Note on Hierarchical Reasoning Models

Compiled by D. Gueorguiev, 8/28/25

On the attached screenshot HRM\_Architecture.png it is depicted the HRM architecture as presented in the paper.

# 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.

Clearly, HRM has recurrent structure.

1 ) the initial input, an initial low‐level weight z^o\_L, and a high‐level weight z^o\_H are randomly initialized.

2 ) The low‐level weight is updated at every step, while the high‐level weight is only updated every T steps.

3 ) Repeating this process for N times yields a total of N×T time‐step iterations.

4 ) One complete N×T time‐step sequence is called a single forward pass of the HRM.

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.

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 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%7C28a78185c1634cd0175b08dde5f4db09%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562009325865%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=9%2BmCI8W0Qr28whsQzq8If3KEY%2BgQ%2FaxJCM0Dup9nHF4%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%7C28a78185c1634cd0175b08dde5f4db09%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562009361285%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=ul7uwAra1RmvN9BljGjf8W6pbYuPXM3oMy4%2Fak4pjFg%3D&reserved=0) (implemented as Markov Decision Process (MDP) defined in page 8 of the paper; see screenshot Q\_Network\_implemented\_as\_MDP.png attached here) to adaptively determine the number of segments.

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%7C28a78185c1634cd0175b08dde5f4db09%7C84df9e7fe9f640afb435aaaaaaaaaaaa%7C1%7C0%7C638919562009376661%7CUnknown%7CTWFpbGZsb3d8eyJFbXB0eU1hcGkiOnRydWUsIlYiOiIwLjAuMDAwMCIsIlAiOiJXaW4zMiIsIkFOIjoiTWFpbCIsIldUIjoyfQ%3D%3D%7C0%7C%7C%7C&sdata=LN6yu9JTjNoOkFDlakl4nkRJTcJtrbQGQXTZTrNjV0c%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.

Appendix

**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 and abolishing RNNs altogether for SMT tasks.

**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)