Working Memory Connections for LSTM

Federico Landi[a,*], Lorenzo Baraldi[a], Marcella Cornia[a], Rita Cucchiara[a]

[a] *Department of Engineering “Enzo Ferrari”, University of Modena and Reggio Emilia, Modena, Italy*

Abstract

Recurrent Neural Networks with Long Short-Term Memory (LSTM) make use of gating mechanisms to mitigate exploding and vanishing gradients when learning long-term dependencies. For this reason, LSTMs and other gated RNNs are widely adopted, being the standard *de facto* for many sequence modeling tasks. Although the memory cell inside the LSTM contains essential information, it is not allowed to influence the gating mechanism directly. In this work, we improve the gate potential by including information coming from the internal cell state. The proposed modification, named Working Memory Connection, consists in adding a learnable nonlinear projection of the cell content into the network gates. This modification can fit into the classical LSTM gates without any assumption on the underlying task, being particularly effective when dealing with longer sequences. Previous research effort in this direction, which goes back to the early 2000s, could not bring a consistent improvement over vanilla LSTM. As part of this paper, we identify a key issue tied to previous connections that heavily limits their effectiveness, hence preventing a successful integration of the knowledge coming from the internal cell state. We show through extensive experimental evaluation that Working Memory Connections constantly improve the performance of LSTMs on a variety of tasks. Numerical results suggest that the cell state contains useful information that is worth including in the gate structure.

Keywords: Long Short-Term Memory Networks, Cell-to-Gate Connections, Gated RNNs, Language Modeling, Image Captioning

1. Introduction

Recurrent Neural Networks (RNNs) (Elman, 1990; Rumelhart et al., 1986) are a family of architectures that process sequential data by means of internal hidden states. The set of parameters of the network is shared across time steps, allowing the RNN to process inputs of variable length. As RNNs suffer from the so-called exploding and vanishing gradient problem (EVGP) (Bengio et al., 1993; Hochreiter, 1991), which hinders the learning of long-term dependencies (Bengio et al., 1994; Pascanu et al., 2013), previous works have proposed to enrich the recurrent cell with

[*] Corresponding author:
Email address: federico.landi@unimore.it (Federico Landi)

This is the authors version of an article accepted for publication in Neural Networks

gating mechanisms (Hochreiter and Schmidhuber, 1997; Jing et al., 2019). For instance, Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) use gates to control the information flow towards and from the memory cell and to regulate the forgetting process (Gers et al., 2000). LSTMs are adopted in a wide number of tasks, such as neural machine translation (Bahdanau et al., 2015; Sutskever et al., 2014), speech recognition (Graves et al., 2013), and also vision-and-language applications like image and video captioning (Vinyals et al., 2015; Xu et al., 2015; Baraldi et al., 2017).

In this paper, we propose a novel cell-to-gate connection that modifies the classic LSTM block. Our formulation is general and improves LSTM overall performance and training stability without any particular assumption on the underlying task. In the *vanilla* LSTM formulation, the gates are controlled by the current input of the block and its previous output, which acts as the hidden state for the network. The long-term memory cell, instead, is employed to store information during the forward pass and provides a safe path for back-propagating the error signal. We argue that the content stored in the memory cell could be useful to regulate the gating mechanisms, too. The key element of our design is a connection between the memory cell and the gates with a protection mechanism that prevents the cell state from being exposed directly. We draw inspiration from the gated *read* operation employed to reveal the cell content at the block output, and enrich it with a learnable projection. In this way, the LSTM block can use the knowledge in the cell (acting as a long-term memory) to control the evolution of the whole network in the short-term.

A similar concept in cognitive psychology and neuroscience is the so-called *working memory* (Ericsson and Kintsch, 1995), a type of memory employed, for instance, *to retain the partial results while solving an arithmetic problem without paper, or to combine the premises in a lengthy rhetorical argument* (Hernandez, 2018). Although definitions are not unanimous, working memory is said to be a cognitive system acting as a third type of memory between long-term and short-term memory. Our connections share this characteristic with working memory. For this reason, we call them *Working Memory Connections* (WMCs).

A first attempt to fuse the information of the cell in the gates was made with the design of *peepholes* (Gers and Schmidhuber, 2000): direct multiplicative connections between the memory cell and the gates. This approach has not been largely adopted in literature, as recent studies report mixed results (Greff et al., 2017) and discourage their use. Since our idea recalls the rationale of peephole connections, we provide a large comparison with this previous work. By doing so, we point out the major issues in the peephole formulation that hinder effective learning and attest that WMCs do not suffer from the same problems. In our experiments, we show that an LSTM equipped with Working Memory Connections achieves better results than comparable architectures, thus reflecting the theoretical advantages of their design. In particular, WMCs surpass vanilla LSTM and peephole LSTM in terms of final performances, stability during training, and convergence time. All these aspects testify the advantage in letting the cell state participate in the gating dynamics. In order to support our conclusions, we conduct a thorough experimental analysis covering a wide area of current research topics.

To sum up, our contribution is mainly three-fold. First, we present a modification of LSTM in which traditional gates are enriched with Working Memory Connections, linking the memory cell with the gates through a protection mechanism. Then, we demonstrate that exposing the LSTM internal state directly and without a proper protection yields unstable training dynamics that compromise the final performance. Finally, we show the effectiveness of the proposed solution in a variety of tasks, ranging from toy problems with very long-term dependencies (adding problem, copy task, and sequential MNIST) to language modeling and image captioning.

2. Related Work

Long Short-Term Memory networks (Hochreiter and Schmidhuber, 1997) aim to mitigate the exploding and vanishing gradient problem (Hochreiter, 1991; Bengio et al., 1994) with the use of gating mechanisms. Since its introduction, LSTM has gained a lot of attention for its flexibility and efficacy in many different tasks. To simplify the LSTM structure, Liu et al. (2020) propose to exploit the content of the long-term memory cell in a recurrent block with only two gates. However, this model neglects the importance of the LSTM output. While this might be useful for simple tasks, it is unlikely to generalize to more complex settings. Arpit et al. (2019) propose to modify the path of the gradients in order to stabilize training with a stochastic algorithm specific to LSTM optimization. This direction of work is not in contrast with our goal, and could possibly be integrated with our proposal since our connection does not require a specific setup to be optimized. Among the LSTM variants, the Gated Recurrent Unit (GRU) (Cho et al., 2014b,a) is the most popular and common architecture (Chung et al., 2014), and features a coupling mechanism between input and forget gates (Greff et al., 2017). A recent line of research aims to tailor the LSTM structure for specific tasks. For instance, Baraldi et al. (2017) propose a hierarchical model for video captioning, while other works incorporate convolutional models into the LSTM structure (Xiao et al., 2020; Li et al., 2018). While these works propose a modification of the LSTM towards a specific goal, we propose a general and powerful idea that adapts to a large set of different tasks.

Recently, models based on self-attention, such as Transformer architecture (Vaswani et al., 2017) and its variants, are achieving state-of-art performances on many different tasks, and also for sequence modeling. For instance, language representations based on BERT (Devlin et al., 2018) can be finetuned with an additional output layer to obtain state-of-art results on many language-based tasks. However, RNNs require much fewer parameters and operations to run than Transformer-based architectures and are still widely adopted. Moreover, LSTMs still have a large market in embedded systems and edge devices for their low computational and memory requirements.

3. Proposed Method

In this section, we present a complete overview of Working Memory Connections. First, we recall the LSTM equations. Second, we explain the modifications introduced in our design. Finally, we motivate the choices behind WMCs *w.r.t.* other approaches. Specifically, we identify key problems in previous cell-to-gate connections that hinder the learning process, and we show that the proposed solution does not suffer from these weaknesses.

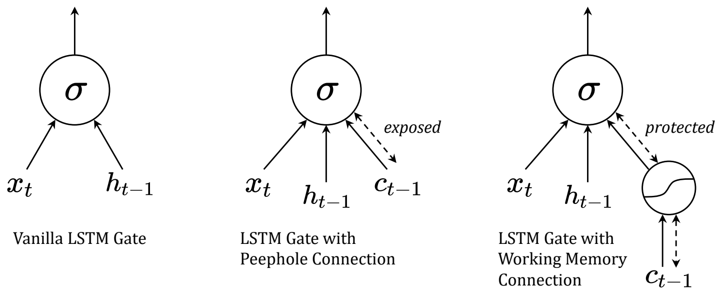

Figure 1: Comparison between a vanilla LSTM gate, a peephole connection, and a Working Memory Connection.

3.1. LSTM

The core idea behind Long Short-Term Memory networks is to create a constant error path between subsequent time steps. Being $\mathbf{x}_t$ the input vector at time t we can write the rollout equations for a vanilla LSTM as:

$$\mathbf{g}_t = \tanh(\mathbf{W}_{gx}\mathbf{x}_t + \mathbf{W}_{gh}\mathbf{h}_{t-1} + \mathbf{b}_g) \tag{1}$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{b}_i) \tag{2}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{b}_f) \tag{3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t \tag{4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{b}_o) \tag{5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t). \tag{6}$$

Here, $\mathbf{g}$ is the block input, $\mathbf{i}$, $\mathbf{f}$, and $\mathbf{o}$ are respectively the input, forget, and output gates, $\mathbf{c}$ represents the memory cell value, and $\mathbf{h}$ is the block output. In this notation, σ is the sigmoid function and $\odot$ denotes element-wise Hadamard product. In its first formulation (Hochreiter and Schmidhuber, 1997), LSTM did not include the multiplicative forget gate. However, being able to forget about past inputs (Gers et al., 2000) allows LSTM to tackle longer sequences while not hindering the back-propagation of the error signal.

3.2. Working Memory Connections

In the following, we introduce Working Memory Connections, which enable the memory cell to influence the value of the gates through a set of recurrent weights. Given a proper design for the connection, we argue that there is a practical advantage in letting the cell state influence the gating mechanisms in the LSTM block directly. In fact, the cell state $\mathbf{c}_t$ provides unique information about the previous time steps that are not present in $\mathbf{h}_t$. For instance, $\mathbf{h}_t$ may be close to zero as a consequence of the output gate saturating towards zero (see Eq. 6), while $\mathbf{c}_t$ may be growing and changing as a result of a sequence of input vectors. In that case, since the cell state cannot control the output gate, the LSTM block is forced to learn which particular value in the input vector is the marker that signals to open the output gate. Instead, with an appropriate connection strategy, the LSTM block could learn a mapping between the cell internal state and the gate values.

Our solution employs a set of recurrent weights $W_{\star c}$, $\star \in \{\mathbf{i}, \mathbf{f}, \mathbf{o}\}$ and a nonlinear activation function to model a connection between memory cell and gates. The application of a non-linearity on the memory cell is coherent with the present LSTM structure: as it can be noticed from Eq. 6, a nonlinear activation function is applied to $\mathbf{c}_t$ before the Hadamard product with $\mathbf{o}_t$[1]. In light of the above-mentioned intuitions, we modify Eq. 2, 3, and 5 by exposing the cell state $\mathbf{c}_t$ at time t through a protection mechanism as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \tanh(\mathbf{W}_{ic}\mathbf{c}_{t-1}) + \mathbf{b}_i) \tag{7}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \tanh(\mathbf{W}_{fc}\mathbf{c}_{t-1}) + \mathbf{b}_f) \tag{8}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \tanh(\mathbf{W}_{oc}\mathbf{c}_t) + \mathbf{b}_o), \tag{9}$$

where $\mathbf{W}_{\star c}\mathbf{c}_t$ denotes a general linear transformation.

At a first glance, Working Memory Connections may seem redundant in the gate structure. In fact, h_{t-1} depends from the value of c_{t-1} (Eq. 6). This impression is misleading, as the proposed connections introduce two main aspects of novelty. First, the non-linear activation function operates on three different projections of the cell state, one for each gate type. Second, Eq. 9 shows that the connection on the output gate depends on c_t, rather than on c_{t-1}, hence allowing for a more responsive control of the output dynamics of the entire LSTM block.

3.3. Advantages of Working Memory Connections

To formally motivate the improvement given by Working Memory Connections, we start by considering the local gradients of the gates in which the cell interaction is added. We limit our formal analysis to the input gate $\mathbf{i}_t$, but our reasoning can be generalized to $\mathbf{f}_t$ and $\mathbf{o}_t$. If we denote by $\bar{\mathbf{i}}_t$ the argument of the sigmoid activation function (Eq. 7) at time t:

$$\bar{\mathbf{i}}_t = \mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \tanh(\mathbf{W}_{ic}\mathbf{c}_{t-1}) + \mathbf{b}_i, \tag{10}$$

then the local gradient of the input gate $\mathbf{i}_t$ is expressed by:

$$\frac{\partial \mathbf{i}_t}{\partial \bar{\mathbf{i}}_t} = \frac{\partial}{\partial \bar{\mathbf{i}}_t}\sigma\left(\bar{\mathbf{i}}_t\right) = diag\left[\sigma\left(\bar{\mathbf{i}}_t\right) \odot \left(\mathbf{1} - \sigma\left(\bar{\mathbf{i}}_t\right)\right)\right], \tag{11}$$

where $\mathbf{1}$ denotes a vector of ones, and $diag[\mathbf{x}]$ indicates a diagonal N × N matrix whose diagonal contains the N elements of vector $\mathbf{x}$.

From here, we can easily derive the local gradients on the recurrent weights $\mathbf{W}_{ix}$,

[1] Previous works (Greff et al., 2017) have also shown that removing this non-linearity leads to a significant loss in terms of performance.

$\mathbf{W}_{ih}$, and $\mathbf{W}_{ic}$ at time t:

$$\frac{\partial \mathbf{i}_t}{\partial \mathbf{W}_{ix}} = \frac{\partial \mathbf{i}_t}{\partial \bar{\mathbf{i}}_t} \otimes \mathbf{x}_t, \tag{12}$$

$$\frac{\partial \mathbf{i}_t}{\partial \mathbf{W}_{ih}} = \frac{\partial \mathbf{i}_t}{\partial \bar{\mathbf{i}}_t} \otimes \mathbf{h}_{t-1}, \tag{13}$$

$$\frac{\partial \mathbf{i}_t}{\partial \mathbf{W}_{ic}} = \delta \hat{\mathbf{i}}_t \otimes \mathbf{c}_{t-1}, \tag{14}$$

where $\otimes$ denotes the outer product of two vectors, and:

$$\delta \hat{\mathbf{i}}_t = \frac{\partial \mathbf{i}_t}{\partial \bar{\mathbf{i}}_t} \odot \left(\mathbf{1} - \tanh^2(\mathbf{W}_{ic} \mathbf{c}_{t-1}) \right). \tag{15}$$

Now, let's consider what happens as t grows: we observe that $\mathbf{x}_t$ and $\mathbf{h}_t$ are bounded to a limited interval. In particular, $\mathbf{x}_t$ is a sample of the input data, and $\mathbf{h}_t$ is bounded in the interval $[-1, 1]$ by construction. Instead, the cell $\mathbf{c}_t$ can grow linearly with the number of recursive steps, making its domain extremely task-dependent. This is a well-known problem, which motivated the introduction of the forget gate in the original LSTM structure (Gers et al., 2000). Despite this, the range of possible values of c_t cannot be restricted to a fixed domain. The hyperbolic tangent non-linearity helps to avoid an excessive influence of the unbounded cell state in the gate mechanics, hence preventing unwanted saturation. As it can be seen in Eq. 7, 8, and 9, the term related to the cell state is bounded in the interval $[-1, 1]$. Additionally, it helps screen the connection weights $\mathbf{W}_{\star c}$ from unstable updates.

Even if $\mathbf{c}_t$ grew linearly with the number of time steps, its influence on the sigmoid argument would be mitigated, and it could not take the sigmoid function into its saturated regime against the other two terms driven by $\mathbf{x}$ and $\mathbf{h}$ respectively. On the other hand, the growth of the cell state would push the hyperbolic tangent towards its own saturated regime. This behavior helps protect the weight matrix employed in the connection from unstable updates.

Peephole Connections and their Limitations. We now turn our attention to a related connection, namely the peephole connection (Gers and Schmidhuber, 2000), which is no longer common in the LSTM formulation. Peephole connections were introduced by Gers and Schmidhuber in (Gers and Schmidhuber, 2000), and enrich the LSTM equations with recurrent weights $\mathbf{W}_{\star c}$, $\star \in \{\mathbf{i}, \mathbf{f}, \mathbf{o}\}$:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{16}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fc}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{17}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{oc}\mathbf{c}_t + \mathbf{b}_o), \tag{18}$$

with $\mathbf{W}_{\star c}$ generally constrained to be diagonal (Graves, 2013; Greff et al., 2017). While this formulation allows for a more precise control of the gates, there are two issues that limit its effectiveness. In this case, the local gradient at time t is expressed by:

$$\frac{\partial \mathbf{i}_t}{\partial \bar{\mathbf{i}}_t} = \frac{\partial}{\partial \bar{\mathbf{i}}_t} \sigma\left(\bar{\mathbf{i}}_t\right) = diag\left[\sigma\left(\bar{\mathbf{i}}_t\right) \odot \left(\mathbf{1} - \sigma\left(\bar{\mathbf{i}}_t\right)\right)\right], \tag{19}$$

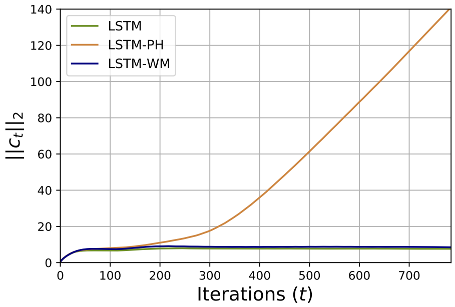

Figure 2: The cell state c_t may grow linearly with the number of time steps. Peephole connections directly expose c_t, creating a key issue (b). Data for this plot is taken from the first training iterations of the sequential MNIST (see § 4.2).

with $\bar{\mathbf{i}}_t$ being the argument of the sigmoid function in Eq 16:

$$\bar{\mathbf{i}}_t = \mathbf{W}_{ix}\mathbf{x}_t + \mathbf{W}_{ih}\mathbf{h}_{t-1} + \mathbf{W}_{ic}\mathbf{c}_{t-1} + \mathbf{b}_i. \tag{20}$$

In light of this difference, Eq. 14 and 15 become:

$$\frac{\partial \mathbf{i}_t}{\partial \mathbf{W}_{ic}} = \frac{\partial \mathbf{i}_t}{\partial \bar{\mathbf{i}}_t} \otimes \mathbf{c}_{t-1}. \tag{21}$$

We observe that, both in Eq. 7 and in Eq 16, the magnitude of the product $\mathbf{W}_{ic}\mathbf{c}_{t-1}$ can in principle grow unbounded. The activation function introduced in WMCs squashes this term into a closed bounded interval. In peephole connections, hovewer, this term is added inside the gate without an adequate protection (see Fig. 1). The result is that, in the peephole formulation, the sigmoid function applied immediately after could be pushed towards its saturating regime independently from the value of $\mathbf{x}_t$ and $\mathbf{h}_t$. In theory, the LSTM block can recover from this situation by setting all the weights in the peephole connection to 0, but in practice this might not happen if the sigmoid gate is saturated most of the time. Even if the two other summands can compensate for the growth of $\mathbf{W}_{\star c}\mathbf{c}$, hence letting gradients flow through the gate, there is still a key issue that hinders learning. In fact, as shown in Eq. 21, the gradients on the recurrent peephole weights grow linearly with $\mathbf{c}$, making updates unstable.

To exemplify this behavior, we report the Euclidean norm of $\mathbf{c}_t$ during the early training stages in Fig. 2. After a small number of time steps, the content of the cell floods the gates of the peephole LSTM. A possible consequence would be that both the input and the forget gates would saturate towards 1. In our example, this aspect leads to an additional and uncontrolled growth of the magnitude of $\mathbf{c}_t$. As it can be seen, Working Memory Connections exhibit a much more regular behavior than peepholes and can prevent the uncontrolled growth of the memory cell.

4. Experiments and Results

The effectiveness of Working Memory Connections and their general benefits can be appreciated in many different tasks. The proposed experiments cover a wide area

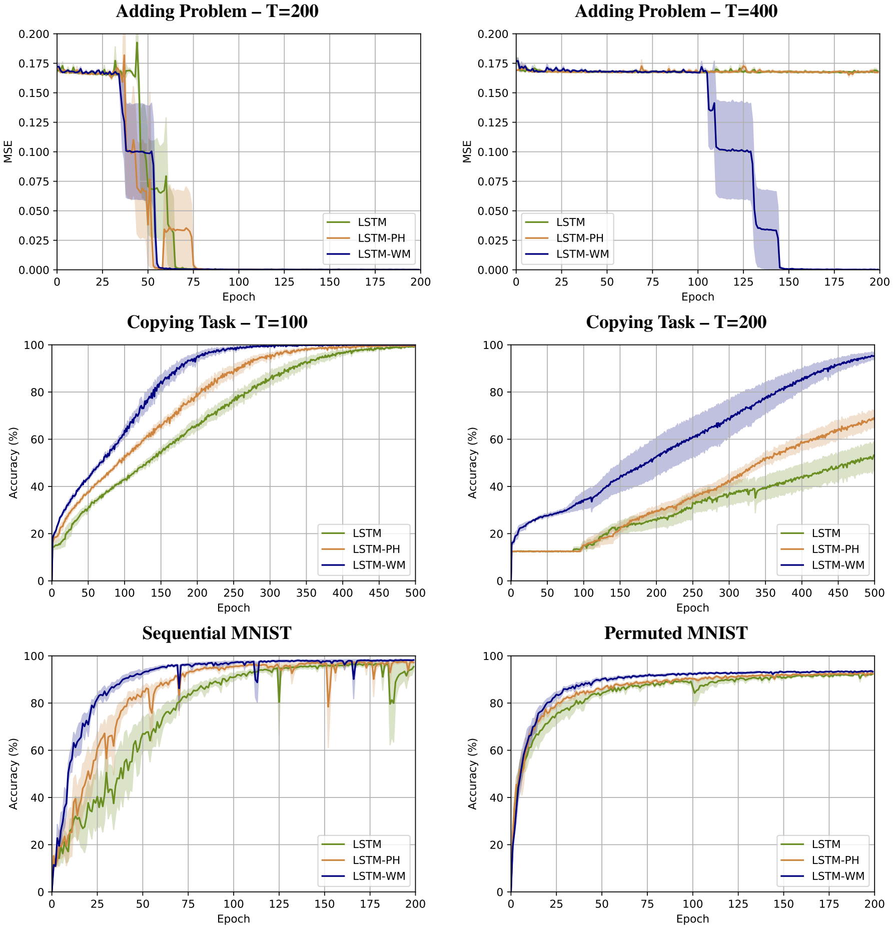

Figure 3: Comparison among traditional LSTM, the proposed LSTM with working memory connections, and peephole LSTM. We investigate three different tasks: the adding problem (top), the copying task (center), and the sequential/permuted MNIST (bottom). In all the plots, shading indicates the standard error of the mean.

of applications: two different toy problems, digit recognition, language modeling, and image captioning. While the analysis on simple tasks helps to clarify the inherent advantages of the proposed approach, results on more challenging real-world applications motivate a wider adoption of our novel connections, especially for long sequences. We compare our model (LSTM-WC) to a traditional LSTM and to an LSTM with peephole connections (LSTM-PH).

4.1. Adding Problem and Copying Tasks

In the adding problem (Hochreiter and Schmidhuber, 1997), the input to the network consists of a series of T pairs (n_t, f_t), with $0 \leq t < T$. The first element n_t is a real-valued number between 0 and 1, and f_t is a corresponding marker. In the entire sequence, only two markers f_i and f_j are set to 1, while the others are set to 0. The goal is to predict the sum of the corresponding real-valued items $n_i + n_j$, for which $f = 1$. In our experiments, we test with $T = 200$ and $T = 400$, and we measure the performance using mean squared error. For this experiment, the networks have hidden size $N = 128$ and train for 200 epochs. We optimize the parameters using SGD with Nesterov update rule. The learning rate is 10^{-2} (momentum factor 0.9) and the batch size is 128. We also clip the gradient norm to 1.0. Results are reported in Fig. 3 (top), where we plot the MSE on the test set for every epoch of training. LSTM-WM achieves the best convergence time for $T = 200$, while the final performance on this setup is similar among the three models. The effectiveness of WMCs is striking in the $T = 400$ setup. In fact, the proposed model solves the adding problem around epoch 145, while the other two architectures cannot learn the task and are stuck on the trivial solution.

In the copying task (Hochreiter and Schmidhuber, 1997), the network observes a sequence of 10 input symbols, waits for T time steps (we use $T = 100$ and $T = 200$), and then must reproduce the same sequence as output. For this experiment, we adopt the same setup described in (Arjovsky et al., 2016). We keep the same implementation details described for the adding problem, except that we train for 500 epochs. In Fig. 3 (center), we plot the test accuracy achieved by the three models at each epoch. In both setups, WMCs play an important role in terms of final performance and convergence time. As in the adding problem, the performance gain given by the proposed architecture is more evident when working on longer sequences: for $T = 200$, WMCs outperform peephole LSTM and vanilla LSTM by around +25% and +40%.

4.2. Permuted Sequential MNIST

The sequential MNIST (sMNIST) (Le et al., 2015) is the sequential version of the MNIST digit recognition task (LeCun et al., 1998). In this task, the image pixels are fed sequentially to the network (from left to right, and top to bottom). The permuted sequential MNIST (pMNIST) is a sequential version of the MNIST digit recognition problem in which the pixels are permuted in a random but fixed order. In both tasks, the goal is to predict the correct digit label after the last input pixel. Following the setup proposed in (Arpit et al., 2019), we use 50k images for training, 10k for validation, and 10k to test our models. The experimental setup is as follows. We set the hidden size to $N = 128$ for all the networks, and train for 200 epochs using SGD with learning rate 10^{-2} and batch size 128 (momentum 0.9 and Nesterov update rule). We clip the gradient norms to 1.0.

Fig. 3 (bottom) reports the mean test accuracy of the three LSTM variants for both setups. We report the standard error of the mean as a shaded area. For the sMNIST task, peephole LSTM performs slightly better than vanilla LSTM. LSTM with Working Memory Connections, instead, outperforms the competing architectures in terms of final accuracy and convergence speed. In particular, our architecture employs only 50 epochs to get above 92% accuracy, while other models are still generally stuck around 65% (vanilla LSTM) and 82% (LSTM-PH). In this experiment, we also find out that WMCs help stabilize training. In fact, the area given by the standard error of the mean is much thicker for our approach than for the other two variants, in particular during the early stages of training. On the pMNIST task, all the models achieve good final results, with LSTM with Working Memory Connections still being the best option.

Table 1: Test accuracy on the sequential MNIST task.

Model	**sMNIST**	**pMNIST**
iRNN (Le et al., 2015)	97.00	82.00
uRNN (Arjovsky et al., 2016)	95.10	91.40
h-detach (Arpit et al., 2019)	98.50	92.30
LSTM (h = 128)	98.16	92.94
LSTM (h = 256)	97.68	**93.97**
LSTM-PH (h = 128)	98.58	93.25
LSTM-PH (h = 256)	98.33	93.40
LSTM-WM (h = 128)	**98.63**	**93.97**

Table 2: Mean test bit per character on the PTB test set. Error range indicates the standard error of the mean.

	Test Bit per Character (BPC)			
	Fixed # Params ($\sim 2.2M$)		Fixed # Hidden Units (512)	
Model	$T_{PTB} = 150$	$T_{PTB} = 300$	$T_{PTB} = 150$	$T_{PTB} = 300$
LSTM	1.334 ± 0.0006	1.343 ± 0.0004	1.386 ± 0.0005	1.395 ± 0.0005
LSTM-PH	1.339 ± 0.0048	1.343 ± 0.0009	1.383 ± 0.0004	1.394 ± 0.0005
LSTM-WM	**1.299 ± 0.0005**	**1.302 ± 0.0008**	**1.299 ± 0.0005**	**1.302 ± 0.0008**

Numerical results, reported in Table 1, confirm that our model outperforms the classic LSTM by a discrete margin (+0.47% and +1.03% on the sequential and permuted MNIST respectively). Since WMCs introduce additional learnable parameters in the LSTM structure, we also compare with vanilla and peephole LSTM with increased hidden size (256 instead of 128). Note that, in this setting, LSTM and LSTM-PH have more than 2× the number of learnable parameters of LSTM-WM. Despite this, LSTM-WC achieves the best results on both tasks. It is worth noting that, while additional parameters in vanilla LSTM improves the results on pMNIST, they are not helpful in the sMNIST task. The flexibility given by WMCs, instead, allows the proposed model to achieve the best result in both setups. Always in Table 1, we compare with two state-of-the-art RNNs (Le et al., 2015; Arjovsky et al., 2016), and with a training algorithm for LSTM (Arpit et al., 2019). The proposed LSTM-WC outperforms the competitors in terms of test accuracy.

4.3. Penn Treebank (PTB) Character-Level Language Modeling

Character-level language modeling requires to predict a single character at each time step given an observed sequence of text. In our experiments on the Penn Treebank (PTB) dataset (Marcus and Marcinkiewicz, 1993), we evaluate the performance of the three different LSTM variants in terms of test mean bits per character (BPC), where lower BPC denotes better performance. We report the results in Table 2, where we compare truncated back-propagation through time (T_{PTB}) over 150 and 300 steps. Since our connection introduces new learnable weights, we consider an additional setup in which we keep a fixed number of parameters for the three networks. For this experiment, we follow the setup proposed by Merity et al. (2018), with the only exception that we employ a single LSTM layer instead of three. The advantage of using Working Memory

Table 3: Image captioning results on COCO test set.

Model	BLEU-1	BLEU-4	METEOR	ROUGE	CIDEr	SPICE
No Attention, ResNet-152						
LSTM	70.9	27.9	24.4	51.7	92.0	17.6
GRU	69.5	26.2	22.7	50.4	82.3	15.6
LSTM-PH	**71.4**	27.8	24.3	51.7	91.1	17.5
LSTM-WM	**71.4**	**28.3**	**24.6**	**52.4**	**94.0**	**17.8**
Attention, Faster R-CNN						
LSTM	75.9	**36.1**	27.4	56.3	111.9	20.3
GRU	76.0	**36.1**	27.0	56.5	111.0	20.2
LSTM-PH	75.8	35.9	27.3	56.3	111.5	20.2
LSTM-WM	**76.2**	**36.1**	**27.5**	**56.5**	**112.7**	**20.4**

Connections is more evident for equal number of hidden units, where the proposed architecture overcomes the vanilla LSTM and peephole LSTM by a significant margin. Even when the number of parameters is fixed for all the models, LSTM-WC outperforms the competitors by 0.035 and 0.041 BPC for $T_{PTB} = 150$ and $T_{PTB} = 300$ respectively. It is worth noting that peephole LSTM performs similarly to or even worse than vanilla LSTM on this task.

4.4. Image Captioning

We evaluate the performance of our LSTM with Working Memory Connections on the image captioning task, which consists of generating textual descriptions for images. We apply our approach to two different captioning models: Show and Tell (Vinyals et al., 2015) and Up-Down (Anderson et al., 2018). The first model includes a single LSTM layer and does not employ attention, while the second is composed of two LSTM layers and integrates attention mechanisms over image regions. We use the Microsoft COCO dataset (Lin et al., 2014) following the splits defined in (Karpathy and Fei-Fei, 2015). To represent images, we employ a global feature vector extracted from the average pooling layer of ResNet-152 (He et al., 2016) for the Show and Tell model, and multiple feature vectors extracted from Faster R-CNN (Ren et al., 2015) for the Up-Down architecture. We train both models with Adam optimizer (Kingma and Ba, 2015) using a learning rate equal to 10^{-4}. All other hyper-parameters are left the same as those suggested in the original papers.

Numerical results are reported in Table 3 using standard captioning evaluation metrics (*i.e.* BLEU-1, BLEU-4 (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004), CIDEr (Vedantam et al., 2015), and SPICE (Anderson et al., 2016)). For all these, higher metric results indicate better performance, with CIDEr being the metric that best correlates with human judgment. In both settings, our LSTM-WM outperforms traditional LSTM and LSTM-PH by a clear margin. Specifically, LSTM-WM improves the vanilla LSTM results by 2.0 CIDEr points on the model without attention and 0.8 CIDEr points on the model with attention over image regions, demonstrating the contribution of WMCs also for this task. As an additional comparison, we replace the LSTM layers with GRU layers. Numerical results suggest that there is not a clear advantage in using GRUs instead of LSTMs for this task. In Fig. 4, we plot

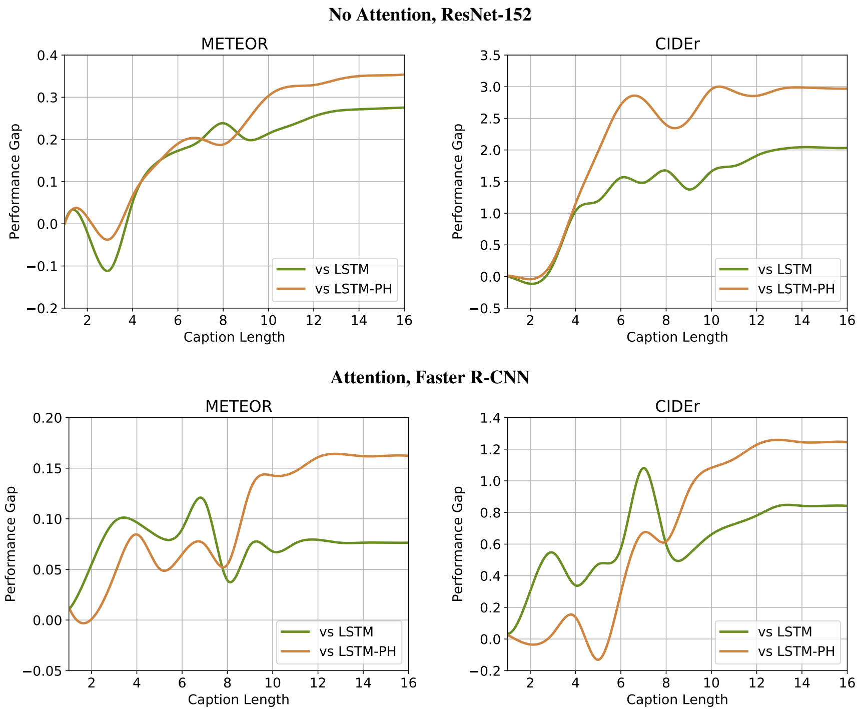

Figure 4: Metric gaps on the image captioning task for increasing instruction lengths.

the metric gap between LSTM-WM and the two competitors in terms of METEOR and CIDEr. On the X-axis we report the length of the generated captions, meaning that we consider the first x words of each predicted sentence. On the Y-axis, a 0 value means that our proposal performs equally, *i.e.* has no performance gap *w.r.t.* the competitor, while a higher value indicates better performance for our model. With this analysis, we aim to check whether the improvement given by WMCs can be restricted to a particular subset of the dataset. As one can observe, the metric gap generally increases with the caption length, especially *w.r.t.* peephole LSTM. We can deduce that the contribution of WMCs escalates with the number of time steps.

5. Discussion

With Working Memory Connections, we show that information stored in the LSTM cell should be accessible in the gate structure. We compare the performance of WMCs to a similar approach named peephole connections (Gers and Schmidhuber, 2000), and to vanilla LSTM. We find out that the structure of WMCs allows for two distinct improvements:

1. *A more precise control of the gates*. The multiplicative gates in the LSTM block must regulate the information flowing through the cell, but they cannot access the

state of that same cell in the traditional LSTM formulation. The presence of the cell state in the multiplicative gates motivates the improvements of LSTM-WM *w.r.t.* vanilla LSTM.

2. *Increased stability during training* compared to peephole connections. Exposing different projections of the cell state without squashing its content seems to be a critical point for the LSTM-PH. This element of novelty in our design explains why WMCs provide a boost in performance even when peepholes fail.

As a consequence of these two improvements, WMCs incorporate the theoretical benefits of peephole connections, originally described by Gers and Schmidhuber (2000), with the training stability and versatility of vanilla LSTM.

It is worth noting that, for tasks that do not require to access the content of the memory cell, Working Memory Connections would not probably bring any benefit in the LSTM formulation, while peepholes might still hinder the whole learning process because of unstable updates.

At the same time, when training stacked LSTMs, the benefits given by WMCs may become less significant. We suppose that this is due to the increased complexity in the network structure, where multiple LSTM blocks can interact through the various layers. Similarly, many architectures employ LSTMs as building blocks together with different components, and the influence of WMCs in these compound deep networks cannot be easily determined. Experiments on image captioning, proposed in this paper, partially answer this question and prove that WMCs afford a small yet existing improvement even in this scenario. However, there are many other complex tasks involving vision, language, and other modalities, that are worth investigating.

6. Conclusion

A current limitation of Long Short-Term Memory Networks consists in not letting the cell state influence the gate dynamics directly. In this paper, we propose Working Memory Connections (WMCs) for LSTM, which provide an efficient way of using intra-cell knowledge inside the network. The proposed design performs noticeably better than the vanilla LSTM and overcomes important issues in previous formulations. We formally motivate this improvement as a consequence of more stable training dynamics. Experimental results reflect the theoretical benefits of the proposed approach and motivate further study in this direction. One future direction might consist in testing the efficacy of Working Memory Connections for an even wider set of tasks.

Acknowledgments

This work has been supported by “Fondazione di Modena” under the project “AI for Digital Humanities” and by the national project “IDEHA: Innovation for Data Elaboration in Heritage Areas” (PON ARS01_00421), cofunded by the Italian Ministry of University and Research.

References

Anderson, P., Fernando, B., Johnson, M., Gould, S., 2016. SPICE: Semantic Propositional Image Caption Evaluation, in: Proceedings of the European Conference on Computer Vision.

Anderson, P., He, X., Buehler, C., Teney, D., Johnson, M., Gould, S., Zhang, L., 2018. Bottom-up and top-down attention for image captioning and visual question answering, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Arjovsky, M., Shah, A., Bengio, Y., 2016. Unitary evolution recurrent neural networks, in: Proceedings of the International Conference on Machine Learning.

Arpit, D., Kanuparthi, B., Kerg, G., Ke, N.R., Mitliagkas, I., Bengio, Y., 2019. h-detach: Modifying the LSTM Gradient Towards Better Optimization, in: Proceedings of the International Conference on Learning Representations.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural Machine Translation by Jointly Learning to Align and Translate, in: Proceedings of the International Conference on Learning Representations.

Banerjee, S., Lavie, A., 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments, in: Proceedings of the Annual Meeting on Association for Computational Linguistics Workshops.

Baraldi, L., Grana, C., Cucchiara, R., 2017. Hierarchical boundary-aware neural encoder for video captioning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Bengio, Y., Frasconi, P., Simard, P., 1993. The problem of learning long-term dependencies in recurrent networks, in: Proceedings of the International Joint Conference on Neural Networks.

Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE Trans. on Neural Networks 5, 157–166.

Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y., 2014a. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. arXiv preprint arXiv:1409.1259 .

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014b. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, in: Proceedings of the Conference on Empirical Methods in Natural Language Processing.

Chung, J., Gulcehre, C., Cho, K., Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. arXiv preprint arXiv:1412.3555 .

Devlin, J., Chang, M.W., Lee, K., Toutanova, K., 2018. BERT: Pre-training of deep bidirectional transformers for language understanding, in: Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics.

Elman, J.L., 1990. Finding Structure in Time. Cognitive Science 14, 179–211.

Ericsson, K.A., Kintsch, W., 1995. Long-Term Working Memory. Psychological Review 102, 211.

Gers, F.A., Schmidhuber, J., 2000. Recurrent nets that time and count, in: Proceedings of the International Joint Conference on Neural Networks.

Gers, F.A., Schmidhuber, J., Cummins, F., 2000. Learning to Forget: Continual Prediction with LSTM. Neural Computation 12, 2451–2471.

Graves, A., 2013. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 .

Graves, A., Mohamed, A.R., Hinton, G., 2013. Speech recognition with deep recurrent neural networks, in: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing.

Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B., Schmidhuber, J., 2017. LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems 28, 2222–2232.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Hernandez, R.F., 2018. Neuroethics, Nootropics, Neuroenhancement: The Ethical Case Against Pharmacological Enhancements. volume 109. LIT Verlag Münster.

Hochreiter, S., 1991. Untersuchungen zu dynamischen neuronalen Netzen. Diploma, Technische Universität München .

Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Computation 9, 1735–1780.

Jing, L., Gulcehre, C., Peurifoy, J., Shen, Y., Tegmark, M., Soljacic, M., Bengio, Y., 2019. Gated Orthogonal Recurrent Units: On Learning to Forget. Neural Computation 31, 765–783.

Karpathy, A., Fei-Fei, L., 2015. Deep visual-semantic alignments for generating image descriptions, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Kingma, D., Ba, J., 2015. Adam: a method for stochastic optimization, in: Proceedings of the International Conference on Learning Representations.

Le, Q.V., Jaitly, N., Hinton, G.E., 2015. A Simple Way to Initialize Recurrent Networks of Rectified Linear Units. arXiv preprint arXiv:1504.00941 .

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

Li, Z., Gavrilyuk, K., Gavves, E., Jain, M., Snoek, C.G., 2018. Videolstm convolves, attends and flows for action recognition. Computer Vision and Image Understanding 166, 41 – 50.

Lin, C.Y., 2004. Rouge: A package for automatic evaluation of summaries, in: Proceedings of the Annual Meeting on Association for Computational Linguistics Workshops.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: Common Objects in Context, in: Proceedings of the European Conference on Computer Vision.

Liu, Y., Hao, X., Zhang, B., Zhang, Y., 2020. Simplified long short-term memory model for robust and fast prediction. Pattern Recognition Letters 136, 81–86.

Marcus, M.P., Marcinkiewicz, M.A., 1993. Building a Large Annotated Corpus of English: The Penn Treebank. Computational Linguistics 19, 313–330.

Merity, S., Keskar, N.S., Socher, R., 2018. An Analysis of Neural Language Modeling at Multiple Scales. arXiv preprint arXiv:1803.08240 .

Papineni, K., Roukos, S., Ward, T., Zhu, W.J., 2002. BLEU: A Method for Automatic Evaluation of Machine Translation, in: Proceedings of the Annual Meeting on Association for Computational Linguistics.

Pascanu, R., Mikolov, T., Bengio, Y., 2013. On the difficulty of training recurrent neural networks, in: Proceedings of the International Conference on Machine Learning.

Ren, S., He, K., Girshick, R., Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks, in: Advances in Neural Information Processing Systems.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1986. Learning representations by back-propagating errors. Nature 323, 533–536.

Sutskever, I., Vinyals, O., Le, Q.V., 2014. Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need, in: Advances in Neural Information Processing Systems.

Vedantam, R., Lawrence Zitnick, C., Parikh, D., 2015. CIDEr: Consensus-based Image Description Evaluation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Vinyals, O., Toshev, A., Bengio, S., Erhan, D., 2015. Show and Tell: A Neural Image Caption Generator, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Xiao, H., Xu, J., Shi, J., 2020. Exploring diverse and fine-grained caption for video by incorporating convolutional architecture into lstm-based model. Pattern Recognition Letters 129, 173 – 180.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, in: Proceedings of the International Conference on Machine Learning.