Connecting Weighted Automata and Recurrent Neural Networks through Spectral Learning

(Supplementary Material)

A Proofs

A.1 Proof of Theorem 2

Theorem. *Any function that can be computed by a vv-WFA with n states can be computed by a linear 2-RNN with n hidden units. Conversely, any function that can be computed by a linear 2-RNN with n hidden units on sequences of one-hot vectors (i.e. canonical basis vectors) can be computed by a WFA with n states.*

More precisely, the WFA $A = (\boldsymbol{\alpha}, \{\mathbf{A}^\sigma\}_{\sigma\in\Sigma}, \boldsymbol{\Omega})$ with n states and the linear 2-RNN $M = (\boldsymbol{\alpha}, \boldsymbol{\mathcal{A}}, \boldsymbol{\Omega})$ with n hidden units, where $\boldsymbol{\mathcal{A}} \in \mathbb{R}^{n\times\Sigma\times n}$ is defined by $\boldsymbol{\mathcal{A}}_{:,\sigma,:} = \mathbf{A}^\sigma$ for all $\sigma \in \Sigma$, are such that $f_A(\sigma_1\sigma_2\cdots\sigma_k) = f_M(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k)$ for all sequences of input symbols $\sigma_1, \cdots, \sigma_k \in \Sigma$, where for each $i \in [k]$ the input vector $\mathbf{x}_i \in \mathbb{R}^\Sigma$ is the one-hot encoding of the symbol σ_i.

Proof. We first show by induction on k that, for any sequence $\sigma_1\cdots\sigma_k \in \Sigma^*$, the hidden state $\mathbf{h}_k$ computed by M (see Eq. (1)) on the corresponding one-hot encoded sequence $\mathbf{x}_1, \cdots, \mathbf{x}_k \in \mathbb{R}^d$ satisfies $\mathbf{h}_k = (\mathbf{A}^{\sigma_1}\cdots\mathbf{A}^{\sigma_k})^\top\boldsymbol{\alpha}$. The case $k = 0$ is immediate. Suppose the result true for sequences of length up to k. One can check easily check that $\boldsymbol{\mathcal{A}} \bullet_2 \mathbf{x}_i = \mathbf{A}^{\sigma_i}$ for any index i. Using the induction hypothesis it then follows that

$$\begin{aligned}\mathbf{h}_{k+1} &= \boldsymbol{\mathcal{A}} \bullet_1 \mathbf{h}_k \bullet_2 \mathbf{x}_{k+1} = \mathbf{A}^{\sigma_{k+1}} \bullet_1 \mathbf{h}_k = (\mathbf{A}^{\sigma_{k+1}})^\top \mathbf{h}_k \\ &= (\mathbf{A}^{\sigma_{k+1}})^\top (\mathbf{A}^{\sigma_1}\cdots\mathbf{A}^{\sigma_k})^\top\boldsymbol{\alpha} = (\mathbf{A}^{\sigma_1}\cdots\mathbf{A}^{\sigma_{k+1}})^\top\boldsymbol{\alpha}.\end{aligned}$$

To conclude, we thus have

$$f_M(\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_k) = \boldsymbol{\Omega}\mathbf{h}_k = \boldsymbol{\Omega}(\mathbf{A}^{\sigma_1}\cdots\mathbf{A}^{\sigma_k})^\top\boldsymbol{\alpha} = f_A(\sigma_1\sigma_2\cdots\sigma_k). \qquad \square$$

A.2 Proof of Theorem 3

Theorem. *Let $f : (\mathbb{R}^d)^* \to \mathbb{R}^p$ be a function computed by a minimal linear 2-RNN with n hidden units and let L be an integer such that $\operatorname{rank}((\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L, L+1\rangle\rangle}) = n$.*

Then, for any $\mathbf{P} \in \mathbb{R}^{d^L\times n}$ and $\mathbf{S} \in \mathbb{R}^{n\times d^L p}$ such that $(\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L,L+1\rangle\rangle} = \mathbf{PS}$, the linear 2-RNN $M = (\boldsymbol{\alpha}, \boldsymbol{\mathcal{A}}, \boldsymbol{\Omega})$ defined by

$$\boldsymbol{\alpha} = (\mathbf{S}^\dagger)^\top (\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L+1\rangle\rangle}, \qquad \boldsymbol{\mathcal{A}} = ((\boldsymbol{\mathcal{H}}_f^{(2L+1)})_{\langle\langle L,1,L+1\rangle\rangle}) \times_1 \mathbf{P}^\dagger \times_3 (\mathbf{S}^\dagger)^\top, \qquad \boldsymbol{\Omega}^\top = \mathbf{P}^\dagger (\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L,1\rangle\rangle}$$

is a minimal linear 2-RNN computing f.

Proof. Let $\mathbf{P} \in \mathbb{R}^{d^L\times n}$ and $\mathbf{S} \in \mathbb{R}^{n\times d^L p}$ be such that $(\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L,L+1\rangle\rangle} = \mathbf{PS}$ Define the tensors

$$\boldsymbol{\mathcal{P}}^* = [\![\boldsymbol{\mathcal{A}}^\star \bullet_1 \boldsymbol{\alpha}^\star, \underbrace{\boldsymbol{\mathcal{A}}^\star, \cdots, \boldsymbol{\mathcal{A}}^\star}_{L-1 \text{ times}}, \mathbf{I}_n]\!] \in \mathbb{R}^{d\times\cdots\times d\times n} \qquad \text{and} \qquad \boldsymbol{\mathcal{S}}^* = [\![\mathbf{I}_n, \underbrace{\boldsymbol{\mathcal{A}}^\star, \cdots, \boldsymbol{\mathcal{A}}^\star}_{L \text{ times}}, \boldsymbol{\Omega}^\star]\!] \in \mathbb{R}^{n\times d\times\cdots\times d\times p}$$

of order $L+1$ and $L+2$ respectively, and let $\mathbf{P}^\star = (\boldsymbol{\mathcal{P}}^*)_{\langle\langle l,1\rangle\rangle} \in \mathbb{R}^{d^l\times n}$ and $\mathbf{S} = (\boldsymbol{\mathcal{S}}^*)_{\langle\langle 1,L+1\rangle\rangle} \in \mathbb{R}^{n\times d^l p}$. Using the identity $\boldsymbol{\mathcal{H}}_f^{(j)} = [\![\boldsymbol{\mathcal{A}} \bullet_1 \boldsymbol{\alpha}, \underbrace{\boldsymbol{\mathcal{A}}, \cdots, \boldsymbol{\mathcal{A}}}_{j-1 \text{ times}}, \boldsymbol{\Omega}^\top]\!]$ for any j, one can easily check the following identities:

$$(\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L,L+1\rangle\rangle} = \mathbf{P}^\star\mathbf{S}^\star, \qquad (\boldsymbol{\mathcal{H}}_f^{(2L+1)})_{\langle\langle L,1,L+1\rangle\rangle} = \boldsymbol{\mathcal{A}}^\star \times_1 \mathbf{P}^\star \times_3 (\mathbf{S}^\star)^\top,$$
$$(\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L,1\rangle\rangle} = \mathbf{P}^\star(\boldsymbol{\Omega}^\star)^\top, \qquad (\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L+1\rangle\rangle} = (\mathbf{S}^\star)^\top\boldsymbol{\alpha}.$$

Let $\mathbf{M} = \mathbf{P}^\dagger \mathbf{P}^\star$. We will show that $\boldsymbol{\alpha} = \mathbf{M}^{-\top}\boldsymbol{\alpha}^\star$, $\boldsymbol{\mathcal{A}} = \boldsymbol{\mathcal{A}}^\star \times_1 \mathbf{M} \times_3 \mathbf{M}^{-\top}$ and $\boldsymbol{\Omega} = \mathbf{M}\boldsymbol{\Omega}^\star$, which will entail the results since linear 2-RNN are invariant under change of basis (see Section 2). First observe that $\mathbf{M}^{-1} = \mathbf{S}^\star \mathbf{S}^\dagger$. Indeed, we have $\mathbf{P}^\dagger \mathbf{P}^\star \mathbf{S}^\star \mathbf{S}^\dagger = \mathbf{P}^\dagger (\boldsymbol{\mathcal{H}}_f^{(2l)})_{\langle\langle l, l+1\rangle\rangle} \mathbf{S}^\dagger = \mathbf{P}^\dagger \mathbf{P} \mathbf{S} \mathbf{S}^\dagger = \mathbf{I}$ where we used the fact that $\mathbf{P}$ (resp. $\mathbf{S}$) is of full column rank (resp. row rank) for the last equality.

The following derivations then follow from basic tensor algebra:

$$\boldsymbol{\alpha} = (\mathbf{S}^\dagger)^\top (\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L+1\rangle\rangle} = (\mathbf{S}^\dagger)^\top (\mathbf{S}^\star)^\top \boldsymbol{\alpha} = (\mathbf{S}^\star \mathbf{S}^\dagger)^\top = \mathbf{M}^{-\top}\boldsymbol{\alpha}^\star,$$

$$\begin{aligned}
\boldsymbol{\mathcal{A}} &= ((\boldsymbol{\mathcal{H}}_f^{(2L+1)})_{\langle\langle L,1,L+1\rangle\rangle}) \times_1 \mathbf{P}^\dagger \times_3 (\mathbf{S}^\dagger)^\top \\
&= (\boldsymbol{\mathcal{A}}^\star \times_1 \mathbf{P}^\star \times_3 (\mathbf{S}^\star)^\top) \times_1 \mathbf{P}^\dagger \times_3 (\mathbf{S}^\dagger)^\top \\
&= \boldsymbol{\mathcal{A}}^\star \times_1 \mathbf{P}^\dagger \mathbf{P}^\star \times_3 (\mathbf{S}^\star \mathbf{S}^\dagger)^\top = \boldsymbol{\mathcal{A}}^\star \times_1 \mathbf{M} \times_3 \mathbf{M}^{-\top},
\end{aligned}$$

$$\boldsymbol{\Omega}^\top = \mathbf{P}^\dagger (\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L,1\rangle\rangle} = \mathbf{P}^\dagger \mathbf{P}^\star (\boldsymbol{\Omega}^\star)^\top = \mathbf{M}\boldsymbol{\Omega}^\star,$$

which concludes the proof. □

A.3 Proof of Theorem 4

Theorem. *Let $(\mathbf{h}_0, \boldsymbol{\mathcal{A}}, \boldsymbol{\Omega})$ be a minimal linear 2-RNN with n hidden units computing a function $f : (\mathbb{R}^d)^* \to \mathbb{R}^p$, and let L be an integer*[7] *such that* $\operatorname{rank}((\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L, L+1\rangle\rangle}) = n$.

Suppose we have access to 3 datasets $D_l = \{((\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_l^{(i)}), \mathbf{y}^{(i)})\}_{i=1}^{N_l} \subset (\mathbb{R}^d)^l \times \mathbb{R}^p$ for $l \in \{L, 2L, 2L+1\}$ where the entries of each $\mathbf{x}_j^{(i)}$ are drawn independently from the standard normal distribution and where each $\mathbf{y}^{(i)} = f(\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_l^{(i)})$.

Then, whenever $N_l \geq d^l$ for each $l \in \{L, 2L, 2L+1\}$, the linear 2-RNN M returned by Algorithm 1 with the least-squares method satisfies $f_M = f$ with probability one.

Proof. We just need to show for each $l \in \{L, 2L, 2L+1\}$ that, under the hypothesis of the Theorem, the Hankel tensors $\hat{\boldsymbol{\mathcal{H}}}^{(l)}$ computed in line 4 of Algorithm 1 are equal to the true Hankel tensors $\boldsymbol{\mathcal{H}}^{(l)}$ with probability one. Recall that these tensors are computed by solving the least-squares problem

$$\hat{\boldsymbol{\mathcal{H}}}^{(l)} = \operatorname*{arg\,min}_{T \in \mathbb{R}^{d \times \cdots \times d \times p}} \|\mathbf{X}(\boldsymbol{\mathcal{T}})_{\langle\langle l,1\rangle\rangle} - \mathbf{Y}\|_F^2$$

where $\mathbf{X} \in \mathbb{R}^{N_l \times d_l}$ is the matrix with rows $\mathbf{x}_1^{(i)} \otimes \mathbf{x}_2^{(i)} \otimes \cdots \otimes \mathbf{x}_l^{(i)}$ for each $i \in [N_l]$. Since $\mathbf{X}(\boldsymbol{\mathcal{H}}^{(l)})_{\langle\langle l,1\rangle\rangle} = \mathbf{Y}$ and since the solution of the least-squares problem is unique as soon as $\mathbf{X}$ is of full column rank, we just need to show that this is the case with probability one when the entries of the vectors $\mathbf{x}_j^{(i)}$ are drawn at random from a standard normal distribution. The result will then directly follow by applying Theorem 3.

We will show that the set

$$\mathcal{S} = \{(\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_l^{(i)}) \mid \ i \in [N_l],\ dim(span(\{\mathbf{x}_1^{(i)} \otimes \mathbf{x}_2^{(i)} \otimes \cdots \otimes \mathbf{x}_l^{(i)}\})) < d^l\}$$

has Lebesgue measure 0 in $((\mathbb{R}^d)^l)^{N_l} \simeq \mathbb{R}^{dlN_l}$ as soon as $N_l \geq d^l$, which will imply that it has probability 0 under any continuous probability, hence the result. For any $S = \{(\mathbf{x}_1^{(i)}, \cdots, \mathbf{x}_l^{(i)})\}_{i=1}^{N_l}$, we denote by $\mathbf{X}_S \in \mathbb{R}^{N_l \times d^l}$ the matrix with rows $\mathbf{x}_1^{(i)} \otimes \mathbf{x}_2^{(i)} \otimes \cdots \otimes \mathbf{x}_l^{(i)}$. One can easily check that $S \in \mathcal{S}$ if and only if $\mathbf{X}_S$ is of rank strictly less than d^l, which is equivalent to the determinant of $\mathbf{X}_S^\top \mathbf{X}_S$ being equal to 0. Since this determinant is a polynomial in the entries of the vectors $\mathbf{x}_j^{(i)}$, $\mathcal{S}$ is an algebraic subvariety of $\mathbb{R}^{dlN_l}$. It is then easy to check that the polynomial $det(\mathbf{X}_S^\top \mathbf{X}_S)$ is not uniformly 0 when $N_l \geq d^l$. Indeed, it suffices to choose the vectors $\mathbf{x}_j^{(i)}$ such

[7] Note that the theorem can be adapted if such an integer L does not exists (see supplementary material).

that the family $(\mathbf{x}_1^{(i)} \otimes \mathbf{x}_2^{(i)} \otimes \cdots \otimes \mathbf{x}_l^{(i)})_{n=1}^{N_l}$ spans the whole space $\mathbb{R}^{d^l}$ (which is possible since we can choose arbitrarily any of the $N_l \geq d^l$ elements of this family), hence the result. In conclusion, $\mathcal{S}$ is a proper algebraic subvariety of $\mathbb{R}^{dlN_l}$ and hence has Lebesgue measure zero [14, Section 2.6.5].

□

B Lifting the simplifying assumption

We now show how all our results still hold when there does not exist an L such that $\text{rank}((\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L,L+1\rangle\rangle}) = n$. Recall that this simplifying assumption followed from assuming that the sets $\mathcal{P} = \mathcal{S} = [d]^L$ form a complete basis for the function $\tilde{f} : [d]^* \to \mathbb{R}^p$ defined by $\tilde{f}(i_1 i_2 \cdots i_k) = f(\mathbf{e}_{i_1}, \mathbf{e}_{i_2}, \cdots, \mathbf{e}_{i_k})$. We first show that there always exists an integer L such that $\mathcal{P} = \mathcal{S} = \cup_{i \leq L}[d]^i$ forms a complete basis for $\tilde{f}$. Let $M = (\boldsymbol{\alpha}^\star, \boldsymbol{\mathcal{A}}^\star, \boldsymbol{\Omega}^\star)$ be a linear 2-RNN with n hidden units computing f (i.e. such that $f_M = f$). It follows from Theorem 2 and from the discussion at the beginning of Section 4.1 that there exists a vv-WFA computing $\tilde{f}$ and it is easy to check that $\text{rank}(\tilde{f}) = n$. This implies $\text{rank}((\boldsymbol{\mathcal{H}}_f)_{(1)}) = n$ by Theorem 1. Since $\mathcal{P} = \mathcal{S} = \cup_{i \leq l}[d]^i$ converges to $[d]^*$ as l grows to infinity, there exists an L such that the finite sub-block $\tilde{\boldsymbol{\mathcal{H}}}_f \in \mathbb{R}^{\mathcal{P} \times \mathcal{S} \times p}$ of $\boldsymbol{\mathcal{H}}_f \in \mathbb{R}^{[d]^* \times [d]^* \times p}$ satisfies $\text{rank}((\tilde{\boldsymbol{\mathcal{H}}}_f)_{(1)}) = n$, i.e. such that $\mathcal{P} = \mathcal{S} = \cup_{i \leq L}[d]^i$ forms a complete basis for $\tilde{f}$.

Now consider the finite sub-blocks $\tilde{\boldsymbol{\mathcal{H}}}_f^+ \in \mathbb{R}^{\mathcal{P} \times [d] \times \mathcal{S} \times p}$ and $\tilde{\mathbf{H}}_f^- \in \mathbb{R}^{\mathcal{P} \times p}$ of $\boldsymbol{\mathcal{H}}_f$ defined by

$$(\tilde{\boldsymbol{\mathcal{H}}}_f^+)_{u,i,v,:} = \tilde{f}(uiv), \quad \text{and} (\tilde{\mathbf{H}}_f^-)_{u,:} = f(u)$$

for any $u \in \mathcal{P} = \mathcal{S}$ and any $i \in [d]$. One can check that Theorem 3 holds by replacing *mutatis mutandi* $(\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L,L+1\rangle\rangle}$ by $(\tilde{\boldsymbol{\mathcal{H}}}_f)_{(1)}$, $(\boldsymbol{\mathcal{H}}_f^{(2L+1)})_{\langle\langle L,1,L+1\rangle\rangle}$ by $\tilde{\boldsymbol{\mathcal{H}}}_f^+$, $(\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L,1\rangle\rangle}$ by $\tilde{\mathbf{H}}_f^-$ and $(\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L+1\rangle\rangle}$ by $\text{vec}(\tilde{\mathbf{H}}_f^-)$.

To conclude, it suffices to observe that both $\tilde{\boldsymbol{\mathcal{H}}}_f^+$ and $\tilde{\mathbf{H}}_f^-$ can be constructed from the entries for the tensors $\boldsymbol{\mathcal{H}}^{(l)}$ for $1 \leq l \leq 2L+1$, which can be recovered (or estimated in the noisy setting) using the techniques described in Section 4.2 (corresponding to lines 2-12 of Algorithm 1).

We thus showed that linear 2-RNNs can be provably learned even when there does not exist an L such that $\text{rank}((\boldsymbol{\mathcal{H}}_f^{(2L)})_{\langle\langle L,L+1\rangle\rangle}) = n$. In this setting, one needs to estimate enough of the tensors $\boldsymbol{\mathcal{H}}^{(l)}$ to reconstruct a complete sub-block $\tilde{\boldsymbol{\mathcal{H}}}_f$ of the Hankel tensor $\boldsymbol{\mathcal{H}}$ (along with the corresponding tensor $\tilde{\boldsymbol{\mathcal{H}}}_f^+$ and matrix $\tilde{\mathbf{H}}_f^-$) and recover the linear 2-RNN by applying Theorem 3. In addition, one needs to have access to sufficiently large datasets D_l for each $l \in [2L+1]$ rather than only the three datasets mentioned in Theorem 4. However the data requirement remains the same in the case where we assume that each of the datasets D_l is constructed from a unique training dataset $S = \{((\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_T^{(i)}), (\mathbf{y}_1^{(i)}, \mathbf{y}_2^{(i)}, \cdots, \mathbf{y}_T^{(i)}))\}_{i=1}^N$ of input/output sequences.

C Leveraging the tensor train structure for computational efficiency

The overall learning algorithm using the TIHT recovery method in TT format is summarized in Algorithm 2. The key ingredients to improve the complexity of Algorithm 1 are (i) to estimate the gradient using mini-batches of data and (ii) to directly use the TT format to represent and perform operations on the tensors $\boldsymbol{\mathcal{H}}^{(l)}$ and the tensors $\boldsymbol{\mathcal{X}}^{(l)} \in \mathbb{R}^{M \times d \times \cdots \times d}$ defined by

$$\boldsymbol{\mathcal{X}}_{i,:,\cdots,:} = \mathbf{x}_1^{(i)} \otimes \mathbf{x}_2^{(i)} \otimes \cdots \otimes \mathbf{x}_l^{(i)} \quad \text{for } i \in [M] \tag{3}$$

where M is the size of a mini-batch of training data ($\boldsymbol{\mathcal{H}}^{(l)}$ is of TT-rank R by design and it can easily be shown that $\boldsymbol{\mathcal{X}}^{(l)}$ is of TT-rank at most M, cf. Eq. (4)). Then, all the operations of the algorithm can be expressed in terms of these tensors and performed efficiently in TT format. More precisely, the products and sums needed to compute the gradient update on line 6 can be performed in $\mathcal{O}\left((R+M)^2(ld+p) + (R+M)^3 d\right)$. After the gradient update, the tensor $\boldsymbol{\mathcal{H}}^{(l)}$ has TT-rank at most $(M+R)$ but can be efficiently projected back to a tensor of TT-rank R using the tensor train rounding operation [33] in $\mathcal{O}\left((R+M)^3(ld+p)\right)$ (which is the operation dominating the complexity of the whole algorithm). The subsequent operations on line 10 can be performed efficiently in the TT format in $\mathcal{O}\left(R^3 d + R^2 p\right)$ (using the method described in [26] to compute the pseudo-inverses of the matrices $\mathbf{P}$ and $\mathbf{S}$). The overall complexity of Algorithm 2 is thus in $\mathcal{O}\left(T(R+M)^3(Ld+p)\right)$ where T is the number of iterations of the inner loop.

Algorithm 2 `2RNN-SL-TT`: Spectral Learning of linear 2-RNNs **in tensor train format**

Input: Three training datasets D_L, D_{2L}, D_{2L+1} with input sequences of length L, $2L$ and $2L+1$ respectively, rank R, learning rate γ and mini-batch size M.

1: **for** $l \in \{L, 2L, 2L+1\}$ **do**

2: Initialize all cores of the rank R TT-decomposition $\boldsymbol{\mathcal{H}}^{(l)} = [\![\boldsymbol{\mathcal{G}}_1^{(l)}, \cdots, \boldsymbol{\mathcal{G}}_{l+1}^{(l)}]\!] \in \mathbb{R}^{d \times \cdots \times d \times p}$ to $\mathbf{0}$.

// Note that all the updates of $\boldsymbol{\mathcal{H}}^{(l)}$ stated below are in effect applied directly to the core tensors $\boldsymbol{\mathcal{G}}_k^{(l)}$, i.e. the tensor $\boldsymbol{\mathcal{H}}^{(l)}$ is never explicitely constructed.

3: **repeat**

4: Subsample a minibatch

$$\{((\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \cdots, \mathbf{x}_l^{(i)}), \mathbf{y}^{(i)})\}_{i=1}^{M} \subset (\mathbb{R}^d)^l \times \mathbb{R}^p$$

of size M from D_l.

5: Compute the rank M TT-decomposition of the tensor $\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{X}}^{(l)}$ (defined in Eq. (3)), which is given by

$$\boldsymbol{\mathcal{X}} = [\![\mathbf{I}_M, \boldsymbol{\mathcal{A}}_1, \cdots, \boldsymbol{\mathcal{A}}_l]\!] \text{ where the cores are defined by } (\boldsymbol{\mathcal{A}}_k)_{i,:,j} = \delta_{ij}\mathbf{x}_k^{(i)} \quad \text{and} \quad (\boldsymbol{\mathcal{A}}_l)_{i,:} = \mathbf{x}_k^{(i)} \tag{4}$$

for all $1 \leq k < l$, $i, j \in [M]$, where δ is the Kroencker symbol.

6: Perform the gradient update using efficient addition and product operations in TT format (see [33]):

$$(\boldsymbol{\mathcal{H}}^{(l)})_{\langle\langle l,1 \rangle\rangle} = (\boldsymbol{\mathcal{H}}^{(l)})_{\langle\langle l,1 \rangle\rangle} + \gamma(\boldsymbol{\mathcal{X}})_{\langle\langle 1,l \rangle\rangle}^{\top}(\mathbf{Y} - (\boldsymbol{\mathcal{X}})_{\langle\langle 1,l \rangle\rangle}(\boldsymbol{\mathcal{H}}^{(l)})_{\langle\langle l,1 \rangle\rangle})$$

7: Project the Hankel tensor $\boldsymbol{\mathcal{H}}^{(l)}$ (which is now of rank at most $R+M$) back onto the manifold of tensor of TT-rank R using the TT rounding operation (see again [33]):

$$\boldsymbol{\mathcal{H}}^{(l)} = \texttt{TT-rounding}(\boldsymbol{\mathcal{H}}^{(l)}, R)$$

8: **until** convergence

9: Let $\mathbf{P} = ([\![\boldsymbol{\mathcal{G}}_1^{(2L)}, \cdots, \boldsymbol{\mathcal{G}}_L^{(2L)}, \mathbf{I}_R]\!])_{\langle\langle L,1 \rangle\rangle}$ and $\mathbf{S} = ([\![\mathbf{I}_R, \boldsymbol{\mathcal{G}}_{L+1}^{(2L)}, \cdots, \boldsymbol{\mathcal{G}}_{2L+1}^{(2L)}]\!])_{\langle\langle 1,L+1 \rangle\rangle}$ (observe that $(\boldsymbol{\mathcal{H}}^{(2L)})_{\langle\langle L,L+1 \rangle\rangle} = \mathbf{PS}$ is a rank R factorization).

10: Return the linear 2-RNN $(\mathbf{h}_0, \boldsymbol{\mathcal{A}}, \boldsymbol{\Omega})$ where

$$\begin{aligned} \boldsymbol{\alpha} &= (\mathbf{S}^{\dagger})^{\top}(\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L+1 \rangle\rangle} \\ \boldsymbol{\mathcal{A}} &= ((\boldsymbol{\mathcal{H}}_f^{(2L+1)})_{\langle\langle L,1,L+1 \rangle\rangle}) \times_1 \mathbf{P}^{\dagger} \times_3 (\mathbf{S}^{\dagger})^{\top} \\ \boldsymbol{\Omega}^{\top} &= \mathbf{P}^{\dagger}(\boldsymbol{\mathcal{H}}_f^{(L)})_{\langle\langle L,1 \rangle\rangle} \end{aligned}$$

by performing efficient computations in TT format for the products [33] and pseudo-inverses (see e.g. [26]).

D Real Data Experiment on Wind Speed Prediction

Besides the synthetic data experiments we showed in the paper, we have also conducted experiments on real data. The data that we use for the these experiments is from TUDelft[8]. Specifically, we use the data from Rijnhaven station as described in [30], which proposed a regression automata model and performed various experiments on the dataset we mentioned above. The data contains wind speed and related information at the Rijnhaven station from 2013-04-22 at 14:55:00 to 2018-10-20 at 11:40:00 and was collected every five minutes. To compare to the results in [30], we strictly followed the data preprocessing procedure described in the paper. We use the data from 2013-04-23 to 2015-10-12 as training data and the rest as our testing data. The paper uses SAX as a preprocessing method to discretize the data. However, as there is no need to discretize data for our algorithm, we did not perform this procedure. For our method, we set the length $L = 3$ and we use the general algorithm described in Appendix B. We calculate hourly averages of the wind speed, and predict one/three/six hour(s) ahead, as in [30]. For our methods we use a linear 2-RNN with 10 states. Averages over 5 runs of this experiment for one-hour-ahead, three-hour-ahead, six-hour-ahead prediction error can be found in Table 1, 2 and 3. The results for RA, RNN and persistence are taken directly from [30]. We see that while TIHT+SGD performs slightly worse than ARIMA and RA for one-hour-ahead prediction, it outperforms all other methods for three-hours and six-hours ahead predictions (and the superiority w.r.t. other methods increases as the prediction horizon gets longer).

Table 1: One-hour-ahead Speed Prediction Performance Comparisons

Method	TIHT	TIHT+SGD	Regression Automata	ARIMA	RNN	Persistence
RMSE	0.573	0.519	0.500	**0.496**	0.606	0.508
MAPE	21.35	18.79	**18.58**	18.74	24.48	18.61
MAE	0.412	0.376	0.363	**0.361**	0.471	0.367

Table 2: Three-hour-ahead Speed Prediction Performance Comparisons

Method	TIHT	TIHT+SGD	Regression Automata	ARIMA	RNN	Persistence
RMSE	0.868	**0.854**	0.872	0.882	1.002	0.893
MAPE	33.98	**31.70**	32.52	33.165	37.24	33.29
MAE	0.632	**0.624**	0.632	0.642	0.764	0.649

Table 3: Six-hour-ahead Speed Prediction Performance Comparisons

Method	TIHT	TIHT+SGD	Regression Automata	ARIMA	RNN	Persistence
RMSE	1.234	**1.145**	1.205	1.227	1.261	1.234
MAPE	49.08	**44.88**	46.809	48.02	47.03	48.11
MAE	0.940	**0.865**	0.898	0.919	0.944	0.923

[8] http://weather.tudelft.nl/csv/