# MC-LSTM: Mass-Conserving LSTM

**Pieter-Jan Hoedt** [*1] **Frederik Kratzert** [*1] **Daniel Klotz** [1] **Christina Halmich** [1] **Markus Holzleitner** [1]
**Grey Nearing** [2] **Sepp Hochreiter** [1 3] **Günter Klambauer** [1]

## Abstract

The success of Convolutional Neural Networks (CNNs) in computer vision is mainly driven by their strong inductive bias, which is strong enough to allow CNNs to solve vision-related tasks with random weights, meaning without learning. Similarly, Long Short-Term Memory (LSTM) has a strong inductive bias toward storing information over time. However, many real-world systems are governed by conservation laws, which lead to the redistribution of particular quantities — e.g. in physical and economical systems. Our novel Mass-Conserving LSTM (MC-LSTM) adheres to these conservation laws by extending the inductive bias of LSTM to model the redistribution of those stored quantities. MC-LSTMs set a new state-of-the-art for neural arithmetic units at learning arithmetic operations, such as addition tasks, which have a strong conservation law, as the sum is constant over time. Further, MC-LSTM is applied to traffic forecasting, modeling a damped pendulum, and a large benchmark dataset in hydrology, where it sets a new state-of-the-art for predicting peak flows. In the hydrology example, we show that MC-LSTM states correlate with real world processes and are therefore interpretable.

## 1. Introduction

**Inductive biases enabled the success of CNNs and LSTMs.** One of the greatest success stories of deep learning are Convolutional Neural Networks (CNNs) (Fukushima, 1980; LeCun & Bengio, 1998; Schmidhuber, 2015; LeCun et al., 2015), whose proficiency can be attributed to their strong inductive bias toward visual tasks (Cohen & Shashua, 2017; Gaier & Ha, 2019). The effect of this inductive bias has been demonstrated by CNNs that solve vision-related tasks with random weights, meaning without learning (He et al., 2016; Gaier & Ha, 2019; Ulyanov et al., 2020). Another success story is Long Short-Term Memory (LSTM) (Hochreiter, 1991; Hochreiter & Schmidhuber, 1997), which has a strong inductive bias toward storing information through its memory cells. This inductive bias allows LSTM to excel at speech, text, and language tasks (Sutskever et al., 2014; Bohnet et al., 2018; Kochkina et al., 2017; Liu & Guo, 2019), as well as timeseries prediction. Even with random weights and only a learned linear output layer, LSTM is better at predicting timeseries than reservoir methods (Schmidhuber et al., 2007). In a seminal paper on biases in machine learning, Mitchell (1980) stated that *"biases and initial knowledge are at the heart of the ability to generalize beyond observed data"*. Therefore, choosing an appropriate architecture and inductive bias for neural networks is key to generalization.

**Mechanisms beyond storing are required for real-world applications.** While LSTM can store information over time, real-world applications require mechanisms that go beyond storing. Many real-world systems are governed by conservation laws related to mass, energy, momentum, charge, or particle counts, which are often expressed through continuity equations. In physical systems, different types of energies, mass or particles have to be conserved (Evans & Hanney, 2005; Rabitz et al., 1999; van der Schaft et al., 1996), in hydrology it is the amount of water (Freeze & Harlan, 1969; Beven, 2011), in traffic and transportation the number of vehicles (Vanajakshi & Rilett, 2004; Xiao & Duan, 2020; Zhao et al., 2017), and in logistics the amount of goods, money or products. A real-world task could be to predict outgoing goods from a warehouse based on a general state of the warehouse, i.e., how many goods are in storage, and incoming supplies. If the predictions are not precise, then they do not lead to an optimal control of the production process. For modeling such systems, certain inputs must be conserved but also redistributed across storage locations within the system. We will refer to conserved inputs as *mass*, but note that this can be any type of conserved quantity. We argue that for modeling such systems, specialized mechanisms should be used to represent locations &

[*]Equal contribution [1]ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria [2]Google Research, Mountain View, CA, USA [3]Institute of Advanced Research in Artificial Intelligence (IARAI). Correspondence to: Pieter-Jan Hoedt <hoedt@ml.jku.at>, Frederik Kratzert <kratzert@ml.jku.at>.

*Proceedings of the* $38^{th}$ *International Conference on Machine Learning*, PMLR 139, 2021. 

whereabouts, objects, or storage & placing locations and thus enable conservation.

**Conservation laws should pervade machine learning models in the physical world.** Since a large part of machine learning models are developed to be deployed in the real world, in which conservation laws are omnipresent rather than the exception, these models should adhere to them automatically and benefit from them. However, standard deep learning approaches struggle at conserving quantities across layers or timesteps (Beucler et al., 2019b; Greydanus et al., 2019; Song & Hopke, 1996; Yitian & Gu, 2003), and often solve a task by exploiting spurious correlations (Szegedy et al., 2014; Lapuschkin et al., 2019). Thus, an inductive bias of deep learning approaches via mass conservation over time in an open system, where mass can be added and removed, could lead to a higher generalization performance than standard deep learning for the above-mentioned tasks.

**A mass-conserving LSTM.** In this work, we introduce Mass-Conserving LSTM (MC-LSTM), a variant of LSTM that enforces mass conservation by design. MC-LSTM is a recurrent neural network with an architecture inspired by the gating mechanism in LSTMs. MC-LSTM has a strong inductive bias to guarantee the conservation of mass. This conservation is implemented by means of left-stochastic matrices, which ensure the sum of the memory cells in the network represents the current mass in the system. These left-stochastic matrices also enforce the mass to be conserved through time. The MC-LSTM gates operate as control units on mass flux. Inputs are divided into a subset of *mass inputs*, which are propagated through time and are conserved, and a subset of *auxiliary inputs*, which serve as inputs to the gates for controlling mass fluxes. We demonstrate that MC-LSTMs excel at tasks where conservation of mass is required and that it is highly apt at solving real-world problems in the physical domain.

**Contributions.** We propose a novel neural network architecture based on LSTM that conserves quantities, such as mass, energy, or count, of a specified set of inputs. We show properties of this novel architecture, called MC-LSTM, and demonstrate that these properties render it a powerful neural arithmetic unit. Further, we show its applicability in real-world areas of traffic forecasting and modeling the damped pendulum. In hydrology, large-scale benchmark experiments reveal that MC-LSTM has powerful predictive quality and can supply interpretable representations.

## 2. Mass-Conserving LSTM

The original LSTM introduced memory cells to Recurrent Neural Networks (RNNs), which alleviate the vanishing

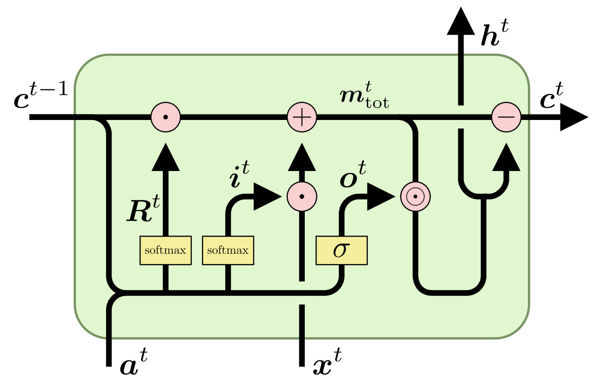


*Figure 1.* Schematic representation of the main operations in the MC-LSTM architecture (adapted from: Olah, 2015).

gradient problem (Hochreiter, 1991). This is achieved by means of a fixed recurrent self-connection of the memory cells. If we denote the values in the memory cells at time $t$ by $\boldsymbol{c}^t$, this recurrence can be formulated as

$$\boldsymbol{c}^t = \boldsymbol{c}^{t-1} + f(\boldsymbol{x}^t, \boldsymbol{h}^{t-1}), \tag{1}$$

where $\boldsymbol{x}$ and $\boldsymbol{h}$ are, respectively, the forward inputs and recurrent inputs, and $f$ is some function that computes the increment for the memory cells. Here, we used the original formulation of LSTM without forget gate (Hochreiter & Schmidhuber, 1997), but in all experiments we also consider LSTM with forget gate (Gers et al., 2000).

MC-LSTMs modify this recurrence to guarantee the conservation of the mass input.The key idea is to use the memory cells from LSTMs as mass accumulators, or mass storage. The conservation law is implemented by three architectural changes. First, the increment, computed by $f$ in Eq. (1), has to distribute mass from inputs into accumulators. Second, the mass that leaves MC-LSTM must also disappear from the accumulators. Third, mass has to be redistributed between mass accumulators. These changes mean that all gates explicitly represent mass fluxes.

Since, in general, not all inputs must be conserved, we distinguish between *mass* inputs, $\boldsymbol{x}$, and *auxiliary* inputs, $\boldsymbol{a}$. The former represents the quantity to be conserved and will fill the mass accumulators in MC-LSTM. The auxiliary inputs are used to control the gates. To keep the notation uncluttered, and without loss of generality, we use a single mass input at each timestep, $x^t$, to introduce the architecture.

The forward pass of MC-LSTM at timestep $t$ can be specified as follows:

$$\boldsymbol{m}^t_{\text{tot}} = \boldsymbol{R}^t \cdot \boldsymbol{c}^{t-1} + \boldsymbol{i}^t \cdot x^t \tag{2}$$

$$\boldsymbol{c}^t = (\boldsymbol{1} - \boldsymbol{o}^t) \odot \boldsymbol{m}^t_{\text{tot}} \tag{3}$$

$$\boldsymbol{h}^t = \boldsymbol{o}^t \odot \boldsymbol{m}^t_{\text{tot}}, \tag{4}$$

where $\boldsymbol{i}^t$ and $\boldsymbol{o}^t$ are the input- and output gates, respectively, and $\boldsymbol{R}$ is a positive left-stochastic matrix, i.e., $\mathbf{1}^T \cdot \boldsymbol{R} = \mathbf{1}^T$, for redistributing mass in the accumulators. The *total mass* $\boldsymbol{m}_{\text{tot}}$ is the *redistributed mass*, $\boldsymbol{R}^t \cdot \boldsymbol{c}^{t-1}$, plus the *mass influx*, or new mass, $\boldsymbol{i}^t \cdot x^t$. The current mass in the system is stored in $\boldsymbol{c}^t$. Finally, $\boldsymbol{h}^t$ is the mass leaving the system.

Note the differences between Eq. (1) and Eq. (3). First, the increment of the memory cells no longer depends on $\boldsymbol{h}^t$. Instead, mass inputs are distributed by means of the normalized $\boldsymbol{i}$ (see Eq. 5). Furthermore, $\boldsymbol{R}^t$ replaces the implicit identity matrix of LSTM to redistribute mass among memory cells. Finally, Eq. (3) introduces $\mathbf{1} - \boldsymbol{o}^t$ as a forget gate on the total mass, $\boldsymbol{m}_{\text{tot}}$. Together with Eq. (4), this assures that no outgoing mass is stored in the accumulators. This formulation has some similarity to Gated Recurrent Units (GRU) (Cho et al., 2014), however MC-LSTM gates are used to split off the output instead of mixing the old and new cell state.

**Basic gating and redistribution.** The MC-LSTM gates at timestep $t$ are computed as follows:

$$\boldsymbol{i}^t = \operatorname{softmax}(\boldsymbol{W}_{\text{i}} \cdot \boldsymbol{a}^t + \boldsymbol{U}_{\text{i}} \cdot \frac{\boldsymbol{c}^{t-1}}{\|\boldsymbol{c}^{t-1}\|_1} + \boldsymbol{b}_{\text{i}}) \quad (5)$$

$$\boldsymbol{o}^t = \sigma(\boldsymbol{W}_{\text{o}} \cdot \boldsymbol{a}^t + \boldsymbol{U}_{\text{o}} \cdot \frac{\boldsymbol{c}^{t-1}}{\|\boldsymbol{c}^{t-1}\|_1} + \boldsymbol{b}_{\text{o}}) \quad (6)$$

$$\boldsymbol{R}^t = \operatorname{softmax}(\boldsymbol{B}_{\text{r}}), \quad (7)$$

where the softmax operator is applied column-wise, $\sigma$ is the logistic sigmoid function, and $\boldsymbol{W}_{\text{i}}$, $\boldsymbol{b}_{\text{i}}$, $\boldsymbol{W}_{\text{o}}$, $\boldsymbol{b}_{\text{o}}$, and $\boldsymbol{B}_{\text{r}}$ are learnable model parameters. The normalization of the input gate and redistribution is required to obtain mass conservation. Note that this can also be achieved by other means than using the softmax function. For example, an alternative way to ensure a column-normalized matrix $\boldsymbol{R}^t$ is to use a normalized logistic, $\tilde{\sigma}(r_{kj}) = \frac{\sigma(r_{kj})}{\sum_n \sigma(r_{kn})}$. Also note that MC-LSTMs directly compute the gates from the memory cells. This is in contrast with the original LSTM, which uses the activations from the previous time step. In this sense, MC-LSTM relies on peephole connections (Gers & Schmidhuber, 2000), instead of the activations from the previous timestep for computing the gates. The accumulated values from the memory cells, $\boldsymbol{c}^t$, are normalized to counter saturation of the sigmoids and to supply probability vectors that represent the current distribution of the mass across cell states. We use this variation e.g. in our experiments with *neural arithmetics* (see Sec. 5.1).

**Time-dependent redistribution.** It can also be useful to predict a redistribution matrix for each sample and timestep, similar to how the gates are computed:

$$\boldsymbol{R}^t = \operatorname{softmax}\left(\mathbf{W}_{\text{r}} \cdot \boldsymbol{a}^t + \mathbf{U}_{\text{r}} \cdot \frac{\boldsymbol{c}^{t-1}}{\|\boldsymbol{c}^{t-1}\|_1} + \boldsymbol{B}_{\text{r}}\right), \quad (8)$$

where the parameters $\mathbf{W}_{\text{r}}$ and $\mathbf{U}_{\text{r}}$ are weight tensors and their multiplications result in $K \times K$ matrices. Again, the softmax function is applied column-wise. This version collapses to a time-independent redistribution matrix if $\mathbf{W}_{\text{r}}$ and $\mathbf{U}_{\text{r}}$ are equal to $\mathbf{0}$. Thus, there exists the option to initialize $\mathbf{W}_{\text{r}}$ and $\mathbf{U}_{\text{r}}$ with weights that are small in absolute value compared to the weights of $\boldsymbol{B}_{\text{r}}$, to favour learning time-independent redistribution matrices. We use this variant in the hydrology experiments (see Sec. 5.4).

**Redistribution via a hypernetwork.** Even more general, a hypernetwork (Schmidhuber, 1992; Ha et al., 2017) that we denote with $g$ can be used to procure $\boldsymbol{R}$. The hypernetwork has to produce a column-normalized, square matrix $\boldsymbol{R}^t = g(\boldsymbol{a}^0, \ldots, \boldsymbol{a}^t, \boldsymbol{c}^0, \ldots, \boldsymbol{c}^{t-1})$. Notably, a hypernetwork can be used to design an *autoregressive* version of MC-LSTMs, if the network additionally predicts auxiliary inputs for the next time step. We use this variant in the pendulum experiments (see Sec. 5.3).

## 3. Properties

**Conservation.** MC-LSTM guarantees that mass is conserved over time. This is a direct consequence of connecting memory cells with stochastic matrices. The mass conservation ensures that no mass can be removed or added implicitly, which makes it easier to learn functions that generalize well. The exact meaning of mass conservation is formalized in the following Theorem.

**Theorem 1** (Conservation property). *Let $m_c^\tau = \sum_{k=1}^K c_k^\tau$ be the mass contained in the system and $m_h^\tau = \sum_{k=1}^K h_k^\tau$ be the mass efflux, or, respectively, the* accumulated mass *in the MC-LSTM storage and the outputs at time $\tau$. At any timestep $\tau$, we have:*

$$m_c^\tau = m_c^0 + \sum_{t=1}^{\tau} x^t - \sum_{t=1}^{\tau} m_h^t. \quad (9)$$

*That is, the change of mass in the memory cells is the difference between the input and output mass, accumulated over time.*

The proof is by induction over $\tau$ (see Appendix C). Note that it is still possible for input mass to be stored indefinitely in a memory cell so that it does not appear at the output. This can be a useful feature if not all of the input mass is needed at the output. In this case, the network can learn that one cell should operate as a collector for excess mass in the system.

**Boundedness of cell states.** In each timestep $\tau$, the memory cells, $c_k^\tau$, are bounded by the sum of mass inputs $\sum_{t=1}^\tau x^t + m_c^0$, that is $|c_k^\tau| \leq \sum_{t=1}^\tau x^t + m_c^0$. Furthermore, if the series of mass inputs converges, $\lim_{\tau\to\infty} \sum_{t=1}^\tau x^\tau =$

$m_x^\infty$, then also the sum of cell states converges (see Appendix, Corollary 1).

**Initialization and gradient flow.** MC-LSTM with $\boldsymbol{R}^t = \boldsymbol{I}$ has a similar gradient flow to LSTM with forget gate (Gers et al., 2000). Thus, the main difference in the gradient flow is determined by the redistribution matrix $\boldsymbol{R}$. The forward pass of MC-LSTM without gates $\boldsymbol{c}^t = \boldsymbol{R}^t \boldsymbol{c}^{t-1}$ leads to the following backward expression $\frac{\partial \boldsymbol{c}^t}{\partial \boldsymbol{c}^{t-1}} = \boldsymbol{R}^t$. Hence, MC-LSTM should be initialized with a redistribution matrix close to the identity matrix to ensure a stable gradient flow as in LSTMs. For random redistribution matrices, the *circular law theorem for random Markov matrices* (Bordenave et al., 2012) can be used to analyze the gradient flow in more detail, see Appendix, Section D.

**Computational complexity.** Whereas the gates in a traditional LSTM are vectors, the input gate and redistribution matrix of an MC-LSTM are matrices in the most general case. This means that MC-LSTM is, in general, computationally more demanding than LSTM. Concretely, the forward pass for a single timestep in MC-LSTM requires $\mathcal{O}(K^3 + K^2(M + L) + KML)$ Multiply-Accumulate operations (MACs), whereas LSTM takes $\mathcal{O}(K^2 + K(M + L))$ MACs per timestep. Here, $M$, $L$ and $K$ are the number of mass inputs, auxiliary inputs and outputs, respectively. When using a time-independent redistribution matrix cf. Eq. (7), the complexity reduces to $\mathcal{O}(K^2M + KML)$ MACs. An empirical runtime comparison is provided in appendix B.6.

**Potential interpretability through inductive bias and accessible mass in cell states.** The representations within the model can be interpreted directly as accumulated mass. If one mass or energy quantity is known, the MC-LSTM architecture would allow to force a particular cell state to represent this quantity, which could facilitate learning and interpretability. An illustrative example is the case of rainfall runoff modelling, where observations, say of the soil moisture or groundwater-state, could be used to guide the learning of an explicit memory cell of MC-LSTM.

## 4. Special Cases and Related Work

**Relation to Markov chains.** In a special case MC-LSTM collapses to a *finite Markov chain*, when $\boldsymbol{c}^0$ is a probability vector, the mass input is zero $x^t = 0$ for all $t$, there is no input and output gate, and the redistribution matrix is constant over time $\boldsymbol{R}^t = \boldsymbol{R}$. For finite Markov chains, the dynamics are known to converge, if $\boldsymbol{R}$ is irreducible (see e.g. Hairer (2018, Theorem 3.13.)). Awiszus & Rosenhahn (2018) aim to model a Markov Chain by having a feed-forward network predict the next state distribution given the current state distribution. In order to insert randomness to the network, a random seed is appended to the input, which allows to simulate Markov processes. Although MC-LSTMs are closely related to Markov chains, they do not explicitly learn the transition matrix, as is the case for Markov chain neural networks. MC-LSTMs would have to learn the transition matrix implicitly.

**Relation to normalizing flows and volume-conserving neural networks.** In contrast to *normalizing flows* (Rezende & Mohamed, 2015; Papamakarios et al., 2019), which transform inputs in each layer and trace their density through layers or timesteps, MC-LSTMs transform distributions and do not aim to trace individual inputs through timesteps. Normalizing flows thereby conserve information about the input in the first layer and can use the inverted mapping to trace an input back to the initial space. MC-LSTMs are concerned with modeling the changes of the initial distribution over time and can guarantee that a multinomial distribution is mapped to a multinomial distribution. For MC-LSTMs without gates, the sequence of cell states $\boldsymbol{c}^0, \ldots, \boldsymbol{c}^T$ constitutes a *normalizing flow* if an initial distribution $p_0(\boldsymbol{c}^0)$ is available. In more detail, MC-LSTM can be considered a *linear flow* with the mapping $\boldsymbol{c}^{t+1} = \boldsymbol{R}^t \boldsymbol{c}^t$ and $p(\boldsymbol{c}^{t+1}) = p(\boldsymbol{c}^t) |\det \boldsymbol{R}^t|^{-1}$ in this case. The gate providing the redistribution matrix (see Eq. 8) is the *conditioner* in a normalizing flow model. From the perspective of normalizing flows, MC-LSTM can be considered as a flow trained in a supervised fashion. Deco & Brauer (1995) proposed volume-conserving neural networks, which conserve the volume spanned by input vectors and thus the information of the starting point of an input is kept. In other words, they are constructed so that the Jacobians of the mapping from one layer to the next have a determinant of 1. In contrast, the determinant of the Jacobians in MC-LSTMs is generally smaller than 1 (except for degenerate cases), which means that volume of the inputs is not conserved.

**Relation to Layer-wise Relevance Propagation.** Layer-wise Relevance Propagation (LRP) (Bach et al., 2015) is similar to our approach with respect to the idea that the sum of a quantity, the relevance $\boldsymbol{Q}^l$ is conserved over layers $l$. LRP aims to maintain the sum of the relevance values $\sum_{k=1}^{K} Q_k^{l-1} = \sum_{k=1}^{K} Q_k^l$ backward through a classifier in order to a obtain relevance values for each input feature.

**Relation to other networks that conserve particular properties.** While a standard feed-forward neural network does not give guarantees aside from the conservation of the proximity of datapoints through the continuity property. The *conservation of the first moments of the data distribution* in the form of normalization techniques (Ioffe & Szegedy, 2015; Ba et al., 2016) has had tremendous success. Here, batch normalization (Ioffe & Szegedy, 2015) could exactly

*Table 1.* Performance of different models on the LSTM addition task in terms of the MSE. MC-LSTM significantly (all $p$-values below .05) outperforms its competitors, LSTM (with high initial forget gate bias), NALU and NAU. Error bars represent 95%-confidence intervals across 100 runs.

| | reference[a] | seq length[b] | input range[c] | count[d] | combo[e] | NaN[f] |
|---|---|---|---|---|---|---|
| MC-LSTM | **0.004** $\pm$ 0.003 | **0.009** $\pm$ 0.004 | **0.8** $\pm$ 0.5 | **0.6** $\pm$ 0.4 | **4.0** $\pm$ 2.5 | 0 |
| LSTM | 0.008 $\pm$ 0.003 | 0.727 $\pm$ 0.169 | 21.4 $\pm$ 0.6 | 9.5 $\pm$ 0.6 | 54.6 $\pm$ 1.0 | 0 |
| NALU | 0.060 $\pm$ 0.008 | 0.059 $\pm$ 0.009 | 25.3 $\pm$ 0.2 | 7.4 $\pm$ 0.1 | 63.7 $\pm$ 0.6 | 93 |
| NAU | 0.248 $\pm$ 0.019 | 0.252 $\pm$ 0.020 | 28.3 $\pm$ 0.5 | 9.1 $\pm$ 0.2 | 68.5 $\pm$ 0.8 | 24 |

[a] training regime: summing 2 out of 100 numbers between 0 and 0.5.
[b] longer sequence lengths: summing 2 out of 1 000 numbers between 0 and 0.5.
[c] more *mass* in the input: summing 2 out of 100 numbers between 0 and 5.0.
[d] higher number of summands: summing 20 out of 100 numbers between 0 and 0.5.
[e] combination of previous scenarios: summing 10 out of 500 numbers between 0 and 2.5.
[f] Number of runs that did not converge.

conserve mean and variance across layers, whereas self-normalization (Klambauer et al., 2017) conserves those approximately. The *conservation of the spectral norm* of each layer in the forward pass has enabled the stable training of generative adversarial networks (Miyato et al., 2018). The *conservation of the spectral norm of the errors* through the backward pass of RNNs has enabled the avoidance of the vanishing gradient problem (Hochreiter, 1991; Hochreiter & Schmidhuber, 1997). In this work, we explore an architecture that exactly *conserves the mass of a subset of the input*, where mass is defined as a physical quantity such as mass or energy.

Similarly, unitary RNNs (Arjovsky et al., 2016; Wisdom et al., 2016; Jing et al., 2017; Helfrich et al., 2018) have been used to resolve the vanishing gradient problem. By using unitary weight matrices, the $L_2$ norm is preserved in both the forward and backward pass. On the other hand, the redistribution matrix in MC-LSTMs assures that the $L_1$ norm is preserved in the forward pass.

**Relation to geometric deep learning.** The field of Geometric Deep Learning (GDL) aims to provide a unification of inductive biases in representation learning (Bronstein et al., 2021). The main tool for this unification is symmetry, which can be expressed in terms of invarant and equivariant functions. From the perspective of GDL, MC-LSTM implements an equivariant mapping on the mass inputs w.r.t shift and scale.

**Relation to neural networks for physical systems.** Neural networks have been shown to discover physical concepts such as the conservation of energies (Iten et al., 2020), and neural networks could allow to learn natural laws from observations (Schmidt & Lipson, 2009; Cranmer et al., 2020b). MC-LSTM can be seen as a neural network architecture with physical constraints (Karpatne et al., 2017; Beucler et al., 2019c). It is however also possible to impose conservation laws by using other means, e.g. initialization, constrained optimization or soft constraints (as, for example, proposed by Karpatne et al., 2017; Beucler et al., 2019c;a; Jia et al., 2019). Hamiltonian Neural Networks (HNNs) (Greydanus et al., 2019) and Symplectic Recurrent Neural Networks (Chen et al., 2019) make energy conserving predictions by using the Hamiltonian, a function that maps the inputs to the quantity that needs to be conserved. By using the symplectic gradients, it is possible to move around in the input space, without changing the output of the Hamiltonian. Lagrangian Neural Networks (Cranmer et al., 2020a), extend the Hamiltonian concept by making it possible to use arbitrary coordinates as inputs.

All of these approaches, while very promising, assume closed physical systems and are thus too restrictive for the application we have in mind. Raissi et al. (2019) propose to enforce physical constraints on simple feed-forward networks by computing the partial derivatives with respect to the inputs and computing the partial differential equations explicitly with the resulting terms. This approach, while promising, does require an exact knowledge of the governing equations. By contrast, our approach is able to learn its own representation of the underlying process, while obeying the pre-specified conservation properties.

## 5. Experiments

In the following, we demonstrate the broad applicability and high predictive performance of MC-LSTM in settings where mass conservation is required[1]. Since there is no quantity to conserve in standard benchmarks for language models, we use benchmarks from areas in which a quantity

[1] Code for the experiments can be found at `https://github.com/ml-jku/mc-lstm`

has to be conserved. We assess MC-LSTM on the benchmarking setting in the area of neural arithmetics (Trask et al., 2018; Madsen & Johansen, 2020; Heim et al., 2020; Faber & Wattenhofer, 2021), in physical modeling on the damped pendulum modeling task by (Iten et al., 2020), and in environmental modeling on flood forecasting (Kratzert et al., 2019b). Additionally, we demonstrate the applicability of MC-LSTM to a traffic forecasting setting. For more details on the datasets and hyperparameter selection for each experiment, we refer to Appendix B.

### 5.1. Arithmetic Tasks

**Addition problem.** We first considered a problem for which exact mass conservation is required. One example for such a problem has been described in the original LSTM paper (Hochreiter & Schmidhuber, 1997), showing that LSTM is capable of summing two arbitrarily marked elements in a sequence of random numbers. We show that MC-LSTM is able to solve this task, but also generalizes better to longer sequences, input values in a different range and more summands. Table 1 summarizes the results of this method comparison and shows that MC-LSTM significantly outperformed the other models on all tests ($p$-value $\leq 0.03$, Wilcoxon test). In Appendix B.1.6, we provide a qualitative analysis of the learned model behavior for this task.

**Recurrent arithmetic.** Following Madsen & Johansen (2020), the inputs for this task are sequences of vectors, uniformly drawn from $[1, 2]^{10}$. For each vector in the sequence, the sum over two random subsets is calculated. Those values are then summed over time, leading to two values. The target output is obtained by applying the arithmetic operation to these two values. The auxiliary input for MC-LSTM is a sequence of ones, where the last element is $-1$ to signal the end of the sequence.

We evaluated MC-LSTM against NAUs and Neural Accumulators (NACs) directly in the framework of Madsen & Johansen (2020). NACs and NAUs use the architecture as presented in (Madsen & Johansen, 2020). That is, a single hidden layer with two neurons, where the first layer is recurrent. The MC-LSTM model has two layers, of which the second one is a fully connected linear layer. For subtraction an extra cell was necessary to properly discard redundant input mass.

For testing, the model with the lowest validation error was used, c.f. early stopping. The performance is measured by the percentage of runs that successfully generalized to longer sequences. Generalization is considered successful if the error is lower than the numerical imprecision of the exact operation (Madsen & Johansen, 2020). The summary in Tab. 2 shows that MC-LSTM was able to significantly outperform the competing models ($p$-value 0.03 for addition and 3e$-$6 for multiplication, proportion test). In Appendix B.1.6, we provide a qualitative analysis of the learned model behavior for this task.

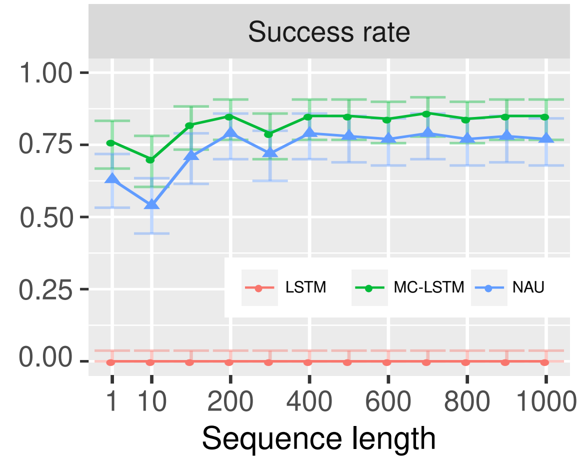


*Figure 2.* MNIST arithmetic task results for MC-LSTM and NAU. The task is to correctly predict the sum of a sequence of presented MNIST digits. The success rates are depicted on the y-axis in dependency of the length of the sequence (x-axis) of MNIST digits. Error bars represent 95%-confidence intervals.

**Static arithmetic.** To enable a direct comparison with the results reported in Madsen & Johansen (2020), we also compared a feed-forward variant of MC-LSTM on the static arithmetic task, see Appendix B.1.3.

**MNIST arithmetic.** We tested that feature extractors can be learned from MNIST images (LeCun et al., 1998) to perform arithmetic on the images (Madsen & Johansen, 2020). This is especially of interest if mass inputs are not given directly, but can be extracted from the available data. The input is a sequence of MNIST images and the target output is the corresponding sum of the labels. Auxiliary inputs are all 1, except the last entry, which is $-1$, to indicate the end of the sequence. The models are the same as in the recurrent arithmetic task with a CNN to convert the images to (mass) inputs for these networks. The network is learned end-to-end. $L_2$-regularization is added to the output of the CNN to prevent its outputs from growing arbitrarily large. The results for this experiment are depicted in Fig. 2. MC-LSTM significantly outperforms the state-of-the-art, NAU ($p$-value 0.002, Binomial test).

### 5.2. Inbound-outbound Traffic Forecasting

We examined the usage of MC-LSTMs for traffic forecasting in situations in which inbound and outbound traffic counts of a city are available (see Fig. 3). For this type of data, a *conservation-of-vehicles* principle (Nam & Drew, 1996) must hold, since vehicles can only leave the city if

*Table 2.* Recurrent arithmetic task results. MC-LSTMs for addition and subtraction/multiplication have two and three neurons, respectively. Error bars represent 95%-confidence intervals.

| | addition | | subtraction | | multiplication | |
|---|---|---|---|---|---|---|
| | success rate[a] | updates[b] | success rate[a] | updates[b] | success rate[a] | updates[b] |
| MC-LSTM | $\mathbf{96}\%^{+2\%}_{-6\%}$ | $4.6 \cdot 10^5$ | $\mathbf{81}\%^{+6\%}_{-9\%}$ | $1.2 \cdot 10^5$ | $\mathbf{67}\%^{+8\%}_{-10\%}$ | $1.8 \cdot 10^5$ |
| LSTM | $0\%^{+4\%}_{-0\%}$ | – | $0\%^{+4\%}_{-0\%}$ | – | $0\%^{+4\%}_{-0\%}$ | – |
| NAU / NMU | $88\%^{+5\%}_{-8\%}$ | $8.1 \cdot 10^4$ | $60\%^{+9\%}_{-10\%}$ | $6.1 \cdot 10^4$ | $34\%^{+10\%}_{-9\%}$ | $8.5 \cdot 10^4$ |
| NAC | $56\%^{+9\%}_{-10\%}$ | $3.2 \cdot 10^5$ | $\mathbf{86}\%^{+5\%}_{-8\%}$ | $4.5 \cdot 10^4$ | $0\%^{+4\%}_{-0\%}$ | – |
| NALU | $10\%^{+7\%}_{-4\%}$ | $1.0 \cdot 10^6$ | $0\%^{+4\%}_{-0\%}$ | – | $1\%^{+4\%}_{-1\%}$ | $4.3 \cdot 10^5$ |

[a] Percentage of runs that generalized to longer sequences.
[b] Median number of updates necessary to solve the task.

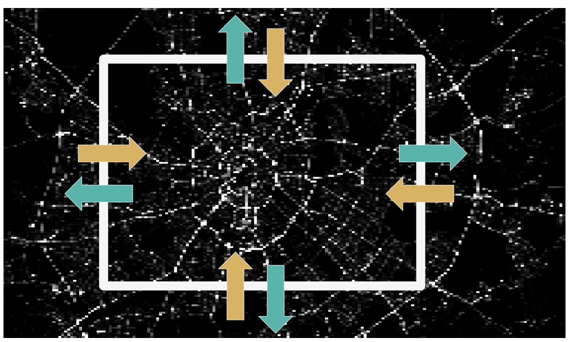

*Figure 3.* Schematic depiction of inbound-outbound traffic situations that require the conservation-of-vehicles principle. All vehicles on outbound roads (yellow arrows) must have entered the city center before (green arrows) or have been present in the first timestep.

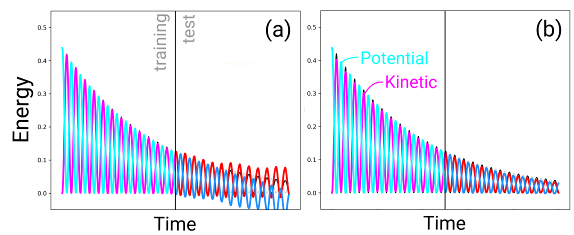


*Figure 4.* Example for the pendulum-modelling exercise. **(a)** LSTM trained for predicting energies of the pendulum with friction in autoregressive fashion, **(b)** MC-LSTM trained in the same setting. Each subplot shows the potential- and kinetic energy and the respective predictions.

they have entered it before or had been there in the first place. Based on data from the traffic4cast 2020 challenge (Kreil et al., 2020), we constructed a dataset to model inbound and outbound traffic in three different cities: Berlin, Istanbul and Moscow. We compared MC-LSTM against LSTM, which is the state-of-the-art method for several types of traffic forecasting situations (Zhao et al., 2017; Tedjopurnomo et al., 2020), and found that MC-LSTM significantly outperforms LSTM in this traffic forecasting setting (all $p$-values $\leq 0.01$, Wilcoxon test). For details, see Appendix B.2.

### 5.3. Damped Pendulum

In the area of physics, we examined the usability of MC-LSTM for the problem of modeling a swinging damped pendulum. Here, the total energy is the conserved property. During the movement of the pendulum, kinetic energy is converted into potential energy and vice-versa. This conversion between both energies has to be learned by the off-diagonal values of the redistribution matrix. A qualitative analysis of a trained MC-LSTM for this problem can be found in Appendix B.3.1.

Accounting for friction, energy dissipates and the swinging slows over time, toward a fixed point. This type of behavior presents a difficulty for machine learning and is impossible for methods that assume the pendulum to be a closed system, such as HNNs (Greydanus et al., 2019) (see Appendix B.3.2). We generated $120$ datasets with timeseries of a pendulum, where we used multiple different settings for initial angle, length of the pendulum, and the amount of friction. We then selected LSTM and MC-LSTM models and compared them with respect to the analytical solution in terms of MSE. For an example, see Fig. 4. Overall, MC-LSTM significantly outperformed LSTM with a mean MSE of $0.01$ (standard deviation $0.02$) compared to $0.07$ (standard deviation $0.14$; with a $p$-value $4.7\text{e}{-10}$, Wilcoxon test). In the friction-free case, no significant difference to HNNs was found (see Appendix B.3.2).

### 5.4. Hydrology: Rainfall Runoff Modeling

We tested MC-LSTM for large-sample hydrological modeling following Kratzert et al. (2019b). An ensemble of

*Table 3.* Hydrology benchmark results. All values represent the median (25% and 75% percentile in sub- and superscript, respectively) over the 447 basins.

| | MC[a] | NSE[b] | $\beta$-NSE[c] | FLV[d] | FHV[e] |
|---|---|---|---|---|---|
| MC-LSTM Ensemble | ✓ | $\mathit{0.744}^{0.814}_{0.641}$ | $-0.020^{0.013}_{-0.066}$ | $-24.7^{31.1}_{-94.4}$ | $\mathbf{-14.7}^{-7.0}_{-23.4}$ |
| LSTM Ensemble | ✗ | $\mathbf{0.763}^{0.835}_{0.676}$ | $-0.034^{-0.002}_{-0.077}$ | $36.3^{59.7}_{-0.4}$ | $\mathit{-15.7}^{-8.6}_{-23.8}$ |
| SAC-SMA | ✓ | $0.603^{0.682}_{0.512}$ | $-0.066^{-0.026}_{-0.108}$ | $37.4^{68.1}_{-31.9}$ | $-20.4^{-12.2}_{-29.9}$ |
| VIC (basin) | ✓ | $0.551^{0.641}_{0.465}$ | $\mathit{-0.018}^{0.032}_{-0.071}$ | $-74.8^{23.1}_{-271.8}$ | $-28.1^{-17.5}_{-40.1}$ |
| VIC (regional) | ✓ | $0.307^{0.402}_{0.218}$ | $-0.074^{0.023}_{-0.166}$ | $18.9^{69.6}_{-73.1}$ | $-56.5^{-38.3}_{-64.6}$ |
| mHM (basin) | ✓ | $0.666^{0.730}_{0.588}$ | $-0.040^{0.003}_{-0.102}$ | $\mathit{11.4}^{65.1}_{-64.0}$ | $-18.6^{-9.5}_{-27.7}$ |
| mHM (regional) | ✓ | $0.527^{0.619}_{0.391}$ | $-0.039^{0.033}_{-0.169}$ | $36.8^{70.9}_{-32.6}$ | $-40.2^{-23.8}_{-51.0}$ |
| HBV (lower) | ✓ | $0.417^{0.550}_{0.276}$ | $-0.023^{0.058}_{-0.114}$ | $23.9^{61.0}_{-25.9}$ | $-41.9^{-17.3}_{-55.2}$ |
| HBV (upper) | ✓ | $0.676^{0.749}_{0.578}$ | $\mathbf{-0.012}^{0.034}_{-0.058}$ | $18.3^{67.5}_{-62.9}$ | $-18.5^{-8.5}_{-27.8}$ |
| FUSE (900) | ✓ | $0.639^{0.715}_{0.539}$ | $-0.031^{0.024}_{-0.100}$ | $\mathbf{-10.5}^{49.2}_{-94.8}$ | $-18.9^{-9.9}_{-27.8}$ |
| FUSE (902) | ✓ | $0.650^{0.727}_{0.570}$ | $-0.047^{-0.004}_{-0.098}$ | $-68.2^{17.1}_{-239.9}$ | $-19.4^{-8.9}_{-27.9}$ |
| FUSE (904) | ✓ | $0.622^{0.705}_{0.527}$ | $-0.067^{-0.019}_{-0.135}$ | $-67.6^{35.7}_{-238.6}$ | $-21.4^{-11.3}_{-33.0}$ |

[a]: *Mass conservation (MC).*
[b]: *Nash-Sutcliffe efficiency:* $(-\infty, 1]$*, values closer to one are desirable.*
[c]: $\beta$*-NSE decomposition:* $(-\infty, \infty)$*, values closer to zero are desirable.*
[d]: *Bottom 30% low flow bias:* $(-\infty, \infty)$*, values closer to zero are desirable.*
[e]: *Top 2% peak flow bias:* $(-\infty, \infty)$*, values closer to zero are desirable.*

10 MC-LSTMs was trained on 10 years of data from 447 basins using the publicly-available CAMELS dataset (Newman et al., 2015; Addor et al., 2017). The mass input is precipitation and auxiliary inputs are: daily min. and max. temperature, solar radiation, and vapor pressure, plus 27 basin characteristics related to geology, vegetation, and climate (described by Kratzert et al., 2019b). All models, apart from MC-LSTM and LSTM, were trained by different research groups with experience using each model. More details are given in Appendix B.4.2.

As shown in Tab. 3, MC-LSTM performed better with respect to the Nash–Sutcliffe Efficiency (NSE; the $R^2$ between simulated and observed runoff) than any other mass-conserving hydrology model, although slightly worse than LSTM.

NSE is often not the most important metric in hydrology, since water managers are typically concerned primarily with extremes (e.g. floods). MC-LSTM performed significantly better ($p = 0.025$, Wilcoxon test) than all models, including LSTM, with respect to high volume flows (FHV), at or above the 98th percentile flow in each basin. This makes MC-LSTM the current state-of-the-art model for flood prediction. MC-LSTM also performed significantly better than LSTM on low volume flows (FLV) and overall bias, however there are other hydrology models that are better for predicting low flows (which is important, e.g. for managing droughts).

**Model states and environmental processes.** It is an open challenge to bridge the gap between the fact that LSTM approaches give generally better predictions than other models (especially for flood prediction) and the fact that water managers need predictions that help them understand not only how much water will be in a river at a given time, but also how water moves through a basin.

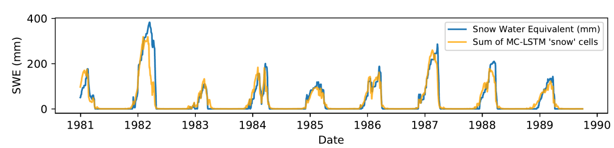


*Figure 5.* Snow-water-equivalent (SWE) from a single basin. The blue line is SWE modeled by Newman et al. (2015). The orange line is the sum over 4 MC-LSTM memory cells (Pearson correlation coefficient $r \geq 0.8$).

Snow processes are difficult to observe and model. Kratzert et al. (2019a) showed that LSTM learns to track snow in memory cells without requiring snow data for training. We found similar behavior in MC-LSTMs, which has the advantage of doing this with memory cells that are *true* mass storages. Figure 5 shows the snow as the sum over a subset of MC-LSTM memory states and snow water equivalent (SWE) modeled by the well-established Snow-17 snow model (Anderson, 1973) (Pearson correlation coefficient $r \geq 0.91$). It is important to note that MC-LSTMs did not have access to any snow data during training. In the best case, it is possible to take advantage of the inductive bias to predict how much water will be stored as snow under different conditions by using simple combinations or mixtures of the internal states. Future work will determine whether this is possible with other difficult-to-observe states and fluxes.

### 5.5. Ablation Study

In order to demonstrate that the design choices of MC-LSTM are necessary together to enable accurate predictive models, we performed an ablation study. In this study, we made changes that disrupt the mass conservation property a) of the input gate, b) the redistribution operation, and c) the output gate. We tested these three variants on data from the hydrology experiments. We chose 5 random basins to limit computational expenses and trained nine repetitions for each configuration and basin. The strongest decrease in performance is observed if the redistribution matrix does not conserve mass, and smaller decreases if input or output gate do not conserve mass. The results of the ablation study indicate that the design of the input gate, redistribution matrix, and output gate, are necessary together to obtain accurate and mass-conserving models (see Appendix Tab. B.8).

## 6. Conclusion

We have demonstrated how to design an RNN that has the property to conserve mass of particular inputs. This architecture is proficient as neural arithmetic unit and is well-suited for predicting physical systems like hydrological processes, in which water mass has to be conserved. We envision that MC-LSTM can become a powerful tool in modeling environmental, sustainability, and biogeochemical cycles.

## Acknowledgments

The ELLIS Unit Linz, the LIT AI Lab, the Institute for Machine Learning, are supported by the Federal State Upper Austria. IARAI is supported by Here Technologies. We thank the projects AI-MOTION (LIT-2018-6-YOU-212), DeepToxGen (LIT-2017-3-YOU-003), AI-SNN (LIT-2018-6-YOU-214), DeepFlood (LIT-2019-8-YOU-213), Medical Cognitive Computing Center (MC3), PRIMAL (FFG-873979), S3AI (FFG-872172), DL for granular flow (FFG-871302), ELISE (H2020-ICT-2019-3 ID: 951847), AIDD (MSCA-ITN-2020 ID: 956832). We thank Janssen Pharmaceutica, UCB Biopharma SRL, Merck Healthcare KGaA, Audi.JKU Deep Learning Center, TGW LOGISTICS GROUP GMBH, Silicon Austria Labs (SAL), FILL Gesellschaft mbH, Anyline GmbH, Google, ZF Friedrichshafen AG, Robert Bosch GmbH, Software Competence Center Hagenberg GmbH, TÜV Austria, and the NVIDIA Corporation.

## References

Addor, N., Newman, A. J., Mizukami, N., and Clark, M. P. The camels data set: catchment attributes and meteorology for large-sample studies. *Hydrology and Earth System Sciences (HESS)*, 21(10):5293–5313, 2017.

Anderson, E. A. National weather service river forecast system: Snow accumulation and ablation model. *NOAA Tech. Memo. NWS HYDRO-17, 87 pp.*, 1973.

Arjovsky, M., Shah, A., and Bengio, Y. Unitary evolution recurrent neural networks. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48, pp. 1120–1128. PMLR, 2016.

Awiszus, M. and Rosenhahn, B. Markov chain neural networks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2261–22617, 2018.

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. arXiv preprint arXiv:1607.06450, 2016.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):1–46, 2015.

Beucler, T., Pritchard, M., Rasp, S., Gentine, P., Ott, J., and Baldi, P. Enforcing analytic constraints in neural-networks emulating physical systems. arXiv preprint arXiv:1909.00912, 2019a.

Beucler, T., Rasp, S., Pritchard, M., and Gentine, P. Achieving conservation of energy in neural network emulators for climate modeling. arXiv preprint arXiv:1906.06622, 2019b.

Beucler, T., Rasp, S., Pritchard, M., and Gentine, P. Achieving conservation of energy in neural network emulators for climate modeling. ICML Workshop "Climate Change: How Can AI Help?", 2019c.

Beven, K. J. *Rainfall-runoff modelling: the primer*. John Wiley & Sons, 2011.

Bohnet, B., McDonald, R., Simoes, G., Andor, D., Pitler, E., and Maynez, J. Morphosyntactic tagging with a meta-bilstm model over context sensitive token encodings. *arXiv preprint arXiv:1805.08237*, 2018.

Bordenave, C., Caputo, P., and Chafai, D. Circular law theorem for random markov matrices. *Probability Theory and Related Fields*, 152(3-4):751–779, 2012.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. arXiv preprint arXiv:2104.13478, 2021.

Chen, Z., Zhang, J., Arjovsky, M., and Bottou, L. Symplectic recurrent neural networks. arXiv preprint arXiv:1909.13334, 2019.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734. Association for Computational Linguistics, 2014.

Cohen, N. and Shashua, A. Inductive bias of deep convolutional networks through pooling geometry. In *International Conference on Learning Representations*, 2017.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P., Spergel, D., and Ho, S. Lagrangian neural networks. arXiv preprint arXiv:2003.04630, 2020a.

Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. Discovering symbolic models from deep learning with inductive biases. In *Advances in Neural Information Processing Systems*, volume 33, 2020b.

Deco, G. and Brauer, W. Nonlinear higher-order statistical decorrelation by volume-conserving neural architectures. *Neural Networks*, 8(4):525–535, 1995. ISSN 0893-6080.

Evans, M. R. and Hanney, T. Nonequilibrium statistical mechanics of the zero-range process and related models. *Journal of Physics A: Mathematical and General*, 38(19): R195, 2005.

Faber, L. and Wattenhofer, R. Neural status registers. arXiv preprint arXiv:2004.07085, 2021.

Freeze, R. A. and Harlan, R. Blueprint for a physically-based, digitally-simulated hydrologic response model. *Journal of Hydrology*, 9(3):237–258, 1969.

Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.

Gaier, A. and Ha, D. Weight agnostic neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pp. 5364–5378. Curran Associates, Inc., 2019.

Gers, F. A. and Schmidhuber, J. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pp. 189–194. IEEE, 2000.

Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with lstm. *Neural Computation*, 12(10):2451–2471, 2000.

Greydanus, S., Dzamba, M., and Yosinski, J. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pp. 15353–15363. Curran Associates, Inc., 2019.

Ha, D., Dai, A., and Le, Q. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Hairer, M. Ergodic properties of markov processes. Lecture notes, 2018.

He, K., Wang, Y., and Hopcroft, J. A powerful generative model using random weights for the deep image representation. In *Advances in Neural Information Processing Systems*, volume 29, pp. 631–639. Curran Associates, Inc., 2016.

Heim, N., Pevný, T., and Šmídl, V. Neural Power Units. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6573–6583. Curran Associates, Inc., 2020.

Helfrich, K., Willmott, D., and Ye, Q. Orthogonal recurrent neural networks with scaled Cayley transform. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pp. 1969–1978. PMLR, 2018.

Hochreiter, S. *Untersuchungen zu dynamischen neuronalen Netzen*. PhD thesis, Technische Universität München, 1991.

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

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 448–456. PMLR, 2015.

Iten, R., Metger, T., Wilming, H., Del Rio, L., and Renner, R. Discovering physical concepts with neural networks. *Physical Review Letters*, 124(1):010508, 2020.

Jia, X., Willard, J., Karpatne, A., Read, J., Zwart, J., Steinbach, M., and Kumar, V. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 558–566. SIAM, 2019.

Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S., LeCun, Y., Tegmark, M., and Soljačić, M. Tunable Efficient Unitary Neural Networks (EUNN) and their application to RNNs. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1733–1741. PMLR, 2017.

Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. In *Advances in neural information processing systems*, volume 30, pp. 971–980, 2017.

Kochkina, E., Liakata, M., and Augenstein, I. Turing at semeval-2017 task 8: Sequential approach to rumour stance classification with branch-lstm. *arXiv preprint arXiv:1704.07221*, 2017.

Kratzert, F., Herrnegger, M., Klotz, D., Hochreiter, S., and Klambauer, G. *NeuralHydrology–Interpreting LSTMs in Hydrology*, pp. 347–362. Springer, 2019a.

Kratzert, F., Klotz, D., Shalev, G., Klambauer, G., Hochreiter, S., and Nearing, G. Towards learning universal, regional, and local hydrological behaviors via machine learning applied to large-sample datasets. *Hydrology and Earth System Sciences*, 23(12):5089–5110, 2019b.

Kreil, D. P., Kopp, M. K., Jonietz, D., Neun, M., Gruca, A., Herruzo, P., Martin, H., Soleymani, A., and Hochreiter, S. The surprising efficiency of framing geo-spatial time series forecasting as a video prediction task–insights from the iarai traffic4cast competition at neurips 2019. In *NeurIPS 2019 Competition and Demonstration Track*, pp. 232–241. PMLR, 2020.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., and Müller, K.-R. Unmasking clever hans predictors and assessing what machines really learn. *Nature communications*, 10(1):1–8, 2019.

LeCun, Y. and Bengio, Y. *Convolutional Networks for Images, Speech, and Time Series*, pp. 255–258. MIT Press, Cambridge, MA, USA, 1998.

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

LeCun, Y., Bengio, Y., and Hinton, G. Deep learning. *Nature*, 521(7553):436–444, 2015.

Liu, G. and Guo, J. Bidirectional lstm with attention mechanism and convolutional layer for text classification. *Neurocomputing*, 337:325–338, 2019.

Madsen, A. and Johansen, A. R. Neural arithmetic units. In *International Conference on Learning Representations*, 2020.

Mitchell, T. M. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers University, Computer Science Department, New Brunswick, NJ, 1980.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. Spectral normalization for generative adversarial networks. In *International Conference on Learning Representations*, 2018.

Nam, D. H. and Drew, D. R. Traffic dynamics: Method for estimating freeway travel times in real time from flow measurements. *Journal of Transportation Engineering*, 122(3):185–191, 1996.

Newman, A., Clark, M., Sampson, K., Wood, A., Hay, L., Bock, A., Viger, R., Blodgett, D., Brekke, L., Arnold, J., et al. Development of a large-sample watershed-scale hydrometeorological data set for the contiguous USA: data set characteristics and assessment of regional variability in hydrologic model performance. *Hydrology and Earth System Sciences*, 19(1):209–223, 2015.

Olah, C. Understanding LSTM networks, 2015. URL `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. Technical report, DeepMind, 2019.

Rabitz, H., Aliş, Ö. F., Shorter, J., and Shim, K. Efficient input—output model representations. *Computer physics communications*, 117(1-2):11–20, 1999.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Rezende, D. and Mohamed, S. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1530–1538. PMLR, 2015.

Schmidhuber, J. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Schmidhuber, J., Wierstra, D., Gagliolo, M., and Gomez, F. Training recurrent networks by Evolino. *Neural Computation*, 19(3):757–779, 2007.

Schmidt, M. and Lipson, H. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Song, X.-H. and Hopke, P. K. Solving the chemical mass balance problem using an artificial neural network. *Environmental science & technology*, 30(2):531–535, 1996.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Tedjopurnomo, D. A., Bao, Z., Zheng, B., Choudhury, F., and Qin, A. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering*, 2020.

Trask, A., Hill, F., Reed, S. E., Rae, J., Dyer, C., and Blunsom, P. Neural arithmetic logic units. In *Advances in Neural Information Processing Systems*, volume 31, pp. 8035–8044. Curran Associates, Inc., 2018.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. Deep image prior. *International Journal of Computer Vision*, 128(7): 1867–1888, 2020.

van der Schaft, A. J., Dalsmo, M., and Maschke, B. M. Mathematical structures in the network representation of energy-conserving physical systems. In *Proceedings of 35th IEEE Conference on Decision and Control*, volume 1, pp. 201–206, 1996.

Vanajakshi, L. and Rilett, L. Loop detector data diagnostics based on conservation-of-vehicles principle. *Transportation research record*, 1870(1):162–169, 2004.

Wisdom, S., Powers, T., Hershey, J. R., Roux, J. L., and Atlas, L. Full-Capacity Unitary Recurrent Neural Networks. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4880–4888. Curran Associates, Inc., 2016.

Xiao, X. and Duan, H. A new grey model for traffic flow mechanics. *Engineering Applications of Artificial Intelligence*, 88:103350, 2020.

Yitian, L. and Gu, R. R. Modeling flow and sediment transport in a river system using an artificial neural network. *Environmental management*, 31(1):0122–0134, 2003.

Zhao, Z., Chen, W., Wu, X., Chen, P. C., and Liu, J. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, 11(2):68–75, 2017.