Recurrent Attention Network

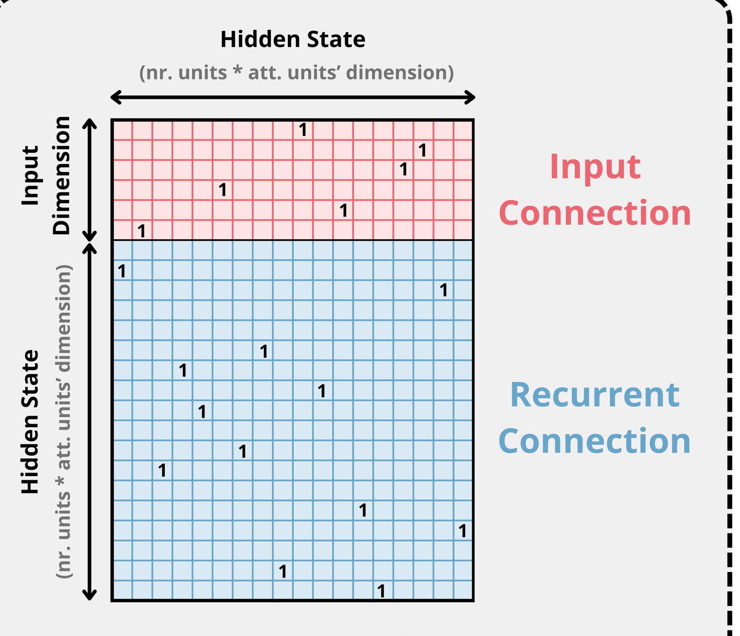
Yannis Bendi-Ouis[1,2,3,4], **Xavier Hinaut**[1,2,3,4]

[1]Bordeaux Univ., CNRS, IMN, UMR 5293, Bordeaux, France
[2]Inria Center of Bordeaux University, Bordeaux, France
[3]LaBRI, Bordeaux Univ., Bordeaux INP, CNRS UMR 5800, France
[4]Neurodegeneratives Diseases Institute, Bordeaux, France

This work is supported by Inria AEx BrainGPT project.
Thanks to Experimental Inria Cluster Plafrim.

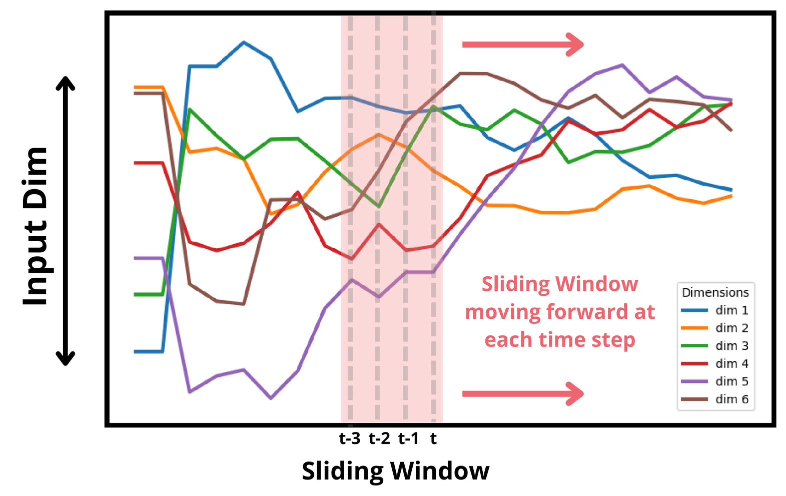
Inspired by Transformers, we're trying to make Reservoir Computing scalable, by using more complex units.

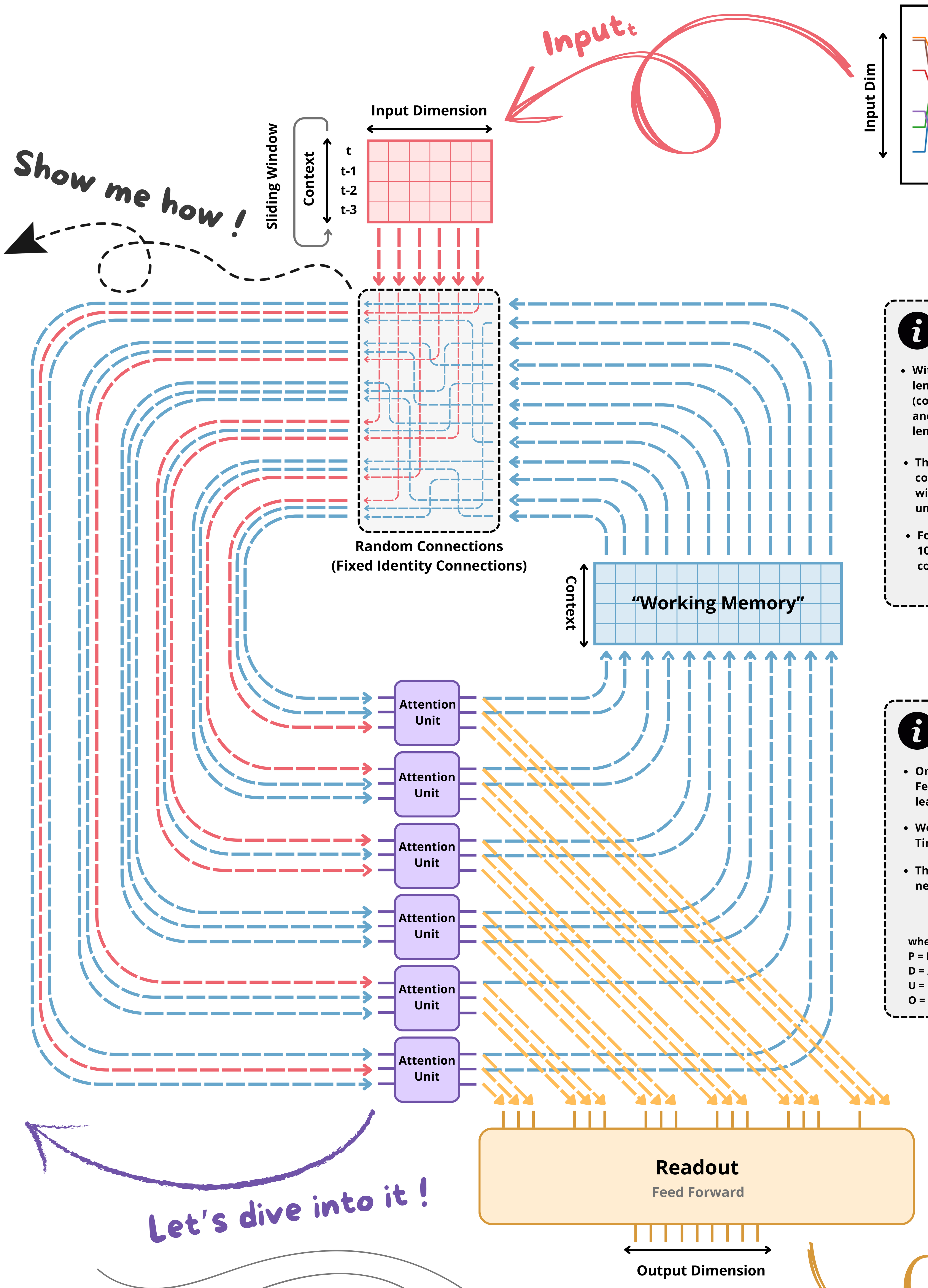
Random Connections (Fixed Identity Connections)



- Each input and recurrent dimensions are mapped to only one input dimension of only one unit.
- Every connections are sparse and topographic / identity connections.

Show me how !





Working Memory and Context Length

- With a "Working Memory" and a fixed context length, we can compute each step in O(1) (constant time) instead of O(n²) (quadratic time) and still access information beyond context length thanks to recurrent connections.
- The shorter the context length is, the faster the computations. It's better to have a lot of units with a very small context length, than a few units with a very large context length.
- For example, one unit of with context length of 100 takes more time than 20 units of with context length of 5 :
  $100^2 > 20*5^2$

Trainable Parameters and Learning

- Only Attention Weight (W_q, W_k, W_v), Unit's Feed-Forward (FF_{in}, FF_{out}) and the Readout are learned.
- We use Truncated BackPropagaton Throught Time with the length of the context.
- The number of parameters trained in the network follows that formula :

$$P = 11D^2U + UDO + 8DU + O$$

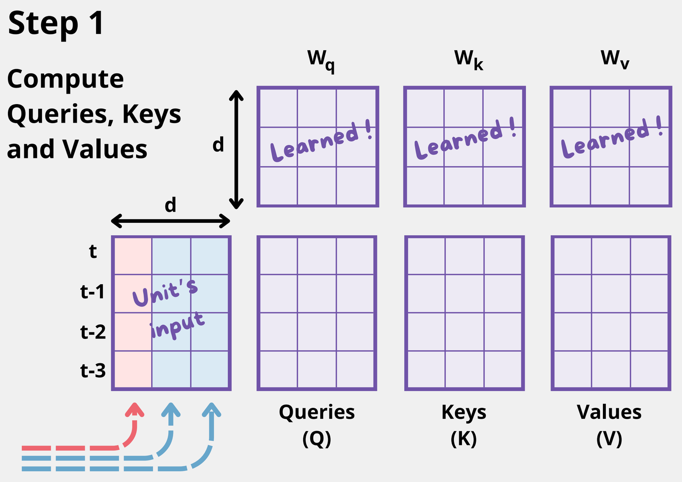
where
P = Number of Trained Parameters
D = Attention Units' Dimension
U = Number of Attention Units
O = Output Dimensions

Inside an Attention Unit

Attention Unit are inspired from Attention Head in [1].

Step 1

Compute Queries, Keys and Values

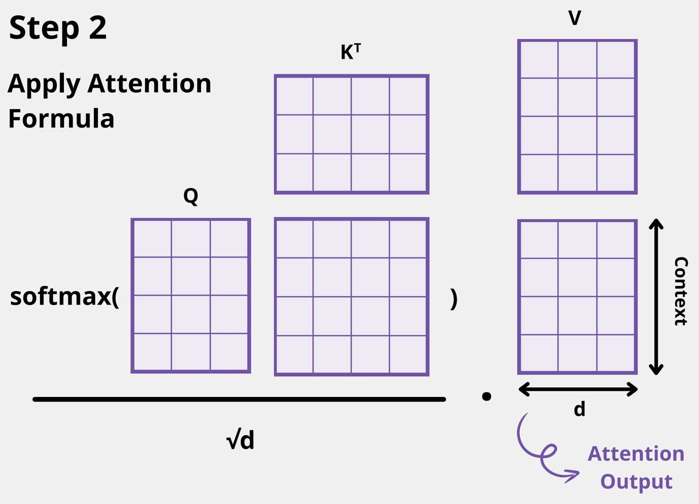


- We could use fixed and random W_q and W_k to compute our Queries and Keys.
- It might extract some diverse patterns and learn to work with it in the same way Reservoir Computing [4] does.
- As we observe in some experiences [3], in doing so, training could converge faster.

PERSPECTIVES

Step 2

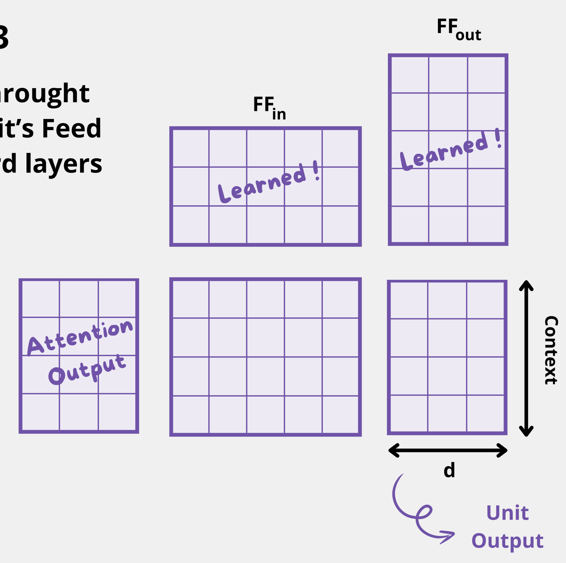
Apply Attention Formula



- Input connections brings new and fresh information of time t, while Recurrent connections brings a mix of old informations computed some steps before.
- Thus, we can apply attention on spatial & temporal informations.

Step 3

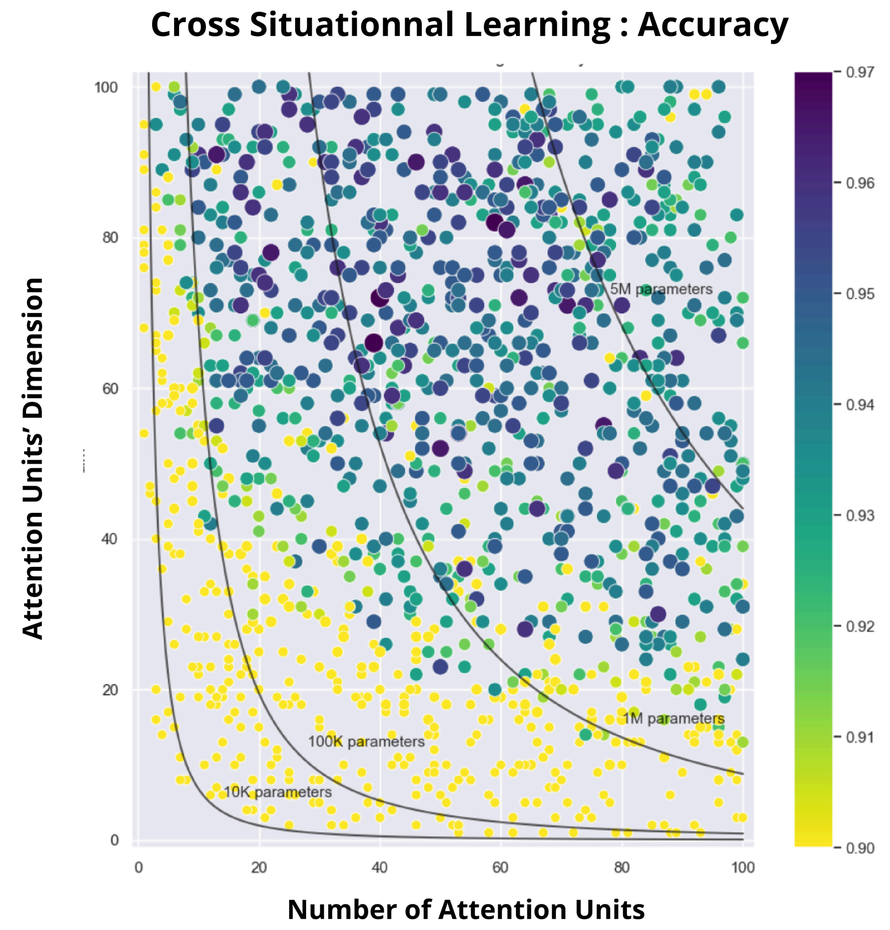
Pass throught the Unit's Feed Forward layers

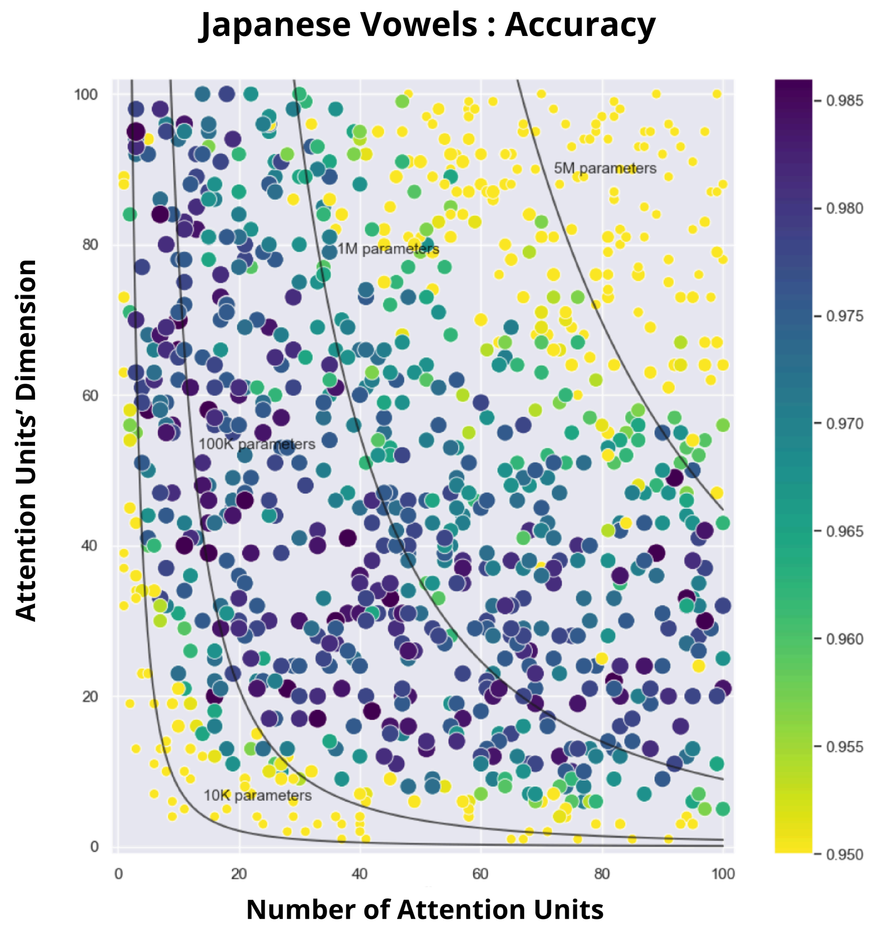


- According to [2], this step retrieves and combines memories.
- We could decompose those Units into two different ones :
  - Attention Only (step 1-2)
  - Memory Only (step 3)
- Thus, some memories could be shared between different Units of Attention Only.
- And, we will be able to tune the model in function of the task's needs : more or less attention or memory units.

PERSPECTIVES

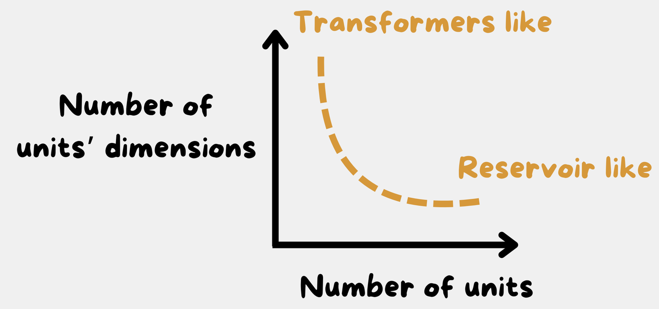
What about the results ?





How to understand the results ?

- This model is a middle ground between Transformers and Reservoir Computing.
- We get closer to Transformers when the number of units is low compared to units' dimension.
- We get closer to Reservoir Computing when number of units is high compared to units' dimension.



References

[1] Vaswani, "Attention is all you need." Advances in Neural Information Processing Systems (2017).
[2] Geva, "Transformer feed-forward layers are key-value memories." arXiv preprint arXiv:2012.14913 (2020).
[3] Léger, "Evolving Reservoirs for Meta Reinforcement Learning." International Conference on the Applications of Evolutionary Computation (2024).
[4] Jaeger, "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note." GMD Technical Report 148.34 (2001).