MC-LSTM: Mass-Conserving LSTM

Pieter-Jan Hoedt¹  Frederik Kratzert¹  Daniel Klotz¹  Christina Halmich¹  Markus Holzleitner¹  
Grey Nearing²  Sepp Hochreiter¹ ³  Günter Klambauer¹

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. 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 & Hä, 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; Lu & Guo, 2019), as well as time-series prediction. Even with random weights and only a learned linear output layer, LSTM is better at predicting time-series 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 &

¹Equal contribution ²ELLIS Unit Linz, LIT AI Lab, Institute for Machine Learning, Johannes Kepler University Linz, Austria ³Google Research, Mountain View, CA, USA ⁴Institute 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 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).
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., 2019; 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 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 $c^t$, this recurrence can be formulated as

$$c^t = c^{t-1} + f(a^t, h^{t-1}),$$

where $a$ and $h$ 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, $a$, and auxiliary inputs, $i$. 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:

$$m^{t}_{tot} = R^t \cdot c^{t-1} + i^t \cdot x^t, \quad (2)$$
$$c^t = (1 - o^t) \odot m^{t}_{tot}, \quad (3)$$
$$h^t = o^t \odot m^{t}_{tot}, \quad (4)$$

![Figure 1. Schematic representation of the main operations in the MC-LSTM architecture (adapted from: Olah, 2015).](image-url)
where $i^t$ and $o^t$ are the input- and output gates, respectively, and $R$ is a positive left-stochastic matrix, i.e., $1^T \cdot R = 1^T$, for redistributing mass in the accumulators. The total mass $m_{\text{tot}}$ is the redistributed mass, $R^t \cdot e^{t-1}$, plus the mass influx, or new mass, $i^t \cdot x^t$. The current mass in the system is stored in $e^t$. Finally, $h^t$ 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 $h^t$. Instead, mass inputs are distributed by means of the normalized $i$ (see Eq. [5]). Furthermore, $R^t$ replaces the implicit identity matrix of LSTM to redistribute mass among memory cells. Finally, Eq. [3] introduces $1 - o^t$ as a forget gate on the total mass, $m_{\text{tot}}$. 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:

$$i^t = \text{softmax}(W_i \cdot a^t + U_i \cdot e^{t-1} + b_i)$$ (5)

$$o^t = \sigma(W_o \cdot a^t + U_o \cdot e^{t-1} + b_o)$$ (6)

$$R^t = \text{softmax}(B_r),$$ (7)

where the softmax operator is applied column-wise, $\sigma$ is the logistic sigmoid function, and $W_i$, $b_i$, $W_o$, $b_o$, and $B_r$ 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 $R^t$ is to use a normalized logistic, $\sigma(r_{kj}) = \frac{\sigma(r_{kj})}{\sum_k \sigma(r_{kj})}$. Also note that MC-LSTMs directly compute the gates from the memory cells. This is in contrast with original LSTM, which uses the activations from the previous timestep. 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, $e^t$, are normalized to counter saturation of the sigmoid 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:

$$R^t = \text{softmax} \left( W_r \cdot a^t + U_r \cdot e^{t-1} + B_r \right),$$ (8)

where the parameters $W_r$ and $U_r$ 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 $W_r$ and $U_r$ are equal to 0. Thus, there exists the option to initialize $W_r$ and $U_r$ with weights that are small in absolute value compared to the weights of $B_r$, 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 $R$. The hypernetwork has to produce a column-normalized, square matrix $R^t = g(a^0,\ldots,a^t,e^0,\ldots,e^{t-1})$. 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 $m^t = \sum_{k=1}^K c^\tau_k$ be the mass contained in the system and $m^t = \sum_{k=1}^K h^\tau_k$ 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:

$$m^\tau = m^0 + \sum_{t=1}^\tau x^t \sum_{i=1}^\tau m^0_i.$$ (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, $c^\tau_k$, are bounded by the sum of mass inputs $\sum_{t=1}^\tau x^t + m^0_k$, that is $|c^\tau_k| \leq \sum_{t=1}^\tau x^t + m^0_k$. Furthermore, if the series of mass inputs converges, $\lim_{\tau \to \infty} \sum_{t=1}^\tau x^\tau = 0$. Therefore, the memory cells will also be bounded in this case.
MC-LSTM

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

**Initialization and gradient flow.** MC-LSTM with $R^t = I$ 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 $R$. The forward pass of MC-LSTM without gates $c^t = R^t c^{t-1}$ leads to the following backward expression $\frac{\partial c^t}{\partial c^{t-1}} = R^t$. 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 $O(K^3 + K^2(M + L) + KML)$ Multiply-Accumulate operations (MACs), whereas LSTM takes $O(K^2 + K(M + L))$ 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 $O(K^2 M + KML)$ 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 $c^0$ 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 $R^t = R$. For finite Markov chains, the dynamics are known to converge, if $R$ 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 $c^0, \ldots, c^T$ constitutes a normalizing flow if an initial distribution $p_0(c^0)$ is available. In more detail, MC-LSTM can be considered a linear flow with the mapping $c^{t+1} = R^t c^t$ and $R(e^{t+1}) = R(e^t) | det R|^{-1}$ 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 $Q^l_i$ is conserved over layers $l$. LRP aims to maintain the sum of the relevance values $\sum_{k=1}^K Q^l_k = \sum_{k=1}^K Q^0_k$ backward through a classifier in order to 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.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Seq Length</th>
<th>Input Range</th>
<th>Count</th>
<th>Combo</th>
<th>Nat</th>
</tr>
</thead>
<tbody>
<tr>
<td>MC-LSTM</td>
<td>0.004 ± 0.003</td>
<td>0.009 ± 0.004</td>
<td>0.8 ± 0.5</td>
<td>0.6 ± 0.4</td>
<td>4.0 ± 2.5</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.008 ± 0.003</td>
<td>0.727 ± 0.169</td>
<td>21.4 ± 0.6</td>
<td>9.5 ± 0.6</td>
<td>54.6 ± 1.0</td>
</tr>
<tr>
<td>NALU</td>
<td>0.060 ± 0.008</td>
<td>0.059 ± 0.009</td>
<td>25.3 ± 0.2</td>
<td>7.4 ± 0.1</td>
<td>63.7 ± 0.6</td>
</tr>
<tr>
<td>NAU</td>
<td>0.248 ± 0.019</td>
<td>0.252 ± 0.020</td>
<td>28.3 ± 0.5</td>
<td>9.1 ± 0.2</td>
<td>68.5 ± 0.8</td>
</tr>
</tbody>
</table>

a: training regime: summing 2 out of 100 numbers between 0 and 0.5.
b: longer sequence lengths: summing 2 out of 1000 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.

Relate to neural networks for physical systems. Neural networks have been shown to discover physical concepts such as the conservation of energy (Iten et al., 2020), and neural networks could allow to learn natural laws from observations (Schmidt & Lipson, 2009; Cranner et al., 2020a). 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; 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. Rass 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. Since there is no quantity to conserve in standard benchmarks for language models, we use benchmarks from areas in which a quantity...
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 precision 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.

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.

<table>
<thead>
<tr>
<th></th>
<th>addition</th>
<th></th>
<th>subtraction</th>
<th></th>
<th>multiplication</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>success</td>
<td>updates</td>
<td>success</td>
<td>updates</td>
<td>success</td>
<td>updates</td>
</tr>
<tr>
<td></td>
<td>rate(^a)</td>
<td></td>
<td>rate(^a)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MC-LSTM</td>
<td>96% ±2%</td>
<td>4.6 · 10(^5)</td>
<td>81% ±6%</td>
<td>1.2 · 10(^5)</td>
<td>67% ±8%</td>
<td>1.8 · 10(^5)</td>
</tr>
<tr>
<td>LSTM</td>
<td>0% ±0%</td>
<td>–</td>
<td>0% ±0%</td>
<td>–</td>
<td>0% ±0%</td>
<td>–</td>
</tr>
<tr>
<td>NAU / NMU</td>
<td>88% ±5%</td>
<td>8.1 · 10(^4)</td>
<td>60% ±9%</td>
<td>6.1 · 10(^4)</td>
<td>34% ±10%</td>
<td>8.5 · 10(^4)</td>
</tr>
<tr>
<td>NAC</td>
<td>56% ±9%</td>
<td>3.2 · 10(^3)</td>
<td>86% ±5%</td>
<td>4.5 · 10(^3)</td>
<td>0% ±4%</td>
<td>–</td>
</tr>
<tr>
<td>NALU</td>
<td>10% ±7%</td>
<td>1.0 · 10(^6)</td>
<td>0% ±4%</td>
<td>–</td>
<td>1% ±4%</td>
<td>4.3 · 10(^5)</td>
</tr>
</tbody>
</table>

\(^a\) 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.

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 \(4.7e^{-10}\), 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
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.3.2

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.

\begin{table}
\centering
\begin{tabular}{llllll}
\hline
 & MC & NSE & $\beta$-NSE & FLV & FHV \\
\hline
MC-LSTM Ensemble & ✓ & 0.744 & -0.020 & -24.7 & -14.7 \\
LSTM Ensemble & ✓ & 0.763 & 0.003 & 26.3 & -15.7 \\
SAC-SMA (basin) & ✓ & 0.603 & -0.066 & -37.4 & -20.4 \\
VIC (basin) & ✓ & 0.551 & -0.018 & -74.8 & -28.1 \\
VIC (regional) & ✓ & 0.307 & -0.074 & 18.9 & -56.5 \\
mHM (basin) & ✓ & 0.666 & -0.040 & 11.4 & -18.6 \\
mHM (regional) & ✓ & 0.527 & -0.039 & 36.8 & -40.2 \\
HBV (lower) & ✓ & 0.417 & -0.022 & 23.9 & -41.9 \\
HBV (upper) & ✓ & 0.676 & -0.012 & 18.3 & -18.5 \\
FUSE (900) & ✓ & 0.639 & -0.031 & -10.5 & -7.5 \\
FUSE (902) & ✓ & 0.650 & -0.047 & -68.2 & -19.4 \\
FUSE (904) & ✓ & 0.622 & -0.067 & -67.6 & -21.4 \\
\hline
\end{tabular}
\caption{Hydrology benchmark results. All values represent the median (25% and 75% percentile in sub- and superscript, respectively) over the 447 basins.}
\end{table}

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.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{snow_water_equivalent.png}
\caption{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.3]).

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), AISNN (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), ADD (MSCA-ITN-2020 ID: 956832). We thank Janssen Pharmaceutica, UCB Biopharma SRL, Merck Healthcare KGaA, AudiJKU 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


