
Master's Thesis Presentation

# Modelling Implicit Acquisition of Sequential Information Using a Neocortical Neural Network:

# Hierarchical Temporal Memory

Date: 29-Jan-2021

By: Taher Habib

# Objective of the Thesis

What are the learning mechanisms used by Brain for acquiring sequential information implicitly?

## Implicit Learning

- Learning in automatic, non-intentional way

  "without any requirements of awareness of either the process or the product of acquisition" (Reber et al., 1991)


- No separate training & testing

# Motivation behind the Thesis

What are the learning mechanisms used by Brain for acquiring sequential information implicitly?

## Why does Implicit Learning Matter?

- Brain Does it ALMOST ALL the time!
- Building Brain-inspired Machine Intelligence could lead to more of:
  - Robustness
  - Generalizability
  - Efficiency – Data Requirement & Learning Energy

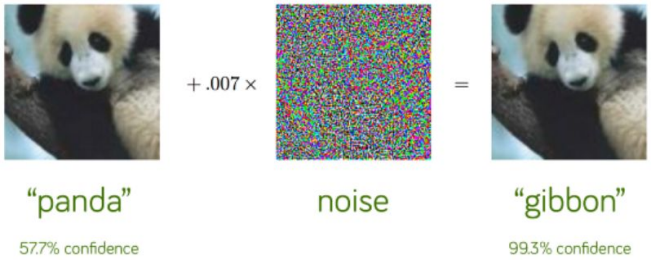



Image Source: Explaining and Harnessing Adversarial Examples, Goodfellow et al, ICLR 2015.

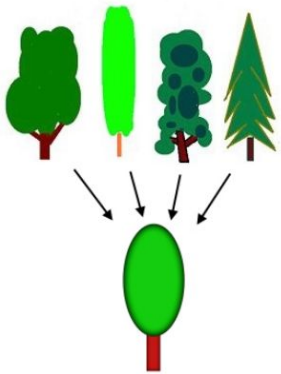


Image Source: Wikipedia contributors. "Concept." *Wikipedia, The Free Encyclopedia.* Wikipedia, The Free Encyclopedia, 14 Jan. 2021. Web. 23 Feb. 2021.

# Past Attempts at Sequence Learning

## Statistical / Classical ML-based Approaches

- Artificial Neural Networks – TDNNs
  (Waibel et al., 1989)
- Recurrent Neural Networks
  (Jordan, 1986; Elman, 1990)
- Hidden Markov Models
  (Fine et al., 1998)
- LSTMs
  (Hochreiter & Schmidhuber, 1997)

**Architecture**
Brain-inspired

**Performance**
High on Complex Sequence Learning

+

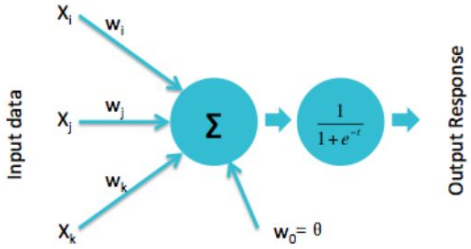
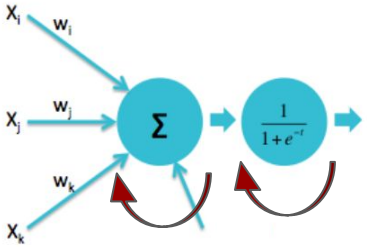



Image Source: On the Origin of Deep Learning, Wang & Raj , 2017, arXiv

**Learning**
NOT (?)
Brain-inspired

−



Backpropagation of Errors

While classical, ML methods achieve a high performance on various Sequence Learning tasks, they hardly attempt to model biology in any detail. As a result, they are not helpful in illuminating on the impressive ability of the neocortical neuronal mechanisms in the brain that perform implicit sequence learning.

# Past Attempts at Sequence Learning

## Brain-inspired Approaches

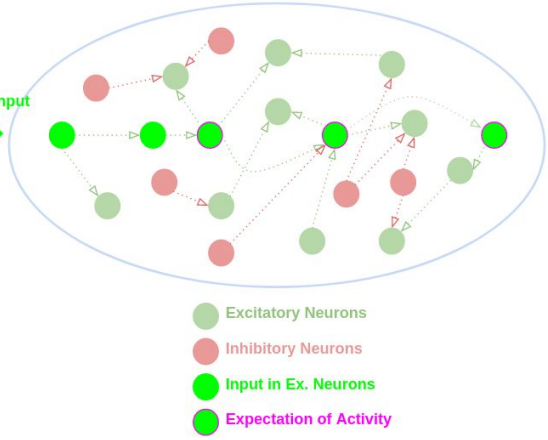
- Recurrently-connected Excitatory Neural Network
  (Rao & Sejnowski, 2001)
- Self Organising Recurrent Network (SORN)
  (Lazar, Pipa & Triesch, 2009)
- Leaky Integrate & Fire Neurons in SORN (LIF-SORN)
  (Klos, Miner & Triesch, 2018)

**Architecture**
Brain-inspired

**Learning**
STDP-based
Hebbian Learning

+

Excitatory Input




**Performance**
(So far) Only
Simple Sequence
Learning

–

While the aforementioned brain-inspired models replicate the neuronal biology in greater detail, they fail to acquire sequential information in more complex tasks with long-range dependencies.

# My Thesis' Attempt

What are the learning mechanisms used by Brain for acquiring sequential information implicitly?

## Neocortical Network: Hierarchical Temporal Memory

Based on Computational Principles found in Brain

- Sparse Coding
- Neurons Detecting Spike-coincidence using Dendrites
  (NOT summing input linearly)

**Architecture**
Neocortex-inspired,
Neurons with
Dendrites

**Learning**
Hebbian Plasticity

**Performance**
Learns Long-term
Dependencies
w/o Supervision

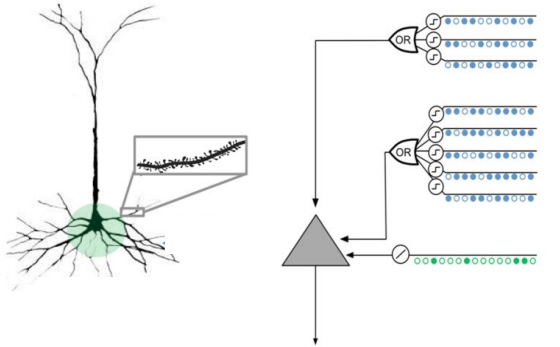



Image Source: Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex, Hawkins & Ahmad, 2016, Neural Circuits

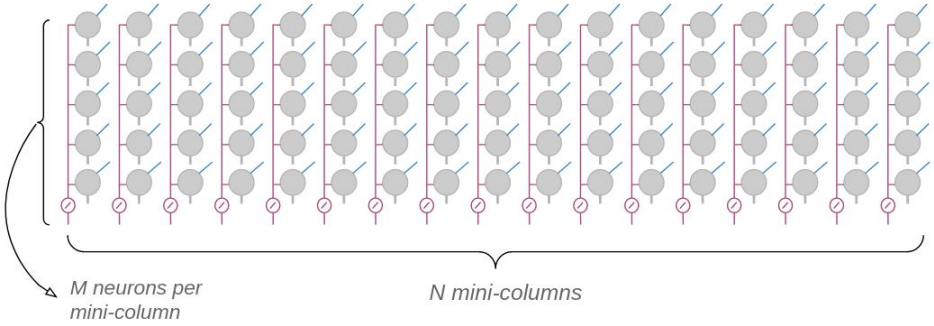
## Learning Task: Artificial Grammar Learning

- Implicit Seq. Learning ≅ Implicit Grammar Learning (Reber, 1967; Petersson & Hagoort, 2010)
- Has nontrivial, higher-order temporal dependencies
  (Petersson et al., 2005)

My thesis' objective is to address the issue of *implicitly* learning sequences with nontrivial, higher-order time-lag dependencies, using a brain-inspired neural network architecture which deploys only biologically-plausible learning mechanisms.

# HTM Network

## Network of HTM Neurons Arranged in Minicolumns



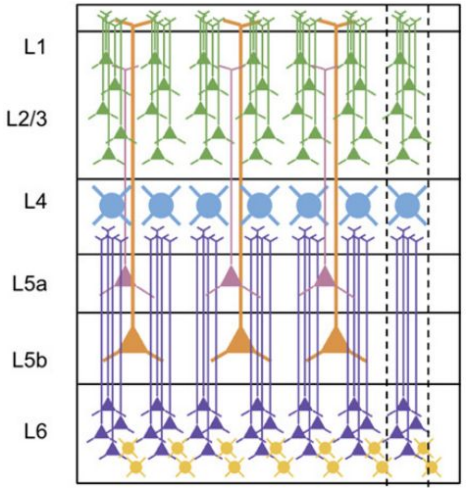




Image Source: An attempt at the Unified Theory of Neocortical Microcircuit in Sensory Cortex, Bennett, 2020, Neural Circuits

# How does HTM Learn Higher-Order Temporal Dependencies?

## Sparse Distributed Representations (SDRs)

- HTM uses SDRs to store the same feedforward input differently in different temporal contexts

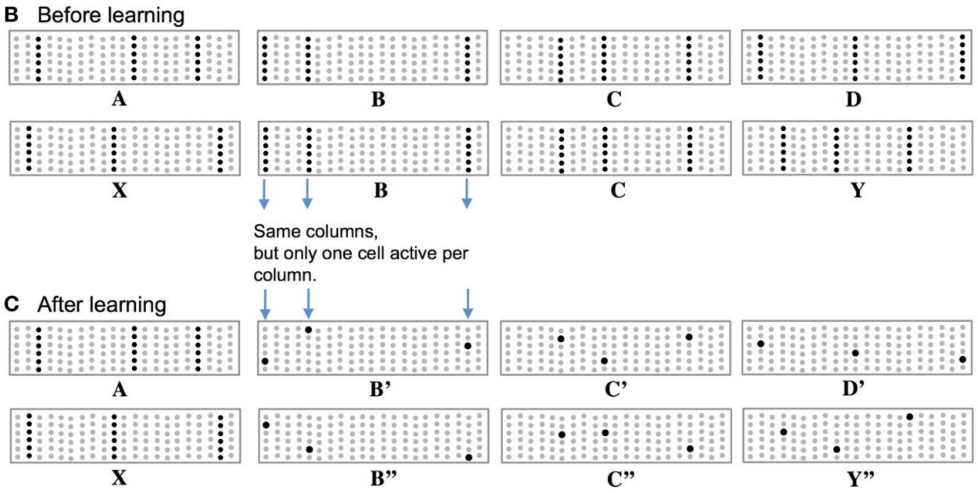



Image Source: Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex, Hawkins & Ahmad, 2016, Neural Circuits

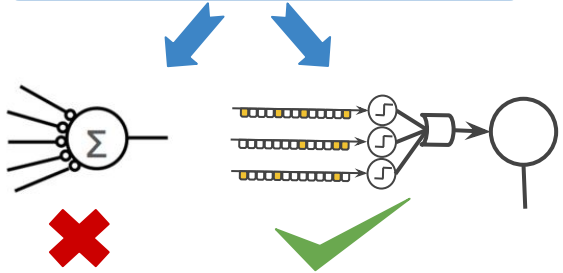
HTM uses sparse activation patterns in its neuronal population to store distinct temporal contexts; thus, lessening the dependency on synaptic connection weights for encoding temporal information.

# How does HTM Learn Higher-Order Temporal Dependencies?

## Spike Coincidence Detection

Use of SDRs to encode information

Neurons to detect synchronized activation patterns in the neuronal assemblies




## Using Dendrites

One cell per activation pattern recognition, impractical

Every cell MUST be able to recognize multiple activation patterns independently

Using Dendrites as independent pattern recognizers (Polsky et al., 2004; Major et al., 2013)

Use of HTM synchronized sparse activation patterns automatically calls for a need to use neurons which act as coincidence detectors. (König et al. 1996)
Use of dendrites that can independently recognize coincident activity dramatically increase the storage capacity of neurons. (Poirazi & Mel, 2001)

# HTM Neuron

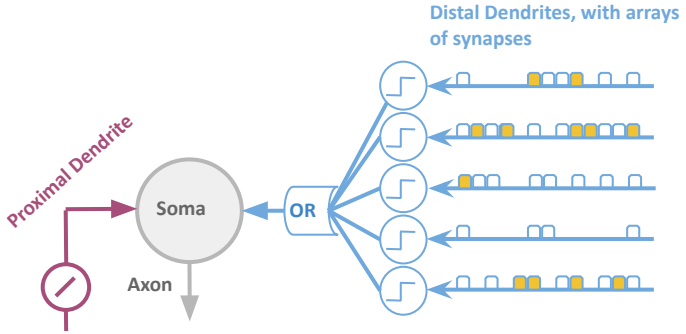
## Dendrites

### Proximal Dendrite

- Feeds External input (feedforward)
- **Activated** when External input is present
- Has no trainable synapses

### Distal Dendrites

- Feeds Lateral input (temporal context)
- **Activated** when more than $\theta$ connected synapses receive presynaptic activation
- Has trainable synapses




## Synapses

Synapse is a *directed* connection from the "axon" of one neuron to a particular dendritic segment of another neuron in the network.

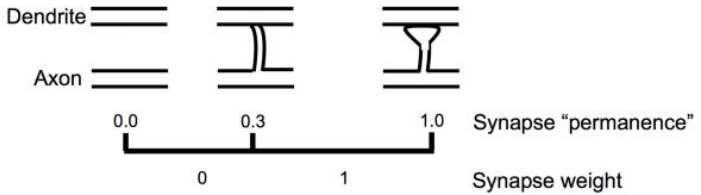
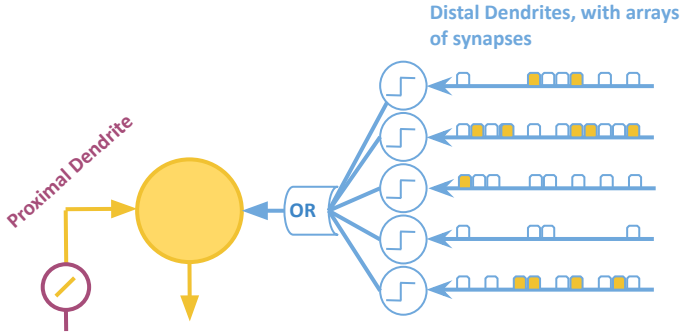



Image Source: Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex, Hawkins & Ahmad, 2016, Neural Circuits

# HTM Neuron

## States of Neuron

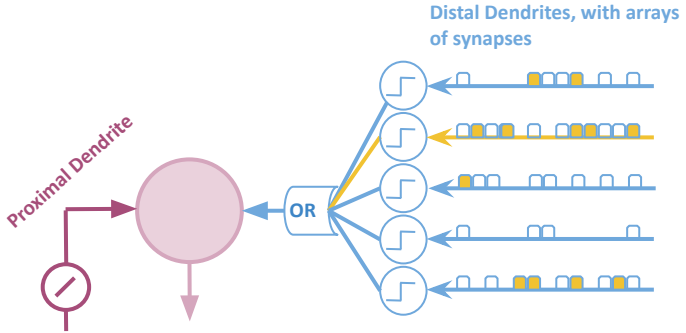
- Active
  - if proximal dendrite active



# HTM Neuron

## States of Neuron

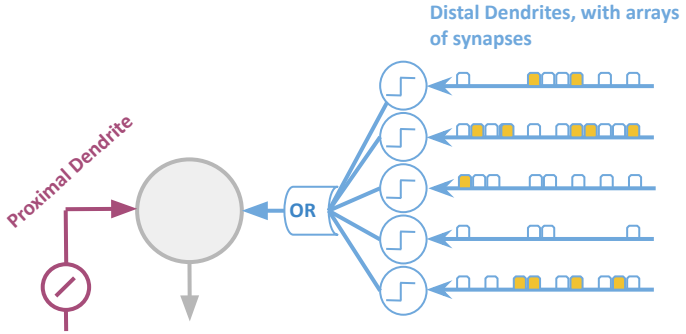
- Active
  - if proximal dendrite active
- Predictive
  - if one/more distal dendrite active



# HTM Neuron

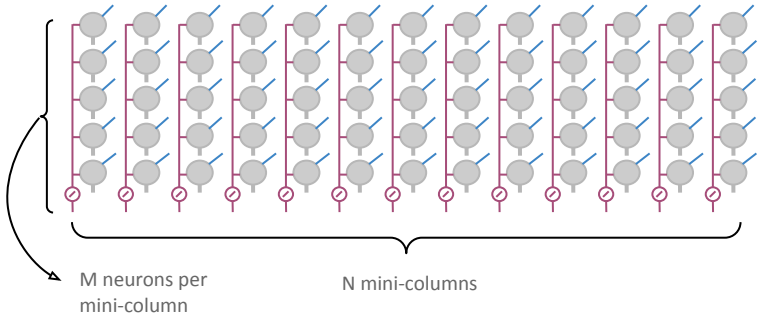
## States of Neuron

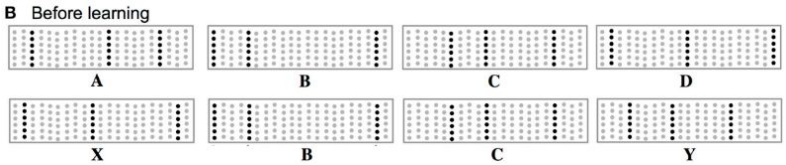
- Active
  - if proximal dendrite active
- Predictive
  - if one/more distal dendrite active
- Inactive
  - otherwise



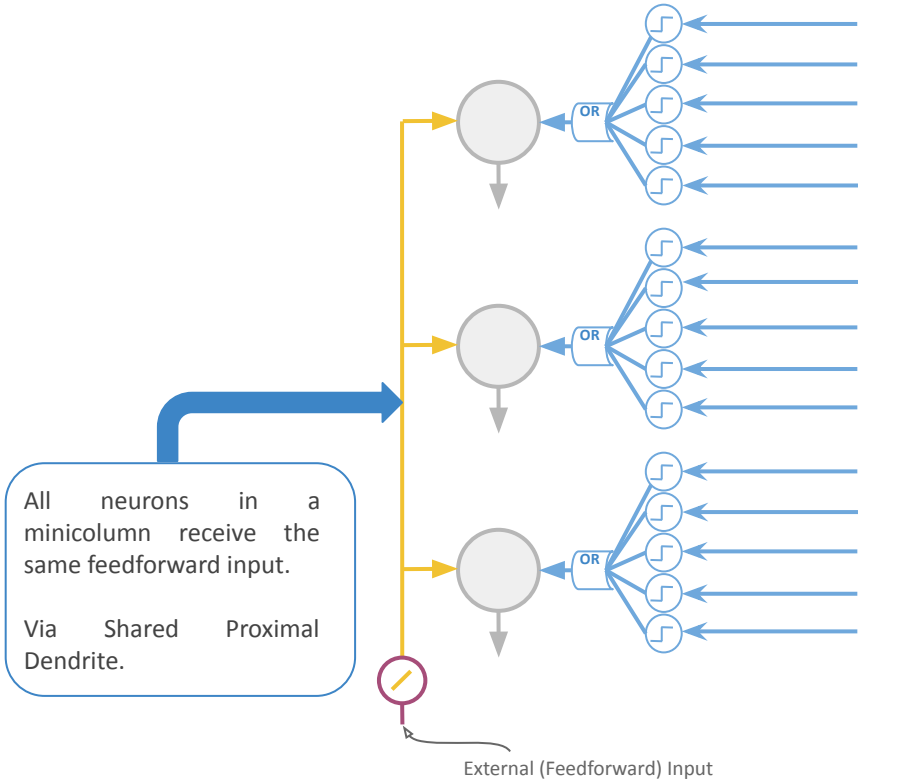
# Minicolumn in HTM

## Network of Neurons Arranged in Minicolumns




## Neurons in a Minicolumn



# Excitation & Inhibition in a Minicolumn in HTM

Let coincidence threshold $\theta = 6$:

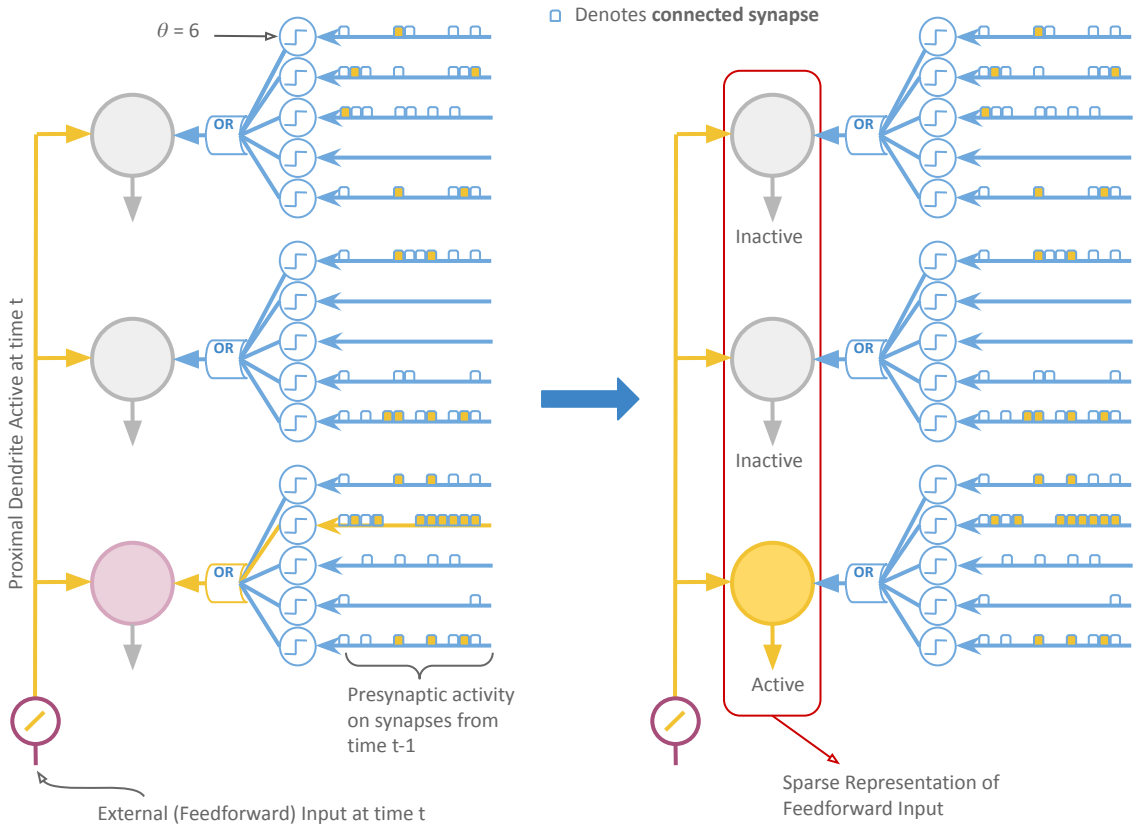
- 2nd distal dendrite of 3rd neuron is active due to enough presynaptic activity from time *t*-1
- 3rd neuron is predictive

If External Input activates the proximal dendrite at time *t*:

- 3rd neuron **inhibits** other two
- Only 3rd neuron becomes active

## States of Neuron

- Active
  - if proximal dendrite active, and
  - *NO inhibition from other neurons*
- Predictive
  - if one/more distal dendrite active
  - *predictive neuron inhibits other neurons in the column*



# Excitation & Inhibition in a Minicolumn in HTM

Let coincidence threshold $\theta = 9$:

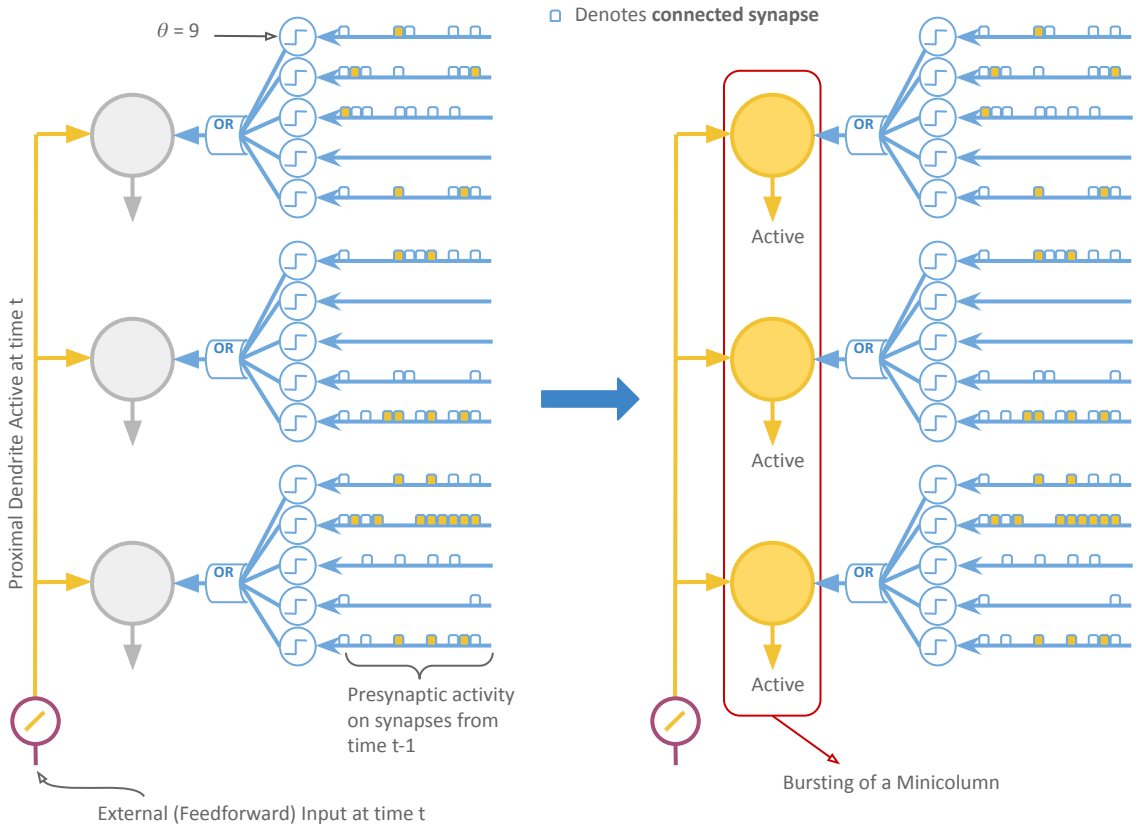
- No distal dendrite is active
- No neuron is predictive

If External Input activates the proximal dendrite at time *t*:

- Since NO neuron is inhibited, all neuron become active

## States of Neuron

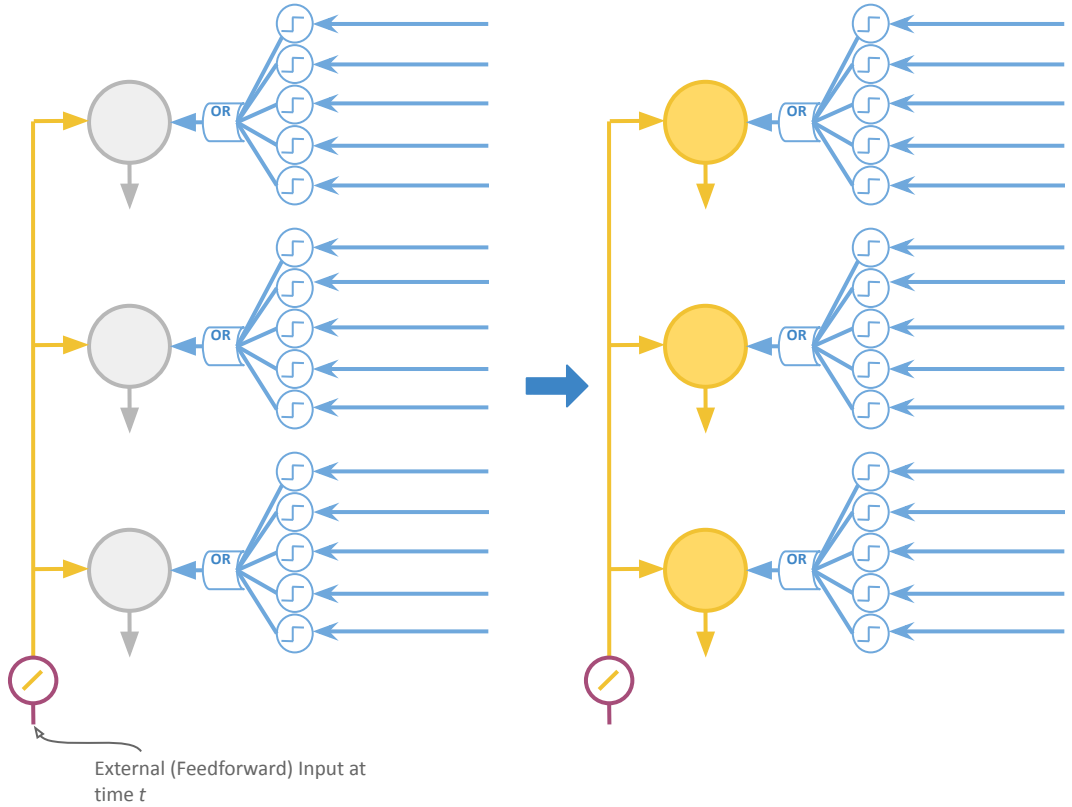
- Active
  - if proximal dendrite active, and
  - NO inhibition from other neurons
- Predictive
  - if one/more distal dendrite active
  - predictive neuron inhibits other neurons in the column



# Learning in HTM

## Initialization

- No Cell has any synapse at any $\varrho$
- All columns burst predominantly



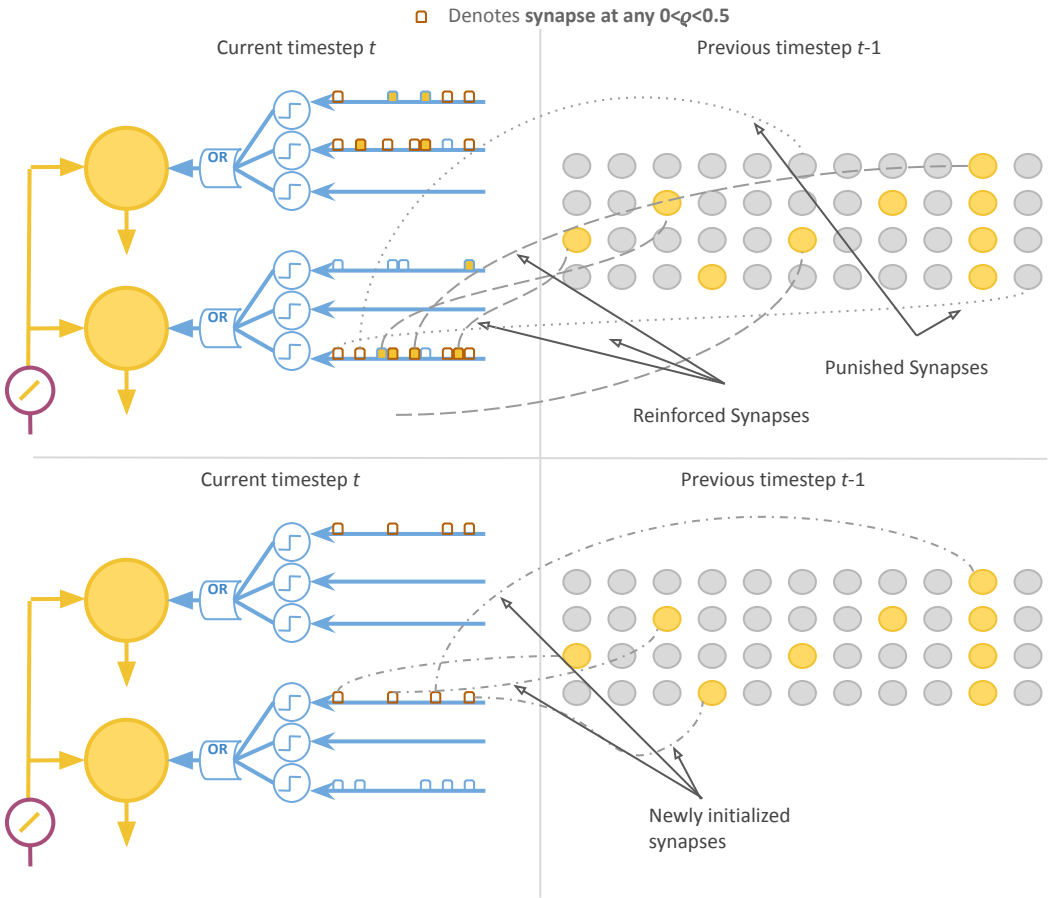
# Learning in HTM

## When a Column Bursts

- Choose a dendrite with most no. of active synapses, at any $\varrho$ value:
  - Reinforce $\varrho$ for active synapses
  - Punish $\varrho$ for inactive synapses

- If no such dendrite exists, choose an unused dendrite randomly and initialize potential synapses to a subset of previously active cells

*Network chooses a single cell in the column to represent the temporal context from previous timestep*

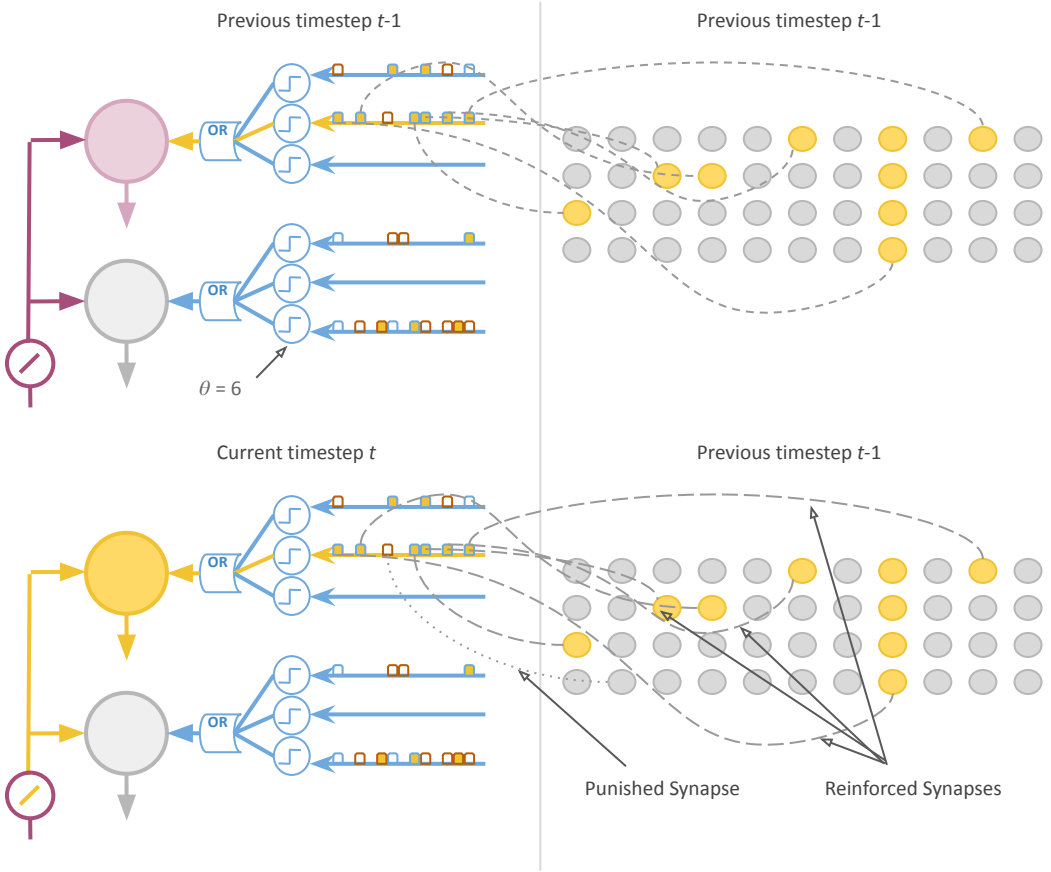


# Learning in HTM

## When a Column has a Correctly Predicted Cell

- Choose active dendrite on the predicted cell:
  - Reinforce $\varrho$ for active synapses
  - Punish $\varrho$ for inactive synapses

*Dendrites become specialized in detecting one particular activation pattern in the network.*

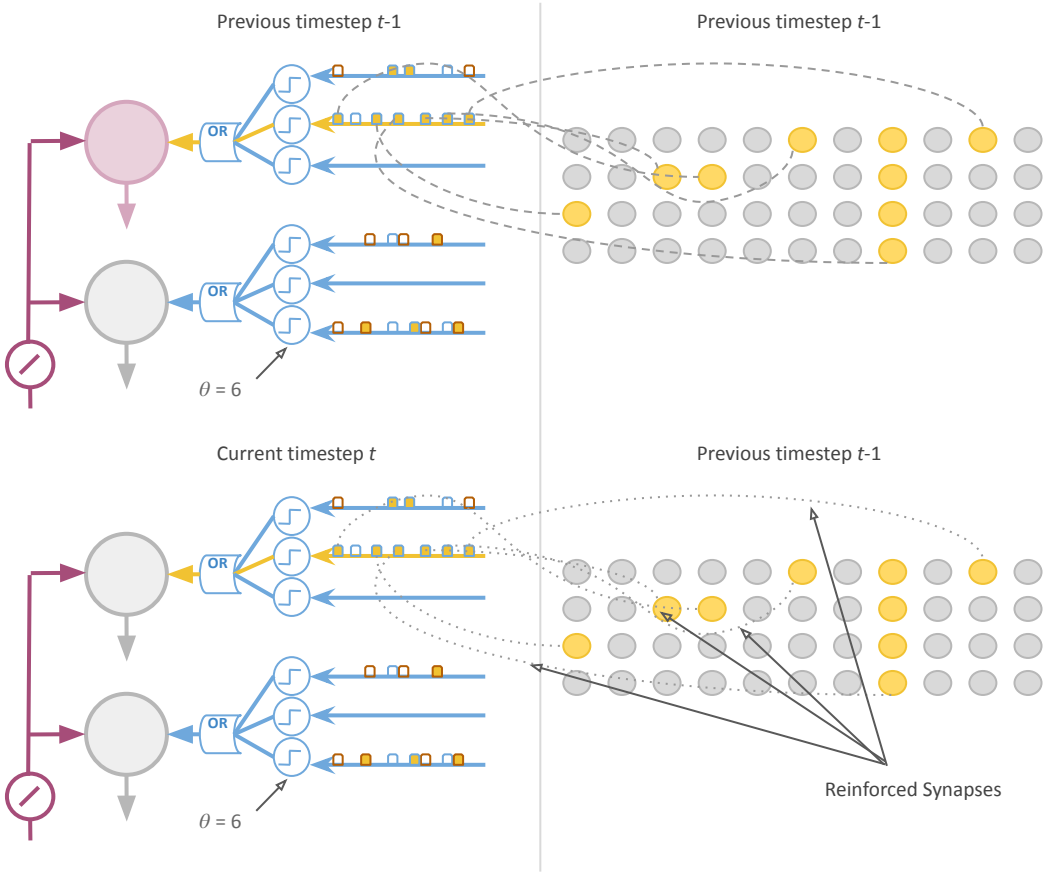


# Learning in HTM

## When a Cell is Incorrectly Predicted

- Choose active dendrite on the predicted cell:
  - Slightly decay $\varrho$ for active synapses

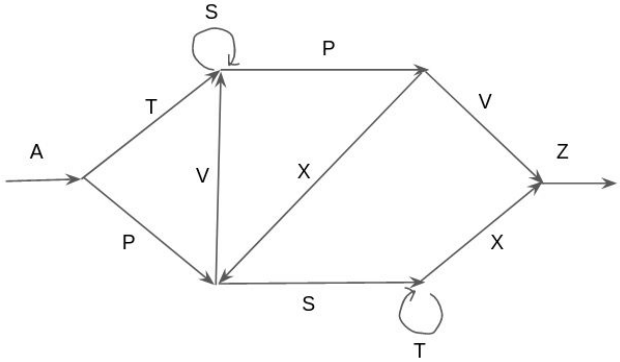
*This frequently happens when multiple predictions for the next timestep are correct.*



# Learning Task – Reber Grammar Acquisition

## Simple Reber Grammar (SRG)

- Finite State Language
  (Chomsky & Miller, 1958)
- Transition from one node to another produces a letter
- Every sentence (string) starts at leftmost node & ends at rightmost node
- In case of multiple possible transitions, all are equally likely
- Infinitely many strings possible




Markov Process generating SRG Strings

Sample Strings

{ ATSSPVZ,
APVPVZ,
ATPXVSPXSTTXZ,
APVSSSSPXSXZ,
... }

SRG is a rule-based – but not entirely deterministic – artificial grammar that *attempts* to model the sequential / syntactical structure of language in humans.

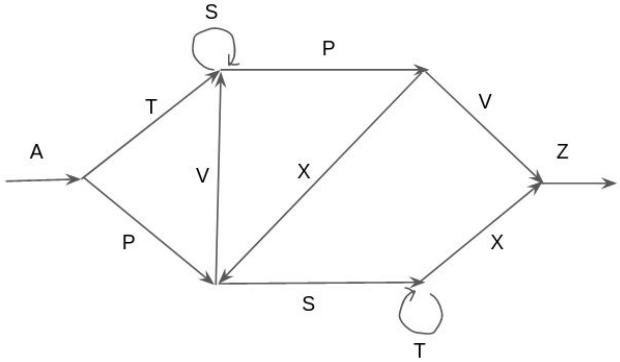
# Learning Task – Reber Grammar Acquisition

## Learning SRG means Learning a Second Order Process

- Given any two consecutive symbol → Probability of occurrence of next symbols can be determined
  (Petersson et al., 2005)
- A 3-gram model is best performing model on SRG
  (Duarte et al., 2014)

$$P_{target} = P(S_t \mid S_{t-1}, S_{t-2})$$

*Any Algorithm that recognizes and acquires a set of trigrams generated from SRG can be said to have learnt the Simple Reber Grammar.* (Petersson et al., 2005)




Markov Process generating SRG Strings

Sample Trigrams

{ ATS, PVZ, XVS, STT, ... }

Learning SRG involves *learning second-order temporal dependencies.*
For a biologically plausible network that learns SRG implicitly, the network should arguably have a representational capacity enough to store every SRG letter distinctly based on the context in its temporal history.

# Model Setup

| Parameters | |
|---|---|
| No. of neurons per column, *M* | 16 |
| No. of columns per letter, *k* | 32 |
| No. of minicolumns, *N* | 32*7 = 224 |
| Max. allowed dendrites / neuron | 128 |
| Max. allowed connected synapses / dendrite | 49 |
| Coincidence Threshold, $\theta$ | 18 |
| Permanence Increment | 0.1 |
| Permanence Decrement | 0.1 |
| Permanence Decay | 0.02 |
| Initial Permanence | 0.25 ± 0.002 |

| | A | T | S | X | P | V | Z |
|---|---|---|---|---|---|---|---|
| **0** | 12 | 10 | 3 | 1 | 8 | 2 | 0 |
| **1** | 13 | 15 | 4 | 21 | 11 | 5 | 7 |
| **2** | 14 | 17 | 9 | 29 | 22 | 6 | 16 |
| **3** | 35 | 18 | 23 | 45 | 27 | 25 | 24 |
| **4** | 36 | 19 | 32 | 49 | 41 | 47 | 26 |
| **5** | 37 | 20 | 42 | 50 | 44 | 55 | 30 |
| **6** | 38 | 28 | 46 | 51 | 48 | 56 | 34 |
| **7** | 40 | 31 | 52 | 64 | 62 | 59 | 53 |
| **8** | 43 | 33 | 54 | 69 | 72 | 74 | 58 |
| **9** | 60 | 39 | 61 | 70 | 77 | 75 | 65 |
| **10** | 80 | 57 | 66 | 73 | 85 | 81 | 68 |
| **11** | 83 | 63 | 71 | 78 | 93 | 88 | 97 |
| **12** | 91 | 67 | 104 | 82 | 94 | 106 | 99 |
| **13** | 92 | 76 | 107 | 84 | 95 | 108 | 110 |
| **14** | 96 | 79 | 109 | 90 | 100 | 112 | 114 |
| **15** | 102 | 86 | 113 | 98 | 103 | 117 | 115 |

| | A | T | S | X | P | V | Z |
|---|---|---|---|---|---|---|---|
| **16** | 105 | 87 | 132 | 111 | 122 | 135 | 120 |
| **17** | 118 | 89 | 134 | 116 | 125 | 138 | 124 |
| **18** | 121 | 101 | 137 | 119 | 127 | 141 | 126 |
| **19** | 123 | 131 | 139 | 143 | 128 | 142 | 145 |
| **20** | 136 | 147 | 144 | 154 | 129 | 161 | 151 |
| **21** | 148 | 159 | 146 | 157 | 130 | 164 | 155 |
| **22** | 152 | 160 | 153 | 162 | 133 | 174 | 166 |
| **23** | 163 | 171 | 158 | 165 | 140 | 175 | 178 |
| **24** | 168 | 173 | 170 | 169 | 149 | 189 | 184 |
| **25** | 172 | 182 | 186 | 187 | 150 | 190 | 194 |
| **26** | 176 | 183 | 192 | 193 | 156 | 198 | 195 |
| **27** | 179 | 197 | 200 | 199 | 167 | 210 | 196 |
| **28** | 185 | 202 | 208 | 203 | 177 | 211 | 201 |
| **29** | 188 | 206 | 209 | 214 | 180 | 215 | 205 |
| **30** | 191 | 207 | 218 | 217 | 181 | 220 | 216 |
| **31** | 204 | 213 | 222 | 223 | 212 | 221 | 219 |

# Learning Single Reber String – *APVPVZ*

The network has:

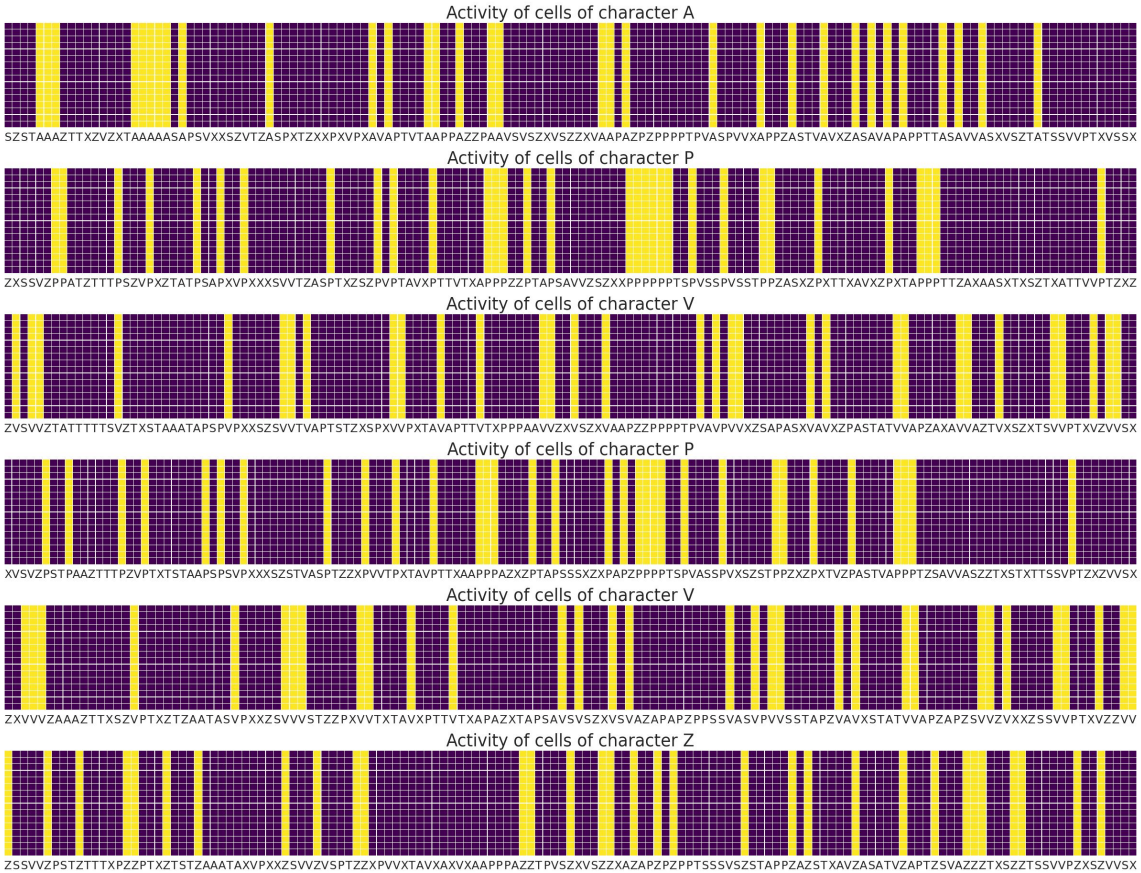
- 32 minicolumns (with 16 neurons each) per character
- Total of 224 (=32*7) minicolumns

## Input

- Single Reber String *APVPVZ* in inputstream
  (50 repetitions)
- Learning *reset* at start of each string
  (*Z*→*A* transition is not learnt)

Activations in network at first 6 timesteps of first repetition of *APVPVZ*

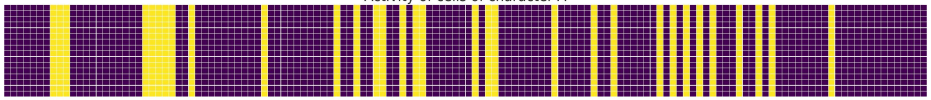
All minicolumns burst for each letter of the string



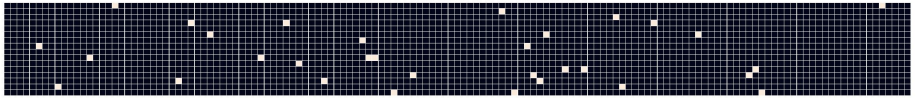
# Learning Single Reber String – *APVPVZ*

Activations and Predictions at 7th repetition of *APVPVZ*

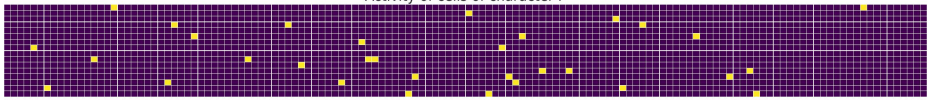
Activity of cells of character A



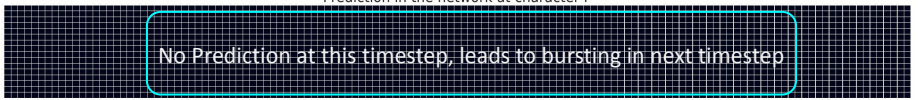
Prediction in the network at character A



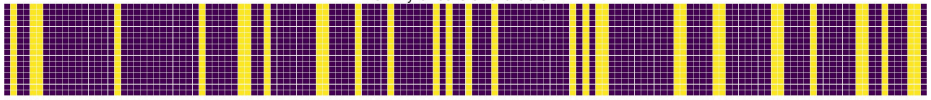
Activity of cells of character P



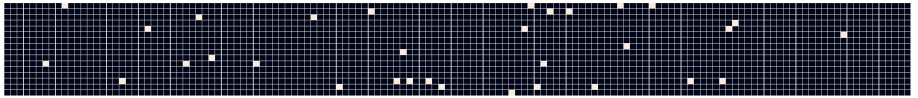
Prediction in the network at character P



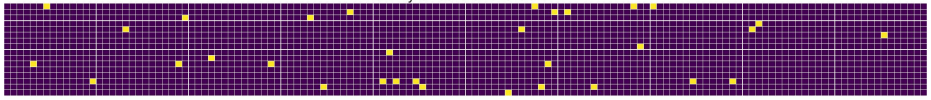

Activity of cells of character V



Prediction in the network at character V

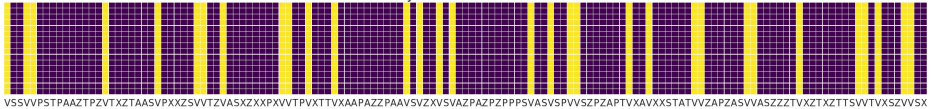


Activity of cells of character P

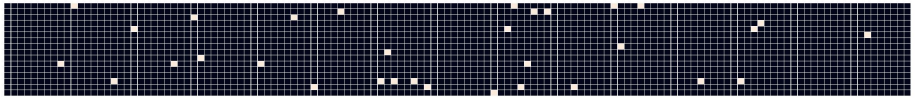


Prediction in the network at character P

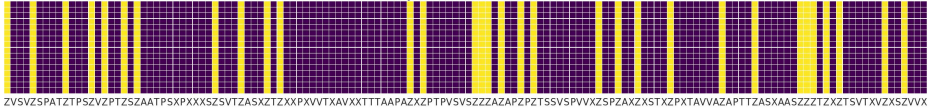

Activity of cells of character V

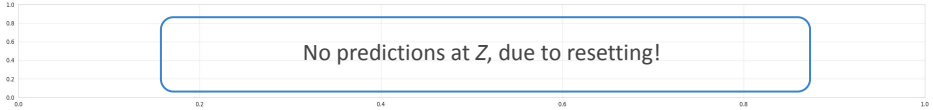


Prediction in the network at character V



Activity of cells of character Z



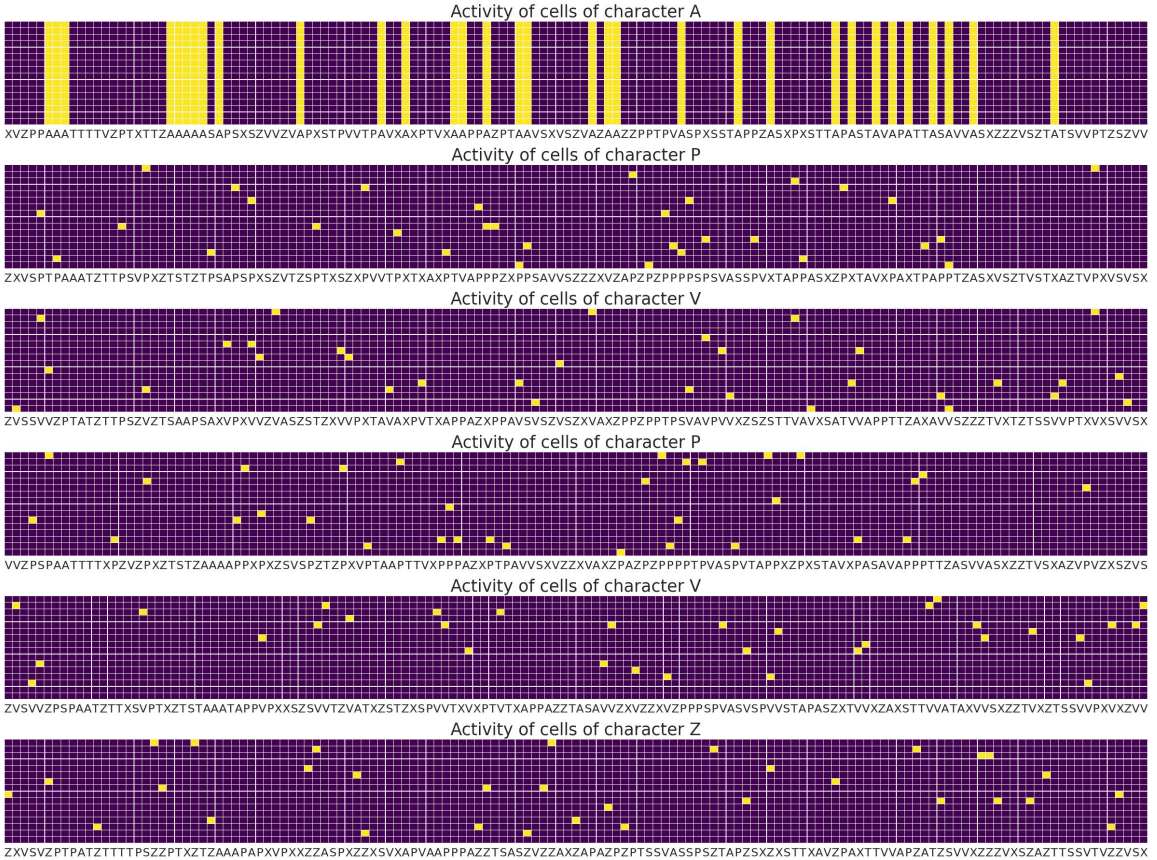


# Learning Single Reber String – *APVPVZ*

Network activations at final 50th repetition

Distinct SDRs learnt for *P* and *V* in distinct temporal contexts

Predictions' SDRs are same as activations shown here!



# Learning Single Reber String – *APVPVZ*

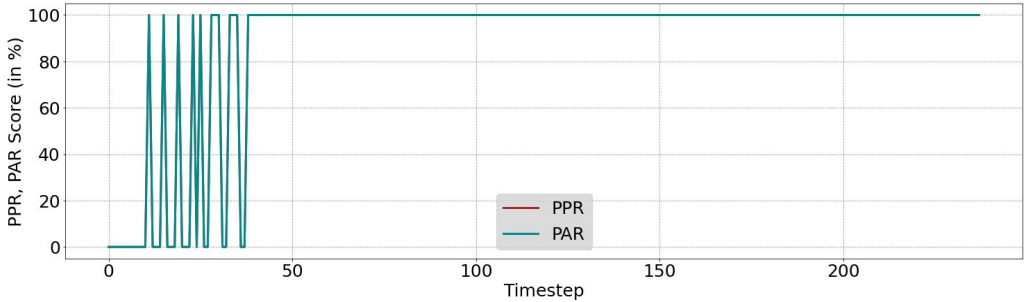
## Performance Evaluation

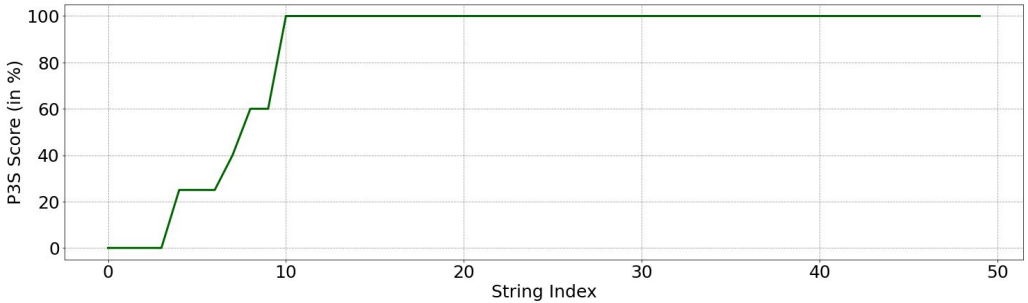
Prediction Accuracy Ratio (PAR) = $\frac{\#\ Correctly\ predicted\ columns}{\#\ Predicted\ columns}$

Prediction Performance Ratio (PPR) = $\frac{\#\ Correctly\ predicted\ columns}{\#\ Expected\ prediction\ columns}$

Prediction Performance Per String (P3S) = *Averaged PPR per string*

## Performance over 50 Repetitions



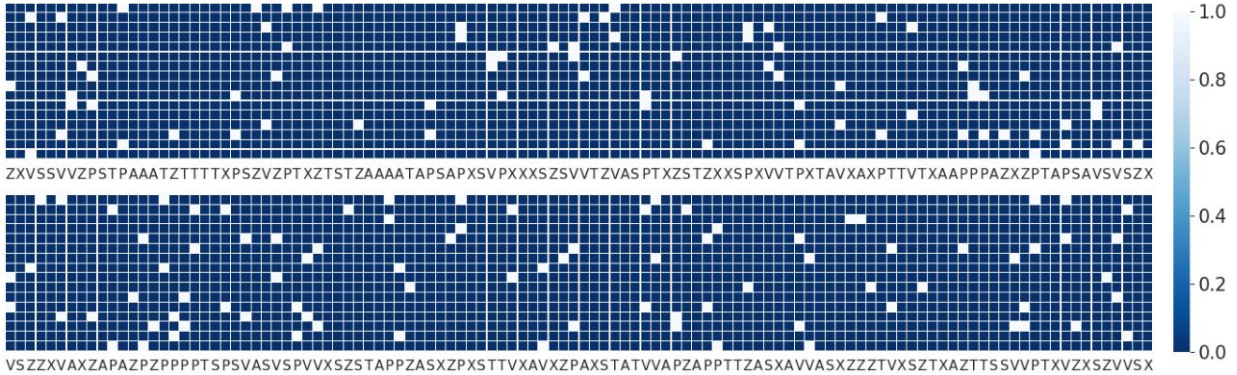




HTM successfully learns *APVPVZ* after 10 repetitions.

# Learning Single Reber String – *APVPVZ*

Dendritic Growth in Network

All 224 minicolums shown in two panels

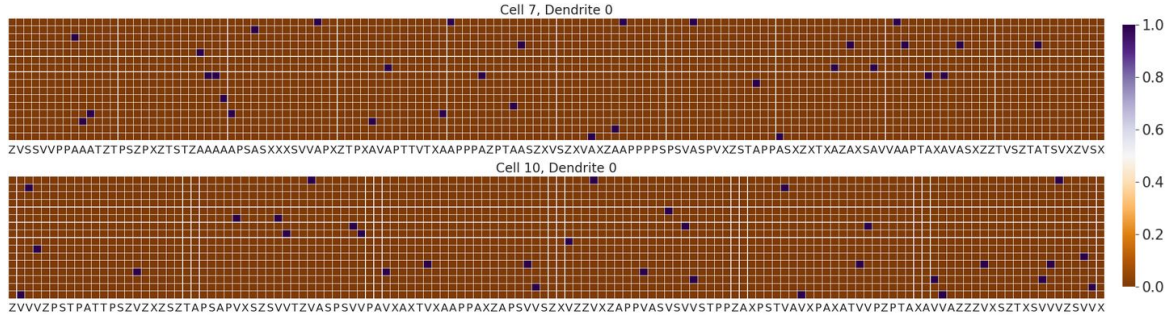



Every *P* and *V* minicolumn has two dendrites created for storing the two transitions – (*A*→*P*) & (*V*→*P*) in case of *P;* (*A*→*P*→*V*) & (*V*→*P*→*V*) in case of *V.* One dendrite is created in every minicolumn of *Z* for storing transition *V*→*Z*.

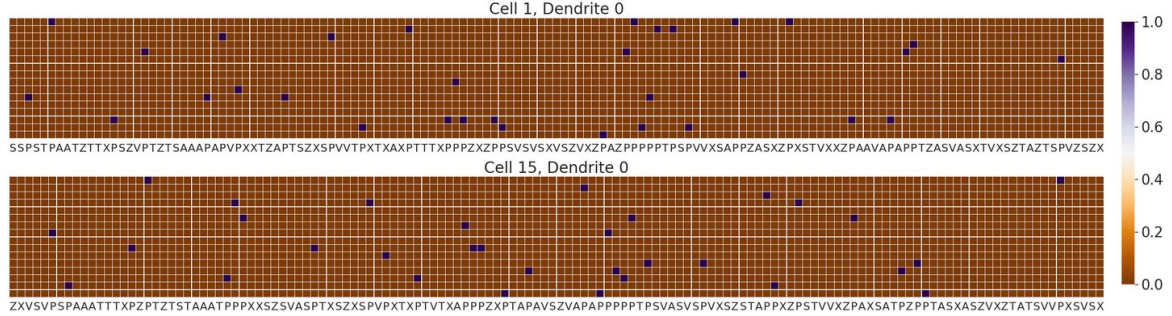
# Learning Single Reber String – *APVPVZ*

## Synaptic Profile of Dendrites

Samples of two dendrites on *P*-cells storing connections to *A* and *V*.




Samples of two dendrites on *V*-cells storing connections to distinct *P*-cells with distinct contexts.



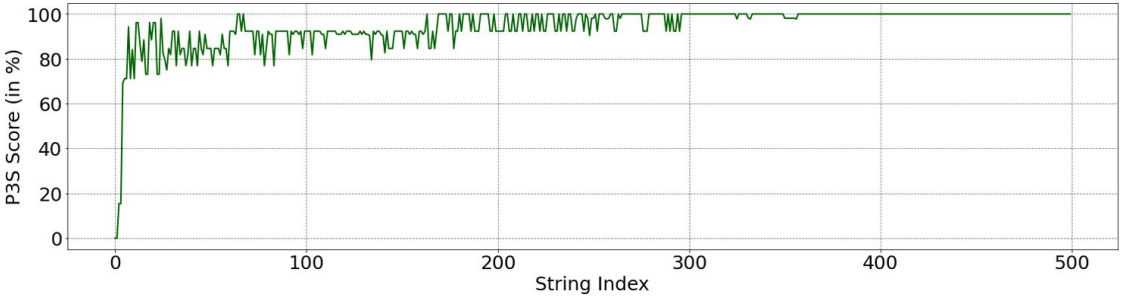

All connected synapses are fully grown i.e., permanence value of 1.

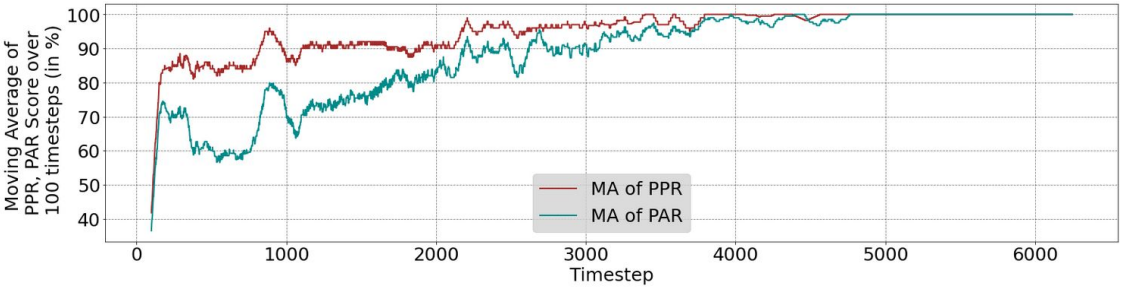
# Learning Long-Range Dependencies (LRDs)

## Input

- An inputstream of 500 strings consisting of equal repetitions of {<*A**S**PVSPXVPXSX**S**Z*>, <*AT**PVSPXVPXSX**T**Z*>, <*AV**PVSPXVPXSX**V**Z*>, <*APVSPXVPXSXZ*>}
- Learning *reset* at start of each string
  (*Z*→*A* transition is not learnt)

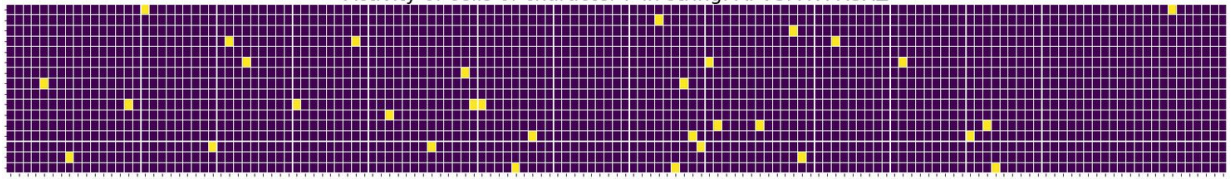
## Performance over 500 LRD strings



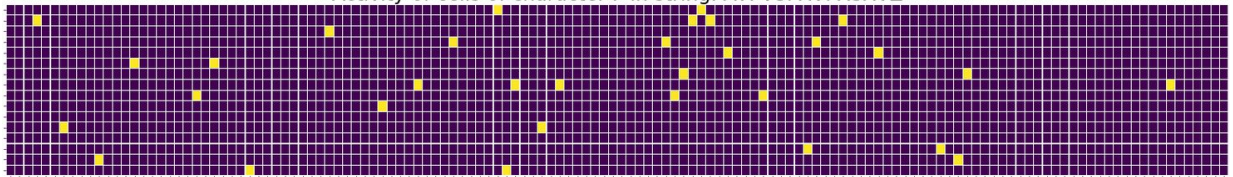
# Learning Long-Range Dependencies (LRDs)

Activity of cells of character P in string: APVSPXVPXSXZ



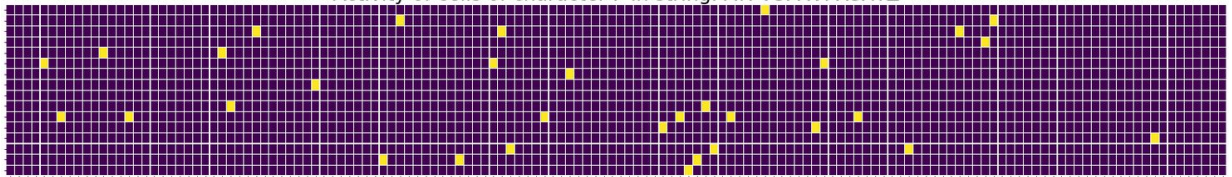
ZXSVPSTPAATZTXPSPTZSTAATPAPSPXXSTVPTXSZXSPXVTPVXAXPVXAPPPAZXPTPSAVSVSZVSZZXVAPZPPPPPPVAVSPVXAPPZASPSTTXVXZPSTATVVAPZPPTTZSXAVVSXZTVXSATTVVPTXVXVX

Activity of cells of character P in string: AVPVSPXVPXSXVZ



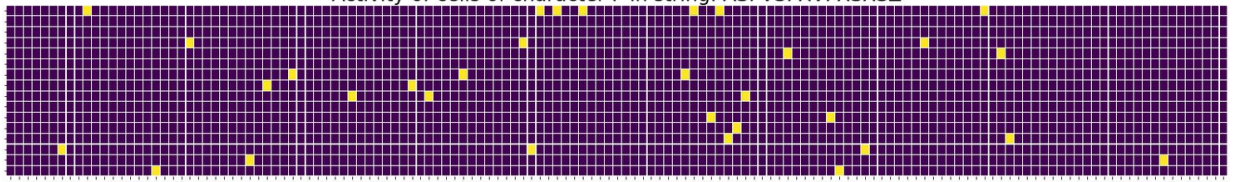
ZSVPSTPAATPSVZPTZTSZAPSPXSVPXXZSVVASPXTZXSPXVTPXVAPTTAAPPPZXPAPAVSVXSZZXAZPPPPPPSPASSPSSTAPPZXPTVZPXTATVAPAPPTTZASXAVVASZZTVZXAZTSSPTXZVVS

Activity of cells of character P in string: ATPVSPXVPXSXTZ



VSSVPSPAZTTPSVPXZTSTAAATAPPXSPXZVVVSPXZSTZXSPVPXTAVXAPTVTPPPAXZPTAPSAVSZZZXAZPZPPPPPTPASSPVZSTAPPZASPVXAXZPSTAVAPZAPPTTSVVSXZTSTXATTSVVPTZXSZVSX

Activity of cells of character P in string: ASPVSPXVPXSXSZ



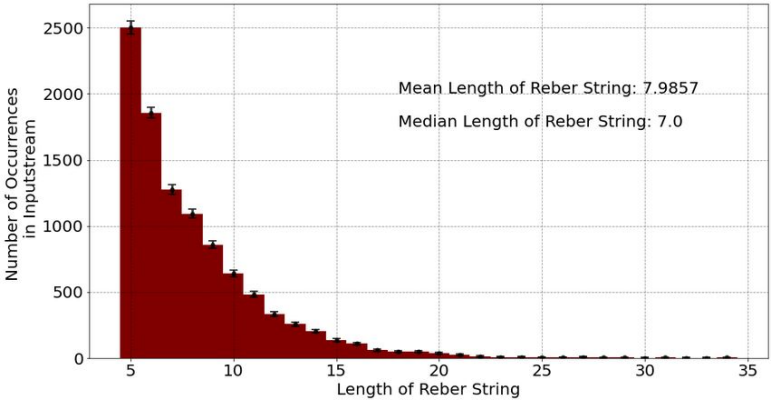
XVSVVZPSTPAATZTTTPSZZPXSTZAAPAPXVPXSVTZAPTZZXXSPXPXTAPTTVTXAPPPAPTAPVSVVZZVAXZAPPZPPPPPASVSPVXZTPPZXPXSTVXAPXSATVAPAPPZASAVVASZTXZTTSSVPTVZXZVS

# Learning Long-Range Dependencies (LRDs)

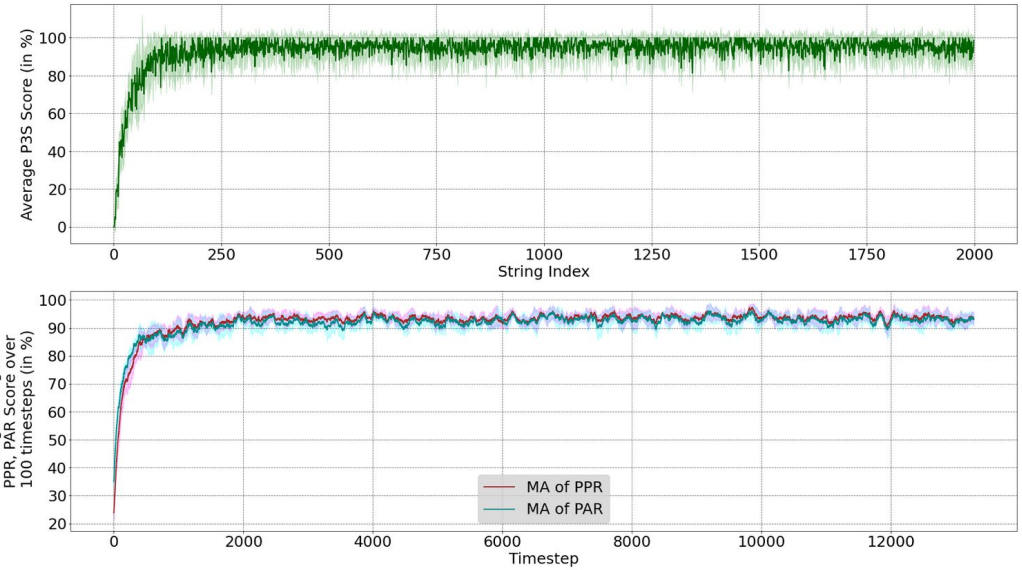
# Learning the SRG

## Input

- An inputstream of 2000 randomly generated Reber Strings (of any length) (10 trials)
- Learning *reset* at start of each string
  (*Z*→*A* transition is not learnt)




## Performance over 2000 Reber Strings (10 trials)



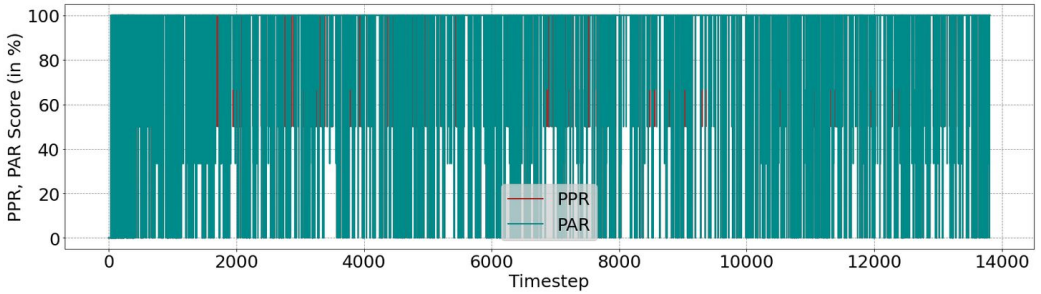

HTM fails to successfully learn the SRG.

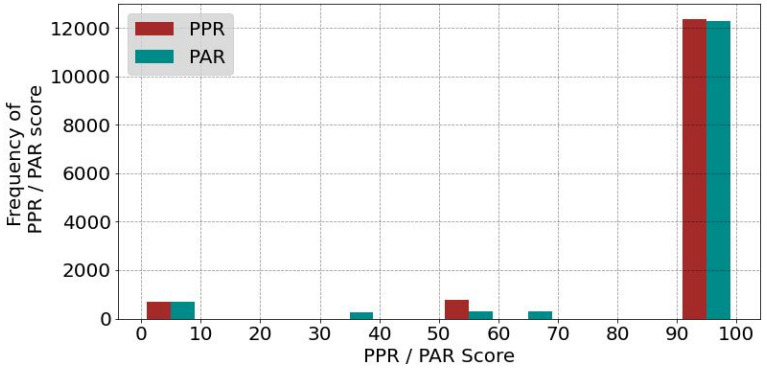
# Reasons for Failure to Learn SRG

- While most of the timesteps have a perfect score, there are 0% scores repeatedly
  - Due to new SDRs being continually learnt, which makes it impossible to have subsequent predictions

- PPR score also often drops to 50%
  - Implying there are imperfect predictions continually
  - Also due to continual learning of new SDRs

PPR & PAR over an Entire Inputstream



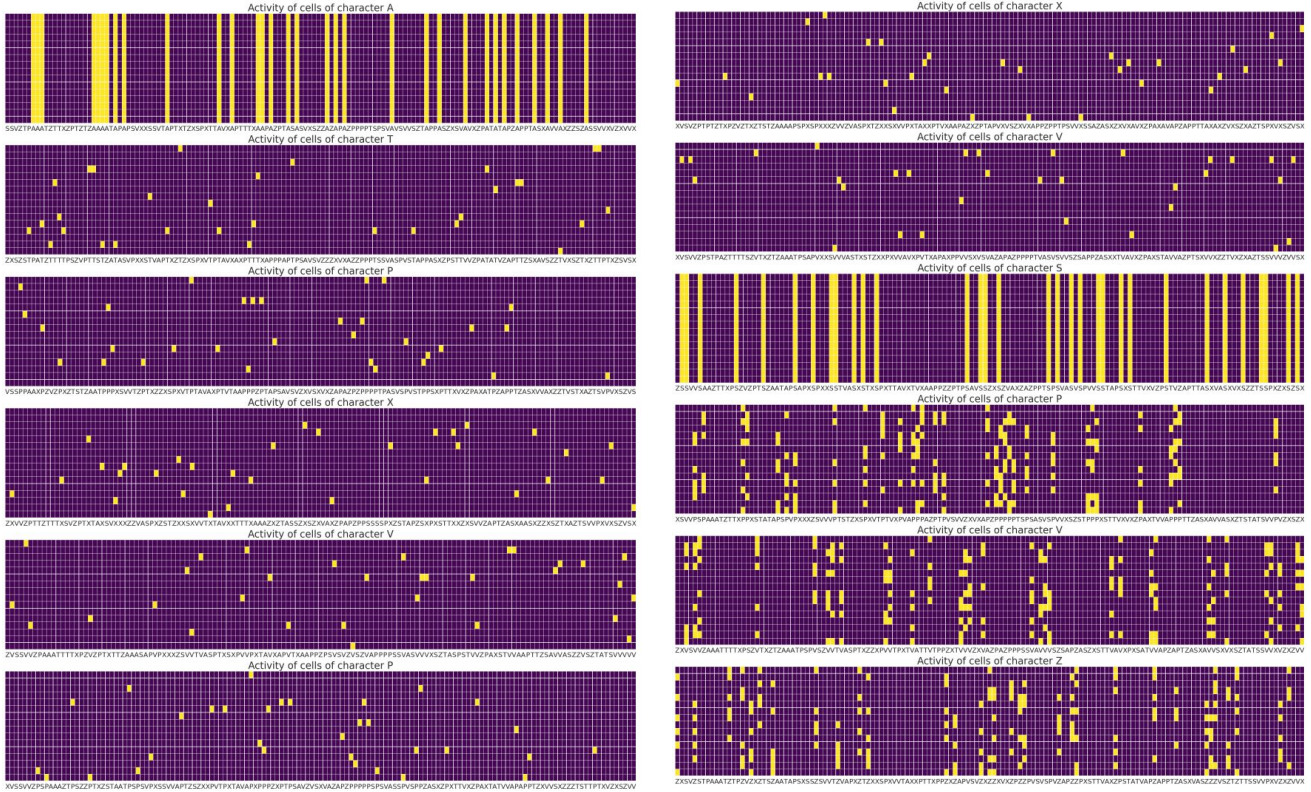
# Reasons for Failure to Learn SRG – Results for string *ATPXVPXVSPVZ*

Activations for *ATPXVPXVSPVZ*

Different SDRs for trigram *"PXV"* in the two repetitions

The SDR for *V* in the 8th step is newly learnt since it results in no prediction for *S*

After bursting in *S*, the network loses the context of the sequence prior to *S*

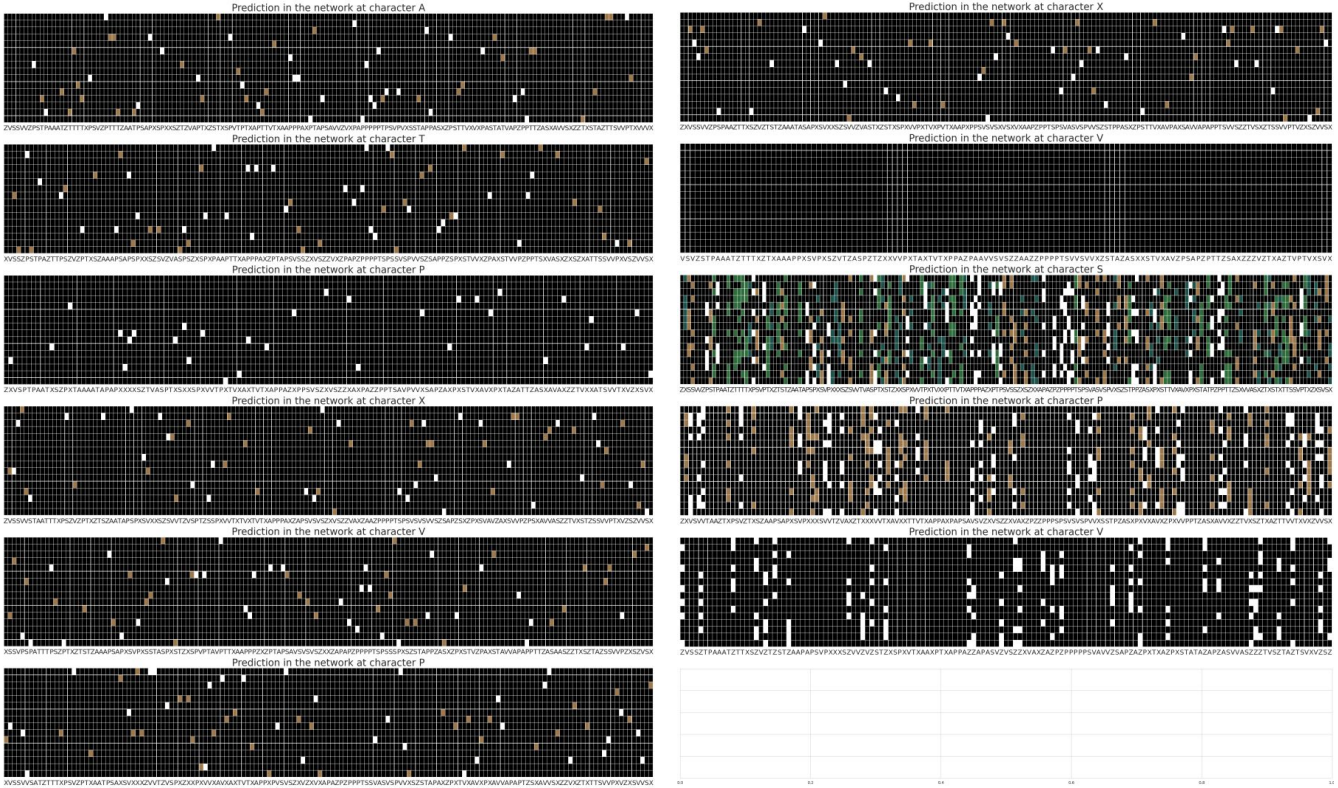


# Reasons for Failure to Learn SRG – Results for string *ATPXVPXVSPVZ*

Predictions for *ATPXVPXVSPVZ*

Prediction of *S* and *P* made at 5th step of *V*, but none at 8th step

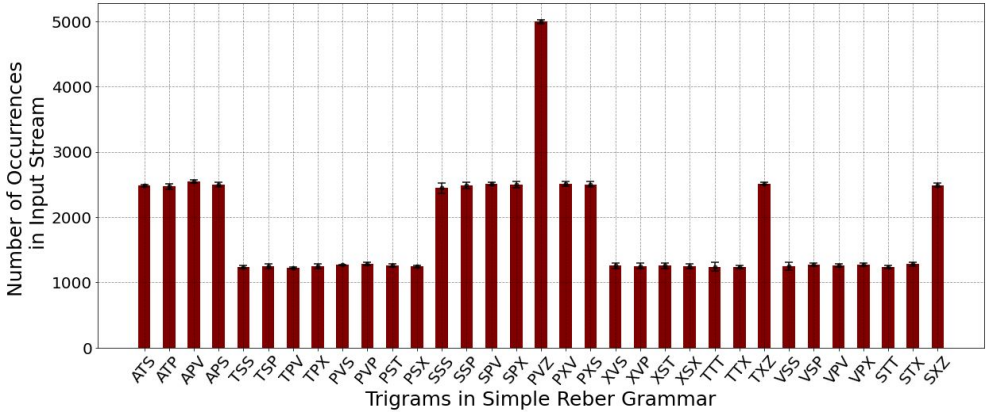
After bursting in *S*, the network predicts all the possible transitions – in all the possible distinct contexts – i.e., prediction of all the possible SDRs of *P*, *S*, *X* and *T*.



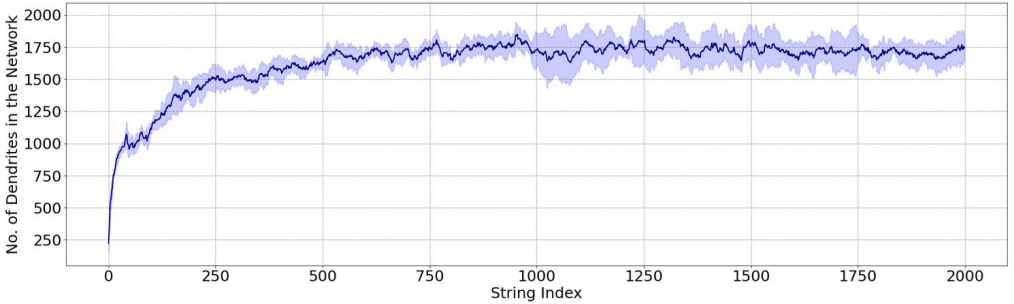

HTM automatically learns higher-order sequences and long-range dependencies with the use of SDRs. But it fails to recognize the “correct order” or “range” of long-range dependencies in a sequence of events.

# Reasons for Failure to Learn SRG – HTM Does NOT Learn Trigrams

- 31 trigrams in the SRG in total
  - Network needs to learn only 31(=33-2) transitions
  - Number of Dendrites required = 31*$k$ = 992.




- Network grows around 1750 dendrites in total.



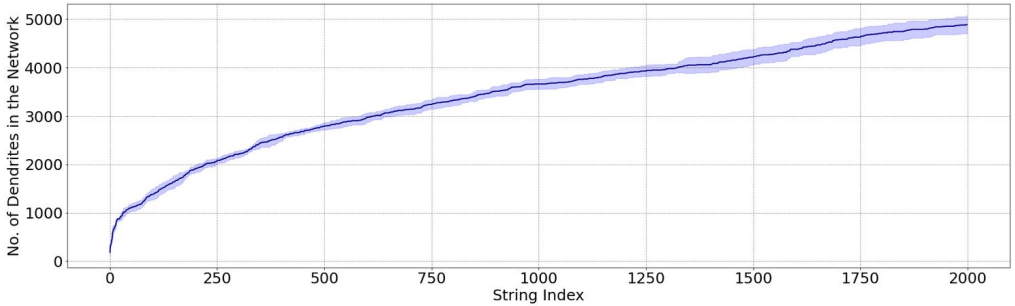

HTM does not learn the Simple Reber Grammar by acquiring its trigrams.

# Reasons for Failure to Learn SRG – HTM Attempts to Memorize Examples

***HYPOTHESIS:***

*If HTM actually learns SDRs in the context of the entire Reber String, then it can be hypothesized that it would keep growing the number of stored SDRs / transitions in the network as it gets exposed to the inputstream since there are potentially an infinite number of Reber strings possible.*

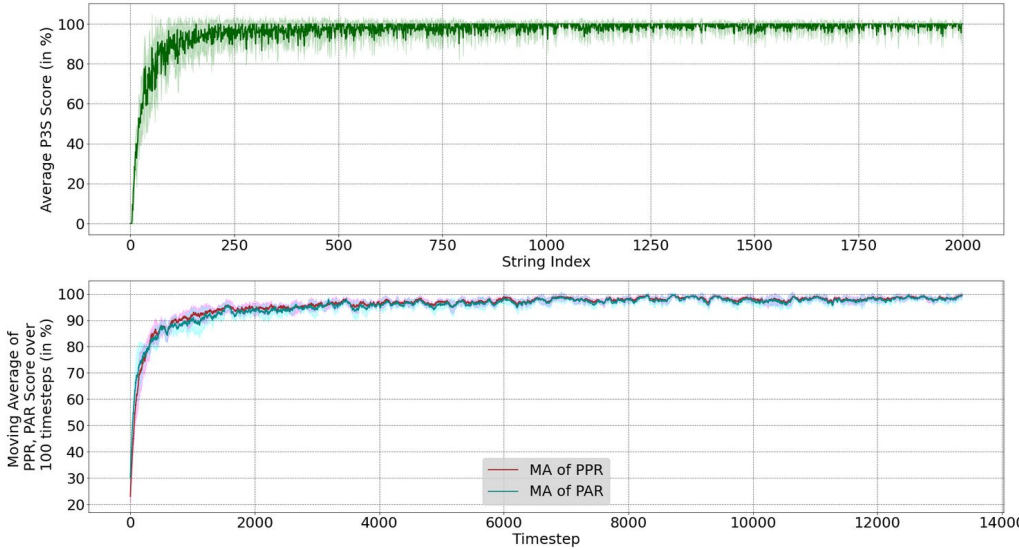
- The number of dendrites in the network keep growing until the end




HTM learns and stores every possible transition in the SRG in the temporal context of the entire Reber string, regardless of its context in a certain trigram.

# Reasons for Failure to Learn SRG – HTM Attempts to Memorize Examples

- Performance also improves gradually as more and more sequences are “memorized”




HTM attempts to memorize the example Reber strings from the inputstream, thereby learning the Reber Grammar in an approximate fashion only.

# Conclusion & Outlook

HTM uses Sparse Distributed Representations and neurons with spike-coincidence detection to implicitly learn sequences of transitions with long-range dependencies.

HTM stores temporal context using SDRs but it comes at the cost of having a variable order memory (use of hierarchical layers?).

HTM lacks a mechanism to learn certain long-range dependencies as single, constituent units in the sequence.

HTM aligns with the Synaptic Clustering Hypothesis of memory formation in the brain. (Kastellakis et al., 2015)

SDRs in HTM are robust & generalizable; and provide explanation for ubiquitous sparsity in the brain. (Yen et al., 2007; Cui et al., 2016)

HTM utilizes binary synaptic weights which are better-suited for long-term memory storage in the brain. (Petersen et al., 1998)

HTM's Superlinear Dendritic Integration Could Potentially Solve the Feature-Binding Problem. (Legenstein & Maass, 2011)

**Thank You for Listening!**

**Questions? Criticisms? Feedback?**