Deep Learning for Language Modelling

Ramaseshan Ramachandran

TABLE OF CONTENTS

TRADITIONAL RNN LIMITATIONS

- The component of the gradient in directions that correspond to long-term dependencies is small[1]
- Gradients shrink over time, making it hard to learn long-term dependencies
- $$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \left(\prod_{k=t+1}^{T} \frac{\partial h_k}{\partial h_{k-1}} \right) \frac{\partial E}{\partial h_t} \frac{\partial h_t}{\partial W}$$
- The component of the gradient in directions that correspond to short-term dependencies is large
- As a result, RNNs can easily learn the short-term but not the long-term dependencies
- **Short-Term Memory**: Effectively remembers information for a few time steps

[1] An empirical exploration of recurrent network architectures - http://dl.acm.org/citation.cfm?id=3045118.3045367

GATING MECHANISM

We require a slowly-decaying error propagation. In other words, the update of weights should enhance/retain distributed properties of a sequence.

- Control the flow of information
- Should the new/old information be allowed or dropped?
- Gates are capable of interrupting, or allowing, the passage of activation values among neurons in the hidden layer
- The ability to manage the flow of information may play a key role arresting or setting up a well-behaved gradient during the back-propagation
- Multiplicative gates provide the protection of memory contents from decaying
- Multiplicative input and output gates protect contents from perturbations

WHAT IS LSTM?

- Long Short-Term Memory (LSTM) networks are a type of Recurrent Neural Network (RNN).
- Designed to overcome the limitations of traditional RNNs, particularly with long-term dependencies.
- The memory cell acts like a conveyor belt for information flow, allowing it to maintain information for long periods.

LSTM I

- In LSTM network[1] is the same as a standard RNN, except that the summation units in the hidden layer are replaced by memory blocks
- Instead of computing h_t from h_{t-1} directly with a linear combination followed by a nonlinearity $(f(W, x_t, h_{t-1}))$, the LSTM directly computes Δh_t, which is then added to h_{t-1} to obtain h_t
- The multiplicative gates allow LSTM memory cells to store and access information over long periods of time, thereby mitigating the vanishing gradient problem[2]
- Along with the hidden state vector, h_t, LSTM maintains a memory vector C_t

LSTM II

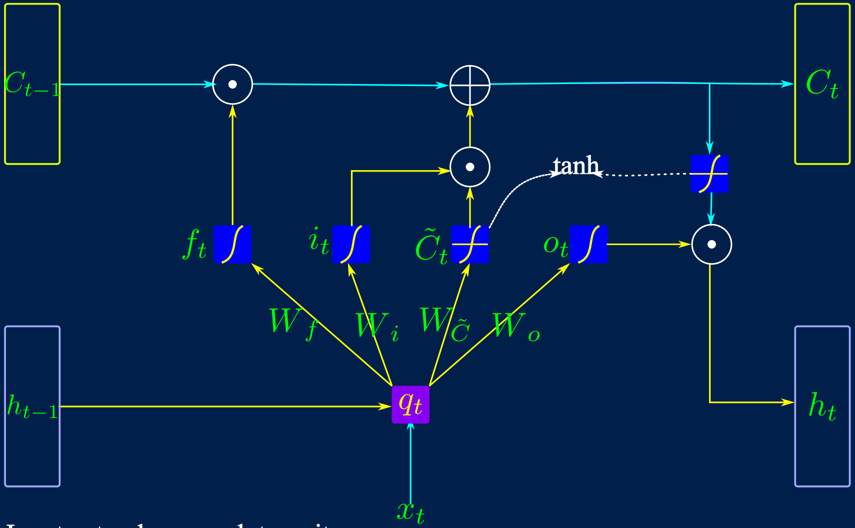
- At each time step the LSTM can choose to read from, write to, or reset the cell using explicit gating mechanisms
- LSTM computes well behaved gradients by controlling the values using the gates
- LSTM turns multiplication into addition
- LSTM uses gates to control how much information to add/erase or include/forget
- LSTM doesn't guarantee that there will be no vanishing/exploding gradient, but it provides a simple way to learn long-distance dependencies

[2] http://dblp.uni-trier.de/db/journals/corr/corr1506.html#KarpathyJL15

LSTM CELL STRUCTURE

- **Cell State (C_t)**: Main pathway for information flow.
- **Forget Gate (f_t)**: Decides what to discard from the cell state.
- **Input Gate (i_t)**: Determines how much new information to add.
- **Candidate Cell State ($\tilde{C}_t$)**: New values that could be added.
- **Output Gate (o_t)**: Controls what parts of the cell state to output.

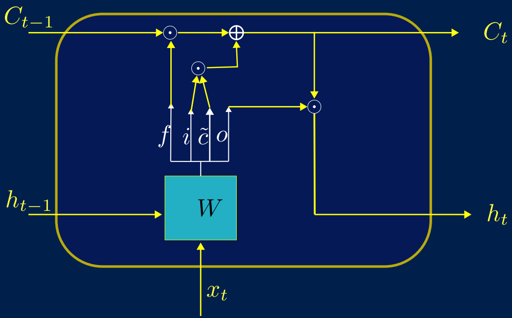
LSTM CELL

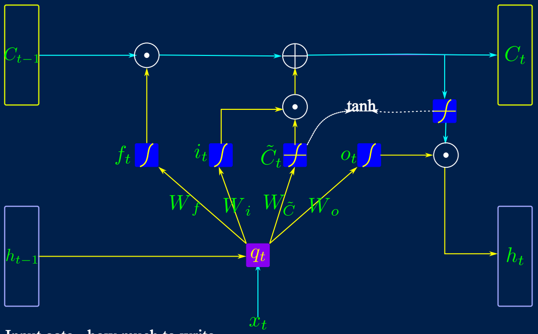


Input gate - how much to write
Forget gate - how much to forget/remember
Output gate - how much to reveal

SIMPLE REPRESENTATION

$$\begin{pmatrix} i \\ f \\ \tilde{C} \\ o \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \tanh \\ \sigma \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \quad (1)$$





Input gate - how much to write
Forget gate - how much to forget/remember
Output gate - how much to reveal

$$f_t = \sigma(W_f q_t + b_f) \tag{2}$$

$$i_t = \sigma(W_i q_t + b_i) \tag{3}$$

$$\tilde{C}_t = \tanh(W_{\tilde{C}_t} q_t) \tag{4}$$

$$C_t = (f_t \otimes C_{t-1}) \oplus (i_t \otimes \tilde{C}_t) \tag{5}$$

$$o_t = \sigma(W_o q_t + b_o) \tag{6}$$

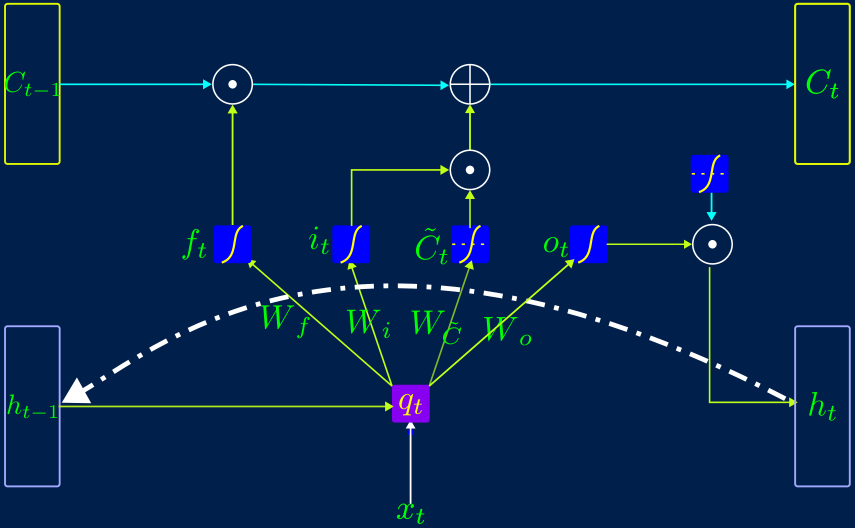
$$h_t = o_t \otimes \tanh(C_t) \tag{7}$$

$$s_t = \tanh(h_t) \tag{8}$$

$$z_t = V z_t \tag{9}$$

$$\hat{y}_t = \mathrm{softmax}(z_t) \tag{10}$$

BACK PROPAGATION THROUGH TIME



Back propagation through all cell weights

FORGET GATE

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

- f_t: Forget gate output.
- σ: Sigmoid function.
- W_f: Weight matrix for the forget gate.
- $[h_{t-1}, x_t]$: Concatenation of previous hidden state and current input.
- b_f: Bias vector for the forget gate.
- **Output Values**: - 0 means "completely forget". - 1 means "completely retain".
- **Learning Process**: Through training, the network learns what information to forget or retain.

IMPORTANCE OF FORGET GATE BIAS IN LSTM INITIALIZATION

- A critical yet often overlooked aspect of LSTMs is the initialization of the **forget gate bias** b_f.
- Standard LSTM initialization typically uses small random weights, which can introduce challenges with long-term dependencies.

EFFECT OF DEFAULT FORGET GATE INITIALIZATION

- Default initialization of LSTM weights often leads to the forget gate bias b_f being close to 0.5.
- This results in a vanishing gradient with a decay factor of approximately 0.5 per timestep.
- Problems with long-term dependencies are especially affected by this vanishing gradient, as seen in [1] and [2]

ADJUSTING THE FORGET GATE BIAS

- To mitigate the vanishing gradient problem, set the forget gate bias b_f to a higher value, such as 1 or 2.
- Initializing b_f to a large value ensures the forget gate is close to 1, enabling better gradient flow.
- This initialization strategy was originally suggested by Gers et al. [3].

$$b_f \approx 1 \text{ or } 2 \tag{11}$$

RISKS OF INCORRECT FORGET GATE INITIALIZATION

- Without proper initialization of b_f, the LSTM might appear incapable of learning tasks with long-range dependencies
- This is a misconception—appropriate initialization of the forget gate enables the LSTM to manage long-term information
- Initializing the forget gate bias b_f to a high value is crucial for effective learning of long-term dependencies in LSTMs
- This adjustment prevents vanishing gradients and enhances LSTM's performance on tasks requiring long-range memory
- Reemphasizing this technique is valuable, as it is often overlooked but significantly impacts the performance of LSTMs

INPUT GATE AND CANDIDATE CELL STATE

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

where i_t and $\tilde{C}_t$ are the Input gate output and the Candidate cell state, respectively

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

where C_t is the Current cell state and $\odot$ is the Element-wise multiplication operator

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$h_t = o_t \odot \tanh(C_t)$$

where o_t is the Output gate and h_t is the output of the Hidden state

IMPACT ON LEARNING

- **Long-Term Dependencies**: Cell state allows gradients to flow back many steps if necessary.
- **Mitigating Vanishing Gradients**: Helps in keeping information unchanged for many steps.

SUMMARY

- LSTMs use forget gates to manage information flow, solving short-term memory issues in traditional RNNs.
- Backpropagation through the cell state allows for effective learning of long-term dependencies.

WHAT IS GRU?

- Gated Recurrent Unit (GRU) is a type of recurrent neural network (RNN) designed to address the vanishing gradient problem
- Introduced as a simpler alternative to Long Short-Term Memory (LSTM) units

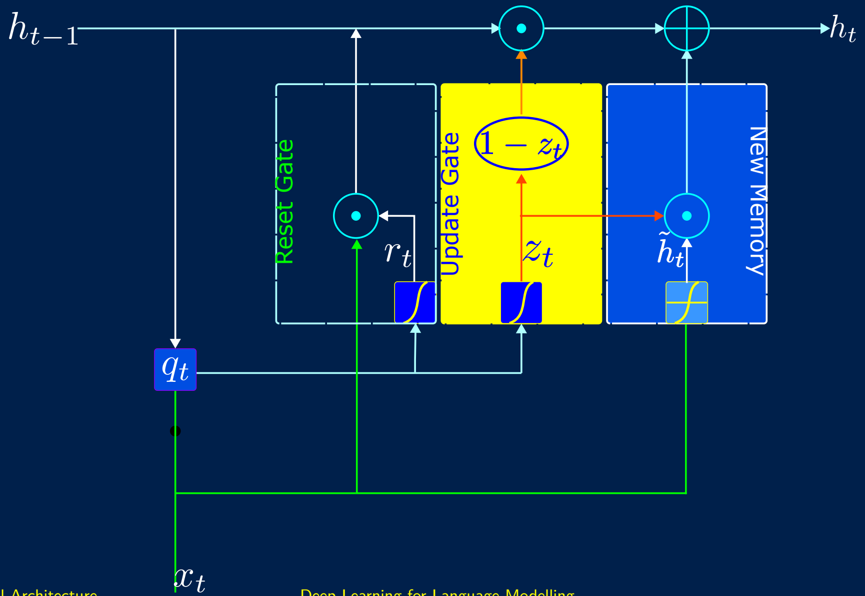
GRU ARCHITECTURE

- **Reset Gate** (r_t):
  - Controls how much of the previous information to forget.
  - $r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$
- **Update Gate** (z_t):
  - Determines how much of the previous information to retain in the current state.
  - $z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$
- **Candidate Activation** ($\tilde{h}_t$):
  - Combines reset gate information to produce a candidate activation.
  - $\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b)$
- **Hidden State Update** (h_t):
  - Updates the hidden state based on the update gate and candidate activation.
  - $h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$

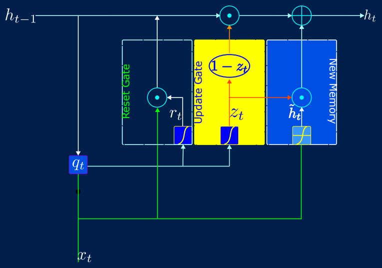
COMPONENTS OF GRU

- **Reset Gate (r_t)**: Decides how much of the past information to forget
- **Update Gate (z_t)**: Determines how much of the previous memory to keep or update.
- **Candidate Activation ($\tilde{h}_t$)**: New memory content

GRU ARCHITECTURE



GRU FORWARD PASS



$$q_t = f(h_{t-1}, x_t) \tag{12}$$

$$z_t = \sigma(W_z, q_t) \tag{13}$$

$$r_t = \sigma(W_r, q_t) \tag{14}$$

$$\tilde{h}_t = \tanh(W.(r_t, q_t)) \tag{15}$$

$$h_t = (1 - z_t) \otimes h_{t-1} \oplus (z_t \otimes \tilde{h}_t) \tag{16}$$

$$s_t = \tanh(h_t) \tag{17}$$

$$\hat{y}_t = \mathrm{softmax}(Vs_t) \tag{18}$$

Intuition

If the reset gate values $\rightarrow 0$, previous memory states are faded and new information is stored. If the z_t is close to 1, the information is copied and retained thereby adjusting the gradient to be alive for the next time step, thereby long-term dependency is stored. BPTT decides the learning of the reset and update gate.

STRUCTURAL DIFFERENCES

- **GRU**: Combines the forget and input gates into a single update gate, and merges the cell state and hidden state
- **LSTM**: Has separate gates for forgetting, inputting, and outputting, with an explicit cell state

COMPUTATIONAL COMPLEXITY

- **GRU**: Fewer parameters and operations, making it computationally lighter
- **LSTM**: More parameters and operations, potentially leading to better performance on complex tasks but at higher computational cost

PERFORMANCE AND USE CASES

- **GRU**: Often performs comparably to LSTM on many tasks, especially when computational efficiency is crucial
- **LSTM**: Generally preferred for tasks requiring longer-term memory or when the additional complexity can be justified by performance gains

WHAT IS BPTT?

- BPTT is a training algorithm for recurrent neural networks (RNNs) like GRUs
- It unfolds the RNN over time, treating each time step as a layer in a deep neural network

GRU FORWARD PASS EQUATIONS

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

BPTT STEPS

1. Forward pass through all time steps.
2. Compute loss at the final time step.
3. Backpropagate through the unfolded network.
4. Update weights considering temporal dependencies.

Loss Function:

$$L = \text{Loss}(h_T, y)$$

where h_T is the final hidden state, and y is the target output.

Initial Derivatives:

$$\frac{\partial L}{\partial h_T} = \text{computed from loss function}$$

BACKPROPAGATION THROUGH TIME (BPTT) IN GRUS

- Like RNNs and LSTMs, GRUs are trained using Backpropagation Through Time (BPTT).
- BPTT involves unrolling the network across time steps and applying backpropagation to compute gradients.
- In GRUs, we must compute gradients for both the **Update Gate** and **Reset Gate**.
- Our goal is to compute partial derivatives with respect to each parameter in order to update them during training.

GRADIENT OF LOSS WITH RESPECT TO OUTPUT

- Let the loss at time t be L_t.
- We aim to compute $\frac{\partial L}{\partial h_t}$ for the current time step.
- Using the chain rule, the total gradient of the loss over time will be:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L_t}{\partial h_t} + \sum_{k=t+1}^{T} \frac{\partial L_k}{\partial h_k} \frac{\partial h_k}{\partial h_t}$$

GRADIENT OF HIDDEN STATE UPDATE h_t

- Recall the hidden state update: $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$.
- The gradient with respect to h_t becomes:

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_t} + \frac{\partial L_t}{\partial h_t}$$

- Expanding $\frac{\partial h_t}{\partial h_{t-1}}$:

$$\frac{\partial h_t}{\partial h_{t-1}} = (1 - z_t) + z_t \cdot r_t \cdot \frac{\partial \tilde{h}_t}{\partial h_{t-1}}$$

GRADIENT OF UPDATE GATE z_t

- The gradient with respect to z_t is crucial for updating the hidden state:

$$\frac{\partial L}{\partial z_t} = \frac{\partial L}{\partial h_t} \cdot (\tilde{h}_t - h_{t-1})$$

- Applying the sigmoid derivative:

$$\frac{\partial z_t}{\partial W_z} = z_t(1 - z_t) \cdot \frac{\partial L}{\partial z_t}$$

GRADIENT OF RESET GATE r_t

- For the reset gate r_t: $\frac{\partial L}{\partial r_t} = \frac{\partial L}{\partial \tilde{h}_t} \cdot h_{t-1}$
- Using the chain rule, we derive: $\frac{\partial r_t}{\partial W_r} = r_t(1 - r_t) \cdot \frac{\partial L}{\partial r_t}$

GRADIENT OF CANDIDATE ACTIVATION $\tilde{h}_t$

- The candidate activation gradient $\frac{\partial L}{\partial \tilde{h}_t}$ is calculated as:

$$\frac{\partial L}{\partial \tilde{h}_t} = \frac{\partial L}{\partial h_t} \cdot z_t$$

- Expanding further, with the $tanh$ activation derivative:

$$\frac{\partial \tilde{h}_t}{\partial W} = (1 - \tilde{h}_t^2) \cdot \frac{\partial L}{\partial \tilde{h}_t}$$

INTERSECTION OF NEUROSCIENCE AND MACHINE LEARNING

- When discussing the biology of forgetting, we enter an intriguing intersection of neuroscience and machine learning.
- Computational models like GRU (Gated Recurrent Unit) and LSTM (Long Short-Term Memory) provide a framework for understanding how information is retained or forgotten.
- Let's explore how these models mimic biological processes in handling forgetting and retaining long-term relationships.

MECHANISMS OF BIOLOGICAL FORGETTING

Biological forgetting can occur due to two primary mechanisms:

- **Decay**:
  - Over time, memories weaken if they are not rehearsed or revisited.
  - This is analogous to how neural network information might degrade if not reinforced.
- **Interference**:
  - New information can interfere with old memories (retroactive interference).
  - Old memories can also interfere with new learning (proactive interference).
  - Similar to how new inputs in RNNs may overwrite or interfere with previous states if not managed properly.

LONG-TERM MEMORY IN BIOLOGY

- Memories that are deemed important or frequently accessed are consolidated into long-term memory (LTM).
- LTM consolidation involves changes in synaptic strength and neural connections.
- This process is analogous to how LSTMs maintain a **cell state** to retain information over extended periods.

LSTM: MECHANISMS FOR FORGETTING AND RETENTION

- **Forget Gate**:
  - Explicitly decides what information to discard from the cell state.
  - Closely mimics biological forgetting, where irrelevant or less important information is discarded.
- **Cell State**:
  - Acts as a form of long-term memory, allowing information to persist across many time steps.
  - Mirrors how humans retain frequently accessed information in LTM.
- **Input and Output Gates**:
  - Control what new information is added to the cell state and what information is output.
  - Analogous to how humans selectively remember or recall information based on context.

GRU: MECHANISMS FOR FORGETTING AND RETENTION

- **Update Gate**:
  - Combines the functionality of LSTM's input and forget gates.
  - Decides how much of the previous memory to retain or update with new information.
  - Less granular than LSTM but effective for many tasks.
- **Reset Gate**:
  - Controls how much past information to forget.
  - Although not as explicit as LSTM's forget gate, it allows a form of resetting past states.

COMPARISON: COMPLEXITY AND EFFICIENCY

- **LSTM**:
  - More complex with separate gates for forgetting, input, and output.
  - Better at handling long-term dependencies, though with more parameters and computational overhead.
- **GRU**:
  - Simpler, with combined gates, which reduces computational cost.
  - Efficient for shorter sequences but may not capture very long-term dependencies as effectively as LSTM.

ROLE OF MUSASHI PROTEINS ROLES IN MEMORY PROCESSES

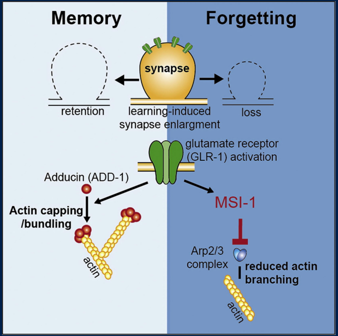


Figure: Musashi protein

- Repetition of events
- Primacy and recency
- Surprise
- Emotional Impact
- Positive or negative actions
- Hypocrisy
- ...

Memory length is regulated cooperatively through the activation of adducin (add-1) and by the inhibitory effect of msi-1[4]. Brain forgets unimportant information in order to remain efficient Can RNN be trained to simulate the actions of adducin and musashi proteins?

MUSHASHI VS RNNS

- Selective Processing: Both Musashi proteins and LSTM/GRU gates selectively process information
- In biological terms - this might mean which mRNAs are translated into proteins
- In neural networks - which information is retained or passed forward.
- Regulation Over Time: Both systems deal with changes over time
- Musashi's role in stem cell maintenance involves long-term regulation, similar to how LSTM maintains cell state over long sequences.

BIOLOGICAL ANALOGY: GRU VS. LSTM

- **LSTM**:
  - LSTM's multiple gates mimic biological systems, where different mechanisms control memory retention, consolidation, and forgetting.
  - The explicit *forget gate* mirrors selective forgetting in the human brain.
- **GRU**:
  - GRU has a simpler architecture, combining some functions, which could be seen as analogous to more streamlined or generalized memory functions in simpler neural systems.
  - While efficient, it lacks the granularity of the forget gate, making it more limited in mimicking biological forgetting.

SUMMARY I

- GRU simplifies LSTM by reducing the number of gates, making it more efficient but potentially less expressive for very complex temporal dependencies
- The choice between GRU and LSTM often depends on the specific requirements of the task, computational resources, and desired model complexity
- Backpropagation in GRUs involves calculating partial derivatives for each gate and updating them based on gradients.
- The BPTT algorithm handles temporal dependencies by summing gradients over time.
- GRUs' simplified structure (with two gates) generally leads to fewer parameters, making backpropagation slightly more efficient compared to LSTMs
- Both LSTM and GRU offer models for understanding forgetting and retention.
- **LSTM** aligns more closely with complex biological systems due to its multiple gates.

SUMMARY II

- **GRU** provides a computationally efficient alternative, balancing simplicity with effective memory retention.
- This comparison highlights how machine learning models draw inspiration from biological processes, enhancing our understanding of memory in both fields.

REFERENCES I

[1] Sepp Hochreiter and Jürgen Schmidhuber. *Long Short-Term Memory*. Cambridge, MA, USA, Nov. 1997. DOI: 10.1162/neco.1997.9.8.1735. URL: https://www.bioinf.jku.at/publications/older/2604.pdf.

[2] Ilya Sutskever et al. *On the importance of initialization and momentum in deep learning*. Atlanta, GA, USA, 2013. URL: https://www.cs.toronto.edu/~fritz/absps/momentum.pdf.

[3] F.A. Gers, J. Schmidhuber, and F. Cummins. *Learning to forget: continual prediction with LSTM*. 1999. DOI: 10.1049/cp:19991218. URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=e10f98b86797ebf6c8caea6f54cacbc5a50e8b34.

REFERENCES II

[4] Nils Hadziselimovic et al. "Forgetting Is Regulated via Musashi-Mediated Translational Control of the Arp2/3 Complex.". In: *Cell* 156.6 (Mar. 2014), pp. 1153–1166. ISSN: 1097-4172. URL: http://view.ncbi.nlm.nih.gov/pubmed/24630719.