DECAYNET: A STUDY ON THE CELL STATES OF LONG SHORT TERM MEMORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

It is unclear whether the extensively applied long short term memory (LSTM) is an optimised architecture for recurrent neural networks. Its complicated design and opaque mechanics make the network hard to analyse and non-immediately clear for its utilities in real-world data. This paper studies LSTMs as systems of difference equations, and takes a theoretical mathematical approach to study consecutive transitions in network variables. Our study shows that the cell state propagation is predominantly controlled by the forget gate. Based on these mathematical insights, we introduce the DecayNet reformulation to calibrate cell state dynamics with a monotonically decreasing forget gate. The reformulation increases LSTM modelling power without the need for introducing new learnable parameters; and also yields more consistent results.

1 INTRODUCTION

Neural networks are powerful universal approximators that are difficult to interpret. This paper presents a numerical study on the forward pass of the *long short term memory* (LSTM) (Hochreiter & Schmidhuber, 1997) recurrent neural network (RNN). We treat LSTMs as systems of difference equations and present visualisations to study consecutive transitions in their variables. The contribution of this paper is two-fold: we clarify the cell state dynamics; and introduce a simple reformulation, **DecayNet**, to increase LSTM modelling power without new learnable parameters.

LSTMs are versatile models that have been used to advance the state-of-the-art for a variety of machine learning problems. This covers handwriting recognition (Graves et al., 2009), speech recognition (Sundermeyer et al., 2012), and text modelling (Kim et al., 2016). There are an abundance of work that extends on the basic LSTM architecture. Popular extensions include the attention mechanism (Bahdanau et al., 2014) and the bidirectional design (Schuster & Paliwal, 1997). It is also not uncommon to blend LSTMs into vision tasks to expand network practicalities (Xu et al., 2015).

However, it is unclear whether LSTMs are optimal architectural designs for RNNs (Jozefowicz et al., 2015). Additionally, the significance of their individual components are unclear, and empirical explorations are required for understanding their utility in real-world data (Karpathy et al., 2015). This lack of interpretability limits our ability in designing better and more *transparent* networks.

RNNs repeatedly integrate new observations into implicit hidden variables to model temporal representations of sequences. Like dynamical systems, the network undergoes iterative computations on a same set of operators. The transitional dynamics can be visualised through cobweb diagrams (Glendinning, 1994); and numerical analyses allow us to predict *shrinkage* and *growth* in variables.

We show that the updating scheme of the LSTM cell state volatilely alternates between a *catch* and a *release* phase. The former controls the shrinkage while the latter controls the growth of cell state magnitudes. Our study reveals that the forget gate dictates the dynamical alterations of the cell state. We hence introduced the DecayNet, a reformulated LSTM with a monotonically decreasing forget gate, for stabilising cell state propagation and for making them more interpretable.

DecayNet-LSTMs produce more consistent results than vanilla LSTMs on the image classifcation tasks of MNIST (LeCun et al., 1998) and Fashion-MNIST (Xiao et al., 2017). Along with recurrent batch normalisation (Cooijmans et al., 2016), DecayNet-LSTMs consistently outperform (best: 97.8%; mean: 97.4%) previous state of the art results on the sequential classification task of permuted pixel-by-pixel MNIST (Le et al., 2015).

2 THE VANILLA LSTM

The reformulated DecayNet-LSTM will be frequently compared to conventional LSTMs. The most commonly implemented LSTM RNN was proposed by Graves & Schmidhuber (2005). This paper will refer to the architecture therein that paper as *the* LSTM.

LSTMs are recursive systems driven by input $\boldsymbol{x}_k$. Input $\boldsymbol{x}_k$ has data-size dimensionality M, and the network propagates for a data-length of D units of time. Subscript k refers to the recursive instance, *i.e.*, $k = 1 \cdots D$, and each instantaneous computation (in abbreviated form) encompasses

System (1): the vanilla LSTM

the gated variables:	$\boldsymbol{f}_k, \boldsymbol{i}_k, \boldsymbol{o}_k =$	$\sigma_s(\mathbf{W}_{\{F,I,O\}}\boldsymbol{x}_k + \mathbf{W}_{R\{F,I,O\}}\boldsymbol{q}_{k-1} + \boldsymbol{b}_{\{F,I,O\}})$,
the internal input:	$\boldsymbol{a}_k =$	$\sigma_t(\mathbf{W}_A\boldsymbol{x}_k + \mathbf{W}_{RA}\boldsymbol{q}_{k-1} + \boldsymbol{b}_A)$,
the cell state:	$\boldsymbol{s}_k =$	$\boldsymbol{f}_k \odot \boldsymbol{s}_{k-1} + \boldsymbol{i}_k \odot \boldsymbol{a}_k$,
the hidden state:	$\boldsymbol{q}_k =$	$\boldsymbol{o}_k \odot \sigma_t(\boldsymbol{s}_k)$,
with dimensionality:	$\boldsymbol{x}_k \in \mathbb{R}^M$,	$\boldsymbol{f}_k, \boldsymbol{i}_k, \boldsymbol{o}_k, \boldsymbol{a}_k, \boldsymbol{q}_k, \boldsymbol{s}_k, \boldsymbol{b}_{\{F,I,O,A\}} \in \mathbb{R}^N$, and
	$\mathbf{W}_{\{F,I,O,A\}} \in \mathbb{R}^{N\times M}$,	$\mathbf{W}_{R\{F,I,O,A\}} \in \mathbb{R}^{N\times N}$.

The input $\boldsymbol{x}_k$ and the hidden state $\boldsymbol{q}_k$ contribute to other variables via $\mathbf{W}$s and via $\mathbf{W}_R$s. Dimensionality N is user-defined and $\odot$ is the operator for element-wise product. Along with biases $\boldsymbol{b}$s, the regressive contributions are activated by either the sigmoidal function σ_s or the hyperbolic tangent function σ_t. During the update of the cell state $\boldsymbol{s}_k$, the forget gate $\boldsymbol{f}_k$ relinquishes fragments of the previous instance $\boldsymbol{s}_{k-1}$, and the input gate $\boldsymbol{i}_k$ fine-tunes replenishment from the internal input $\boldsymbol{a}_k$. Finally, variable $\boldsymbol{q}_k$, which also serves as the output of the network, is a transformation of $\boldsymbol{s}_k$ synchronised with the output gate $\boldsymbol{o}_k$.

3 LSTMS AS DIFFERENCE EQUATIONS

Difference equations are recurrent relations with discrete propagation; and the term dynamic refers to the dependency between time and geometric space. Dynamical behaviours of a difference equation can be studied to determine whether variables will grow or shrink. This allows us to explain behaviours of individual LSTM variables and clarify network mechanics.

This section discusses three key concepts – the importance for studying the LSTM cell state; the two dynamic regimes for cell state values to grow and to shrink; and that the forget gate can be controlled to yield interpretable and predictable cell state motions. Cobweb diagrams will be used to support concept visualisations, and a brief introduction to the technique can be found in **Appendix A**.

3.1 THE CELL STATE REVISITED

This paper postulates the cell state as the source of LSTM's modelling capability. Furthermore, we regard gated variables as secondary variables. Our justifications are as follows.

First, gated variables only exist to read and to set the hidden variables of $\boldsymbol{q}_k$ and $\boldsymbol{s}_k$. That is, they exist to assist LSTMs to memorise better, but are not a part of the memories. Second, σ_t is a one-to-one function, thus dynamic properties of $\boldsymbol{q}_k$ are inherited from $\boldsymbol{s}_k$. Last, $\boldsymbol{a}_k$ is included in the update of $\boldsymbol{s}_k$, and acts as a small component to the grander dynamic entailed in $\boldsymbol{s}_k$. For these reasons, we postulate the cell state as the most important variable and study it as a difference equation.

3.2 A PIECEWISE ANALYSIS ON THE CELL STATE

We study the cell state, $\boldsymbol{s}_k = \boldsymbol{f}_k \odot \boldsymbol{s}_{k-1} + \boldsymbol{i}_k \odot \boldsymbol{a}_k$, separately as $\boldsymbol{s}_p = \boldsymbol{f}_k \odot \boldsymbol{s}_{k-1}$ and $\boldsymbol{s}_q = \boldsymbol{i}_k \odot \boldsymbol{a}_k$. Since we asserted gated variables as secondary variables, we can think of $\boldsymbol{f}_k$ and $\boldsymbol{i}_k$ as constant vectors with elements between 0 and 1. That is, their individual dimensions have cobweb outlines that are analogous to **Figures 1(a)** and **1(b)**.

For the rest of this paper, we will refer to marginal dimensions of $\boldsymbol{f}_k$, $\boldsymbol{i}_k$, $\boldsymbol{a}_k$, $\boldsymbol{x}_k$, and $\boldsymbol{s}_k$ as f_{mk}, i_{mk}, a_{mk}, x_{mk}, and s_{mk}, respectively. For instance, each dimension of s_q is $i_{mk}\ a_{mk}$ and inherits the diverse characteristic outline of **1(b)** through σ_t from a_{mk}.

Consecutive transitions in network variables are shown as the dotted trajectories in **Figure 1**. They start from the rhombuses and ends on the circles. The characteristic outlines of the sub-dynamics are the solid lines, and the dashed lines represent the 45-degree $y = x$ lines.

The two subfigures address qualitatively different dynamic regimes. The origin of **Figure 1(a)** is a sink that attracts propagation trajectories and decreases variable magnitudes. Conversely, the origin of **1(b)** is a source that repels propagation trajectories and increases variable magnitudes. We refer to **1(a)** as the dynamic catch regime; and refer to **1(b)** as the dynamic release regime.

The update of the LSTM cell state is thus a composition of two opposing modifiers. With one that dedicates to its growth, and the other specialised in its shrinkage. It is thus crucial to understand which one, between $\boldsymbol{s}_p$ and $\boldsymbol{s}_q$, serves as the more dominant evolution component. We will then reformulate the more dominant component with deterministic properties to redefine the network with interpretable and affirmative qualities.

A key difference between the opposing regimes of **Figure 1** is the boundedness of the characteristic outlines. The sub-dynamic of $\boldsymbol{s}_q$ is limited between ± 1, while that of $\boldsymbol{s}_p$ is unbounded. This implies that $\boldsymbol{s}_q$ contributes less to the additive relationship of $\boldsymbol{s}_k$, and is less impactful than $\boldsymbol{s}_p$. With the dominant component identified, our next target is to understand how $\boldsymbol{s}_p$ affects vanilla LSTMs. The insight will allow us to propose sensible modifications to make LSTM mechanics more interpretable.

3.3 The forget gate and the catch-and-release dynamics

Base on the two insights of, first, that $\boldsymbol{s}_p = \boldsymbol{f}_k \odot \boldsymbol{s}_{k-1}$ serves as the dominant modifier to $\boldsymbol{s}_k$, and second, as each of its dimensional outline $s_{m(k-1)}$ is modified by gradient f_{mk}, we conjecture that the forget gate controls the *overall* gradient of the characteristic outline of the cell state. That is, $\boldsymbol{f}_k$ predominantly controls the stability of the origin, and dictates the dynamic alterations between the catch and the release phases. As a consequence and regardless of $\boldsymbol{s}_q$, values of the cell state are more likely to grow with large $\boldsymbol{f}_k$, and more likely to shrink with small $\boldsymbol{f}_k$.

These inferences will now be visualised with the three rows of **Figure 2**. The first row simulates the near-random dynamics of a vanilla LSTM cell state, whereas the second and the third rows illustrate that transitions in cell state values can be made predictable under controlled forget gates.

3.3.1 The near-random dynamics of a vanilla LSTM cell state

Gated units of LSTMs are not hardwired with deterministic propagation. As a result, they evolve in a near-chaotic fashion. The first row of **Figure 2** simulates behaviours of $\boldsymbol{s}_k$ under unregulated $\boldsymbol{f}_k$.

The three subfigures of the row serves as consecutive instances of a propagation, *i.e.*, $k = [1 \dots 3]$. The presentation follows **Figure 1**, and dot-dash lines are appended to instances $k = 2$ and 3 to show the characteristic outlines of their previous instances. Parameters $f_{m(1\dots3)} = [1, 0.25, 1]$ and $i_{m(1\dots3)} = [1, 0, 0.75]$ are carefully chosen to exaggerate impacts brought to the vanilla LSTM cell state by the near-chaotic evolution of gates. The real scenarios and experimental set-ups that justify the choice of these demonstrative parameters can be found in **Appendix B**.

These settings lead to sudden catches and abrupt releases. The disordered dynamics make it difficult to predict growth and shrinkage in cell state magnitudes. It is also unclear how such drastic behaviours can assist us to understand the learned features of a trained / optimised LSTM.

3.3.2 Interpretable cell state dynamics under controlled forget gates

Let us now consider cell state dynamics under controlled $\boldsymbol{f}_k$s. The second row has $f_{mk} = 0.9$ with $i_{m(1\dots3)} = [1, 0.75, 0.5]$; and the third row has $f_{mk} = 0.25$ with $i_{m(1\dots3)} = [1, 0.75, 0.15]$. Consistent growth and consistent shrinkage are observed in rows two and three respectively.

The magnitudes evolve monotonically because the controlled forget gates expose cell states to homogeneous dynamic regimes. Large constant f_{mk}s expose the cell state to consistent (*global*) releases; whereas small constant f_{mk}s expose s_{mk} to *global* catches.

Dynamic reversals may occur in $\boldsymbol{s}_k$ through $\boldsymbol{s}_q$, but only under the unlikely incidents where combinations of extreme values of $\boldsymbol{i}_k$ and $\boldsymbol{a}_k$ are simultaneously presented. In addition, the dynamic reversals are merely temporary (*local*) due to the boundedness of values in $\boldsymbol{s}_q$.

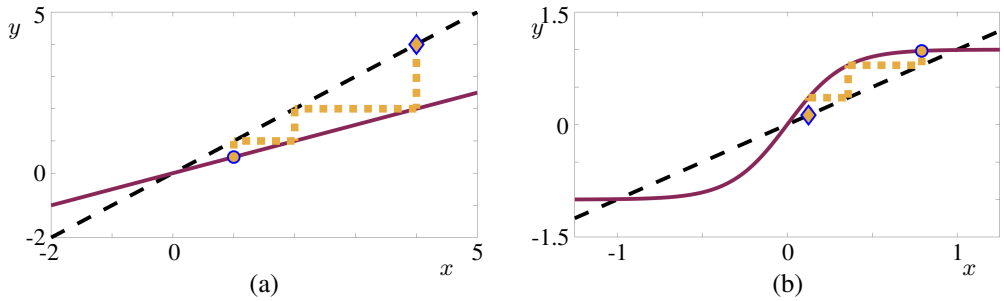


Figure 1: Visualisations of difference equations variable propagation

The dynamic regimes of $\boldsymbol{s}_p$ and $\boldsymbol{s}_q$ behave analogously to the respective difference equations of **(a)**: $x_k = T(x_{k-1}) = 0.5x_{k-1}$ and **(b)**: $x_k = T(x_{k-1}) = \sigma_t(3x_{k-1})$. The propagation, in dots, start from rhombuses and end on circles. The dashed lines represent $y = x$ and the solid lines denotes the characteristic outlines of $y = T(x)$. Vertical mappings from $y = x$ to $y = T(x)$ reflect instantaneous variable computations of $T(x_{k-1})$; and horizontal mappings from $y = T(x)$ to $y = x$ represent the recursive nature of difference equations to re-insert the last output as new input $x_k = T(x_{k-1})$. The origin of **(a)** is a sink which attracts propagation and diminishes variable magnitudes; that of **(b)** is a source which repels propagations and increases variable magnitudes.

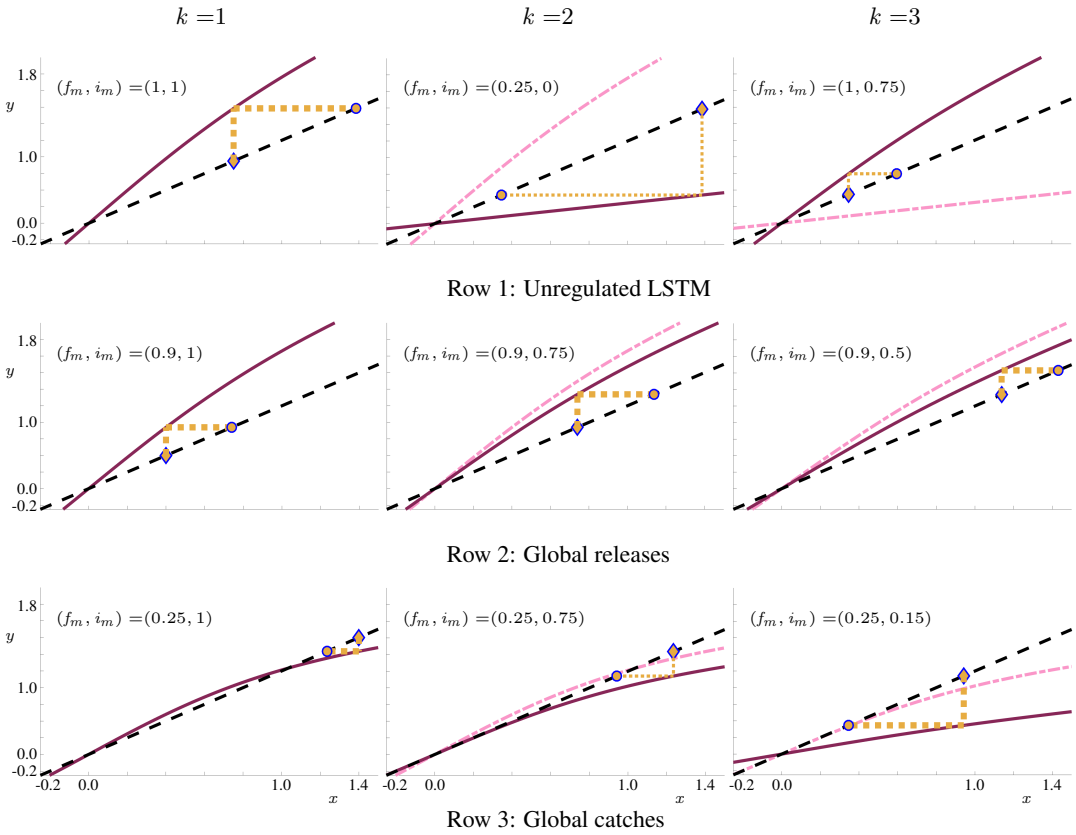


Figure 2: Cell state propagation of controlled and uncontrolled forget gates

Three qualitatively different scenarios on cell state propagation are investigated.
The subfigures follow the presentation format of **Figure 1**; additional dot-dash lines are appended as the characteristic shape of the $k-1$th instance. The characteristic shape of interest is the update of the cell state $y = T(x) = f_{mk}\, x + i_{mk}\, \sigma_t(x)$. From top to bottom, the rows represent for cell state dynamics under an unregulated LSTM, under a large constant forget gate, and under a small constant forget gate, respectively. The latter two scenarios exhibit monotonic propagation because the constant forget gates expose cell states to homogeneous dynamical regimes. The motions are thus more interpretable than those of the unregulated LSTM, where abrupt shrinkage and sudden growth are observed.

4 THE DECAYNET REFORMULATION

This section presents the DecayNet reformulation[1] to calibrate cell state dynamics according to mathematical insights therein **Subsection 3.3**. From **Figure 2**, it is evident that forget gates deprived of random evolution yield predictable and stable cell state propagation. For this reason, we propose the idea to hard-wire forget gate outputs with a continuous and monotonic decrease.

The DecayNet modifies the forget gate according to

System (2): The DecayNet reformulation on the forget gate

the forget-polar input: $\boldsymbol{p}_k = \boldsymbol{p}_{k-1} - \frac{\pi}{2}\frac{1}{6D}(\mathbf{W}_F\boldsymbol{x}_k + \mathbf{W}_{RF}\boldsymbol{q}_{k-1} + 3)$
purposedly presented this way

with $\boldsymbol{p}_0 = \frac{\pi}{2} \cdot \tilde{\mathbf{1}}$,

forget gate elements: $\boldsymbol{f}_{k_i} = \Phi(\boldsymbol{p}_{k_i}) = \begin{cases} 1 & \text{for } \boldsymbol{p}_{k_i} > \frac{\pi}{2} \\ 0 & \text{for } \boldsymbol{p}_{k_i} < 0 \\ \sin(\boldsymbol{p}_{k_i}) & \text{otherwise} \end{cases}$

for i in $1, 2, \ldots N$,

and inherits all remaining variables and their corresponding dimensionalities from **System (1)**. Note, no new learnable parameters are introduced to this system.

The DecayNet initialises the forget gate as a vector of ones,
$\boldsymbol{f}_{0_i} = \sin(\boldsymbol{p}_{0_i}) = \sin(\frac{\pi}{2}) = 1$,
and allows the said ones to decrease over time. For each input shown, variables of the DecayNet-LSTM propagate for D units of time. We want a design that, over time, decays a maximised cumulative amount of 1, and a minimised cumulative amount of 0, to constrain the final value $\boldsymbol{f}_{k_D} \in [0, 1]$.

We implement the sinusoidal function as the "activation function" of the forget gate. In addition, we engineer "pre-activated neurons" with *most* values within the bounds of $[0, \frac{\pi}{2}]$. Let us elaborate.

We replace σ_t with $\sin$ for an ergodic decay over time. The available amount of instantaneous decay needs to be uniform throughout time to ensure that all instantaneous lost in values are equally important. The logistic outline of σ_t yields less changes when magnitudes of pre-activated values are extreme. We remove this undesirable property with the substitution of $\sin$ with radian-like inputs.

Elements of the pre-activated neurons are *mostly* in the bound of $[0, \frac{\pi}{2}]$ because of weights $\mathbf{W}_F$ and $\mathbf{W}_{RF}$. We ran numerous simulations over $\mathbf{W}_F\boldsymbol{x}_k + \mathbf{W}_{RF}\boldsymbol{q}_{k-1}$, with entries of the weights initialised with uniform sampling within the interval of ± 1, and observe that the empirical distributions are *mostly* bounded within the limits of $[-3, 3]$. That is, *ideally* we have

$$\begin{aligned} &\mathbf{W}_F\boldsymbol{x}_k + \mathbf{W}_{RF}\boldsymbol{q}_{k-1} && \in [-3, 3] \text{ and} \\ &\tfrac{\pi}{2}\tfrac{1}{6}(\mathbf{W}_F\boldsymbol{x}_k + \mathbf{W}_{RF}\boldsymbol{q}_{k-1} + 3) && \in [0, \tfrac{\pi}{2}]. \end{aligned}$$

The design in mind is thus, for each instance, a maximum amount of $\frac{\pi}{2D}$ is lost from $\boldsymbol{p}_{k_i}$'s initialised $\frac{\pi}{2}$. In other words, over an input-length of D units of time, an accumulation, minimised at 0 and maximum at $\frac{\pi}{2}$, is lost in the forget-polar input. The former induces the final value $\boldsymbol{p}_{D_i} = \frac{\pi}{2}$ (equivalently $\boldsymbol{f}_{D_i} = 1$), and the latter gives $\boldsymbol{p}_{D_i} = 0$ (equivalently $\boldsymbol{f}_{D_i} = 0$).

DecayNet-LSTMs learn to dichotomise cell state values conditioned on the input shown. One subset of the cell state maintains large magnitudes, and the other decreased to small magnitudes. The former subset corresponds to dimensions of the forget gate that maintain their large magnitude over time; whereas the latter subset corresponds to those that diminish.

More importantly, the monotonic decrease in the forget gate serves as an interpretation to LSTM's inferential decision mechanism. This is because the update of the cell state,
$\boldsymbol{s}_k = \boldsymbol{s}_p + \boldsymbol{s}_q = \boldsymbol{f}_k \odot \boldsymbol{s}_{k-1} + \boldsymbol{i}_k \odot \boldsymbol{a}_k$, is composed of recurrent memory $\boldsymbol{s}_p$ and feedforward memory $\boldsymbol{s}_q$. The irreversible decay indicates that, dimensions of the cell state, that correspond to forget gates that diminish, gradually lose their recurrent nature over time, and effectively function as feedforward neurons. Large forget gate values thus serve as clear indicators for, those neural dimensions, that the DecayNet-LSTM prefer to reiterate, for the input-driven optimisation.

[1] Implementation of our algorithm will be available at https://github.com/anonymous_author/decaynet/

5 RELATED STUDIES

The forward pass of LSTMs (and RNNs) consists high dimensional variables with coupled relationships; and is mostly understood with observational studies and search studies. The former explores practical network usages, and the latter finds optimal combinations of network variables.

Observational studies suggest that cell states develop specific functionality. In text (Karpathy et al., 2015), they are observed to act as line length counters and quotation machines; whereas in speech (Wu & King, 2016), they are highly correlated to the Mel-Cepsrtal coefficient. Search studies (Greff et al., 2017; Chung et al., 2014; Jozefowicz et al., 2015) found that, no particular combinatory sets of variables yield an architecture, that consistently outperforms LSTM in all experimental conditions.

Though rigorous and informative, these studies are based on *exhibited qualities of optimised LSTMs*. Karpathy et al. (2015) conjectured the functionality of trained LSTM cell states through examining post-activated cell state values $\sigma_t(\boldsymbol{s}_k)$. Greff et al. (2017) observed that gated recurrent units (GRUs) (Cho et al., 2014) are capable of performing well on tasks unoptimised by those LSTMs deprived of output gates. Hence they conjectured that GRUs' coupling of the input gate and the forget gate, and the output gate of LSTMs prevent unbounded cell state. In contrast, our theoretical mathematical approach therein **Sections 3** and **4** are conducted prior to the training of LSTMs.

Mathematical properties of neural networks are mostly studied through the lens of statistical learning theory. This paper took an alternative approach. We treated neural networks as dynamical systems, and offered the perspective of understanding LSTMs as systems of difference (discrete) equations.

Statistical studies and dynamical studies are complementary. Burger & Neubauer (2003) took an inverse problem approach and integrated Tikhonov regularisations to enhance neural network stability. Tallec & Ollivier (2018) offers another theoretical study on the forward pass. Incremental updates of LSTMs are treated as differential equations, and the effects of multiple scales are analysed.

The mathematical analyses therein this paper differs to existing literature because, we propose modifications on *a specific* variable, to yield *affirmative / deterministic* outcome. The DecayNet reformulation coerces, the LSTM *forget gate*, to *monotonically decrease* over time.

A *hard* regularisation emerges from the reformulation. As mentioned in **Section 4**, the network learns to *irreversibly* suppress the recurrent nature of a selection of cell state values conditioned on the input shown. This differs to soft regularisations such as Dropout (Srivastava et al., 2014) and Zoneout (Krueger et al., 2016), where the respective nullified and frozen activation are recoverable.

6 EXPERIMENTS AND RESULTS

6.1 IMAGE CLASSIFICATION

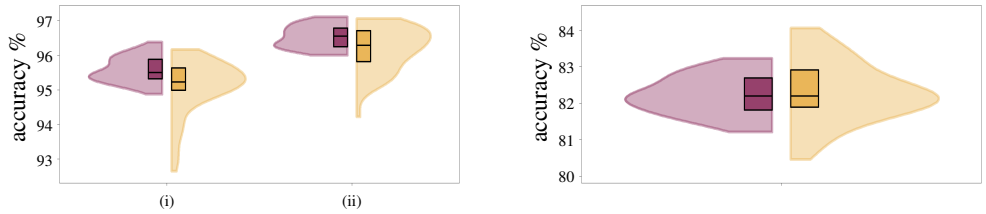
We compare the DecayNet-LSTM against the vanilla LSTM on MNIST (LeCun et al., 1998) and on Fashion-MNIST (Xiao et al., 2017). MNIST is a database of handwritten digits of numbers [0-9], and Fashion-MNIST contains images of fashion products from 10 categories. Both database have 60K and 10K images for training and testing respectively; all images are in 28×28-pixel formats.

Networks of this subsection have 1 hidden layer, and are trained on the Adam optimiser (Kingma & Ba, 2014) with learning rate 0.01. A softmax classifier is attached to produce prediction from the final hidden state $\boldsymbol{q}_D$. Prior to training, no normalisation is applied to MNIST, and a normalisation of mean and standard deviation (MSD)$= (0.5, 0.5)$ is applied to Fashion-MNIST. The networks are not regulated through other means. For each task, the accuracy of 50 randomly initialised simulations are collected from each competing model. The results and statistics are listed in **Table 1**, with important numbers in bold font. Note, the term MSD will reappear later.

6.1.1 EXPERIMENT 1: COMPARISONS WITH SAME AMOUNT OF PARAMETERS

We conducted two comparisons on DecayNets and LSTMs. The first under the setting of 32 hidden dimensions (HD) and the second with 64 HD. All networks were trained on MNIST for 1 epoch.

Within each HD-group, the best performancing DecayNets outperformed LSTMs. The results, in prediction accuracy, were, 96.4% vs 96.2% for 32 HD, and 97.1% vs 97.0% for 64 HD.



(a) (b)

Figure 3: Back-to-back violin plots with side-by-side box plots

Result visualisations for Experiment 1 and Experiment 2, with purple for DecayNet and yellow for LSTM. Subplot **(a)** corresponds to Experiment 1 for MNIST. Comparisons (i) and (ii) are on 32 and 64 HD networks. The best performing DecayNet outperforms the best performing LSTM; and DecayNets have more consistent results than LSTMs. Subplot **(b)** corresponds to Experiment 2 for 48 HD DecayNets vs 64 HD LSTMs on Fashion-MNIST. The best performing LSTM outperforms the best performing DecayNet; but arbitrary performances of DecayNets are comparable with LSTMs, and are more consistent.

Table 1: Results of Experiments 1 & 2

Data	Model	Accuracy (%)	t-test (p)	Surpass	Best result (%)
MNIST	LSTM (32 HD)	95.1 ± 0.2	- -	- -	96.2
	DecayNet (32 HD)	$\mathbf{95.6 \pm 0.1}$	$< 10^{-4}$	$\surd$	96.4
	LSTM (64 HD)	96.2 ± 0.2	- -	- -	97.0
	DecayNet (64 HD)	$\mathbf{96.5 \pm 0.1}$	$< 10^{-4}$	$\surd$	97.1
Fashion-MNIST	LSTM (64 HD)	82.3 ± 0.3	- -	- -	84.1
	DecayNet (48 HD)	$82.2 \pm \mathbf{0.2}$	0.5197	- -	83.2

Two independent sample t-tests showed that DecayNets yielded **significantly statistically greater** accuracy than LSTMs. For 32 HD models, DecayNets gave MSD = (95.6%,0.0037) against LSTMs' MSD = (95.1%,0.0078), with p-values $< 10^{-4}$. For 64 HD models, DecayNets gave MSD = (96.5,0.0033) against LSTMs' MSD = (96.2%,0.0064), with p-values $< 10^{-4}$. We present back-to-back violin plots with side-by-side box plots in **Figure 3(a)**. Purple is used to denote DecayNets, and yellow is for LSTMs. The box plots show that the inter-quatile ranges of DecayNets are smaller than those of LSTMs, implying that the DecayNets yielded more consistent and more reliable results.

6.1.2 Experiment 2: Comparisons with different amount of parameters

For Fashion-MNIST, we compare 48 HD DecayNets against 64 HD LSTMs over 1 epoch of training. The DecayNets of this task have 37.44% less parameters than their rivaling LSTMs.

The best performing LSTMs outperformed DecayNets, in prediction accuracy, by 84.1% vs 83.2%. However, two independent sample t-test showed that there were **no statistical differences** in the mean accuracy for DecayNets and LSTMs. DecayNets have MSD = (82.2%,0.0054) against LSTMs' MSD = (82.3%,0.0084), with p-values > 0.5. The result visualisations in **Figure 3(b)**, again, shows that DecayNets yield more consistent results than LSTMs.

6.2 Sequence classification

A different type of training is performed on MNIST. Pixels of MNIST images are processed, one at a time, as a $784 (= 28 \times 28)$-unit pixel-by-pixel sequence to RNNs therein this subsection. Following Le et al. (2015), we apply a fixed random permutation over the pixels for setting up the task commonly known as permuted sequential MNIST (Perm-SeqMNIST).

RNNs of this subsection have a single 100 HD layer trained on the RMSProp optimiser (Tieleman & Hinton, 2012) over 150 epoches. The learning rate is 0.001, with 0.9 momentum and no weight decay. Gradient clipping is applied at 1 to avoid exploding gradients (Pascanu et al., 2012). All weights are initialised with uniform sampling within the interval of ± 1, and all biases are intialised

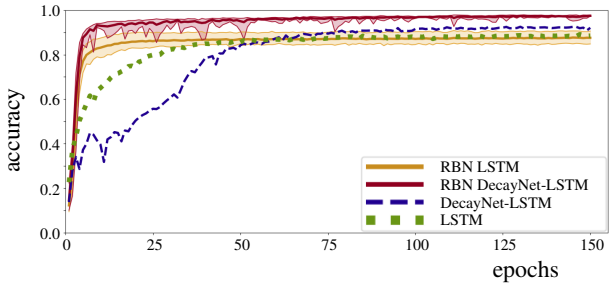


Figure 4: Testing accuracy of the RNNs on Perm-SeqMNIST

Purple and blue denotes DecayNet-LSTM, while green and yellow is used for LSTM. Performances of RBN RNNs are thick solid lines over shades. Bottoms and tops of the shades are the worst and the best performance of all simulations; and the thick central line is the average performance.

Table 2: Existing competitive results on Perm-SeqMNIST

Model	Accuracy (%)	Source
RBN DecayNet-LSTM	Best: **97.8** ; Mean: **97.4**	This paper
Temporal Convolutional Network	97.2	Bai et al. (2018)
RBN Zoneout LSTM	95.9	Krueger et al. (2016)
3D tLSTM+CN	95.7	He et al. (2017)
Dilated GRU and Dilated CNN	94.6 and 96.7	Chang et al. (2017)
iRNN	82.0	Le et al. (2015)

as zeros. No normalisation is performed on the data prior to training, and a softmax classifier is attached to produce prediction from the final hidden state $\boldsymbol{q}_D$.

The best prediction accuracy for unregulated RNNs is 92.1% for DecayNet-LSTM and 89.5% for LSTM (which roughly matches the baseline model of 89.8% in Krueger et al. (2016)). We then integrated recurrent batch normalisation (RBN) (Cooijmans et al., 2016) to DecayNet-LSTM and found that both the best performing RBN DecayNet-LSTM, 97.8%, and the average simulatory performance, 97.4%, yield better results than the state of the art performance reported in Bai et al. (2018) at 97.2%. **Table 2** presents a list of competitive results in existing literature, and **Figure 4** shows the test set accuracy. Purple and blue denote DecayNet-LSTMs, while yellow and green are used for LSTMs. The best unregulated networks are dashed and dotted; and 10 simulations of RBN RNNs are presented as thick solid lines on shades. Bottoms and tops of the shades are the worst and best performances of all simulations; the central lines are the mean performances. The RBN LSTM results differ to those reported in Cooijmans et al. (2016) and Krueger et al. (2016), we speculate this as the differences in weights and biases initialisation; also, weight decay was not applied.

7 Conclusion

This paper proposes the DecayNet reformulation for LSTMs. The motivation and modification are based on theoretical mathematical clarifications of the LSTM cell state mechanics. The update of the cell state is studied as a difference equation, and we showed that the update alternatives between a catch and a release phase to control shrinkage and growth in its values. We found that the dynamic alteration is predominately controlled by the forget gate and hence proposed the idea to stabilise the cell state through a monotonically decreasing forget gate. The reformulation is easy to implement and introduces no extra learnable parameters. Along with RBN, the DecayNet-LSTM achieved better results than previously reported state of the art performance on the Perm-SeqMNIST task.

However, the true potential of the DecayNet lies in its ability to clarify network dynamics. For instance, it would be interesting to treat decaying patterns of this reformulation as features to analyse interactions among speech utterances and other natural language processing tasks (Suominen, 2014).

LSTMs are still opaque. The authors of this manuscript believe that more mathematical analyses should be focused on component s_q of the cell state. We believe this less controllable, and more diverse sub-dynamic of the cell state controls bifurcations (Glendinning, 1994) in LSTM dynamics.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *International Conference on Learning Representations arXiv preprint arXiv:1409.0473*, 2014.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Geoff Boeing. Visual analysis of nonlinear dynamical systems: chaos, fractals, self-similarity and the limits of prediction. *Systems*, 4(4):37, 2016.

Martin Burger and Andreas Neubauer. Analysis of tikhonov regularization for function approximation by neural networks. *Neural Networks*, 16(1):79–90, 2003.

Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, Xiaodong Cui, Michael Witbrock, Mark A Hasegawa-Johnson, and Thomas S Huang. Dilated recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 77–87, 2017.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Tim Cooijmans, Nicolas Ballas, César Laurent, Çağlar Gülçehre, and Aaron Courville. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025*, 2016.

Paul Glendinning. *Stability, instability and chaos: an introduction to the theory of nonlinear differential equations*, volume 11. Cambridge university press, Cambridge, 1994.

Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

Alex Graves, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE transactions on pattern analysis and machine intelligence*, 31(5):855–868, 2009.

Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2017.

Zhen He, Shaobing Gao, Liang Xiao, Daxue Liu, Hangen He, and David Barber. Wider and deeper, cheaper and faster: Tensorized lstms for sequence learning. In *Advances in Neural Information Processing Systems*, pp. 1–11, 2017.

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

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning*, pp. 2342–2350, 2015.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *AAAI*, pp. 2741–2749, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

David Krueger, Tegan Maharaj, János Kramár, Mohammad Pezeshki, Nicolas Ballas, Nan Rosemary Ke, Anirudh Goyal, Yoshua Bengio, Aaron Courville, and Chris Pal. Zoneout: Regularizing rnns by randomly preserving hidden activations. *arXiv preprint arXiv:1606.01305*, 2016.

Quoc V Le, Navdeep Jaitly, and Geoffrey E Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR, abs/1211.5063*, 2012.

Mike Schuster and Kuldip K Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

Hanna Suominen. Guest editorial: Text mining and information analysis of health documents. *Artificial intelligence in medicine*, 61(3):127–130, 2014.

Corentin Tallec and Yann Ollivier. Can recurrent neural networks warp time? *In International Conference on Learning Representations arXiv preprint arXiv:1804.11188*, 2018.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Zhizheng Wu and Simon King. Investigating gated recurrent neural networks for speech synthesis. *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing (ICASSP) arXiv preprint arXiv:1601.02539*, 2016.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pp. 2048–2057, 2015.

A A BRIEF INTRODUCTION TO COBWEB DIAGRAMS

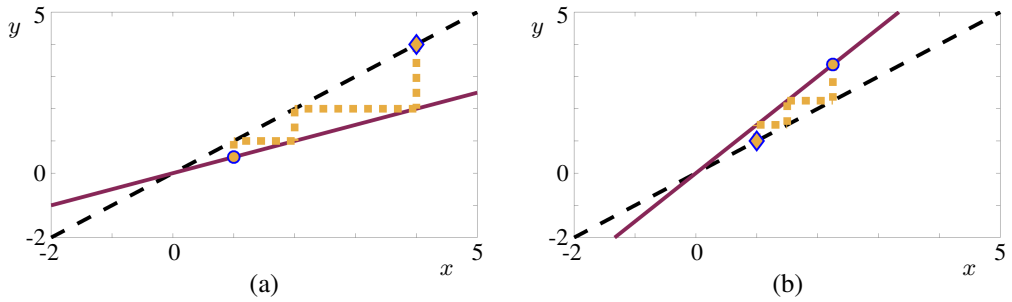


Figure 5: Simple examples of cobwebs

Two simple examples of cobwebs are presented.
The subplots are **(a)** for $y = 0.5x$ and **(b)** for $y = 1.5x$. The presented examples show that the gradients around the vicinity of the equilibria, which are the origins for these scenarios, dictate the stability of the said equilibria. The gradient around the origin of **(a)** is < 1 and hence is a sink that attracts solution / neural trajectoreis; whereas its counterpart in **(b)** is > 1 and hence is a source that repels trajectories. The attractiveness and repulsiveness of the dynamic singularities dictate the growth and shrinkage in variables.

This appendix should only serve as a brief introduction to cobweb diagrams. Interested readers should consult Glendinning (1994) for a broad and complete view on cobweb diagrams, and other dynamical techniques / implications.

Cobweb diagram serves as an easy technique to visualise dynamical motions for variables in low dimensional difference equations. Note, not differential equations. Let us first revisit the concept of difference equations, and then introduce cobweb diagrams.

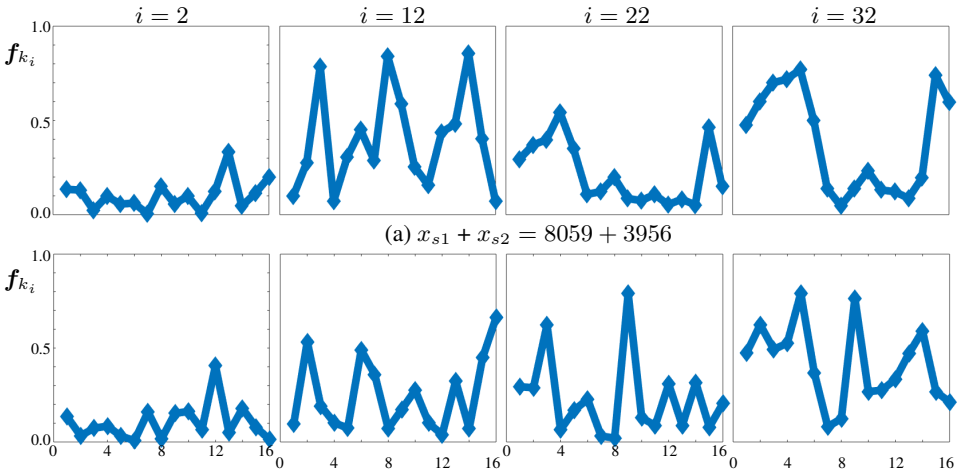
The simplest form of a difference equation is $x_k = T(x_{k-1})$ with $x_0 = \alpha$. Difference equations are simple recursive networks where variable x_k serves as both the input and the output. Subscript k represents computational instances, and T is some arbitrary function.

A cobweb diagram consists 4 fundamental elements. The 45-degree $y = x$ line, the characteristic shape of the operator $y = T(x)$, an initial point $x_0 = \alpha$, and a trail of propagation trajectories. **Figure 5** will be used to support the narrative therein this appendix. The dashed line is $y = x$, the solid line is $y = T(x)$, the initial positions are in rhombuses, the solution trajectories are dotted, and the final positions are in circles.

Cobwebbing is useful because it visualises the repeated insertion of functional outputs. This form of repetition is visualised by two qualitatively different types of mappings: a vertical mapping, and a horizontal mapping. Vertical mappings initiate from $y = x$ and terminate on $y = T(x)$, whereas horizontal mappings start from $y = T(x)$ and end on $y = x$. Vertical mappings represent for functional computations, *i.e.*, given the input of $x_{k-1} \rightarrow$ compute for the output of $T(x_{k-1})$. Their counterparts of horizontal mappings represent for system updates, *i.e.*, given the finished computation of $T(x_{k-1}) \rightarrow$ update the network variable of $x_k = T(x_{k-1})$. This is the reason why the variable trajectories shown in **Figure 5** are zig-zag-like. Not all trajectories are zig-zagged, interested readers can find descriptions of tent map propagation in Boeing (2016).

Another important feature to cobwebbing is the labelling of dynamic singularities known as equilibria. Equilibria of difference equations are the intersections of $y = x$ and $y = T(x)$. These are the positions where, once a trajectory maps into, or starts on them, will never leave. Trajectories move towards attractive / stable equilibria known as sinks, and moves away from repulsive / unstable equilibria known as sources. For instance, the origin of **Figure 5(a)** is a stable equilibria, and that of **5(b)** is an unstable equilibria. While travelling towards the sink, the variable of **5(a)** decreases in magnitudes, i.e., magnitudes of the y-coordinates of the trajectories decrease; and while travelling away the source, the variable of **5(b)** increases in magnitudes. From these two examples, it is clear that the stability of the equilibria are highly dependent on the gradients around their vicinities. The gradient around the origin of **5(a)** and **5(b)** are < 1 and > 1 respectively. More interesting dynamics may emerge with more complicated outlines of $y = T(x)$.

B A JUSTIFICATION ON THE DEMONSTRATIVE PARAMETERS



(a) $x_{s1} + x_{s2} = 8059 + 3956$

(b) $x_{s1} + x_{s2} = 8409 + 29882$

Figure 6: Time series of $\boldsymbol{f}_k$ in an optimised LSTM

We inserted binary forms of inputs x_{s1} and x_{s2} to a LSTM trained to perform summation. In response, the 2nd, 12th, 22nd, and 32nd dimensions of the forget gate exhibited near-random dynamics. We observe extreme dynamic alterations with no clear patterns.

This appendix serves as a justification to the chosen demonstrative parameters in the first row of **Figure 2**. The justification involves 3 items: the vanilla LSTM (see **System (1)**), an additive problem, and records of time series of the forget gate.

The binary string additive problem defined:

The vanilla LSTM is provided with two numbers x_{s1} and x_{s2}, in their 16-digit binary form, and is trained to generate the correct summation output y_s, also in the 16-digit binary form. Some constraints are applied to ensure that values of both the input and the output do not exceed maximum capacity of the 16-digit binary form. The maximum value of x_{s1} is limited to the rounded value of $\frac{2^{16}-1}{2}$, and that of x_{s2} is limited to the floored value of $\frac{2^{16}-1}{2}$.

The LSTM in this appendix is a one layer network with 2 input dimensions, 32 hidden dimensions, and 1 output dimension. The network is set to run for 16 units of time, and during each instance, one digit from each of the paired input is forwarded to the LSTM. The binary digits are presented from back to forth. For every instance, a feedforward regressive weight is applied onto the hidden state $\boldsymbol{q}_k$, then activated by the σ_t, and further rounded to compute for the output digit of the instance.

The network is trained in gradient descent with backpropagation through time. The learning rate is set as 0.1, and the network is not regularised by other means.

Time series of the forget gate:

After the network is optimised, we randomly generated two pairs of x_{s1} and x_{s2} and inserted them to the network. Time series of the 2nd, 12th, 22nd, and 32nd dimension of the forget gate is shown in **Figure 6**.

Marginal dimensions of the forget gate oscillate in aperiodic and near-random fashions. There are no clear nor repeating patterns, and it is common to have extreme alterations in oscillatory magnitudes. The demonstrative parameters $f_{m(1...3)} = [1, 0.25, 1]$ mimic these behaviours via an initial decrease followed by a sharp increase. In order to provide clear illustrations on the impact to $\boldsymbol{s}_k$ from $\boldsymbol{f}_k$, the three values are chosen to be extreme alterations.