An Operator Theoretic Approach for Analyzing Sequence Neural Networks Supplementary Material

Ilan Naiman, Omri Azencot

Department of Computer Science,
Ben-Gurion University of the Negev, Beer Sheva, Israel
naimani@post.bgu.ac.il, azencot@cs.bgu.ac.il

A Unigram and bigram highlighting in sentiment analysis

Several examples of reviews in which the positive and negative unigrams are highlighted by the projection magnitude of the hidden states are shown in Fig. 1. In particular, we note the first negative review, where the network decreases the projection magnitude when it identifies a positive word (`excellent`), and the magnitude increases significantly when the word `bad` appears. To further assess the eigenvectors role in identifying positive and negative unigrams in the reviews, we perform the following statistical evaluation. We employ a Bag Of Words (`BOW`) algorithm on the vocabulary to classify the sentiment on the word level. Using the `BOW` classification, we extract the positive and negative words from each review, and we compute how many of these words attain high projection magnitude values. Specifically, given a batch H, we compute the following averages

$$a_p(H) = \frac{1}{|T_p|} \sum_{t \in Tp} \lfloor s(1, h_t) \rceil \ , \quad a_n(H) = \frac{1}{|T_n|} \sum_{t \in T_n} \lfloor s(3, h_t) \rceil \ , \tag{1}$$

where $s(j, h_t) = |\hat{h}_t(j)| \in [0, 1]$ is the projection magnitude of the state h_t onto the eigenvector u_j, T_p and T_n are the indices of positive and negative elements, respectively, and the operator $\lfloor \cdot \rceil$ rounds a number to the closest integer. Thus, $a_p(H)$ and $a_n(H)$ hold the average of positive and negative words whose projection magnitude is a at least .5 or higher. Consequently, we can view Eq. (1) as the percentage of identified positive and negative unigrams. We show in Fig. 5 histograms of $a_p(H)$ and $a_n(H)$ over the entire test set. We find that on average, 62.3% and 74.0% of the positive and negative words, respectively, are identified correctly by the eigenvectors. These statistics support and reinforce our observation about the role of the eigenvectors in counting positive/negative words.

B The case of general n-grams for $n > 2$

In addition to unigram and bigram highlighting, we also consider `KANN` in the general case of n-grams where $n > 2$. We note that n-grams with large n are less common in sentiment analysis, both from a semantic perspective (what would a 5-gram mean?) and also from a statistics viewpoint (e.g., even 2-grams are scarce in the IMDB dataset). Nevertheless, we would like to show in the following that `KANN` indeed extends to the general case, and we test our method on 3-gram phrases. To this end, we created a batch of 3-grams by adding amplifiers to 2-grams, e.g., "not bad" was changed to "not extremely bad" or "not very bad" and etc., and we repeated our analysis. Specifically, we extract a batch $H \in \mathbb{R}^{k \times n \times m}$ and obtain an operator C. Then, we project the batch on the eigenvectors as described in Eq.(10) for each j eigenvector of C while we sum over time and average over batch:

$$\xi_j(H) = \frac{1}{k} \sum_{h \in H} \sum_{t \in T_h} s(j, h_t) \tag{2}$$

where each $h \in H$ is a sequence of hidden states corresponds to an example in the batch, k is the size of the batch and T_h are the indices of the sequence. We sort $\xi(H)$ from high to low values to better visualize the importance of the Koopman eigenvectors and their ordering. We plot the resulting graph in Fig. 2 where a hierarchical behavior is shown, and each group of eigenvectors have a different role. The first group with labels $[3, 2, 4, 5]$ with the highest projection values captures 1-grams as discussed in the main text. The second group with labels $[6, 7]$ encodes 2-grams, and the third group with labels $[10, 9]$ encodes 3-grams as we show in Fig. 3. Specifically, using the same method we described in the main text, we obtain highlighting of 3-gram components in the review when projected onto eigenvector 10. Importantly, in our analysis on 3-grams we did not re-train the network. Rather, we simply created a batch with 3-gram components and analyzed the results obtained with `KANN`.

C Shuffled reviews

We will consider the following two reviews: "not bad and very good" and "very bad and not good". These reviews serve as a great example to understand the behavior of the network as they contain the same words in a different order, and the overall meaning is opposite. Considered from a linear dynamical systems viewpoint, these reviews seem to pose a challenge: given that the tokens are the same but in a different order, how will a linear system be able to distinguish between them? Indeed, assuming a linear system in the space of tokens would be problematic in this case. However, we would like to emphasize that our main claim in the paper is that the dynamics in the *latent space* are sufficiently linear to allow for Koopman analysis. As the network is recurrent and nonlinear, and the tokens are processed one-by-one, the above reviews can be distinguished in practice while not breaking the linearity in the latent space. Specifically, each of these reviews is embedded completely differently. The first example starts with the word "not", whereas the second example starts with the word "very". In practice, the network embeds these latent states in completely different locations of the latent space. Moreover, the embedding of the other states are related to the initial locations. Thus, in practice, the network has two completely different trajectories for the example reviews in discussion. From a (linear) dynamical systems perspective, we have two trajectories of different initial conditions. Such cases can be typically differentiated and identified using linear dynamical systems.

To show it numerically, we generated these two reviews (five words each), and we repeated our analysis. Specifically, we projected their latent trajectories (as obtained from the network) onto the first two dominant `PCA` modes (as was done in (Maheswaranathan et al. 2019)), and we report the obtained paths in Tab. 1. Keeping in mind that the network is mostly one-dimensional in a `PCA` representation, the above paths clearly show that the reviews are correctly classified. Namely, the first

review starts in the negative part of the x-axis (-0.716) and finishes in the positive part of the x-axis (0.713). In comparison, the second review starts in the positive part of the x-axis (1.478) and finishes in the negative part of the x-axis (-0.572). Computing the paths using our `KANN` representation via the matrix C, we obtain the paths reported in Tab. 1. While the values are different (as our linear approximation exhibits some error), the initial positions and trend are the same for the nonlinear representation and our Koopman linear representation. In particular, the trajectories end on the same side of the x-axis for each review, exactly as we have in the nonlinear network.

	"not bad and very good"		"very bad and not good"	
Time	`Network`	`KANN`	`Network`	`KANN`
$t=1$	$(-0.716,\ 0.547)$	$(-0.284,\ 0.649)$	$(1.478,\ 0.355)$	$(1.094,\ 0.522)$
$t=2$	$(-1.080,\ 0.280)$	$(-0.589,\ 0.612)$	$(0.313,\ 0.078)$	$(0.128,\ 0.619)$
$t=3$	$(-0.440),\ -0.093)$	$(0.003,\ 0.479)$	$(0.392,\ -0.282)$	$(0.335,\ 0.503)$
$t=4$	$(0.807,\ -0.319)$	$(0.708,\ 0.419)$	$(-0.895,\ -0.311)$	$(-0.530,\ 0.571)$
$t=5$	$(0.713,\ 0.036)$	$(0.598,\ 0.474)$	$(-0.572,\ -0.291)$	$(-0.237,\ 0.542)$

Table 1: The nonlinear network as well as our linear representation are able to differentiate between the reviews "not bad and very good" and "very bad and not good", and to correctly classify them. Specifically, we show the trajectories of the hidden states as obtained from the network and our method when projected to the first dominant `PCA` modes. The results above show similar initial conditions and trend, i.e., both start and end on the same side of the x-axis. We conclude that the network learns a representation which is sufficiently linear in the latent space, allowing to methods such as ours to expose its dynamics.

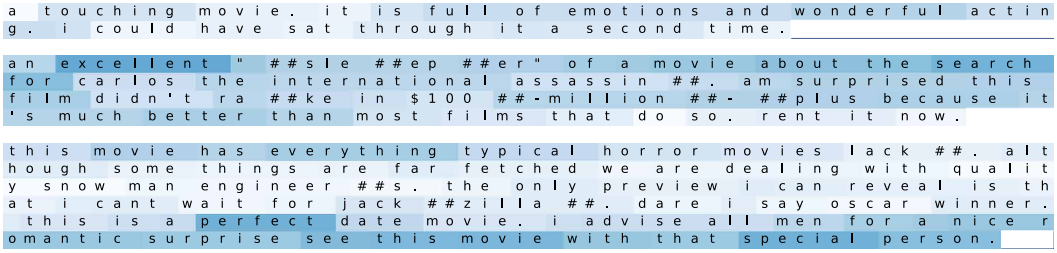
D Projecting normal beat signals onto `PCA` components

In Sec. 4.2 in the main text, we discover that the dominant Koopman eigenvectors are capable of identifying the salient features in the beat signals (marked by dashed black lines in Fig. 4). To compare our results with `PCA` and `KernelPCA` (using `rbf` kernel), we now repeat the same experiment, but instead of projecting onto Koopman modes, we project the hidden states H to the first four PCs and first four eigenvectors of the centered kernel matrix respectively. We provide both qualitative and quantitative comparison with both methods. Fig. 4 shows the resulting graphs, clearly demonstrating that `PCA` and `KernelPCA` fail to encode the dynamics. In Tab. 2 we provide a quantitative comparison of our method to `PCA` and `KernelPCA`. Specifically, for every method, we compute the mode with the minimal distance to the salient features located at times $t=3,\ 35\leq t\leq 75,\ t=103$ and $t=133$. The results clearly show that `KANN` attains the lowest error for each of the salient features.

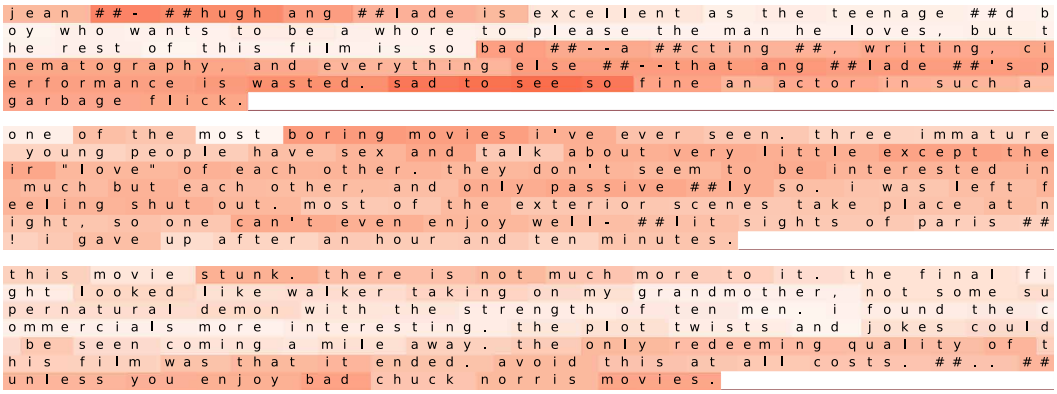
Method	$t=3$	$35\leq t\leq 75$	$t=103$	$t=133$
`PCA`	1.5817	1.2197	0.3356	1.1685
`KernelPCA`	0.0762	1.1045	0.4095	0.7217
`KANN`	**0.0317**	**0.5107**	**0.1724**	**0.0871**

Table 2: For every salient feature at times $t=3,\ 35\leq t\leq 75,\ t=103$ and $t=133$, we compute the distance between the signal and its reconstruction using the principal modes of `PCA`, `KernelPCA` and `KANN`. Our approach exhibits the minimal error in comparison to `PCA` and `KernelPCA`.

Positive unigram reviews:



Negative unigram reviews:



Bigram reviews:

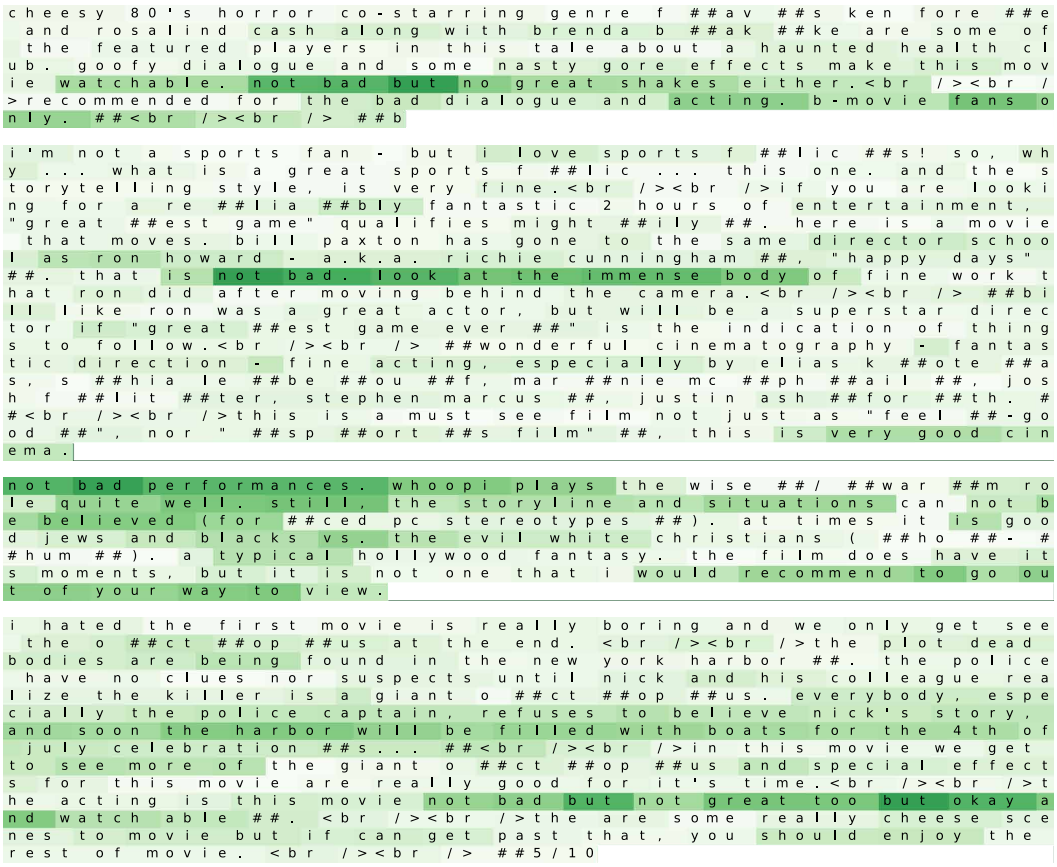


Figure 1: Several examples of highlighted unigrams and bigrams.

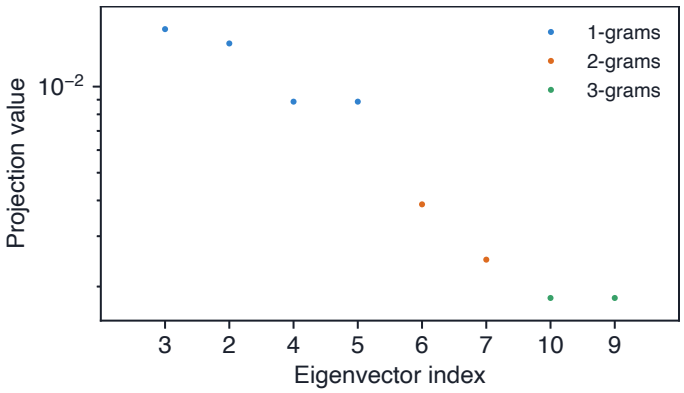


Figure 2: Projecting the batches of reviews onto the Koopman eigenvectors as specified in Eq. (2) reveals a hierarchical ordering where subspaces of eigenvectors attain different roles. In particular, the first group of eigenvectors colored blue and with indices$[3, 2, 4, 5]$ highlights 1-grams. Similarly, the second group of eigenvectors with the orange color and indices $[6, 7]$ identifies 2-grams. Finally, the last group in green with indices $[10, 9]$ highlights 3-grams. We note that a similar structure was identified across different batches.

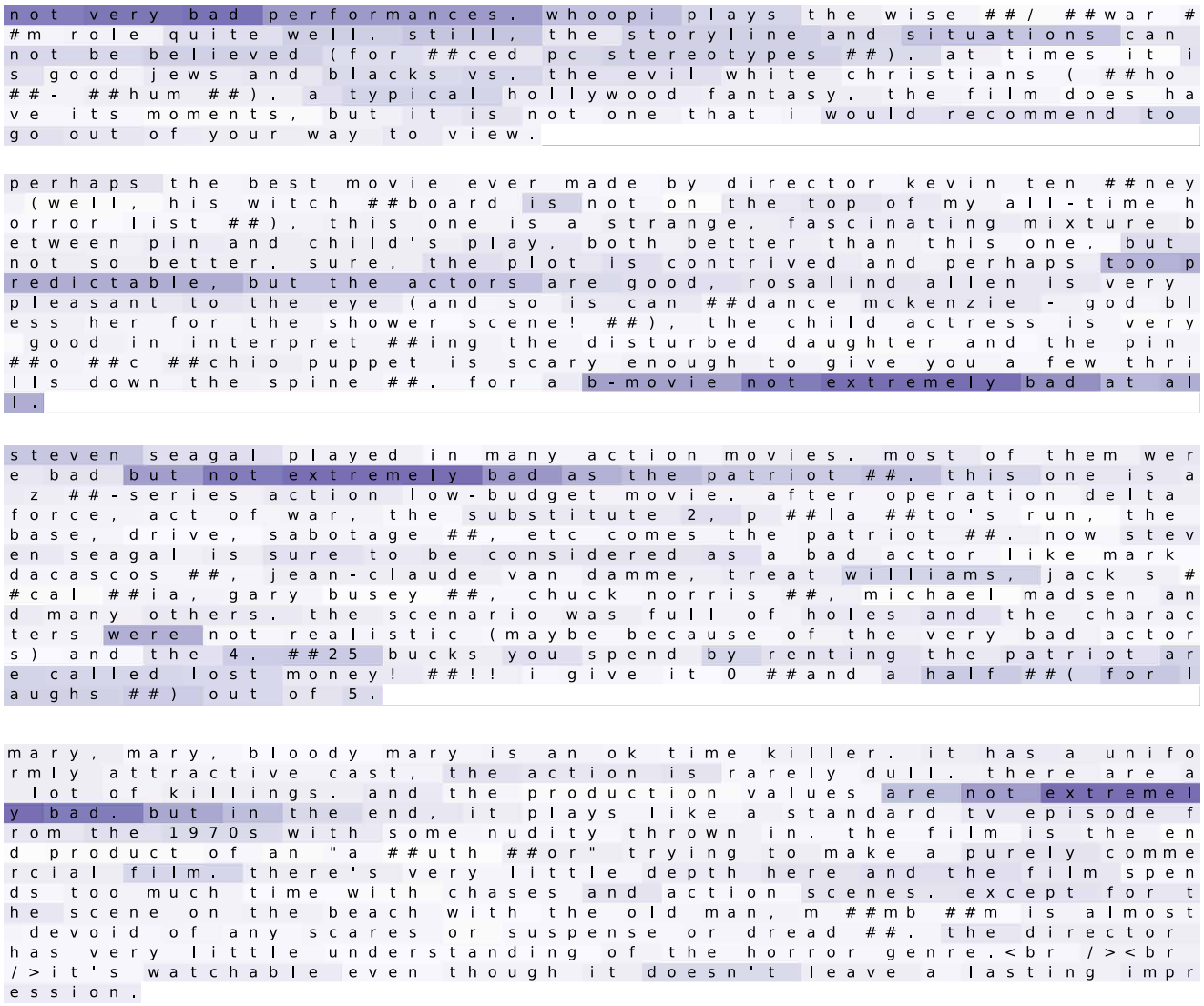


Figure 3: Projecting the hidden states to eigenvector 10 highlights 3-gram components, similarly to the highlighting of 1-grams and 2-grams in Fig. 1.

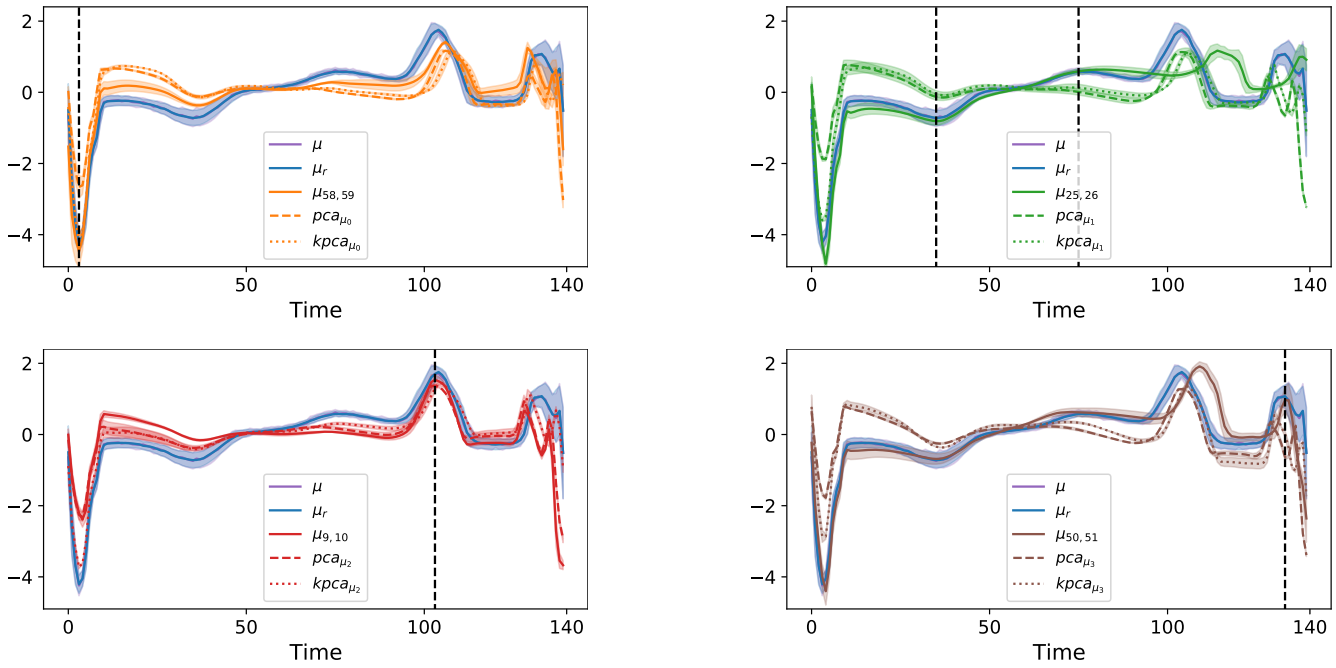


Figure 4: We show the first four principal modes of `KANN` (solid lines), `PCA` (dashed lines), and `KernelPCA` (dotted lines). The above graphs show that our method is better at matching the salient features of beat signals which are marked by black dashed lines in comparison to `PCA` and `KernelPCA`. We conclude that the network mainly focuses on reconstructing these salient features, allowing the user to easliy distinguish between normal and anomalous beats during post-prcoessing.

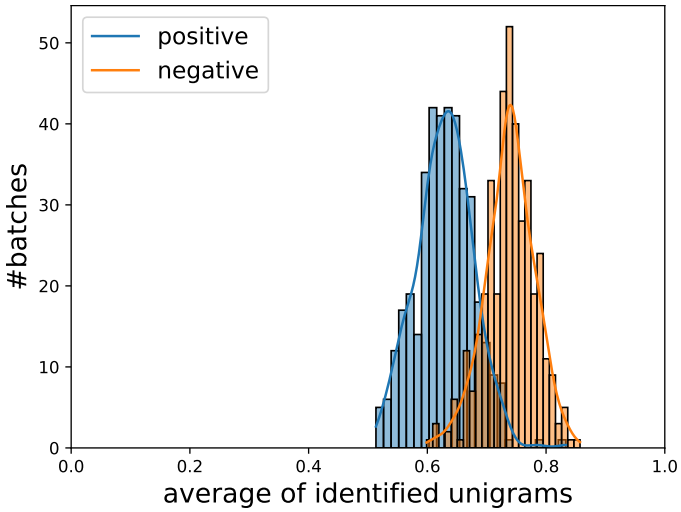


Figure 5: The above histograms show the average percentage of identified positive (blue) and negative (orange) unigrams per batch. It follows that negative words are better identified by the network (74.0%) in comparison to positive words (62.3%).

E Different basis and network architectures

Choice of basis. We will now demonstrate the robustness of our approach to the choice of basis. The first step to computing the matrix C involves the projection of the given states onto a basis. In our work, we mostly experimented with the truncated SVD modes obtained by decomposing the hidden states tensor. In what follows, we additionally show that the principal component analysis (PCA), and Fourier transform (FFT) bases lead to quantitatively similar results on the sentiment analysis task. We note that while the bases are linear in terms of projection, SVD and PCA are data-driven, whereas FFT is data-agnostic, i.e., the basis elements are independent of the data. First, we compute the relative error as in Sec. 4.4, and we obtain $0.0347, 0.0347, 0.973$ for SVD, PCA, and FFT, respectively. The somewhat poor result of FFT is expected, as it is data-agnostic. Second, we compare the dominant eigenvalues of the different C matrices computed using the bases. It follows that across all bases, the dominant eigenvalues correspond to one another. In particular, the average error between corresponding eigenvalues is 0.003 for PCA, and 0.02 for FFT, when measured from the eigenvalues of SVD. We additionally plot the dominant eigenvalues in Fig. 7 where the x-axis is the real part, and the y-axis is the imaginary part. Finally, we also show how the dominant eigenvectors have the same semantic role in highlighting the positive words in the same review. Indeed, we show in Fig. 6 that the positive words obtain large projection magnitudes in all bases. See the words e.g., amazing, special, good. Overall, the results show robustness to linear bases.

Results extend across architectures. In addition to robustness to the basis, we also verify that our results qualitatively extend across different architectures. Specifically, we trained a vanilla recurrent neural network (RNN) (Elman 1990), a long short term memory model (LSTM) (Hochreiter and Schmidhuber 1997), and a gated recurrent unit (GRU) network on the sentiment analysis problem. Then, we extract a single batch from the test set, and evaluate our KANN approach on the trained models. We find that our analysis yields similar results in all cases. In particular, the dominant eigenvalues of each of the models attain related values as can be seen in Fig. 7 (right). Moreover, we find that the dominant eigenvectors share the same role of highlighting positive and negative unigrams. To verify this, we computed the histograms of identified positive and negative words as in Fig. 5. We observe that on average 62%, 76%, and 62% positive words are discovered by the projection magnitude of the RNN, LSTM, and GRU models. Similarly, the negative unigrams are highlighted in an average of 55%, 83%, and 74% for RNN, LSTM, and GRU. Indeed, there is a large variation in the statistics of the models, where LSTM obtains the best averages, followed by GRU, and RNN is last. Nevertheless, in all cases, the average identification of unigrams is above 50%, and given that BOW is noisy by itself, we believe these statistics are qualitatively similar. In addition, we plot in Fig. 8 a few examples of highlighted reviews obtained with the models.

F Results on the copy task

The copy task was designed to test the memory retaining capabilities of recurrent units (Hochreiter and Schmidhuber 1997). In this task, the network is expected to memorize the first few characters in the input array and copy them to the end of the output vector which is otherwise filled with blanks. For instance, the input-output structure reads $928{-}{-}{-}{:}{-}{-} \mapsto_{\varphi} {-}{-}{-}{-}{-}{-}928$, if the model is required to remember three digits across three blanks. Thus, the challenge increases with more digits to remember and when the amount of blanks is higher. We trained a dtriv architecture (Casado 2019) on the copy task with three characters to remember and 30 blanks for 500 iterations. The dtriv model is similar to a vanilla RNN with the exception that its hidden-to-hidden transformation is *orthogonal*. The network converges to an accuracy of 100% on the training and test data as it is a relatively easy setting. The following analysis is based on a test batch of size 32, yielding a states tensor $H \in \mathbb{R}^{32\times 36\times 48}$ where the middle dimension is the sequence length, and the last dimension is the hidden state size.

The latent structure of the copy task is measure-preserving. Our first analysis result deals with the geometric structure of the learnt dynamics. Before discussing our results, we make the following three observations. First, the copy problem with its unknown dynamics φ which maps inputs to outputs, is isometric. Indeed, for many choices of norms, e.g., L^2, it follows that $d(x_1, x_2) = d(y_1 = \varphi(x_1), y_2 = \varphi(x_2))$ where x_1, x_2 are two input vectors, and y_1, y_2 are two output vectors. Thus, φ belongs to the class of *measure-preserving* dynamical systems. Second, while dtriv uses orthogonal hidden-to-hidden matrices, the overall network transformation is not necessarily isometric due to the nonlinear activation layers. Indeed, the analysis in (Arjovsky, Shah, and Bengio 2016) which also applies to dtriv only establishes an upper bound on the gradient norms, and there is no lower bound. In practice, since dtriv uses modReLU (Arjovsky, Shah, and Bengio 2016), it may cause a non-isometric transformation to the latent space. Third, Koopman theory establishes a connection between the algebraic properties of the linear operator to the geometric structure of the dynamics as we show next. The following result is not novel (Eisner et al. 2015), but we prove it below for completeness.

Proposition 1 *Let φ be an invertible measure preserving dynamical system on a compact, inner-product domain $\mathcal{M}$. Then its associated Koopman operator is unitary.*

Proof. Let $\varphi : \mathcal{M} \to \mathcal{M}$ be a map on the compact, inner-product space $\mathcal{M}$. We denote by μ the continuous measure on $\mathcal{M}$, and its induced metric $\|z\|$. The map φ is measure preserving, i.e., $\mu(\varphi^{-1}A) = \mu(A)$ for every measurable set $A \subset \mathcal{M}$. Let $\mathcal{K}_\varphi$ be the Koopman operator of φ acting on the function space of square integrable function L^2. Given the indicator function 1_A for the set A, we have that

$$\mathcal{K}_\varphi 1_A(z) = 1_A(\varphi \circ z) = 1_{\varphi^{-1}A}(z) \ ,$$

and thus

$$\int_{\mathcal{M}} \mathcal{K}_\varphi 1_A \,\mathrm{d}\,\mu = \mu(\varphi^{-1}A) = \mu(A) = \int_{\mathcal{M}} 1_A \,\mathrm{d}\,\mu \,.$$

Moreover, positive functions converge to a representation using simple indicator functions. Consequently, we have that $\int_{\mathcal{M}} \mathcal{K}_\varphi f \,\mathrm{d}\,\mu = \int_{\mathcal{M}} f \,\mathrm{d}\,\mu$ for general $f \in L^2$ since it can be written as the difference of the integrable negative and positive components of f.

The Koopman operator is linear and it is pointwise multiplicative, i.e., $\mathcal{K}_\varphi(\alpha f + \beta g) = \alpha\mathcal{K}_\varphi(f) + \beta\mathcal{K}_\varphi(g)$ and $\mathcal{K}_\varphi(f\,g) = \mathcal{K}_\varphi(f)\mathcal{K}_\varphi(g)$, where $\alpha, \beta \in \mathbb{R}$, and $f, g \in L^2$. Due to these observations, it follows that $\mathcal{K}_\varphi$ preserves the inner product of functions, namely, for every $f, g \in L^2$

$$\langle f, g\rangle = \int_{\mathcal{M}} f\,g \,\mathrm{d}\,\mu = \int_{\mathcal{M}} \mathcal{K}_\varphi(f\,g) \,\mathrm{d}\,\mu = \langle \mathcal{K}_\varphi(f),\, \mathcal{K}_\varphi(g)\rangle \,.$$

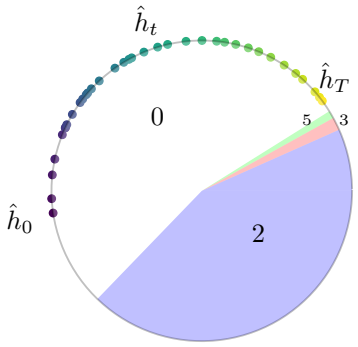
Thus, the Koopman operator in this case is an isometry, since

$$\mathrm{d}(f,\, g) = \|f - g\| = \langle f - g,\, f - g\rangle^{\frac{1}{2}} = \langle \mathcal{K}_\varphi(f - g),\, \mathcal{K}_\varphi(f - g)\rangle^{\frac{1}{2}} \,.$$

Finally, if φ is invertible then $\mathcal{K}_\varphi^*\mathcal{K}_\varphi = \mathcal{K}_\varphi\mathcal{K}_\varphi^*$ where $\mathcal{K}_\varphi^*$ is the adjoint operator, and thus $\mathcal{K}_\varphi$ is unitary.

Given H as specified above and its corresponding C, we find that C is approximately orthogonal, i.e., $C^TC \approx \mathrm{id}$. Specifically, the relative error $|C^TC - \mathrm{id}|^2/|C|^2 = 0.0625$. Our findings align with prior work (Rustamov et al. 2013) which shows that approximate Koopman operators are approximately orthogonal for measure-preserving maps. Therefore, although `dtriv` is not guaranteed to learn a measure-preserving latent map, it does so in practice as our method reveals.

Eigenvectors span multiple digits in the copy task. Our second analysis result on the copy problem focuses on the eigendecomposition of C. We find that most of the eigenvalues, 44 out of 47, are approximately unit length, i.e., $|\lambda_j - 1| < 5\mathrm{e}{-2}$, which also reinforces the above findings. Based on Eq. (9) it follows that the eigenvectors of those eigenvalues have long memory horizons, e.g., $\tau = 418$ for $\epsilon = 1\mathrm{e}{-1}$. This is well beyond the required memory horizon for this task which is 30 as the number of blanks. Additionally, we find that all eigenvectors have the capacity to represent several characters, depending on the root of unity they are multiplied with. Namely, computing the output of the state $\tilde{h} = \mathrm{Re}(zv_j)$ for several z values, yields various digits. For instance, the inset shows a specific eigenvector and its associated digits with their respective span of the unit ball. We also plot in shaded dots the coefficients of a particular input over time. Evidently, the shown eigenvector is responsible to output the blank part of the output since the coefficients are located in the zero regime. Qualitatively similar results were obtained for the other eigenvectors as we show in Fig. 9 the sets of digits for select eigenvectors. For instance, v_6 spans the digits $\{0, 5, 7\}$, depending on the root of unity z_i we multiply with v_6. Thus, the network essentially splits the latent space onto digit regions. Then, given an input such as `928---:--`, the network generates its latent trajectory by carefully scaling the eigenvectors to point to the required output for every time sample.



Quantitative results on the copy task. We briefly recall that `RENN` uses the hidden state tensor H to generate a set of fixed points, i.e., points h^* for which the dynamical system $h_t = F(h_{t-1}, x_t)$ is stationary $h^* \approx F(h^*, 0)$ (Sussillo and Barak 2013). Then, they derive their analysis using the input and recurrent Jacobians of F, $\mathcal{J}^{\mathrm{inp}}$ and $\mathcal{J}^{\mathrm{rec}}$, evaluated at a single point $(h^*,\, x^* \equiv 0)$. We show in Fig. 10 the resulting Jacobian matrices using `RENN` where $\mathcal{J}^{\mathrm{rec}} \approx \mathrm{id}$ matrix (left). This is actually the expected result—as the blanks are mapped to zeros in this task, using $x^* \equiv 0$ means we look for fixed points h^* related to a blank input. However, the output for a blank input should be blank as well, and thus the hidden states converge to a section of the manifold which is indifferent to the inputs. Indeed, in (Sussillo and Barak 2013; Maheswaranathan and Sussillo 2020; Maheswaranathan et al. 2020), the authors discuss approaches to select input dependent initial points x^*, however, it remains unclear how to avoid the above issue since any chosen point is related to a particular potential input. For reference and comparison, we show in Fig. 10 (middle) the algebraic structure of our C matrix.

To assess the information encoded in $\mathcal{J}^{\mathrm{rec}}$ and $\mathcal{J}^{\mathrm{inp}}$ vs. C, we perform the following experiment. Let $\{h_t\}$ denote the nonlinear path of hidden states obtained from the copy task network. Given a certain threshold $l = 1, ..., T$, we split the path to two segments $\{h_t\}_{t=1}^{l}$ and $\{B\,C^k\tilde{h}_l\}_{k=1}^{T-l}$. That is, the first segment is simply the original states, and the second segment includes linear predictions with C^k while always using h_l. We denote by $H_l^{\mathtt{KANN}}$ the union of the paths, i.e.,

$$H_l^{\mathtt{KANN}} = \{h_1, ..., h_l, B\,C\tilde{h}_l, .., B\,C^{T-l}\tilde{h}_l\} \,. \tag{3}$$

For every admissible l, we generate $H_l^{\mathtt{KANN}}$, and we compute the accuracy obtained by the network using the path $H_l^{\mathtt{KANN}}$. Fig. 10 (right) shows in blue the accuracy for the nonlinear path which is simply 100%. We show in orange the accuracy obtained for

several l/T values. The accuracy results of KANN are extremely good, even when the percentage is high, i.e., most of the path does *not* use the states provided by the network, but rather, their linear prediction. Further, we emphasize that the orange point marks the percentage for three hidden states. That is, our method gets more than 80% accuracy exactly when all the non-blank input digits are implicitly available in the states. Therefore, our results highlight that C truly mimics the nonlinear dynamics as it is the minimal set of necessary inputs for a meaningful prediction.

In comparison, RENN can be used in a similar fashion to generate H_l^{RENN} using the following formula

$$h_{t+1}^{\text{RENN}} := h^* + \mathcal{J}^{\text{rec}}(\bar{h}_t - h^*) + \mathcal{J}^{\text{inp}} x_t \tag{4}$$

$$\approx \bar{h}_t + \mathcal{J}^{\text{inp}} x_t \ , \tag{5}$$

where $\bar{h}_t$ can be the original h_t or h_t^{RENN} depending on l, and the bottom formula is relevant when $\mathcal{J}^{\text{rec}} \approx \text{id}$ matrix. The green curve in Fig. 10 shows the accuracy results of RENN. Due to the trivial nature of $\mathcal{J}^{\text{rec}}$, RENN achieves zero accuracy in most cases, and it significantly improves when the last three states become available (marked by the green point). Thus, RENN requires almost the entire sequence of ground-truth hidden states to produce good accuracy measures in this scenario.

G Training Information

In Tab. 3 we add details regarding the models training process across each architecture and task. In the tasks column, SA, ECGC, and CT are acronyms for Sentiment Analysis, ECG Classification, and Copy Task, respectively. In addition, we used weight decay regularization in both ECG classification and Sentiment Analysis tasks.

Table 3: The following hyperparameters per task and model were used during training.

Task	Architecture	#epochs	#units	Optimizer	LR	LR Scheduler	Clip
SA	RNN	7	128	Adam	5e−3	ExpLR, $\gamma = 0.6$	15
SA	GRU	5	256	Adam	5e−3	ExpLR, $\gamma = 0.5$	15
SA	LSTM	5	256	Adam	1e−3	ExpLR, $\gamma = 0.3$	5
ECGC	GRU	150	64	Adam	1e−3	–	−1
ECGC	LSTM	150	64	Adam	1e−3	–	−1
CT	dtriv	500	48	RMSprop	1e−3	–	−1
CT	RNN	$10k$	64	RMSprop	5e−3	ExpLR, $\gamma = 0.85$	5
CT	GRU	285	48	RMSprop	1e−2	–	−1
CT	LSTM	$6.5k$	48	RMSprop	5e−3	–	10

H KANN reproduces the latent dynamics

We performed a quantitative study of the ability of C to truly capture the latent dynamics. We show that indeed, KANN is able to reproduce the nonlinear dynamics of the network in Eq.(1) from the main text, to a high degree of precision, and thus we achieve the empirical justification to replace F with C. To this end, we consider the following two metrics:

1. **Relative error** of hidden states: let $\{h_{s,t}\}$ be a collection of states over samples $s = 1, ..., S$ and across time $t = 1, ..., T$. We generate the predicted collection $\{h_{s,t}^{\text{KANN}}\}$ using Eq.(6), and we compute

$$e_{\text{rel}}(\{h_{s,t}^{\text{KANN}}\}, \{h_{s,t}\}) = \frac{1}{T \cdot S} \sum_{s,t} |h_{s,t}^{\text{KANN}} - h_{s,t}|_2^2 \ / \ |h_{s,t}|_2^2 \ .$$

2. **Accuracy error**: let G be the neural network component that takes a state and produces the output of the model, i.e., $G(h_t) = \tilde{y}_t$. We denote by $\tilde{c}_t$ the category predicted by $\tilde{y}_t$, for instance $\tilde{c}_t = \arg\max(\tilde{y}_t)$. We compare the difference between $\tilde{c}_t$ and $\tilde{c}_t^{\text{KANN}}$, obtained from $G(h_t^{\text{KANN}}) = \tilde{y}_t^{\text{KANN}}$.

We show in Fig.4 of the main text, the results of our quantitative study. For the sentiment analysis problem (Fig.4, left), we obtain $> 99\%$ correspondence with the classification of the network over *all* the test set, as is shown for the True Positive (TP) and True Negative (TN) columns vs. the False Positive (FP) and False Negative (FN) columns. In the ECG classification task (Fig.4, right), we reconstruct 145 signals of the normal test set and compute their loss. There is a noticeable yet small shift in the loss histogram between the network reconstruction (blue) in comparison to our reconstruction (orange). However, the threshold for this problem set at 26 during training (black dashed line) yields $> 97\%$ agreement in classification. In particular, the false classification of normal signals (around loss 90) appear both in the network output and in ours. Finally, we also computed the relative error of the hidden states for each of the tasks, and we show the results in Tab. 4. Overall, the results demonstrate that KANN faithfully represents the latent dynamics.

Table 4: Relative error of hidden states

Task	Copy task	Sentiment analysis	ECG classification
#batch	32	64	145
e_{rel}	0.021	0.095	0.0056

References

Arjovsky, M.; Shah, A.; and Bengio, Y. 2016. Unitary Evolution Recurrent Neural Networks. In *International Conference on Machine Learning*, volume 48, 1120–1128.

Casado, M. L. 2019. Trivializations for Gradient-Based Optimization on Manifolds. In *Advances in Neural Information Processing Systems*, 9154–9164.

Eisner, T.; Farkas, B.; Haase, M.; and Nagel, R. 2015. *Operator theoretic aspects of ergodic theory*, volume 272. Springer.

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

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Maheswaranathan, N.; and Sussillo, D. 2020. How recurrent networks implement contextual processing in sentiment analysis. *arXiv preprint arXiv:2004.08013*.

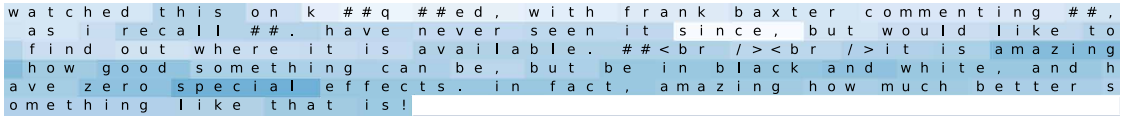
Maheswaranathan, N.; Sussillo, D.; Metz, L.; Sun, R.; and Sohl-Dickstein, J. 2020. Reverse engineering learned optimizers reveals known and novel mechanisms. *arXiv preprint arXiv:2011.02159*.

Maheswaranathan, N.; Williams, A.; Golub, M.; Ganguli, S.; and Sussillo, D. 2019. Reverse engineering recurrent networks for sentiment classification reveals line attractor dynamics. In *Advances in Neural Information Processing Systems*, 15696–15705.

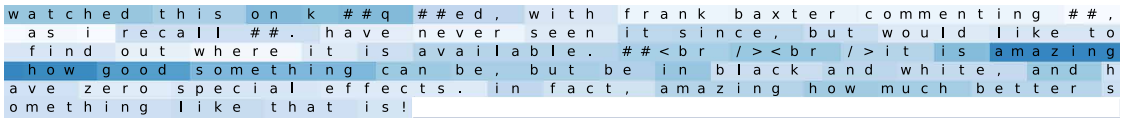
Rustamov, R. M.; Ovsjanikov, M.; Azencot, O.; Ben-Chen, M.; Chazal, F.; and Guibas, L. 2013. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4): 1–12.

Sussillo, D.; and Barak, O. 2013. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3): 626–649.

Using SVD:



Using PCA:



Using FFT:

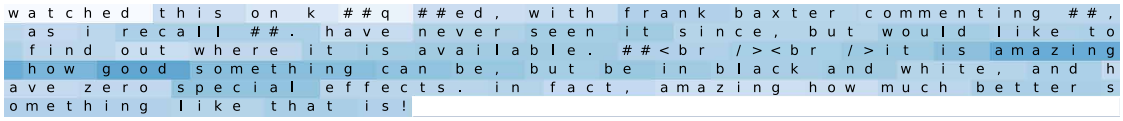


Figure 6: Computing C using SVD, PCA, and FFT yield eigenvectors with the same semantic role. Indeed, projections using different bases highlight various positive words in the same review.

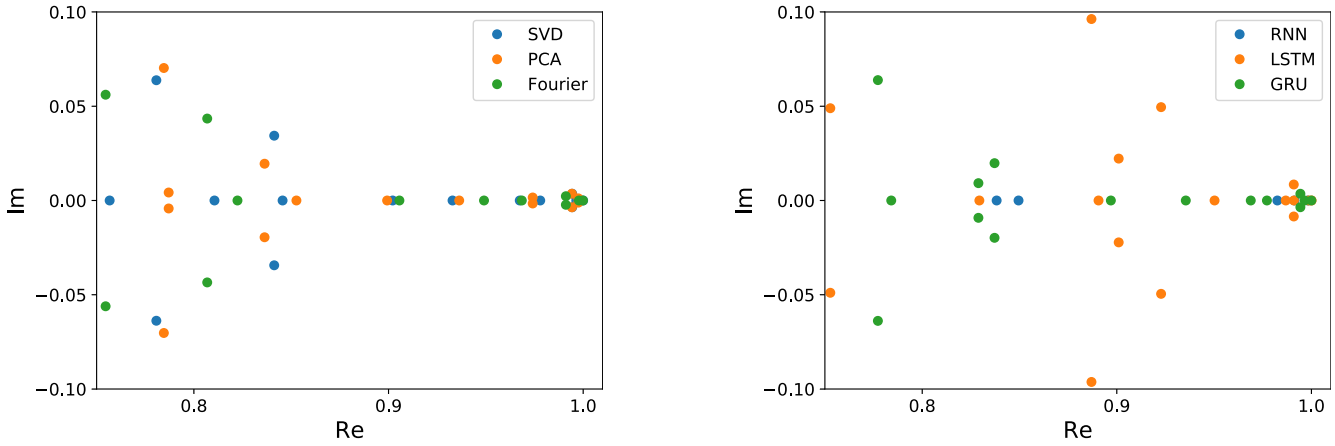
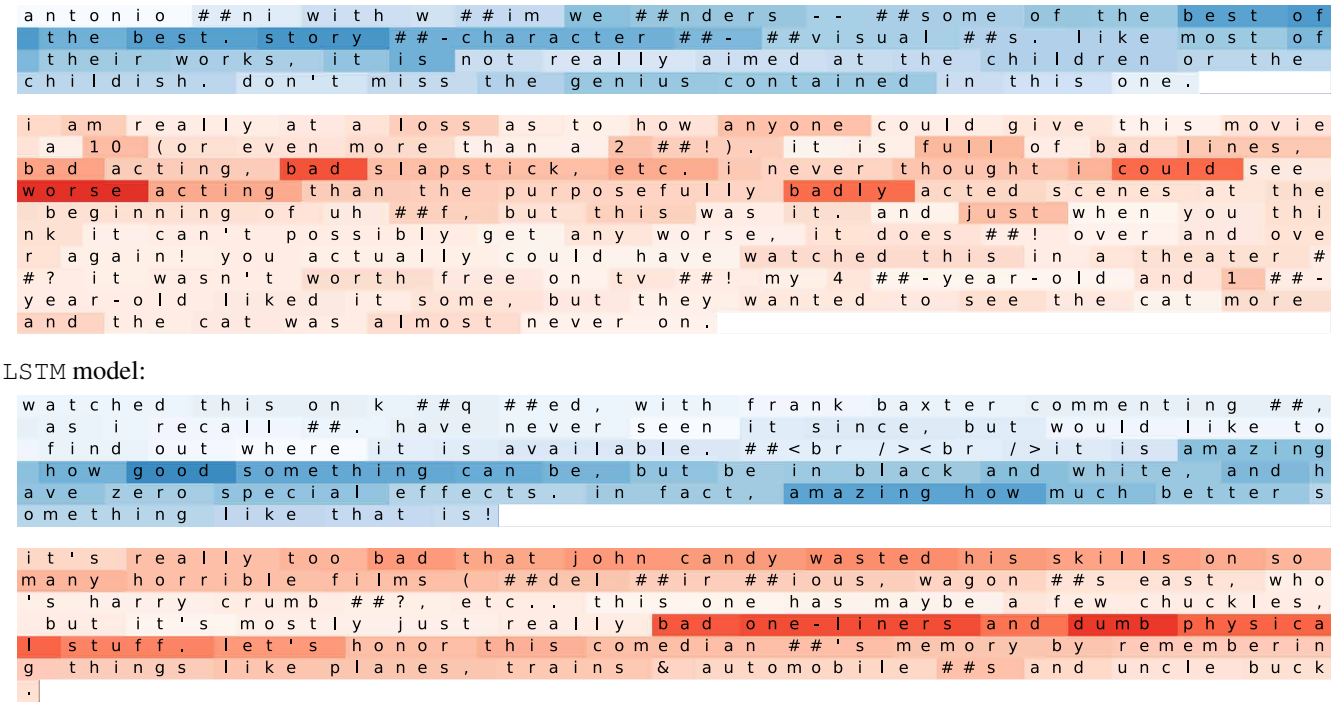


Figure 7: We show the dominant eigenvalues of various C matrices in the complex plane (Re, Im). Most of the eigenvalues correspond when different bases such as SVD, PCA, FFT are used (left). Similarly, using recurrent components such as RNN, LSTM, GRU leads to related spectra (right).

RNN model:



LSTM model:

Figure 8: Examples of highlighted unigrams obtained from the trained RNN and LSTM models.

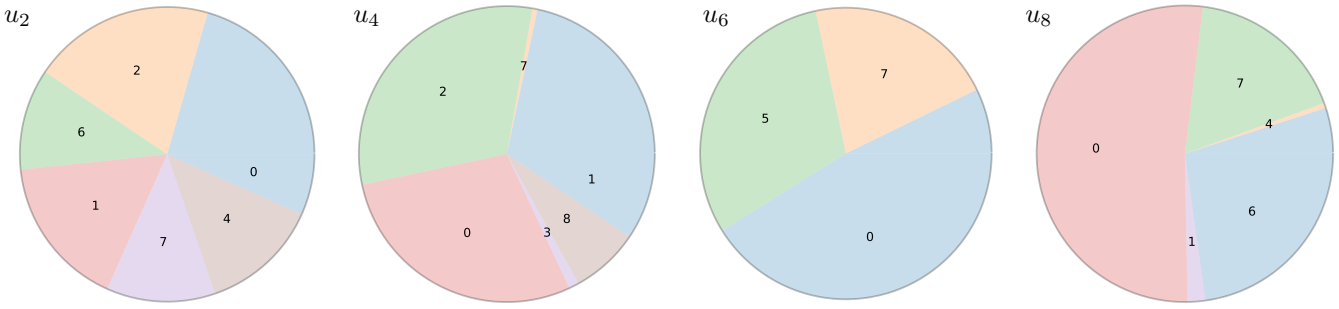


Figure 9: Every eigenvector in the copy task span multiple characters in the alphabet, allowing it contribute to the propagation of the initial digits over the sequence.

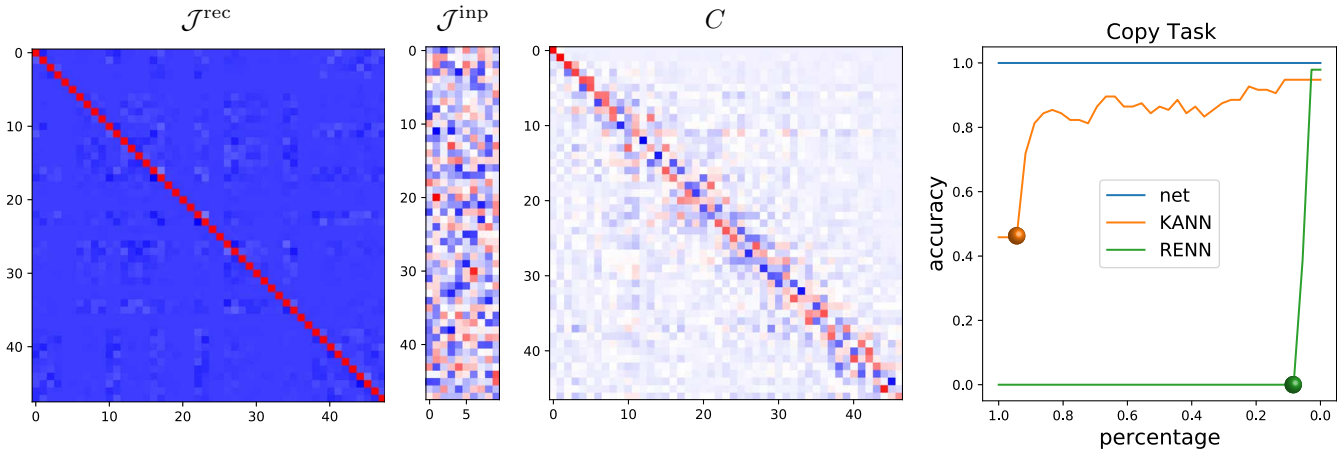


Figure 10: Computing RENN components for the copy task leads to an almost identity recurrent Jacobian, $|\mathcal{J}^{\text{rec}} - \text{id}| = 0.11$ relative error. In comparison, our matrix C is approximately orthogonal and it exhibits a diagonally-dominant structure. Our KANN approach attains good accuracy results when used to predict the states path. See the text.