_{k+1} - X_k)^{\otimes 2}, \frac{1}{3!}(X_{k+1} - X_k)^{\otimes 3}, ..., \frac{1}{n!}(X_{k+1} - X_k)^{\otimes n}, ...\right),$$
$$\tag{16}$$

where $\otimes$ denotes the tensor product. Let $\widehat{X}_k * \widehat{X}_{k+1}$ denote the *concatenation* of $\widehat{X}_k$ and $\widehat{X}_{k+1}$. That is, $\widehat{X}_k * \widehat{X}_{k+1} : [t_k, t_{k+2}] \to \mathbb{R}^d$ is given by

$$\widehat{X}_k * \widehat{X}_{k+1}(t) = \begin{cases} \widehat{X}_k(t) & t \in [t_k, t_{k+1}] \\ \widehat{X}_{k+1}(t) & t \in (t_2, t_{k+2}]. \end{cases} \tag{17}$$

The signature of the concatenation $\widehat{X}_k * \widehat{X}_{k+1}$ is given by *Chen's relation*, whose proof is in [51]. To state this result, we first note that $S(\widehat{X})_{s,t}^n$ can be interpreted as an element of the *extended tensor algebra* of $\mathbb{R}^d$:

$$T((\mathbb{R}^d)) = \left\{(a_0, ..., a_n, ...) : a_n \in \mathbb{R}^{d \otimes n}\right\}. \tag{18}$$

16

Proposition A.4 (Chen's Relation). *The following identity holds:*

$$S(\widehat{X}_k * \widehat{X}_{k+1})_{t_k,t_{k+2}} = S(\widehat{X}_k)_{t_k,t_{k+1}} \otimes S(\widehat{X}_{k+1})_{t_{k+1},t_{k+2}}, \tag{19}$$

where for elements $A, B \in T((\mathbb{R}^d))$ with $A = (A_0, A_1, A_2, ...)$ and $B = (B_0, B_1, B_2, ...)$ the tensor product $\otimes$ is defined

$$A \otimes B = \left(\sum_{j=0}^{k} A_j \otimes B_{k-j} \right)_{k \geq 0}. \tag{20}$$

Let $\mathbf{X} = (X_0, ..., X_L)$ be a time-series. Then the linear interpolation $\tilde{X} : [0, T] \to \mathbb{R}^d$ can be represented as the concatenation of a finite number of linear paths:

$$\tilde{X} = \widehat{X}_0 * \cdots * \widehat{X}_{L-1}. \tag{21}$$

Hence, the signature is

$$S(\tilde{X})_{0,T} = S(\widehat{X}_0)_{0,t_1} \otimes \cdots \otimes S(\widehat{X}_{L-1})_{t_{L-1},T}. \tag{22}$$

B Related Work, Experimental Choices, and Impact Statement

Continuous-time models. Since their introduction in [12], Neural ODEs were extended in various ways to facilitate modelling continuous time-series data [70, 60, 34, 40, 79]. While Neural ODEs and their extensions are successful in certain tasks they are burdened with a high computational cost, which makes them scale very poorly to long sequences in the time-series setting. Various authors propose methods and augmentations to vanilla Neural ODEs to decrease their computational overhead [24, 6]. Other approaches to augmenting deep learning methods to modelling continuous data include implicit neural representations [80, 29], continuous kernel convolutions [69], or Fourier neural operators [50, 63].

Transformers. First proposed in [86], the Transformer has been exceptionally successful in discrete sequence modelling tasks such as natural language processing (NLP). Key to the success of the Transformer in NLP is the attention mechanism, which extracts long-range dependencies. There are a number of extensions to improve efficiency and decrease the computation cost of the attention mechanism [49, 88, 22, 41, 16].

Signatures in machine learning. The path signature originates from theoretical stochastic analysis [51] and has since become a popular tool in machine learning. Path signatures are regarded as effective feature transformations for sequential data [64, 27, 44]. Additionally, signatures help mitigate the computational cost of Neural CDEs in long time-series [57] and non-Markovian stochastic control problems [39]. Other more recent works in this direction include [18, 87]. Approaches such as randomized signatures [21, 19] and the signature kernel [47, 77] have been developed to mitigate the curse of dimensionality inherent in path signature computations. Rough Transformers provide a first step towards incorporating path signatures for continuous-time sequence modelling using Transformers. [6]

We also note that contemporary work [84] employs a Transformer architecture with signature features for the task of deep hedging. However, our work differs in several key aspects. First, we introduce the multi-view attention mechanism, which uses signatures to extract both global and local information, which we found to be necessary in our experimentation, as Transformers are known to struggle in extracting local information (see Figure 7), whereas their work just uses a global signature. Moreover, their work computes the signature at every time step, strictly dilating input data. This is particularly problematic for long, multi-variate sequences, for reasons discussed above, and can actually negatively impact performance. Our work, however, *compresses* data using the multi-view signature transform, and we find that this compressed representation can actually improve performance. Finally, their work relies on the assumption that data is regularly sampled, as the signature is computed at every time step, in contrast to our work which is robust to irregular sampling.

---

[6]For a preliminary version of this paper, we also direct the reader to [56].

17

Long-Range Sequence modelling. A highly relevant line of research related to enhancing recurrent neural networks' capability to capture long-term dependencies involves the development of various models. These include Unitary RNNs [1], Orthogonal RNNs [38], expRNNs [48], chronoLSTM [82], antisymmetric RNNs [10], Lipschitz RNNs [25], coRNNs [71], unicoRNNs [72], LEMs [73], waveRNN [42], Linear Recurrent Units [62], and Structured State Space Models [32, 31]. While we utilize many benchmarks and synthetic tasks from these works to test our model, it is important to note that our work is not intended to compete with the state-of-the-art in these tasks. Therefore, we do not directly compare our model with the models mentioned above. Instead, this paper seeks to show that the baseline Transformer architecture can benefit from the use of signatures by (i) becoming more computationally efficient, (ii) being invariant to the sampling rate of the signal, and (iii) having a good inductive bias for temporal and spatial processing. Furthermore, we highlight that `RFormer` brings alternative benefits, such as the ability to perform spatial processing effectively, which is a setting in which long-range sequence models typically struggle.

Efficient Attention Variants. There are several efficient self-attention variants that have emerged over the years, including Sparse Transformer [14], Longformer [5], Linear Transformers [41], BigBird [90], Performer [16], or Diffuser [26]. In our setting, we highlight that a central part of this paper is to showcase how signatures significantly reduce the computational requirements of vanilla attention and empirically demonstrate that this also results in improved learning dynamics and invariance to the sampling frequency of the signal. Given the large efficiency gains that we observed with this approach when employed on vanilla attention, we did not consider that further experimentation on other forms of "approximate" attention was needed. Since most variants of attention seek to make the operation more efficient through several approximations (e.g., linearization or sparsification techniques), we believe that a first attempt at showcasing the power of multi-view signatures on vanilla attention is already significant. However, other variants of attention (such as the ones outlined before) could be added on top of the signature representations to obtain even better efficiency gains.

Limitations and Future Work. While we found RFormer to be very performant in our experiments, much of this performance gain relies on heavy hyperparameter tuning, especially when it comes to the choice of window sizes and signature level. However, this could be handled using Neural Architecture Search (NAS) techniques, such as those employed in [76]. Furthermore, despite the computational gains we achieve for low-dimensional sequences, additional work would be required to scale this method to larger dimensions. We should also note that the experiments and results presented in this paper are constrained by the relatively small scale of the models studied.

C   Experimental Details

All experiments are conducted on an NVIDIA GeForce RTX 3090 GPU with 24,564 MiB of memory, utilizing CUDA version 12.3. Hyperparameters used to produce the results in Table 2 are reported in Tables 6. The timings presented in all tables are obtained by executing each model independently for each dataset and averaging the resulting times across 100 epochs.

Table 6: Hyperparameters used for Table 2, where G and L refer to the Global and Local signature components, respectively.

	SCP1	SCP2	MI	EW	ETC	HB
Batch Size	20	10	50	5	10	20
Embedded Dim	10	5	20	20	20	5
Multi-View Terms	[G]	[G]	[L]	[L]	[G]	[G, L]
Learning Rate	4.08e-3	1.38e-3	4.08e-3	6.73e-3	1.00e-3	7.72e-3
Num. Heads	3	3	3	1	1	3
Num. Layers	2	3	3	2	1	3
Num. Sig Windows	100	50	200	10	400	30
Sig Level	2	3	2	2	1	2
Univariate	true	true	true	false	false	true
Num. Epoch	110	10	26	39	200	16

18

Table 7: Hyperparameters validation on remaining datasets.

Dataset	Learning Rate	Number of Windows	Sig. Depth	Sig. Type	Univariate/Multivariate Sig.
Sinusoidal	1×10^{-3}	75	6	Multi-View	-
HR	1×10^{-3}	75	4	Local	Multivariate

To prevent excessive growth in signature terms, we use the univariate signature in LOB datasets. As an alternative, one could employ randomized signatures [19] or low-rank approximations [9, 11] .

D   Baselines Validation

This section collects the validation of Step and Depth for the Neural-RDE model. Optimal values are selected for evaluation on test-set. Early-stopping is used with the same criteria as [57].

Table 8: Validation accuracy on the sinusoidal dataset.

Acc. Val	Step	Depth	Memory Usage (Mb)	Elapsed Time (s)
17.26	2	2	778.9	6912.7
12.21	2	3	770.3	1194.43
16.35	4	2	382.2	2702.48
19.27	4	3	386.16	574.97
20.99	8	2	193	1321.36
24.02	**8**	**3**	**194.17**	**332.17**
17.15	16	2	97.13	136.43
21.59	16	3	98.17	156.93
17.46	24	2	65.96	105.94
20.59	24	3	66.68	98.97

Table 9: Validation accuracy on the long sinusoidal dataset.

Acc. Val	Step	Depth	Memory Usage (Mb)	Elapsed Time (s)
11.10	2	2	4017.22	2961.98
9.59	2	3	4008.33	2779.52
10.39	4	2	2001.76	1677.78
10.19	4	3	2006.80	1615.64
14.03	8	2	1004.07	665.55
15.34	**8**	**3**	**1005.72**	**723.41**
1.61	16	2	503.66	125.85
1.92	16	3	505.28	120.63
1.51	24	2	339.80	58.87
2.12	24	3	341.90	69.35

19

Table 10: Validation accuracy on the EW dataset.

Acc. Val	Step	Depth	Memory Usage (Mb)	Elapsed Time (s)
84.62	2	2	5799.40	21289.99
87.18	**2**	**3**	**6484.93**	**25925.80**
79.49	4	2	2891.61	11449.14
82.05	4	3	3240.99	9055.12
82.05	8	2	1446.94	4143.26
76.92	8	3	1624.73	3616.43
82.05	16	2	724.35	1909.69
76.92	16	3	817.04	1924.27
79.49	24	2	483.92	1098.21
74.36	24	3	543.78	987.02

Table 11: Validation loss on the HR dataset.

Acc. Val	Step	Depth	Memory Usage (Mb)	Elapsed Time (s)
2.44	**2**	**2**	**5044.44**	**56492.33**
3.03	2	3	5059.28	39855.19
3.67	4	2	2515.40	10765.58
16.04	4	3	2531.44	7157.20
5.35	8	2	1259.30	3723.94
2.70	8	3	1268.60	18682.82
3.58	16	2	632.08	3518.96
3.64	16	3	636.64	7922.96
3.86	24	2	422.74	3710.95
3.55	24	3	426.83	6567.02

Table 12: Validation loss on the LOB dataset (1K), included as an additional experiment in Appendix **??**.

Val Loss	Step	Depth	Memory Usage (Mb)	Elapsed Time (s)
0.58	**2**	**2**	**1253.55**	**180.79**
1.74	2	3	1447.57	308.52
1.58	4	2	623.87	71.18
32.90	4	3	754.05	87.81
2.94	8	2	317.40	61.27
4.84	8	3	406.88	62.71
2.24	16	2	164.70	18.67
6.26	16	3	234.92	24.20
3.82	24	2	112.80	12.69
15.35	24	3	176.68	14.92

E Long Temporal Datasets Details

Table 13 summarises the long temporal modeling datasets from the UEA time series classification archive [3] used in Section 4.

20

Table 13: Summary of datasets used in the long time-series classification task.

Dataset	#Sequences	Length	#Classes	#Dimensions
SelfRegulationSCP1 (SCP1)	561	896	2	6
SelfRegulationSCP2 (SCP2)	380	1152	2	7
MotorImagery (MI)	378	3000	2	64
EigenWorms (EW)	259	17984	5	6
EthanolConcentration (ETC)	524	1751	4	3

F Ablation Studies

F.1 Global and Local Signature Components

In this section, we ablate the use of the multi-view signature transform over both global and local transformations of the input signal. The results for the sinusoidal datasets are shown in Figure 7. In most cases, the use of both local and global components improves the performance of RFormer. This choice, however, can be seen as a hyperparameter and will be dataset-dependent.
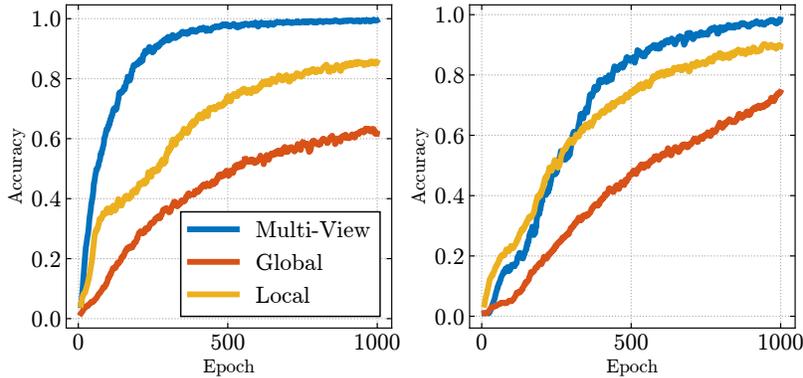
Figure 7: Ablation of local and local components of the multi-view signature for the sinusoidal datasets. **Left:** Sinusoidal dataset. **Right:** Long Sinusoidal dataset.

F.2 Signature Level and Naive Downsampling

One of the main points of the paper is that the shorter representation of the time-series endowed by the signatures helps to significantly reduce the computational cost of the self-attention operation with minimal information loss (and with improved performance in many of the experiments). By equation (16), one sees that the first level of the signature of a linear function is the difference between its endpoints. Hence, using multi-view attention with signature level one operates on the increments of piecewise-linear interpolated data, which corresponds to naive downsampling. To test that higher levels of the signature provide improvements in performance, we compare the result of using the signature on the datasets tested in Table 14 below.

Table 14: Comparative performance of different methods on datasets.

Dataset	Linear-Interpolation + Vanilla	Rough Transformer with sig level (n)	Improvement
EigenWorms	64.10%	90.24% (2)	40.77%
HR	10.56	2.66 (4)	74.81%

There is a significant performance gain in considering higher levels of the signature because one can capture the higher-order interactions between the different time-series.

21

G   Additional Experiments and Comparisons

G.1   Random Drop Experiments

Furthermore, we conduct a new set of experiments in which we dropped 30% and 70% of the dataset for RFormer. Note that even with a 70% drop rate in the EigenWorms dataset, the vanilla Transformer fails to run due to memory limitations. Therefore, to provide results for the Transformer model on the EigenWorms dataset, we conduct experiments with an 85% drop rate. This comparison highlights the performance gap between the vanilla Transformer and our proposed model under these conditions, with the RFormer model yielding superior results. All results are computed across five seeds and are summarized in the tables and figure below.

Table 15: Performance of models under various data drop scenarios for EW dataset.

Model	Full	30% Drop	50% Drop	70% Drop	85% Drop
Transformer	OOM	OOM	OOM	OOM	$72.45\% \pm 3.36$
RFormer	$90.24\% \pm 2.15$	$87.86\% \pm 3.28$	$87.69\% \pm 4.97$	$83.35\% \pm 2.86$	$82.74\% \pm 2.13$

Table 16: Performance consistency of RFormer under data drop scenarios for HR dataset.

Model	Full	30% Drop	50% Drop	70% Drop
RFormer	2.66 ± 0.21	2.72 ± 0.19	2.82 ± 0.05	2.98 ± 0.08

Table 17: Epoch-wise performance under different data drop scenarios for the sinusoidal dataset.

	Epoch 100	Epoch 250	Epoch 500	Epoch 1000
30% Drop	48.6%	82.5%	91.4%	99.3%
70% Drop	35.7%	56.8%	64.9%	67.8%

Table 18: Epoch-wise performance under different data drop scenarios for the long sinusoidal dataset.

	Epoch 100	Epoch 250	Epoch 500	Epoch 1000
30% Drop	39.1%	72.6%	96.2%	98.2%
70% Drop	27.5%	66.7%	78.5%	85.3%
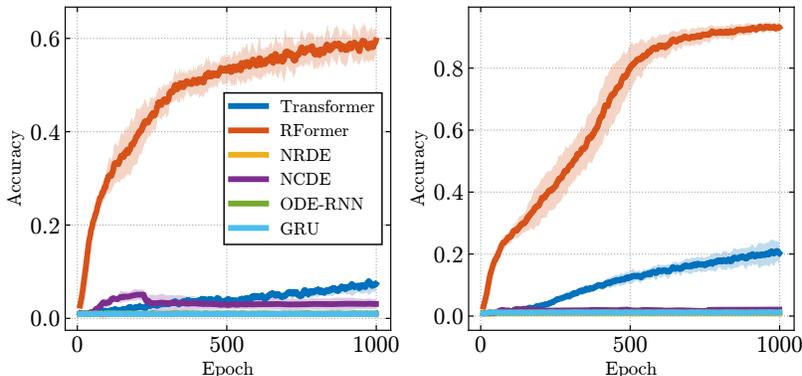
Figure 8: Test accuracy per epoch for the frequency classification task across three random seeds for sinusoidal datasets with 50% random drop per epoch. **Left:** Sinusoidal dataset. **Right:** Long Sinusoidal dataset.

22

Finally, Table 19 compares CRU and RFormer in an irregularly sampled synthetic data setting, featuring shorter sinusoids and fewer classes than the experiments in Section 4.1. Additionally, Table 20 presents the hyperparameter validation for CRU (see Table 21 for training time analysis). These experiments demonstrate that recurrent models perform well with short sequences. Note that despite RFormer's superior performance, our model is significantly faster than other continuous-time models, as shown in Appendix G.2, particularly in Table 21.

Table 19: Comparison of RFormer and CRU (two best and simplest performing instances [Num.basis/Bandwidth= 20/3]) at different random drop percentages.

L	Random Drop	RFormer	CRU (LSD=10)	CRU (LSD=20)
	0%	100.00%	100%	100.00%
	30%	98.60%	65.90%	99.60%
100	50%	97.80%	34.40%	94.70%
	70%	96.10%	43.00%	78.60%
	85%	85.50%	32.30%	57.30%
	0%	100.00%	100.00%	100%
	30%	99.90%	42.95%	94.90%
250	50%	99.40%	43.65%	77.30%
	70%	98.30%	45.40%	94.40%
	85%	86.20%	38.80%	83.60%
	0%	100.00%	100.00%	OOM
	30%	99.90%	47.15%	OOM
500	50%	99.70%	48.80%	OOM
	70%	99.30%	55.15%	OOM
	85%	87.70%	46.50%	OOM

Table 20: CRU's hyperparameters ($L = 100$) (latent state dimension (LSD), number of basis matrices (Num.basis), and their bandwidth).

LSD	Num. basis	Bandwidth	Acc (30 Epochs)
	15	3	78%
10	15	10	-
	20	3	100%
	20	10	-
	15	3	81.30%
20	15	10	91.70%
	20	3	100%
	20	10	99.90%
	15	3	99.90%
40	15	10	97.50%
	20	3	100%
	20	10	100%

G.2 Additional Efficiency Experiments and Discussion

We conduct additional experiments to compare the runtime of Rough Transformers with other models. In this experiment, we use the synthetic sinusoidal dataset considered in our paper and compute the runtime per epoch for varying sequence lengths. We demonstrate results for two variants of RFormer: "online", which corresponds to computing the signatures of each batch during training (resulting in significant redundant computation), and "offline", which corresponds to computing the signatures in one go at the beginning of training. We include a recent RNN-based model as a basis for comparison with high-performing RNN baselines. In addition to the models discussed in Section 4, we introduce Continuous Recurrent Units (CRU) [78] as a new baseline. See Table 21 for a summary of the results.

23

Table 21: Seconds per epoch for growing input length and for different model types on the sinusoidal dataset.

Model	S/E for Varying Context Length ↓							
	L=100	**L=250**	**L=500**	**L=1000**	**L=2500**	**L=5000**	**L=7.5k**	**L=10k**
NRDE	5.87	11.67	20.27	44.01	103.11	201.21	312.31	467.47
NCDE	42.59	121.82	225.14	458.09	1126.77	2813.42	4199.50	5345.39
GRU	1.56	1.55	1.65	1.63	1.78	2.37	3.65	4.79
CRU	59.22	199.15	789.28	OOM	OOM	OOM	OOM	OOM
ContiFormer	61.36	248.31	1165.02	OOM	OOM	OOM	OOM	OOM
Transformer	0.75	0.79	0.82	0.95	1.36	5.31	9.32	16.32
RFormer (Online)	0.75	0.88	0.94	1.03	1.28	1.55	1.83	2.35
RFormer (Offline)	0.67	0.64	0.63	0.65	0.60	0.59	0.62	0.60

We remark that previous running times are obtained with a batch size of 10. Further, the `ContiFormer` model could be run for $L = 1000$ if decreasing the batch size to 2 (which significantly affects the parallelization process), avoiding OOM issues and resulting in 4025 seconds/epoch, which is several orders of magnitude larger than `RFormer`. As an additional experiment, we tested the epoch time (S/E) of `RFormer` for extremely oversampled sinusoidal time series. We show our results in the table below.

Table 22: Seconds per epoch for very large input length.

Model	S/E for Varying Context Length ↓			
	L=25k	**L=50k**	**L=100k**	**L=250k**
RFormer (Online)	5.39	9.06	19.95	45.20
RFormer (Offline)	0.60	0.61	0.60	0.63

Thus, the time needed to compute the signature is inconsequential when compared with the time required to train standard models on the full or even downsampled datasets, since this step has to be carried out only once. To put this into context with an example, we note that it takes 4s to compute the signature representations for the HR dataset (which is about half the time it takes for the Vanilla Transformer to go through one epoch) and results in a 26× increase in computational speed for RFormer when compared to the vanilla Transformer.

Table 23: Processing times for different sizes on the sinusoidal dataset.

Size	**100**	**250**	**500**	**1k**	**2.5k**	**5k**	**7.5k**	**10k**	**25k**	**50k**	**75k**	**100k**
Time	0.15 s	0.21 s	0.24 s	0.39 s	0.42 s	0.51 s	0.70 s	1.09 s	1.64 s	2.94 s	4.49 s	5.74 s

To showcase that this is the case for not only sequences of moderate length but also extremely long sequences, we also carry out the following experiment where we compute the signature representation for the sine dataset, with a progressively increasing number of datapoints. As seen in Table 23, this does not cause an explosion in computational time.

24
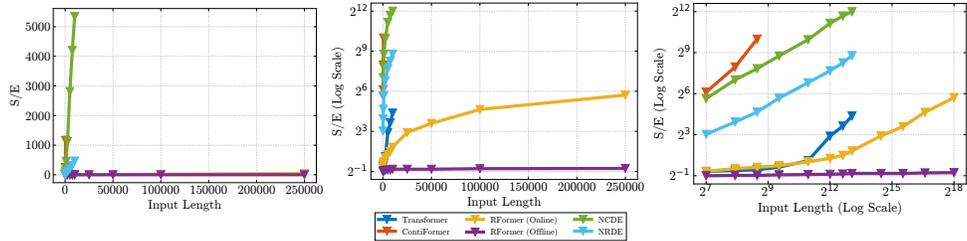
Figure 9: Seconds per epoch for growing input length and for different model types on the sinusoidal dataset for extremely long lengths (up to 250k) **Left:** Log Scale. **Middle:** Regular Scale. **Right:** Log-log scale. When a line stops, it indicates an OOM error.

G.3 Additional ContiFormer Comparisons

Also, to provide some context of the performance of `ContiFormer` compared with our method (and not only results on complexity and training times), we run the model on the sinusoidal classification task for signals of length $L = 100$ and $L = 250$. Due to the slow running time of the `ContiFormer` model, we did not consider sequence lengths of $L > 250$. We evaluate the `ContiFormer` model using one head. However, given the subpar results we obtain, we also test it with four heads, using the hyperparameters originally used in the paper for their irregularly sampled time series classification experiments. By contrast, all variations of `RFormer` tested in this paper for this experiment employ only one head, but reported significantly better results.

Table 24: Model performance for $L = 100$.

Model	Epoch 100	Epoch 250	Epoch 500
ContiFormer (1 Head)	2.3%	2.8%	3.1%
ContiFormer (4 Heads)	8.5%	17.3%	20.0%
Transformer (1 Head)	13.7%	40.1%	82.8%
RFormer (1 Head)	38.7%	81.1%	92.3%

25