Short-Term Plasticity Neurons Learning to Learn and Forget

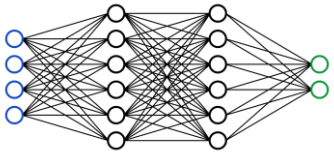
Hector Garcia Rodriguez

Qinghai Guo

Timoleon Moraitis

Motivation: Biological vs Artificial Neural Networks

Artificial NNs

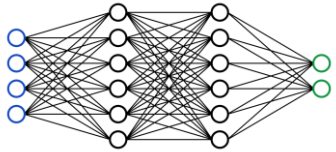


- Abstract inspiration from biology.

Biological NNs

Motivation: Biological vs Artificial Neural Networks

Artificial NNs

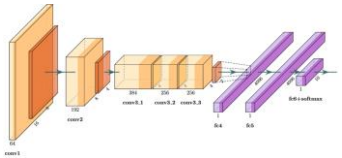


Biological NNs

- Abstract inspiration from biology.
- Limitations compared to animals and humans.
    1. Long training, big data

    2. Difficulty in dynamic environments

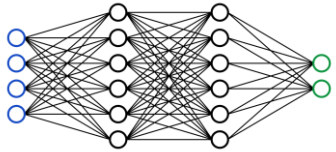
    3. Ad hoc, task-specific architectures



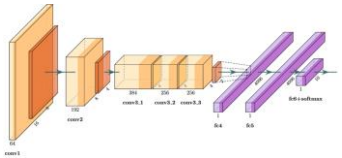
    4. High computational demands

Motivation: Biological vs Artificial Neural Networks

Artificial NNs

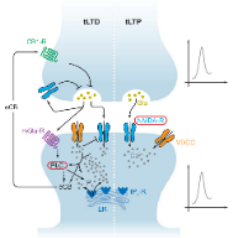


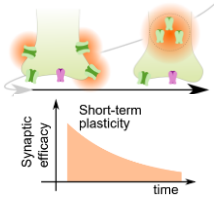
- Abstract inspiration from biology.
- Limitations compared to animals and humans.
  1. Long training, big data
  2. Difficulty in dynamic environments
  3. Ad hoc, task-specific architectures
  4. High computational demands

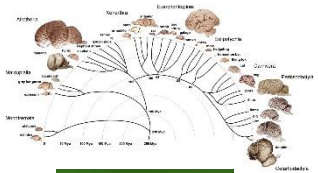


Biological NNs

- Unique mechanisms, missing from ANNs.



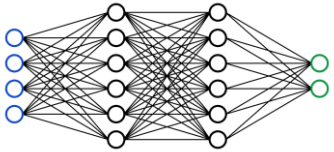




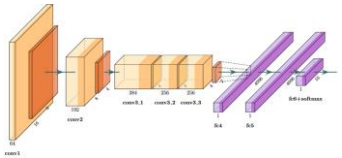
1. Complex synaptic plasticity for learning
2. Dynamically changing chemical concentrations
3. Shared principles across brain areas and species
4. Extreme energy efficiency in neurons and synapses

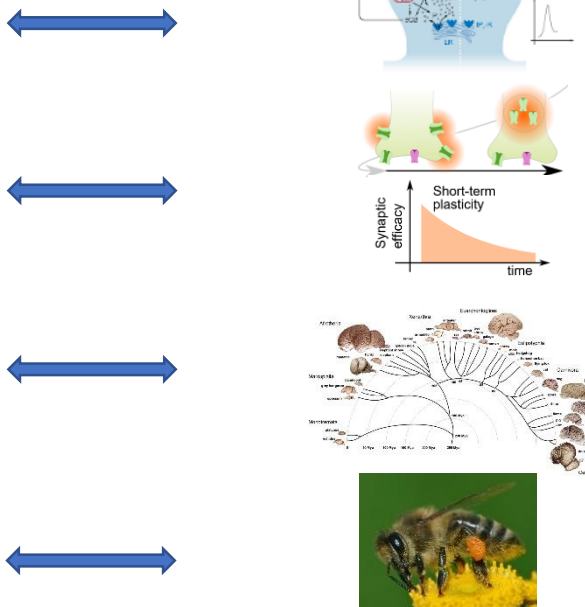
Motivation: Biological vs Artificial Neural Networks

Artificial NNs



- Abstract inspiration from biology.
- Limitations compared to animals and humans.
  1. Long **training**, big data
  2. Difficulty in **dynamic** environments
  3. Ad hoc, **task-specific** architectures
  4. High computational **demands**





Biological NNs

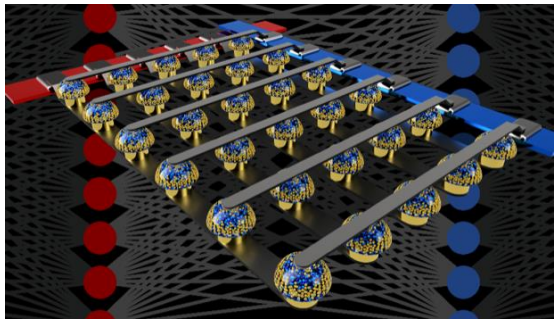
- Unique mechanisms, missing from ANNs.
- The mechanisms are closely related to the advantages.
  1. Complex synaptic **plasticity** for learning
  2. **Dynamically** changing chemical concentrations
  3. **Shared** principles across brain areas and species
  4. Extreme energy **efficiency** in neurons and synapses

Background: Neuromorphic Computing

Conversion of biophysical operations into (hardware) models of computation.

Background: Neuromorphic Computing

Conversion of biophysical operations into (hardware) models of computation.

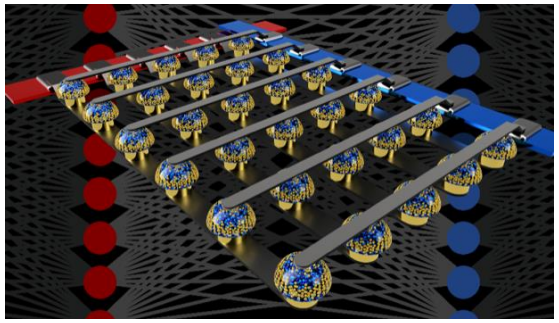


Ambrogio et al., 2018

Energy efficiency through physics of electronics.

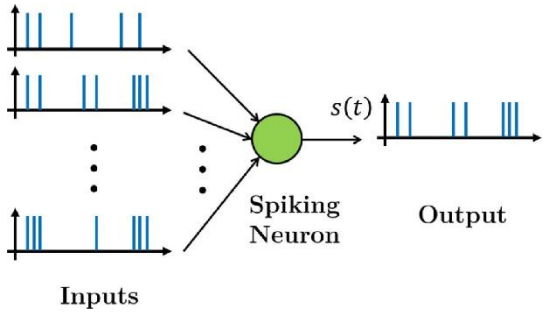
Background: Neuromorphic Computing

Conversion of biophysical operations into (hardware) models of computation.



Ambrogio et al., 2018

Energy efficiency through physics of electronics.

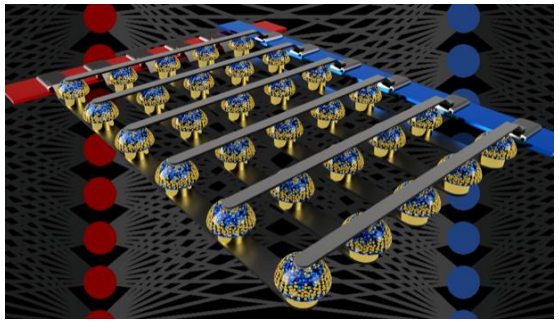


Anwani & Rajendran, 2019

The field has focused on Spiking Neural Networks.

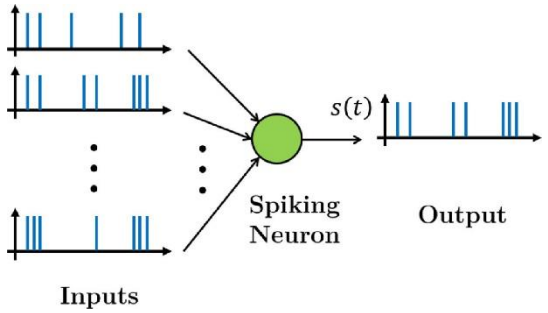
Background: Neuromorphic Computing

Conversion of biophysical operations into (hardware) models of computation.



Ambrogio et al., 2018

Energy efficiency through physics of electronics.



Anwani & Rajendran, 2019

The field has focused on Spiking Neural Networks.

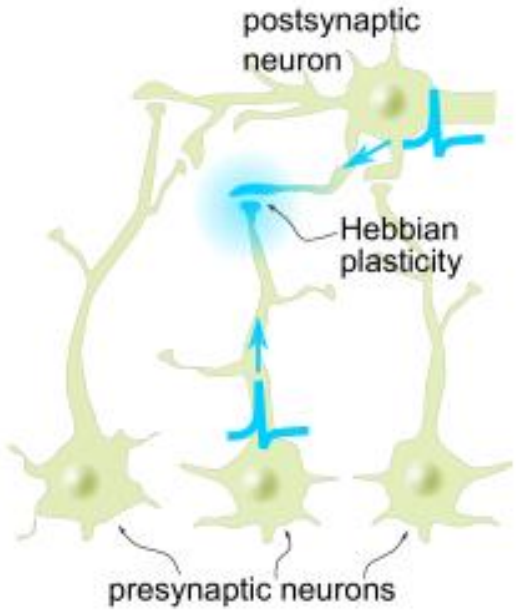
- We also used a neuromorphic mechanism (STP).

But:

- We improved **both *efficiency* and *proficiency***.
- We showed advantages **beyond SNNs, in state-of-the-art** ML.

Background: Plasticity

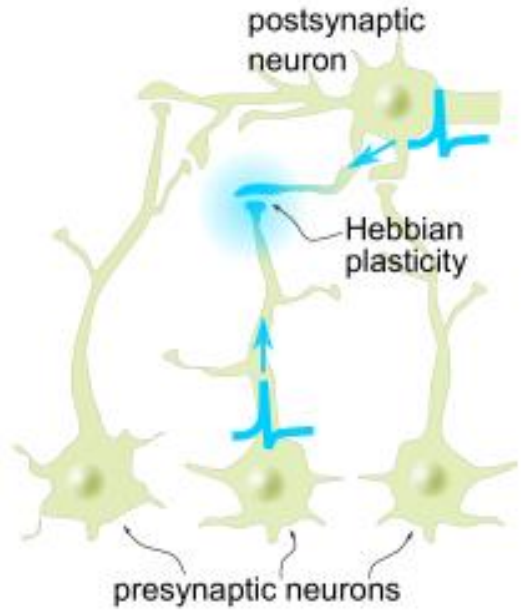
- E.g. Hebbian plasticity, “Fast Weights”
- Local learning in the brain



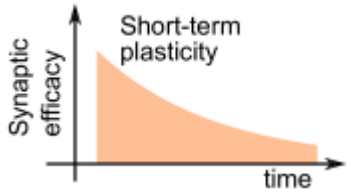
Moraitis et al., 2020

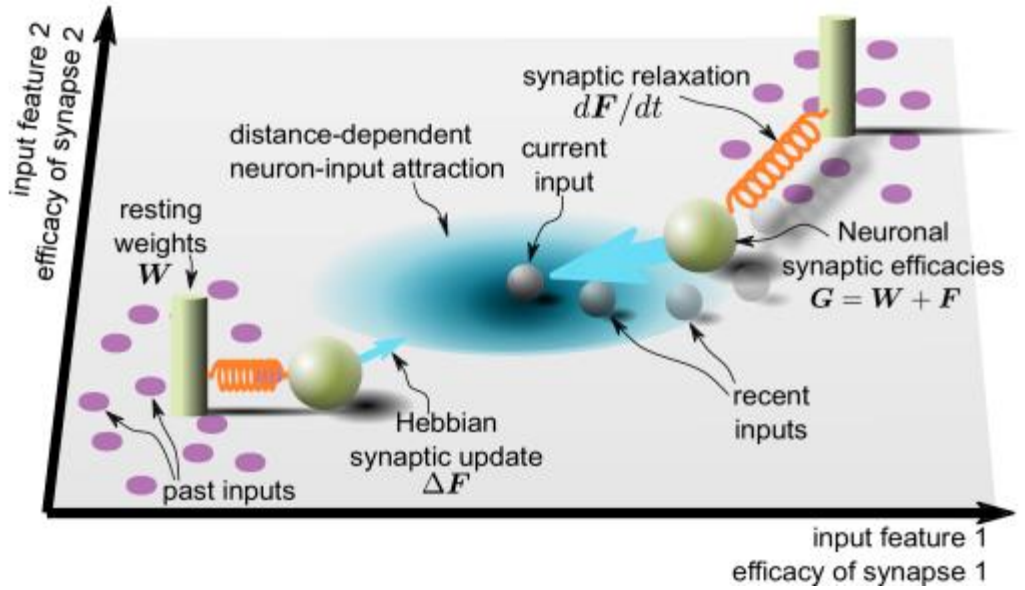
Background: Plasticity and Short-Term Plasticity

- E.g. Hebbian plasticity, “Fast Weights”
- Local learning in the brain



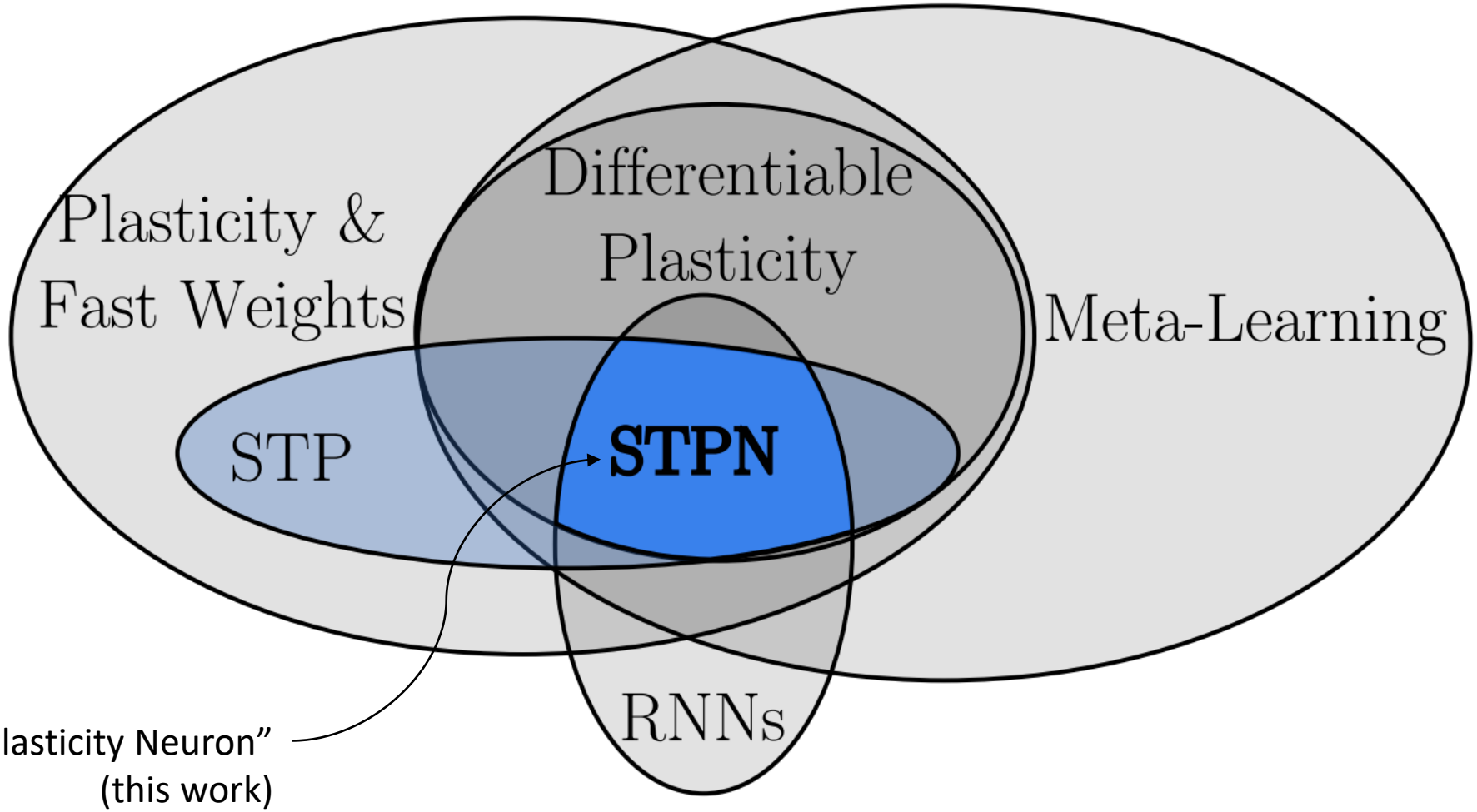
- Short-term plasticity (STP)
- Decaying effect of weight updates





➢ Hebbian STP: models certain dynamic environments optimally, in theory (Moraitis et al. 2020).

Background concepts



The Short-Term Plasticity Neuron (STPN)

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

- Split synaptic efficacy G into long-term weight W and short-term component F

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$$

The Short-Term Plasticity Neuron (STPN)

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

- Split synaptic efficacy G into long-term weight W and short-term component F

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$$

- Update short-term component with Hebbian product and decay dynamics

$$\boldsymbol{F}^{(t+1)} = \underbrace{\boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)})}_{\text{Hebbian update}}$$

learning

The Short-Term Plasticity Neuron (STPN)

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

- Split synaptic efficacy G into long-term weight W and short-term component F

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$$

- Update short-term component with Hebbian product and decay dynamics

$$\boldsymbol{F}^{(t+1)} = \underbrace{\boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)})}_{\text{Hebbian update}} + \underbrace{\boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}}_{\text{Decay dynamics}}$$

learning forgetting

The Short-Term Plasticity Neuron (STPN)

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

- Split synaptic efficacy G into long-term weight W and short-term component F

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$$

- Update short-term component with Hebbian product and decay dynamics

$$\boldsymbol{F}^{(t+1)} = \underbrace{\boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)})}_{\text{Hebbian update}} + \underbrace{\boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}}_{\text{Decay dynamics}}$$

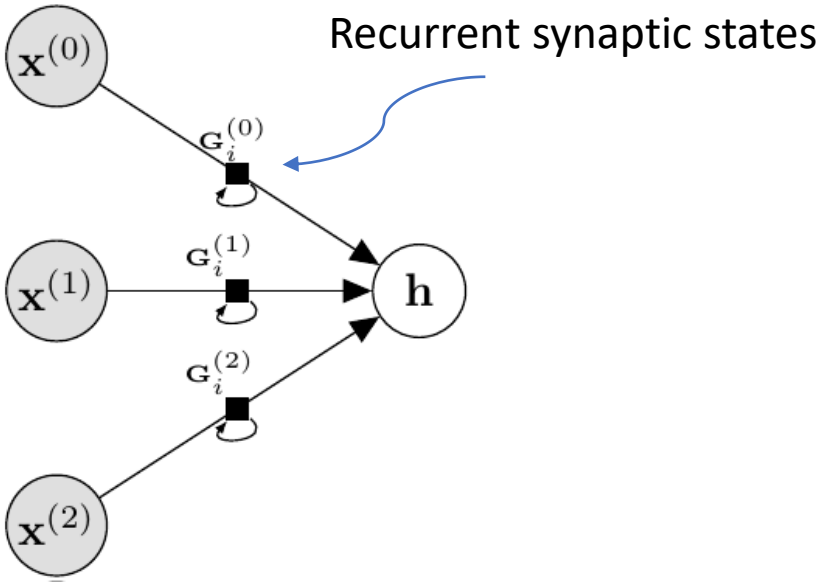
learning

forgetting

Recurrent synaptic state

The Short-Term Plasticity Neuron (STPN)

STPN-F (feed-forward)

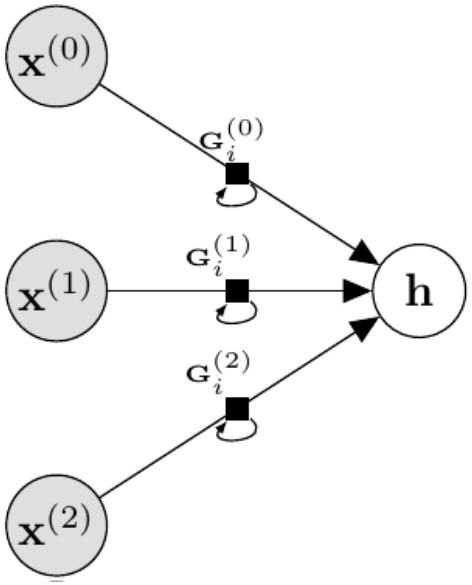


$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

$$\boldsymbol{F}^{(t+1)} = \boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)}) + \boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}$$

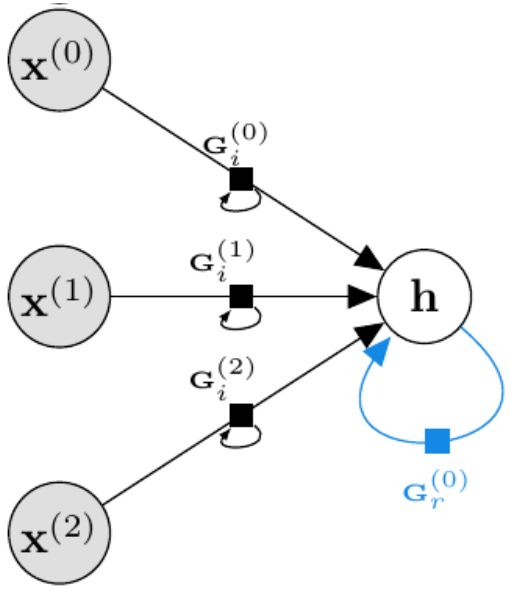
The Short-Term Plasticity Neuron (STPN)

STPN-F (feed-forward)



$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$
$$\boldsymbol{F}^{(t+1)} = \boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)}) + \boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}$$

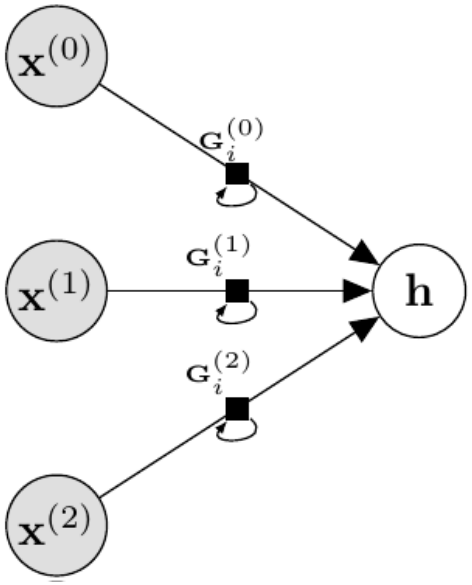
STPN-R (recurrent)



$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}[\boldsymbol{x}^{(t)}; \boldsymbol{h}^{(t-1)}])$$
$$\boldsymbol{F}^{(t+1)} = \boldsymbol{\gamma} \odot ([\boldsymbol{x}^{(t)}; \boldsymbol{h}^{(t-1)}] \otimes \boldsymbol{h}^{(t)}) + \boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}$$

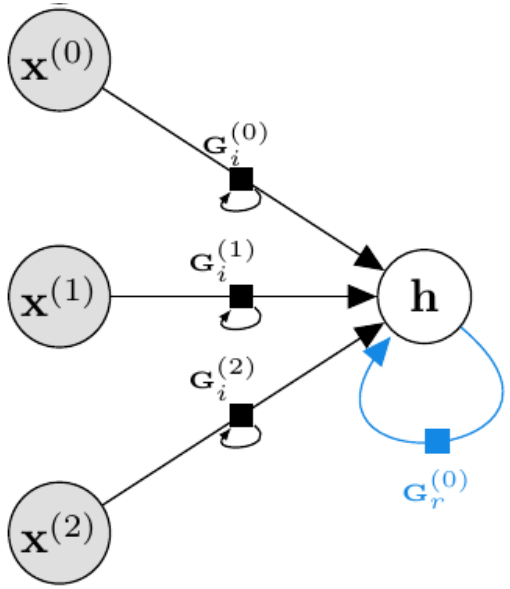
The Short-Term Plasticity Neuron (STPN)

STPN-F (feed-forward)



$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$
$$\boldsymbol{F}^{(t+1)} = \boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)}) + \boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}$$

STPN-R (recurrent)



$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}[\boldsymbol{x}^{(t)}; \boldsymbol{h}^{(t-1)}])$$
$$\boldsymbol{F}^{(t+1)} = \boldsymbol{\gamma} \odot ([\boldsymbol{x}^{(t)}; \boldsymbol{h}^{(t-1)}] \otimes \boldsymbol{h}^{(t)}) + \boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}$$

…STPN-L? (LSTM-like)

The Short-Term Plasticity Neuron (STPN)

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$$

STP on F (and G):

$$\boldsymbol{F}^{(t+1)} = \underbrace{\boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)})}_{\text{Hebbian update}} + \underbrace{\boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}}_{\text{Decay dynamics}}$$

learning

forgetting

The Short-Term Plasticity Neuron (STPN)

$$\boldsymbol{h}^{(t)} = \sigma(\boldsymbol{G}^{(t)}\boldsymbol{x}^{(t)})$$

$$\boldsymbol{G}^{(t)} = \boldsymbol{W} + \boldsymbol{F}^{(t)}$$

STP on F (and G):

$$\boldsymbol{F}^{(t+1)} = \underbrace{\boldsymbol{\gamma} \odot (\boldsymbol{x}^{(t)} \otimes \boldsymbol{h}^{(t)})}_{\text{Hebbian update}} + \underbrace{\boldsymbol{\lambda} \odot \boldsymbol{F}^{(t)}}_{\text{Decay dynamics}}$$

learning forgetting

BPTT on γ and λ (and W):

learning to learn and forget

Experiments: Tasks

- Supervised Learning
- Reinforcement Learning
- 4 tasks

Experiments: Tasks

Associative Retrieval Task
Ba et al., 2016

input

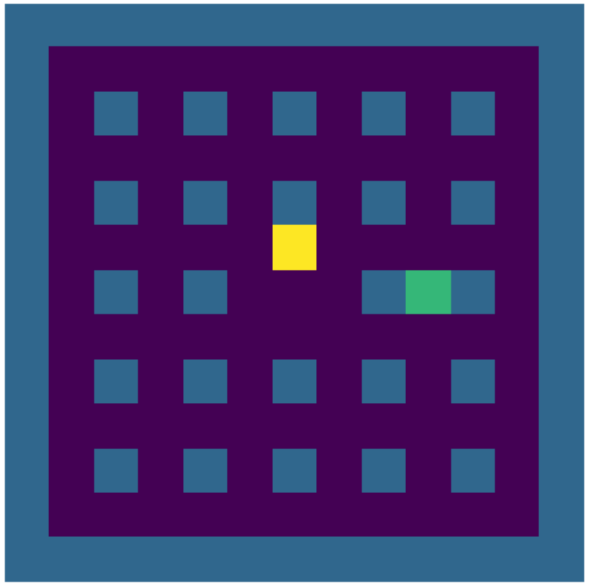
c9 k8 j3 f1 ?? k

target

8

- Memorisation and retrieval of associations presented sequentially

Maze exploration
Miconi et al., 2018

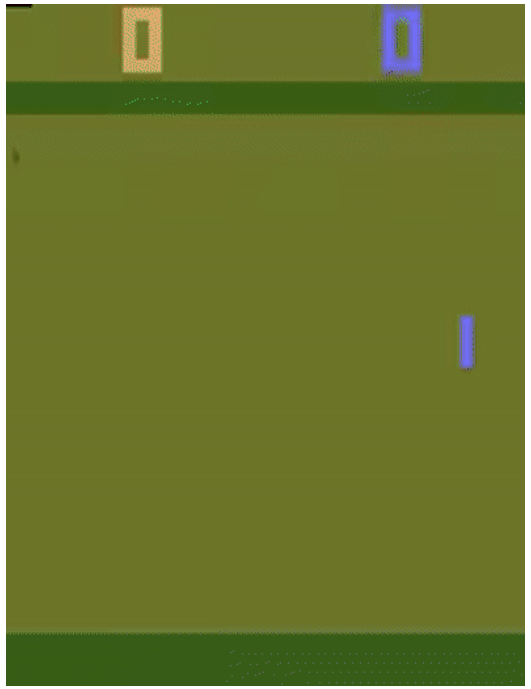


- Meta-learning to:
  - Adapt to variations in the maze
  - Change between exploration and exploitation

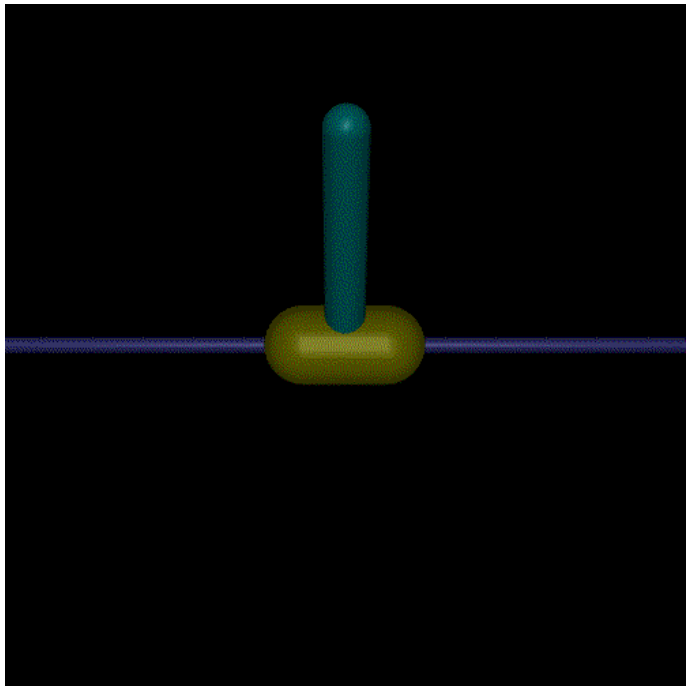
- Previously used to test models with *fast weights*

Experiments: Tasks

Atari Pong
Bellemare et al., 2013



MuJoCo Inverted Pendulum
Todorov et al., 2012

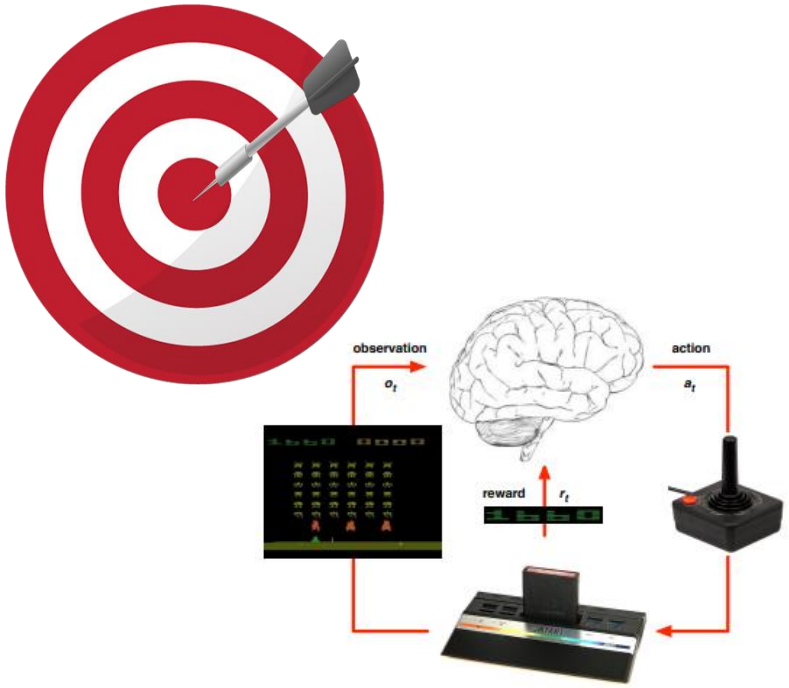


- More complex tasks, previously not tested for models with *fast weights*
- RNNs part of competitive solutions

Experiments: Metrics

We evaluated:

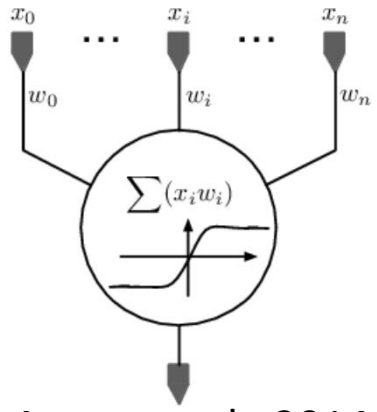
- Proficiency (Accuracy, reward)



- Efficiency (1/energy)
  - Power consumption of weighting operations
  - Assuming analog biological implementation/neuromorphic hardware

$$P = V^2 g$$

Electric power consumption — Input (voltage) — Synaptic efficacy (conductance)



Amant et al., 2014

Experiments: Baselines

We compared STPN with:

- *Fast weights* SOTA models
  - Fast weights RNN (Ba et al., 2016)
  - Differentiable plasticity (Miconi et al., 2019)
- Commonly used NNs
  - LSTM
  - RNN
  - MLP

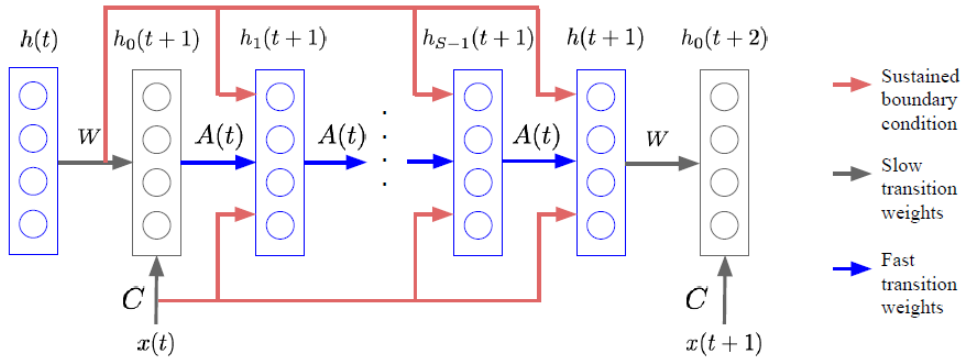
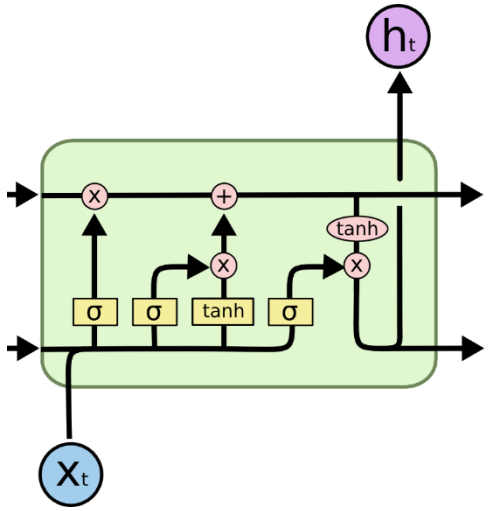


Figure 1: The fast associative memory model.

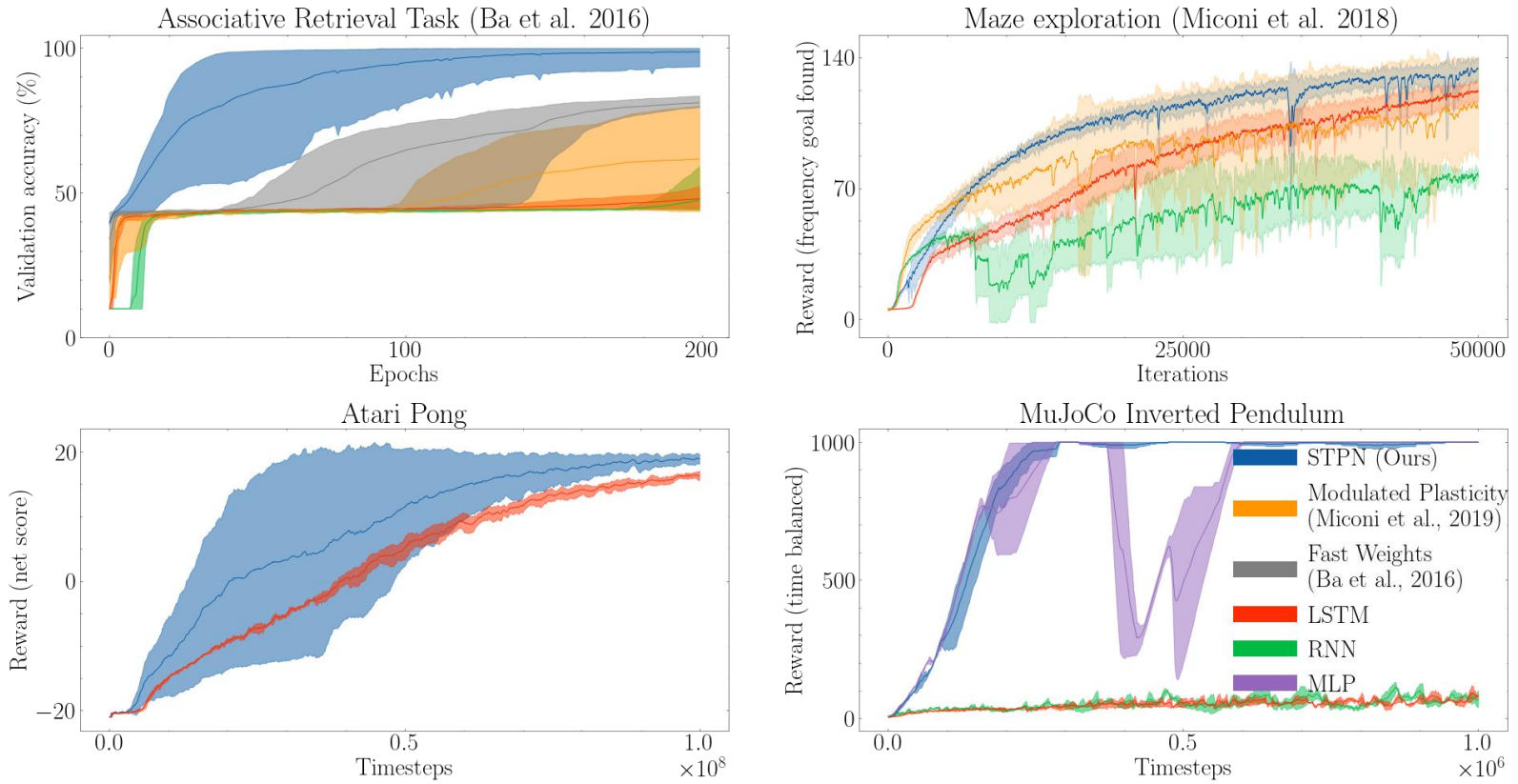
Ba et al., 2016



LSTM Illustration by Olah C., 2015

Results: Proficiency

- STPN obtained highest accuracy and reward
  - Than all baselines
  - In all tasks



Results: Efficiency

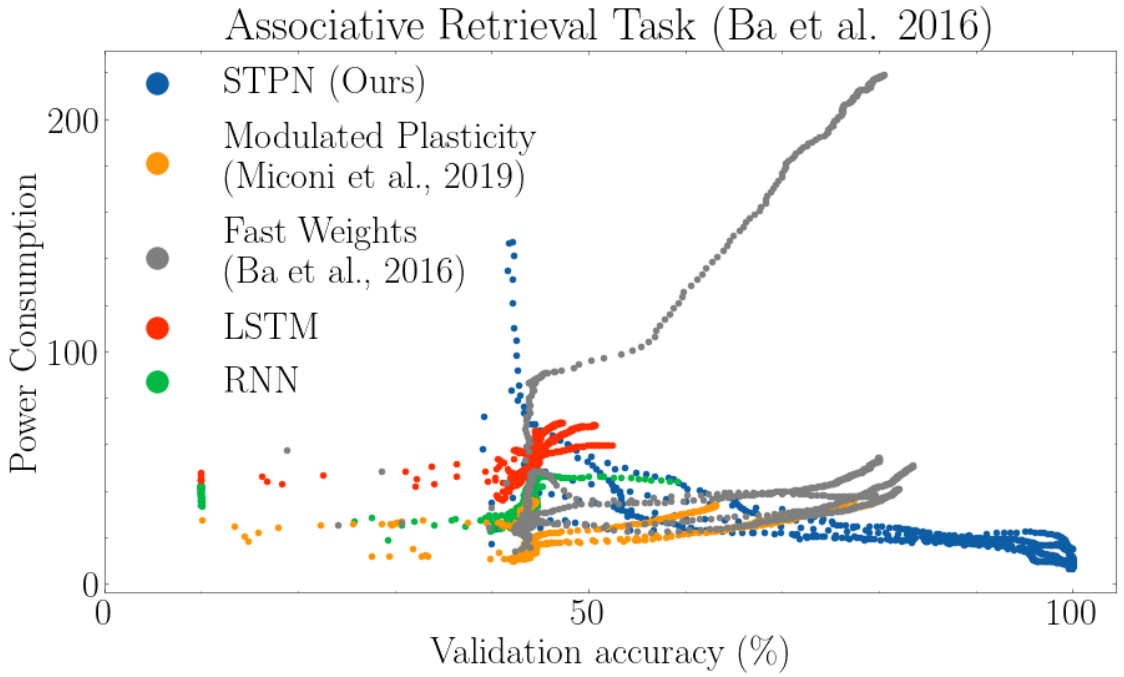
- STPN is also significantly more energy-efficient, up to 10x
  - Than all baselines
  - In all tasks

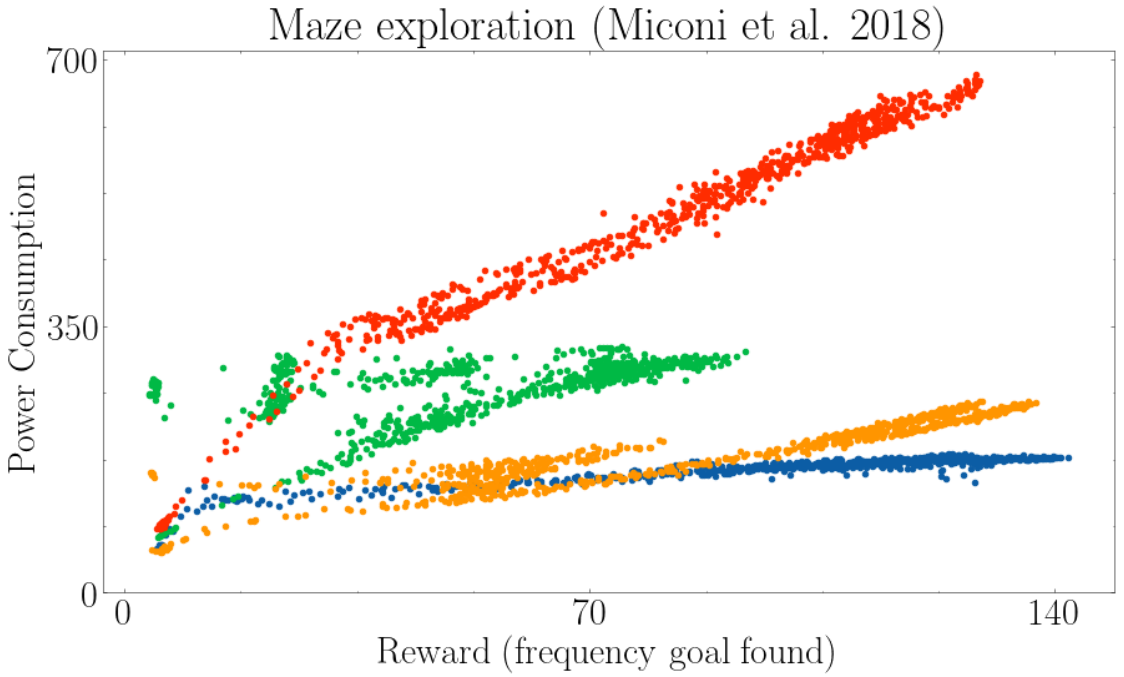
Table 4. Power consumption of trained models during inference. Average over multiple inference runs and timesteps within a run, reported mean and standard deviation over seeds.

Model	Associative Retrieval	Maze Exploration	Pong	Inverted Pendulum
STPN (Ours)	**10.9 ± 3.8**	**181.1 ± 2.9**	**52.7 ± 6.4**	**7.9 ± 2.5**
LSTM	65.6 ± 3.4	649.1 ± 15.3	576.8 ± 33.5	758.8 ± 81.7
RNN	43.0 ± 4.9	330.6 ± 24.3	-	143.0 ± 22.4
Modulated plasticity	32.0 ± 7.4	231.6 ± 23.9	-	-
Fast Weights	80.6 ± 69.4	-	-	-
MLP	-	-	-	115.8 ± 33.9

Results: Efficiency

- Power consumption remains low even as proficiency increases.
  - Despite no explicit optimization of efficiency.
  - Learning to depress irrelevant synapses improves both.





Summary

- STPN: a **new recurrent model** founded on ML theory and neuroscience
- Trainable STP through **recurrency within each synapse**
- **Meta-learning** to learn ***and forget*** through BPTT + STP
- **Highest accuracy** and reward
  - In **multiple tasks**
  - Than **SOTA** recurrent and plastic models
- A new method for **energy efficiency in neuromorphic** hardware
- Increases importance of **STP's role in biology**
- Learning to forget might tackle ***catastrophic* forgetting**