Note on Hierarchical Reasoning Models

Compiled by D. Gueorguiev, 8/28/25

Interesting new Small Language Model with different (recurrent) architecture than the classic transformer-based LLM. I think we should try this model sooner rather than later because **a)** it shows promise, **b)** it is easy/inexpensive to train (only 1000 samples) and **c)** there is already a well documented[github repo](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fsapientinc%2FHRM&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985392203%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=6uS3WYv%2Bb3bhbucov50cRqsAUP%2B8QOipMDOY6GjHs%2B8%3D&reserved=0" \o "Protected by Outlook: https://github.com/sapientinc/HRM. Click or tap to follow the link." \t "_blank) for this model.

**Hierarchical Reasoning Model (HRM)**  
It comes from a small company in Singapore - Sapient, Inc.  
It is a new recurrent architecture for Language Models inspired by the hierarchical and multi-timescale processing in the human brain.  
More on this interesting architecture in the comments of this thread.

The significant claim about HRM is:  
*"With only****27******million parameters****, HRM achieves exceptional performance on complex reasoning tasks using only 1000 training samples. The model operates without pre-training or CoT data, yet achieves nearly perfect performance on challenging tasks including complex Sudoku puzzles and optimal path finding in large mazes."*

Also:

*"Furthermore, HRM outperforms much larger models with significantly longer context windows on the Abstraction and Reasoning Corpus (*[*ARC*](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farcprize.org%2F&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985413252%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=1lY6sEBP76b0ksysU9iNxwKJQBIZMy5VAH2nK7EhQ1M%3D&reserved=0)*), a key benchmark for measuring artificial general intelligence capabilities. These results underscore HRM's potential as a transformative advancement toward universal computation and general-purpose reasoning systems."*

*"HRM scored 40.3% in*[*ARC-AGI-1*](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farcprize.org%2Farc-agi%2F1%2F&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985427254%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=C28jHowM3vVWUzQ1iVTKe4JxwfLsOvaNXFUuhGiQ7DQ%3D&reserved=0)*, compared with 34.5% for OpenAI's o3-mini-high, 21.2% for Anthropic's Claude 3.7 and 15.8% for Deepseek R1. In the tougher*[*ARC-AGI-2*](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farcprize.org%2Farc-agi%2F2%2F&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985440897%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=WgKk08ODRFgeOGv%2F18hZ1dGS%2F9UagBm01cicuIzf5eE%3D&reserved=0)*test, HRM scored 5% versus o3-mini-high's 3%, Deepseek R1's 1.3% and Claude 3.7's 0.9%."*

paper: [https://arxiv.org/abs/2506.21734](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farxiv.org%2Fabs%2F2506.21734&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985454841%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=6m%2BDKMyCma64pmez94U0J8hcfq2fYKcRlcpHVangPLo%3D&reserved=0)  
github repo: [https://github.com/sapientinc/HRM](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fsapientinc%2FHRM&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985467890%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=R%2BMA3Ryef7GT%2BHPRTiAgsBqK3R9gMjhH6SrFfHM4kq8%3D&reserved=0)

I am very curious about these statements and if wonder if we can verify some of those.  
The instructions on the Github repo to run the tests are very clear and simple (see the [README.md](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fsapientinc%2FHRM%2FREADME.md&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985481057%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=W59XQ6ccZxzCrc4C2HhyaARSo3bNpfnZ%2Bnak2o3FkQY%3D&reserved=0) section [Run Experiments](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fsapientinc%2FHRM%3Ftab%3Dreadme-ov-file%23run-experiments&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985494556%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=xPXeJZbpyrOyAGQNmNTy1E2wPVCLESPrD%2FjbWD7f66Y%3D&reserved=0) and also [Full-scale Experiments](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgithub.com%2Fsapientinc%2FHRM%3Ftab%3Dreadme-ov-file%23launch-experiments&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985507558%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=bg6Ee2fQfKeIknfzHMb4OAIkEjyyRn80DtACWzt9qKc%3D&reserved=0))  
The full-scale training assumes 8 GPUs and it lasts approx. 2 hours. I am wondering if I run the same training workload on my Google Colab Pro account with single A100 (which comes with at least 40GB of VRAM) how long it will take.

[https://medium.com/@mlshark/killing-lmms-with-a-tiny-27m-model-hierarchical-reasoning-model-explained-hrm-briefly-explained-bf1121a97e77](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fmedium.com%2F%40mlshark%2Fkilling-lmms-with-a-tiny-27m-model-hierarchical-reasoning-model-explained-hrm-briefly-explained-bf1121a97e77&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985520384%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=grz7gx3OcZ7Q0aT4RNnUAAFfD5EXkn2wkLUO8chcy60%3D&reserved=0)

If I have to summarize my impression on this new HRM model with one sentence it would be:

***"Back to the good ol' Recurrent Neural Nets to train with less data compared to the Attention-based Encoder-Decoders (aka Seq2Seq architecture)."***

So it makes sense to start with:

**A bit of history on Recurrent Neural Networks (RNNs)**:

Before the advent of the Transformers utilizing the Attention mechanism in 2017 the RNN were the state-of-the-art (SOTA) design for statistical machine translation (SMT). ( **Note**: SMT models are exactly the kind of models which we would use for the GenAI ***text-to-text generation*** use case. ) For details on RNN see the best possible introduction by Andrej Karpathy - [1]. Key historical papers on RNNs are [2], [3], [4] and [5].

The problems with RNN for SMT tasks are as follows:  
i ) highly non-linear behavior due to the recurrent nature of these networks which manifests itself as:  
    i.1 ) **vanishing gradients problem** - during training in the backprop step the gradients become extremely small and as a result it becomes increasing hard to learn from earlier parts of the input sequence  
    i.2 ) **exploding gradients problem** - during training in the backprop step the gradients become very large and the weight coefficients updates become unstable/erratic which makes it difficult to learn.  
    i.3 ) **long-term dependencies problem** - sentences can be long, and words appearing early in a sentence can significantly influence words much later in the same sentence. Vanilla RNNs struggle to maintain information across these long distances.  
ii ) computational and architectural issues  
    ii.1 ) **sequential processing** - RNNs process information sequentially due to the recurrent (loop-like) structure of the network. They are difficult to parallelize in matrix notation and be trained /run inferences on GPU.  
   ii.2 ) **fixed-length embedding vector space** - In early RNN-based neural machine translation models, the entire source sentence was encoded into a single, fixed-length vector. This "bottleneck" struggled to retain all necessary information for long sentences, leading to information loss.

RNN built with Long-Short-Term Memory (LSTM, see [6] and [7]) or Gated Recurrent Units (GRU, see [8]) fix i.1), i.2). However, classical RNNs do not address i.3) and ii.2). The problem ii.2 ) stays with almost all RNN designs and is overcome by the use of the Attention mechanism.  
The presence of i.3) and ii.2) in practically all RNN designs was the main reason for the widespread adoption of the Transformers **with the Attention mechanism** (which guarantees linear (loop-free) behavior and is quite paralelizable). Thus after 2017 the RNNs were gradually abolished in SMT tasks.

# I. HRM Architecture

HRM is inspired by three fundamental principles of neural computation observed in the brain:

• Hierarchical processing: The brain processes information across a hierarchy of cortical areas. Higher-level areas integrate information over longer timescales and form abstract representations, while lower-level areas handle more immediate, detailed sensory and motor processing

• Temporal Separation: These hierarchical levels in the brain operate at distinct intrinsic timescales, reflected in neural rhythms (e.g., slow theta waves, 4–8 Hz and fast gamma waves, 30–100 Hz). This separation allows for stable, high-level guidance of rapid, low-level computations.

• Recurrent Connectivity: The brain features extensive recurrent connections. These feedback loops enable iterative refinement, yielding more accurate and context-sensitive representations at the cost of additional processing time. Additionally, the brain largely avoids the problematic deep credit assignment problem associated with Backprop through time (BPTT).

HRM is built with the following 4 learnable components:  
**Input Network** , Low-Level Recurrent Module (denoted as **L-Module**, basically an RNN), High-Level Recurrent Module (denoted as **H-Module**, basically an RNN), **Output Network**.  
Refer to the attached screenshot *HRM\_Architecture.png*. HRM has recurrent structure. We begin with the initial input, an initial low‐level weight z^o\_L, and a high‐level weight z^o\_H, all randomly initialized.

1. The low‐level weight is updated at every step, while the high‐level weight is only updated every T steps.
2. Repeating this process for N times yields a total of N×T time‐step iterations.
3. One complete N×T time‐step sequence is called a single forward pass of the HRM.

**A diagram of a system

AI-generated content may be incorrect.**

Figure:the HRM architecture as presented in the paper.

# II. Hierarchical convergence of HRM and why it is better than classic RNN

Although convergence is crucial for recurrent networks, standard RNNs are fundamentally limited by their tendency to converge too early. As the hidden state settles toward a fixed point, update magnitudes shrink, effectively stalling subsequent computation and capping the network’s effective depth. To preserve computational power, we actually want convergence to proceed very slowly–but engineering that gradual approach is difficult, since pushing convergence too far edges the system toward instability.

HRM is explicitly designed to counteract this premature convergence through a process we term hierarchical convergence. During each cycle, the L-module (an RNN) exhibits stable convergence to a local equilibrium. This equilibrium, however, depends on the high-level state z\_H supplied during that cycle. After completing the T steps, the H-module incorporates the sub-computation’s outcome (the final state z\_L) and performs its own update. This z\_H update establishes a fresh context for the L-module, essentially “restarting” its computational path and initiating a new convergence phase toward a different local equilibrium. This process allows the HRM to perform a sequence of distinct, stable, nested computations, where the H-module directs the overall problem-solving strategy and the L-module executes the intensive search or refinement required for each step. Although a standard RNN may approach convergence within T iterations, the hierarchical convergence benefits from an enhanced effective depth of N\*T steps. This mechanism allows HRM both to maintain high computational activity (forward residual) over many steps (in contrast to a standard RNN, whose activity rapidly decays) and to enjoy stable convergence. This translates into better performance at any computation depth,

# III. Approximate Gradient computation in HRM

HRM’s architecture is similar to that of an RNN. When updating an RNN’s weights, you must store every intermediate hidden state at each time step. If producing the output takes T steps, you need to retain T hidden states for gradient descent. This procedure is known as Backpropagation Through Time (BPTT).

The authors proposed a more efficient approach. Research shows that ***as an RNN processes its input, its hidden state often converges to a fixed point, so after a certain time step, subsequent hidden states barely change***. By treating all intermediate high- and low-level hidden states as constant vectors, it dramatically boost efficiency. The memory requirement then drops from O(T) to just O(1).  
Thus, the gradient propagates as follows: output head → final state of the H-module → final state of the L-module → input embedding. Refer to screenshot *Gradient\_path\_in\_Approx\_Gradient\_Algorithm.png* for the gradient path.

A close-up of a text

AI-generated content may be incorrect.

# IV. Adaptive computational time (ACT)

One question is: when should we stop HRM? How many forward passes do we need? If you’re familiar with the classic **System 1** versus **System 2** distinction from [Daniel Kahneman’s](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farchive.org%2Fdetails%2Fthinking-fast-and-slow-daniel-kahneman%2Fpage%2Fn1%2Fmode%2F2up&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985669307%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=Y1VKmrFcQYPIUFdzksJZUml%2FLM9fd1FHWlv3Cofw0VE%3D&reserved=0)[*Thinking, Fast and Slow*](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farchive.org%2Fdetails%2Fthinking-fast-and-slow-daniel-kahneman%2Fpage%2Fn1%2Fmode%2F2up&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985682912%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=%2BaWdyO07ZuMZyFiZbKSwLrpk%2Fnww45p%2FgEzxuin56Kc%3D&reserved=0), you’ll know the human way of thinking typically alternate between a default-mode network for automatic thinking (**System 1**) and the other one for deliberate reasoning (**System 2**). Neuroscientific evidence shows that these cognitive modes share overlapping neural circuits, particularly within regions such as the prefrontal cortex and the default mode network. This indicates that the brain dynamically modulates the “runtime” of these circuits according to task complexity and potential rewards. Inspired by the above mechanism, we incorporate an adaptive halting strategy into HRM that enables “thinking, fast and slow”. This integration leverages deep supervision and uses the [Q-learning algorithm](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FQ-learning&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985697803%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=aJvE7R%2BlIVsD7oOv4P3oipxd%2FMODr0QoRN3KwLHDybk%3D&reserved=0).

Short diversion on the Q-Learning family of algorithms

Q Learning is a kind of Reinforcement Learning algorithm which usually is defined via [Markov Decision Process](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FMarkov_decision_process&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985711261%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=8WCpWZ406yAPGrS1Bhmc%2FspYG%2BtByfkAYfPM9QGcwJo%3D&reserved=0)(finite or infinite). This specific Q Learning algorithm is implemented as a finite Markov Decision Process (MDP) which updates the state and makes decision as defined in page 8 of the paper; see screenshot *Q\_Network\_implemented\_as\_MDP.png* attached here) to adaptively determine the number of segments.  
**Note1**: *Q-Network* is specific implementation of a given Q Learning algorithm.  
**Note2**: DQN (Deep Q-Network) is a specific implementation of a given Q Learning algorithm which (not surprisingly) involves Deep Learning Network.  Think of of DL network specifically designed and trained to discover the best available approximation of the optimal policy for this specific Q-Learning problem. Recall, the optimal policy aims to maximize the total cumulative reward using the defined Reward function for the RL problem at hand. More on DQNs - [here](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fmedium.com%2F%40samina.amin%2Fdeep-q-learning-dqn-71c109586bae&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985724459%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=O2zvDKTD%2FO8YQpqsBifX0faF59PI3UGddtE5KKMosEI%3D&reserved=0). Also look at [Gymnasium environment for specific examples .e.g. balancing the cartpole](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fgymnasium.farama.org%2Fenvironments%2Fclassic_control%2Fcart_pole%2F&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985737727%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=DEQJu0rXDDu0tcjZhdot5WLx%2FE8xmxF3NfUNxu48zec%3D&reserved=0), etc.

**Adaptive Halting in HRM**  
Based on neuroscientific findings, the authors employ an **adaptive halting strategy** to regulate HRM’s thinking speed. Refer to screenshot *Adaptive\_Halting\_Strategy.png*. They append another [Q-network](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.geeksforgeeks.org%2Fdeep-learning%2Fdeep-q-learning%2F&data=05%7C02%7C%7C3b069b32fe244a6a49d308dde7bf875c%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638921531985750968%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=CFXdnBFVQZVe6q5HiMfBlfp8eQt%2B4yqzB34EXQqUF9E%3D&reserved=0) to decide whether the HRM should stop or continue. The maximum number of thinking steps M\_max⁡ is fixed, while the minimum M\_min, is set to 1 with probability ε and uniformly sampled from {2,…,M\_max⁡} with probability 1−ε, where ε is a small number between 0–1.

At each forward pass, the Q-Network receives a reward of 1 when it halts with a correct answer and 0 when it choose to continue. At the same time, the HRM’s other components are updated.  
So the overall loss combines both the Q-head loss and the sequence-to-sequence loss - see screenshot *Loss\_function\_of\_the\_HRM\_Learning.png*

# References

[1] [The Unreasonable Effectiveness of Recurrent Neural Networks, A. Karpathy's blog, 2015](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fkarpathy.github.io%2F2015%2F05%2F21%2Frnn-effectiveness%2F&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558874066%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=ZSHIAuyEFv2Jia%2BLYJSF%2BANsgSBSF0SqrSrbqpGG8bo%3D&reserved=0)  
[2] [Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, K. Cho et al, 2014](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farxiv.org%2Fabs%2F1406.1078&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558900705%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=Ngs5NOViLN9s%2FoObYQ5qTvTRhjLQHd5DGycS1jbWiBI%3D&reserved=0)  
[3] [Sequence to Sequence Learning with Neural Networks, I. Sutsekever et al, 2014](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farxiv.org%2Fabs%2F1409.3215&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558915686%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=5T%2BxbGq58hL%2FZq4OH3BMskTLBP0Ti4M7TZDEde%2BkpT4%3D&reserved=0)  
[4] [Generating Sequences With Recurrent Neural Networks, A. Graves, 2013](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farxiv.org%2Fabs%2F1308.0850&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558929809%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=ncUbLQiBtxJq%2FSJ28MkYtcGX9hkvkYU%2BYEa82RN%2Fllo%3D&reserved=0)  
[5] [Recurrent Continuous Translation Models, N. Kalchbrenner and P. Blunsom, 2013](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Faclanthology.org%2FD13-1176.pdf&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558944954%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=s0XytBpGxNr5ZGmoyNArsWuHi0xlb1EJu6LC%2BpIiXxA%3D&reserved=0)  
[6] [Long Short Term Memory , Sepp Hochreiter, J. Schmidhuber, 1997](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fwww.bioinf.jku.at%2Fpublications%2Folder%2F2604.pdf&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558960785%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=%2BYzpvboxWM%2BEgsDcjEcn%2B%2Fu1vl2Ro9927dAP08IbZbg%3D&reserved=0)  
[7] [Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks, R. Staudemeyer, 2019](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Farxiv.org%2Fabs%2F1909.09586&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558975871%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=4Dz%2BG1seEYc0hv9CTvQEOlW5VtnWy6wmSNUJCXbWExY%3D&reserved=0)  
[8] [Gated Recurrent Unit, Wikipedia](https://na01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FGated_recurrent_unit&data=05%7C02%7C%7C0346b3173cd34d18537508dde5f4fcf2%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562558991898%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=g38W9egMJqrBcvfIvXVr%2Bi6SUzeBHCU%2Bq9gd%2FI1wTv4%3D&reserved=0)

[9] [All You Need to Know about the Attention Mechanism, Damien Benveniste, 2025](https://github.com/dimitarpg13/transformers_intro/blob/main/articles_and_books/attention/All_You_Need_To_Know_About_The_Self-Attention_Layer_Beneveniste_2025.pdf)