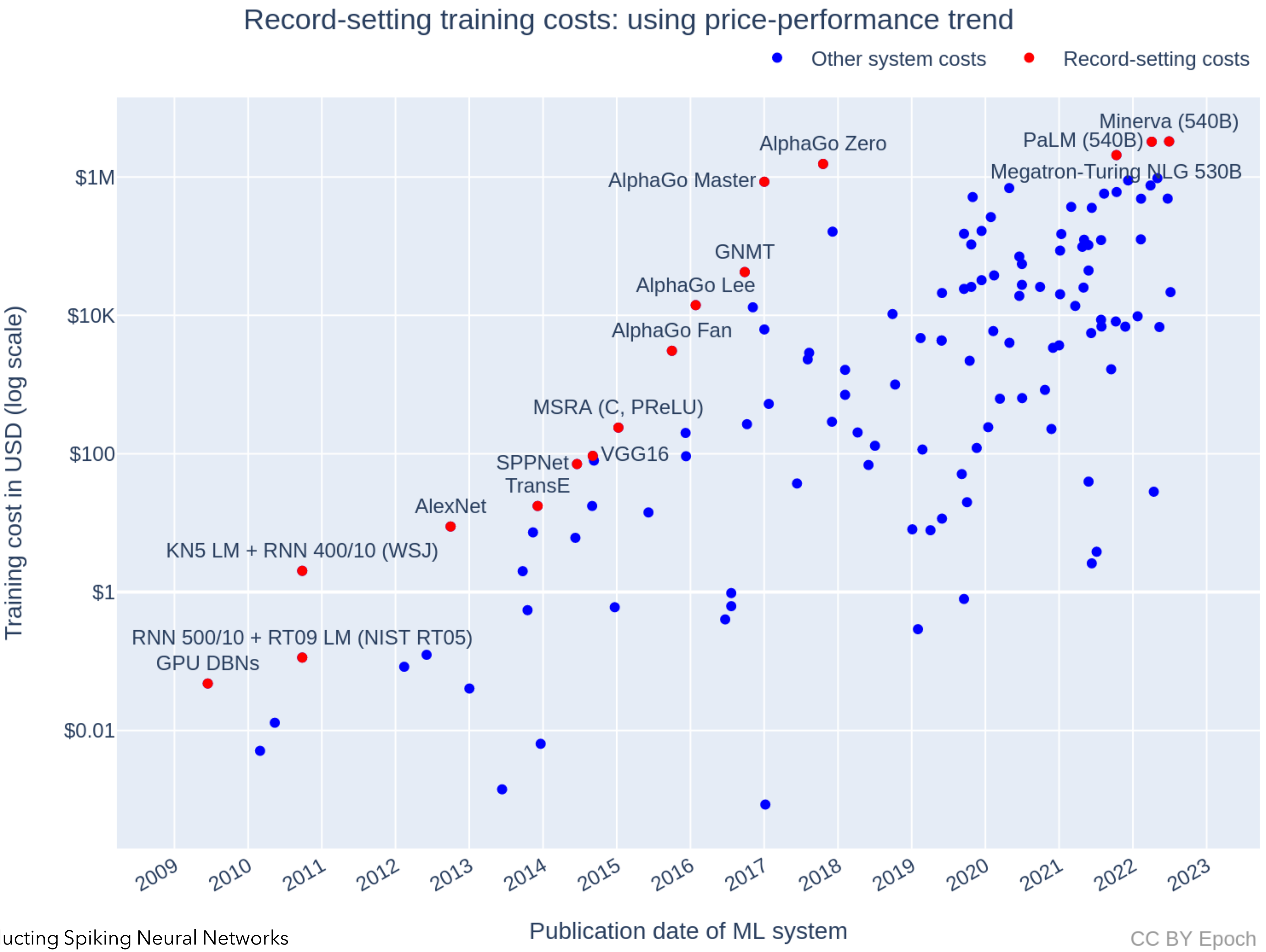
Colgate University, Physics and Astronomy, 2023
Daniel A. Espinosa, Prof. Kenneth (Ken) Segall

Time-encoded Superconducting Spiking Neural Networks

Large artificial neural networks are expensive in data, energy, money & time

(1)

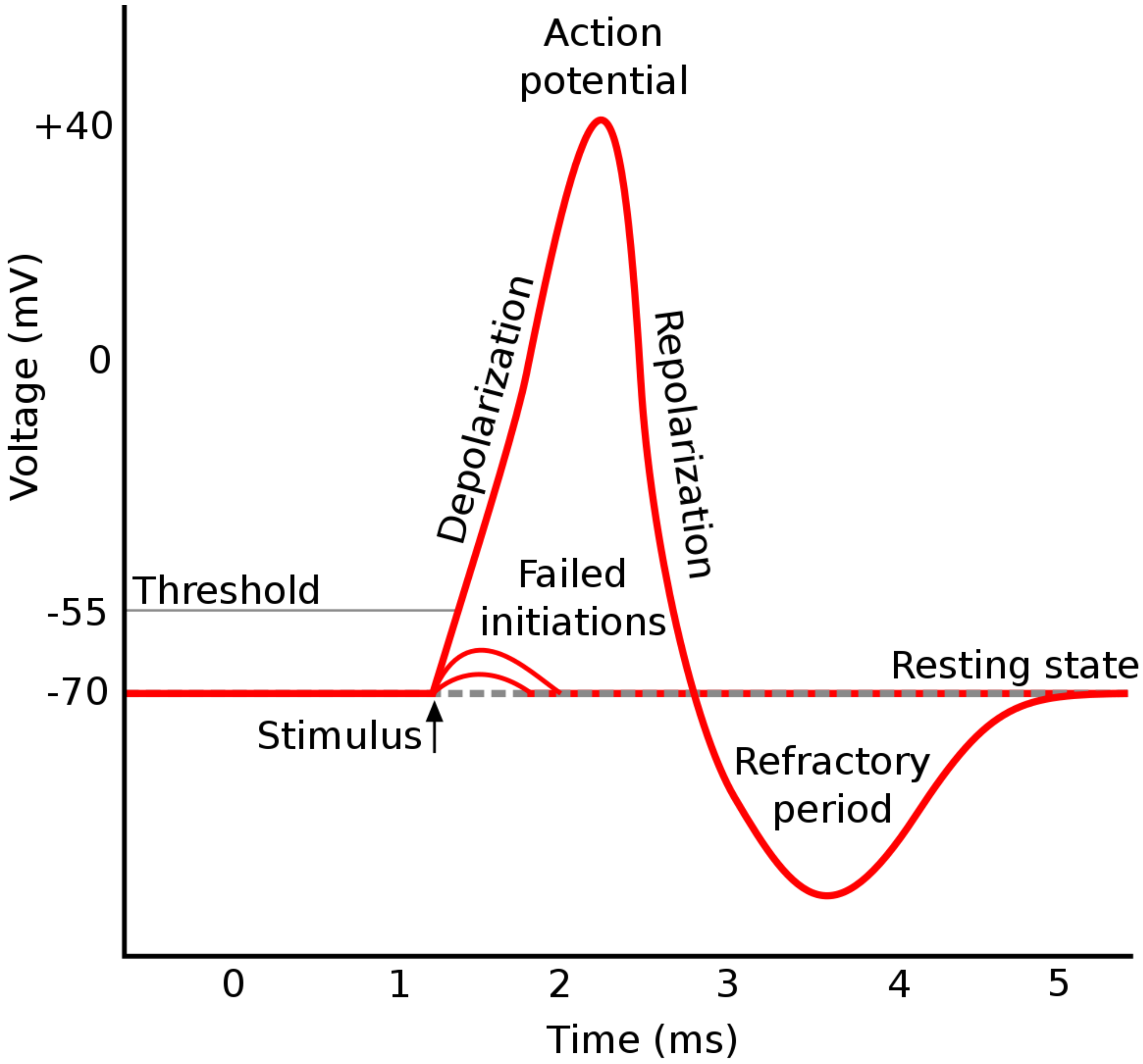

Record-setting training costs: using price-performance trend
Other system costs
Record-setting costs
Training cost in USD (log scale)
$1M
$10K
$100
$1
$0.01
Minerva (540B)
PaLM (540B)
AlphaGo Zero
Megatron-Turing NLG 530B
AlphaGo Master
GNMT
AlphaGo Lee
AlphaGo Fan
MSRA (C, PReLU)
SPPNet
VGG16
TransE
AlexNet
KN5 LM + RNN 400/10 (WSJ)
RNN 500/10 + RT09 LM (NIST RT05)
GPU DBNs
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
Publication date of ML system
CC BY Epoch

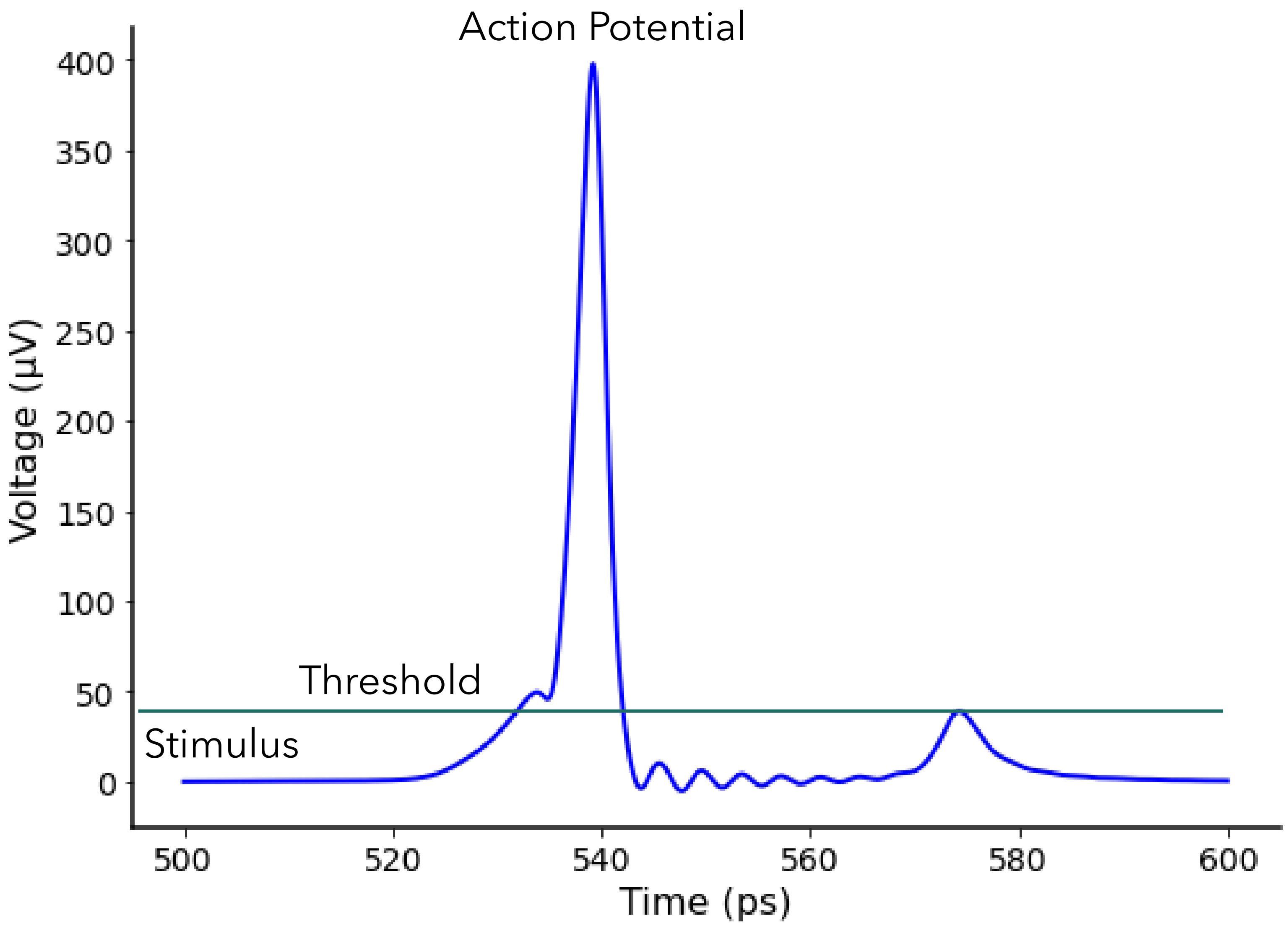
Why use superconducting hardware?

Neuromorphic

Fast

Energy efficient


Action potential
+40
0
-55
-70
Voltage (mV)
Depolarization
Repolarization
Threshold
Failed initiations
Resting state
Stimulus
Refractory period
0
1
2
3
4
5
Time (ms)


Action Potential
Threshold
Stimulus
400
350
300
250
200
150
100
50
0
Voltage (µV)
500
520
540
560
580
600
Time (ps)

Superconducting spikes use about 10^{-18} Joules each.

Human brain? About 10^{-11} Joules each.

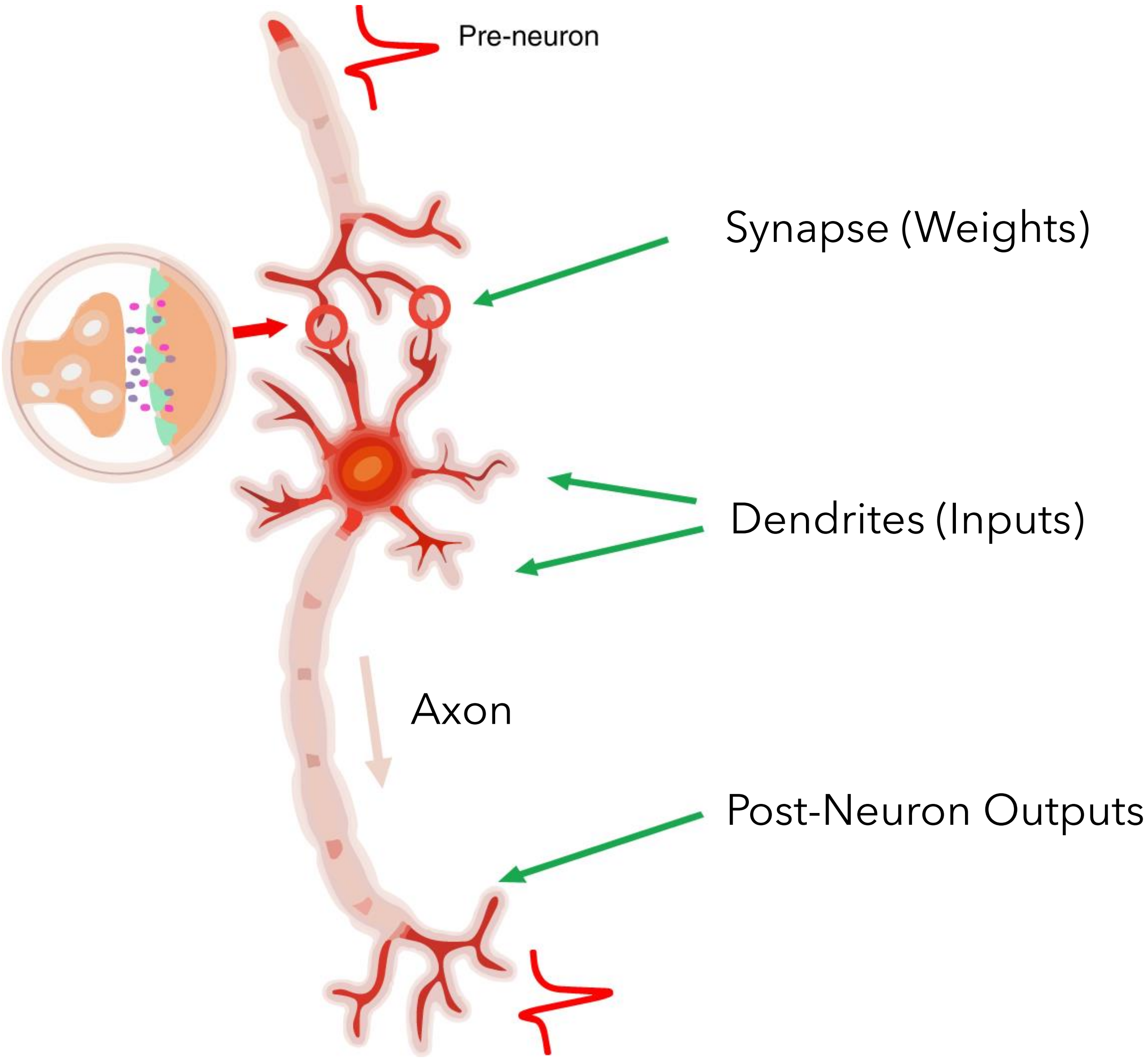
On average a 6 to 7 order of magnitude improvement. (!!)

(2-3)

Neural network basics

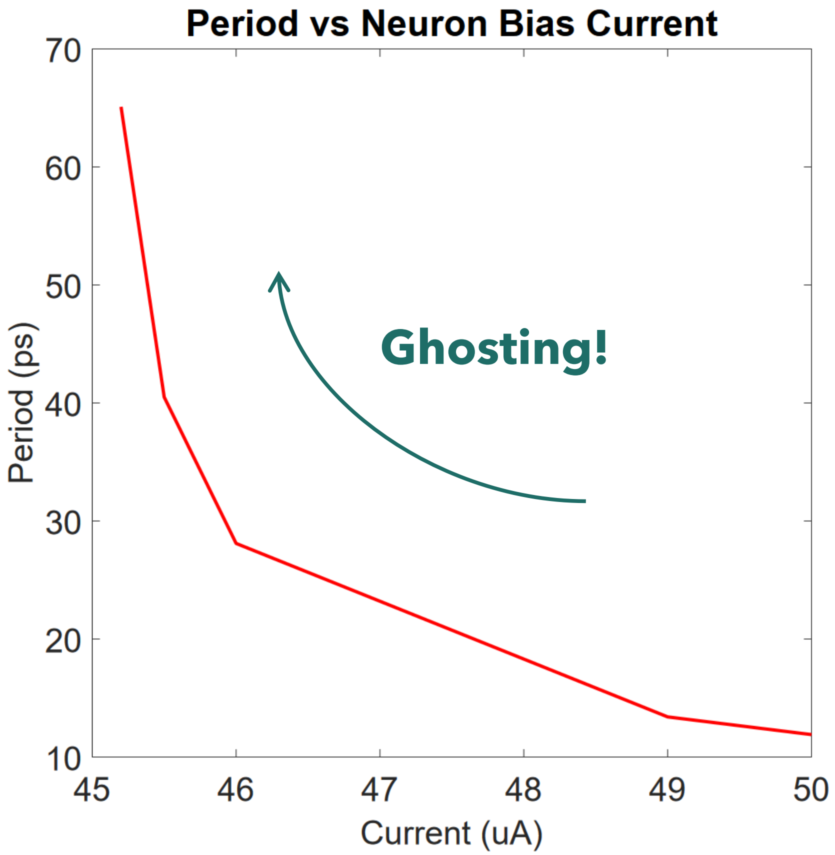
As universal approximators, artificial neural networks are remarkably versatile at tackling diverse tasks, such as image recognition and natural language processing.

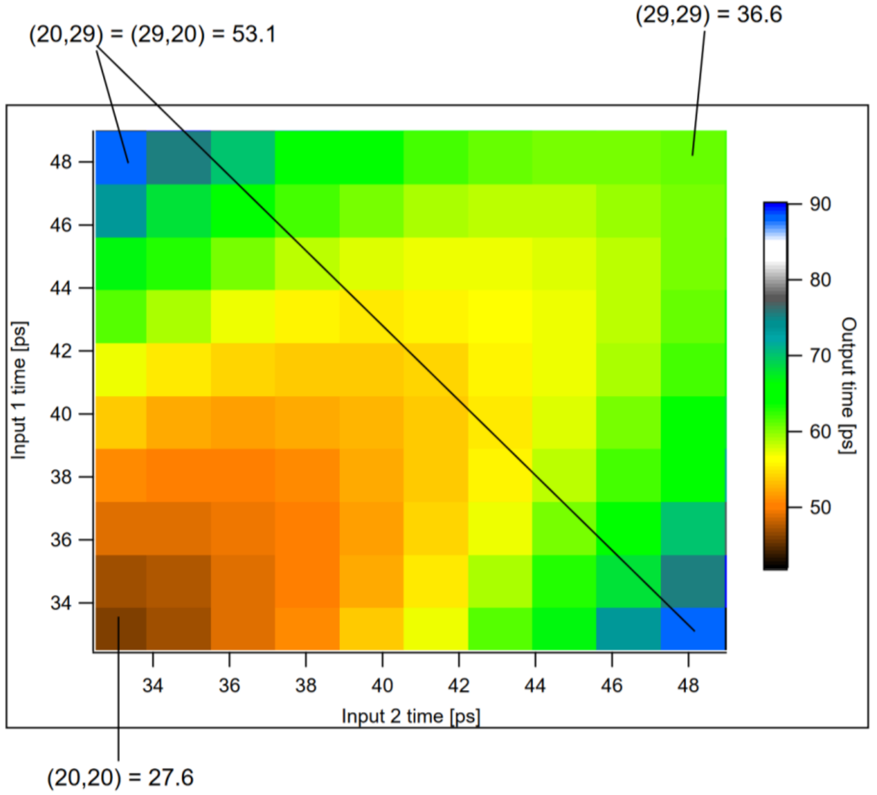
(2)



Picture Credit: https://www.nature.com/articles/s41467-020-17215-3

Missing piece: Time Encoding





So, you can make a network… how can we tell if it is a useful one?

The XOR Separability Problem!

Logical Truth Table

A	B	OUT
0	0	0
0	1	1
1	0	1
1	1	0

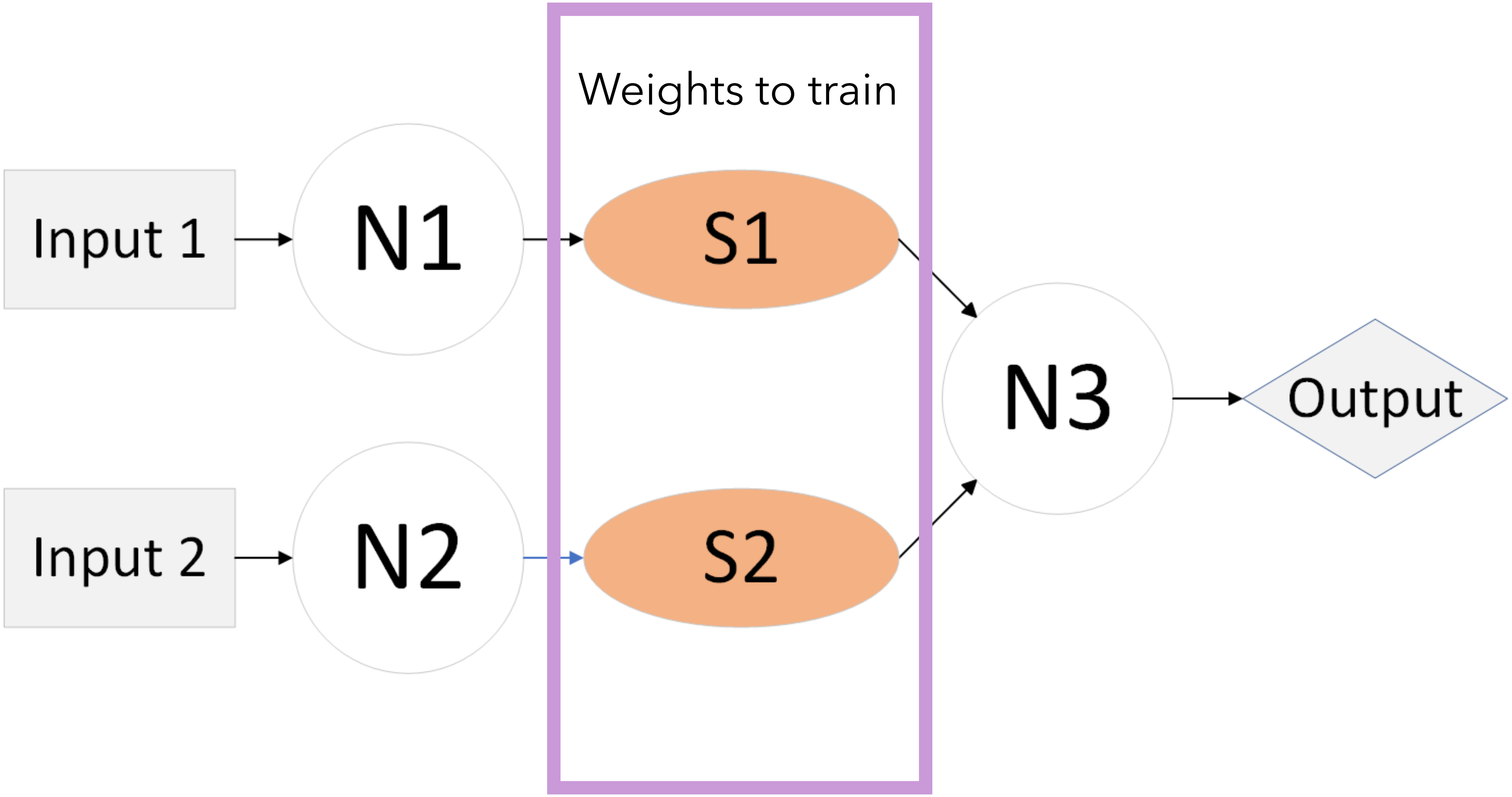
Time Encoded Table

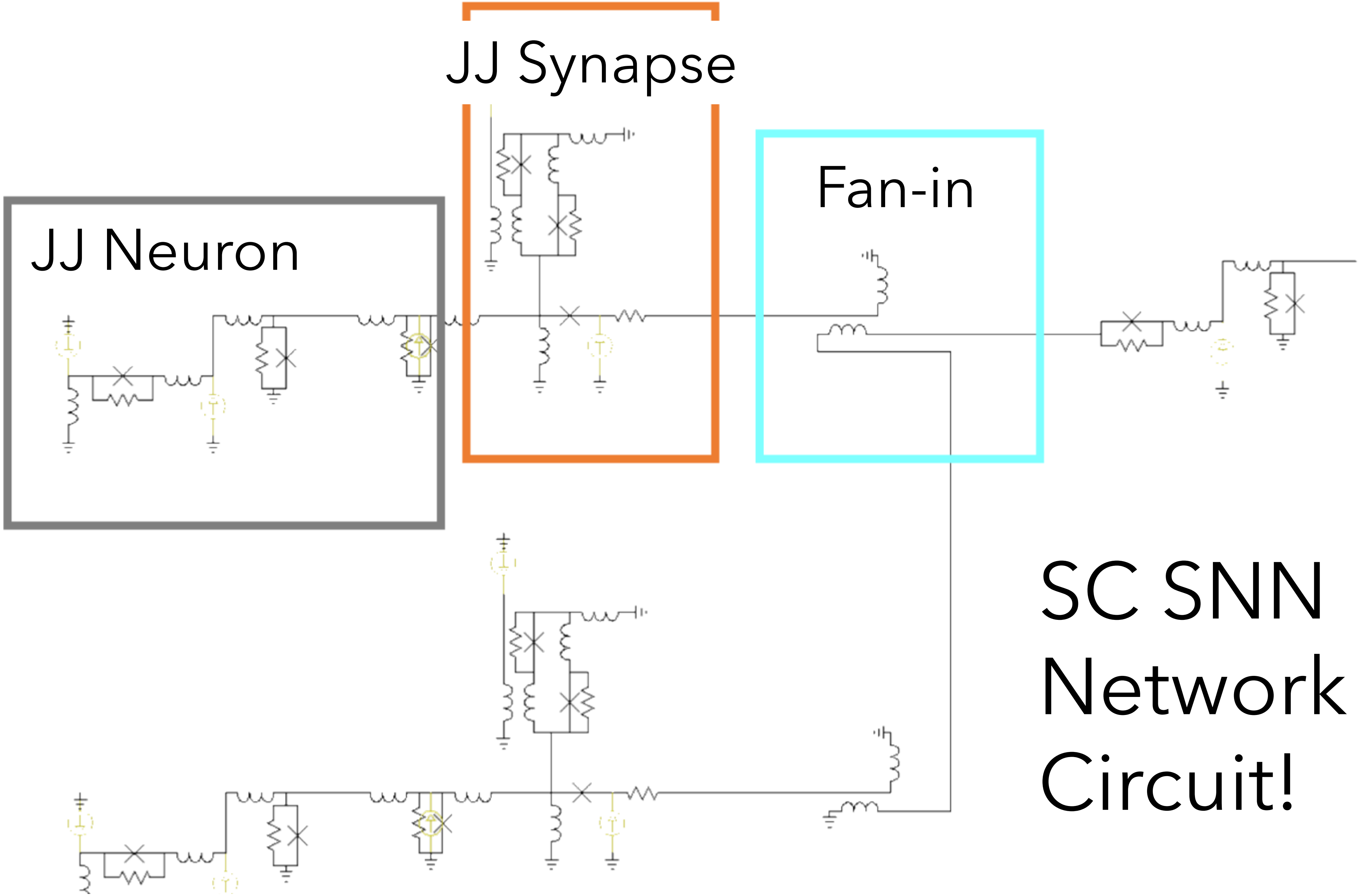
A	B	OUT
T0	T0	T0’
T0	T1	T1’
T1	T0	T1’
T1	T1	T0’

Time-to-First-Spike

A	B	OUT
500ps	500ps	535ps
500ps	550ps	550ps
550ps	500ps	550ps
550ps	550ps	585ps

The SC spiking neural network





SC SNN Network as Circuit!

Can we train this network?

Yes, we can!

Loss Function

$$L_1(y, \hat{y}) = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

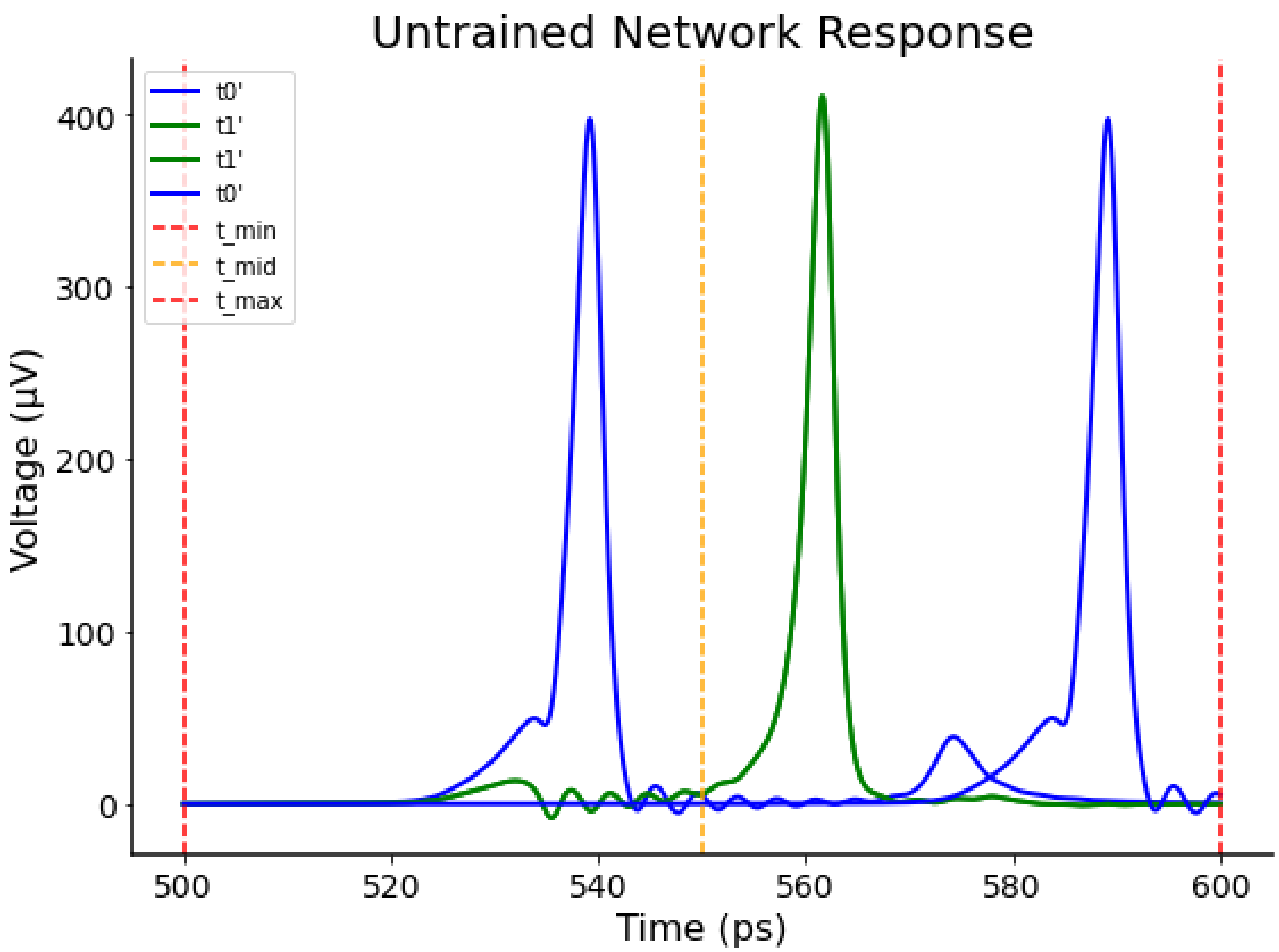
y = true values
ŷ = predicted values

Gradient Descent

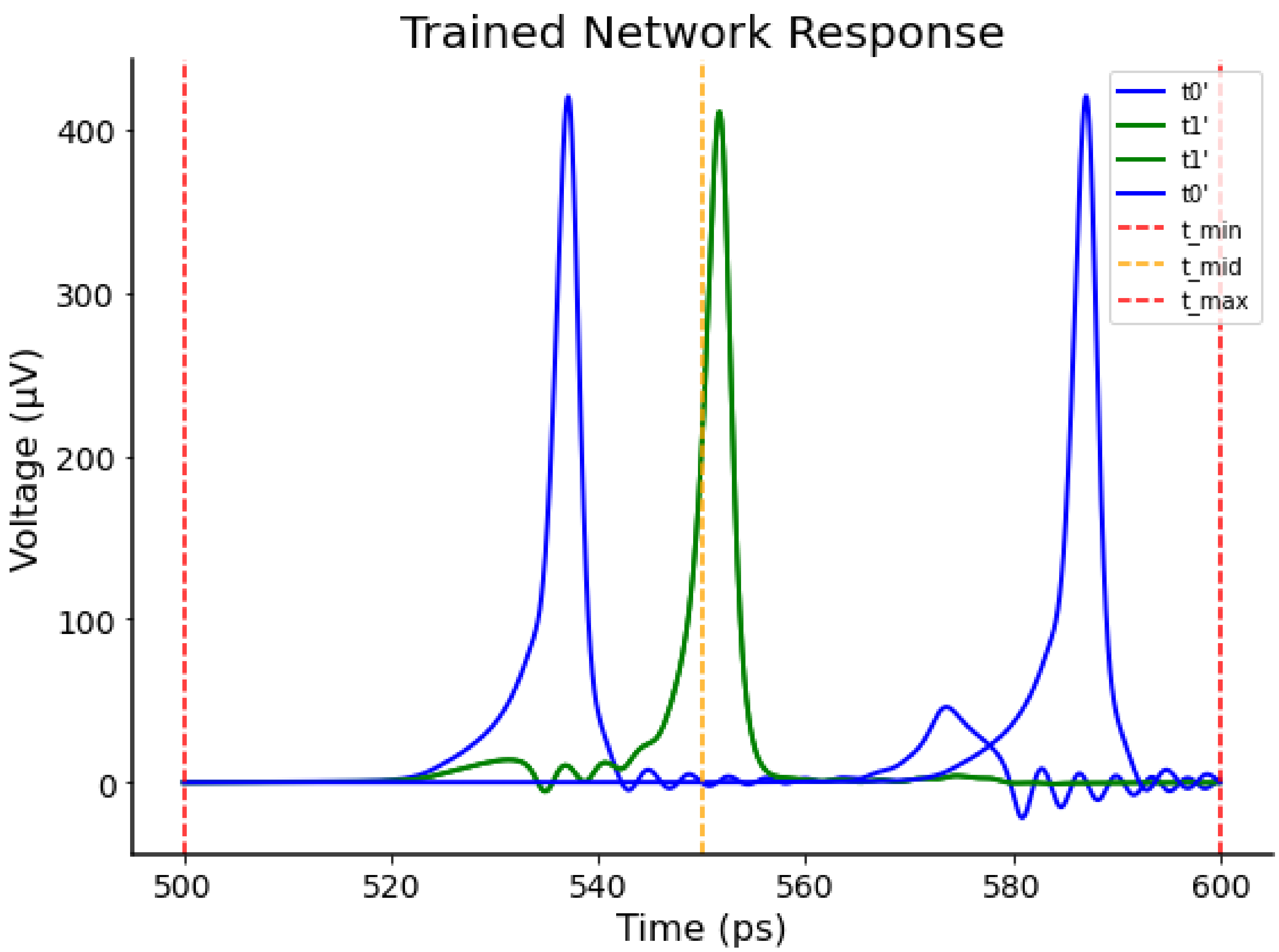
$$\theta_{i+1} = \theta_i - \alpha \nabla L(\theta_i)$$

θ = weight
α = learning rate
L(θ) = loss of model using weight θ

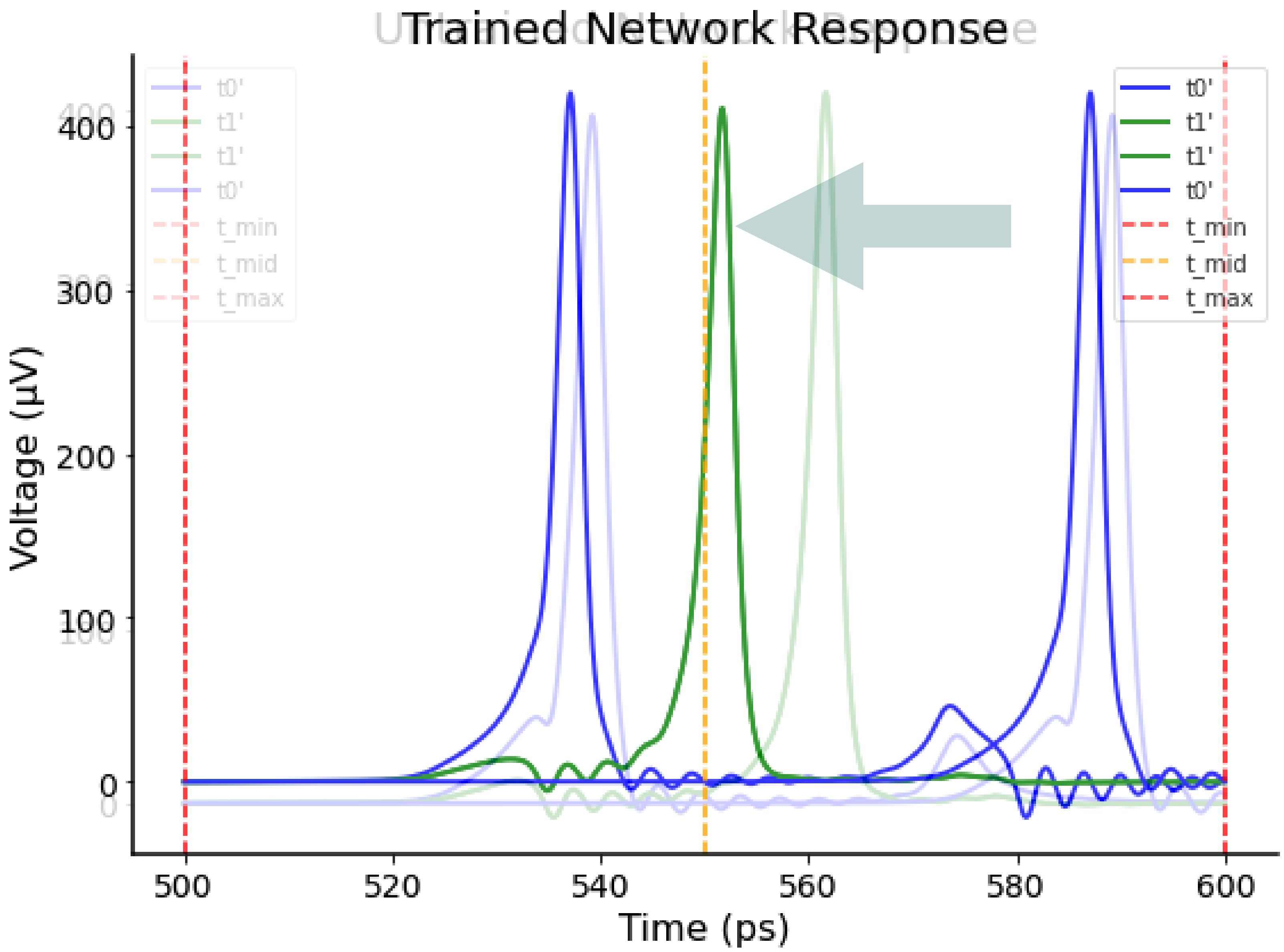
Before Training



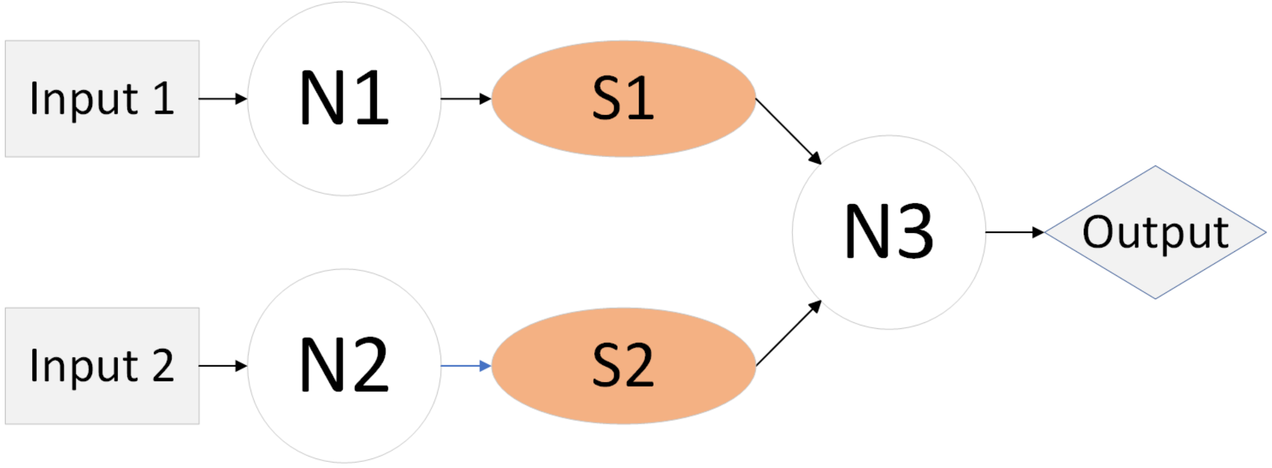
After Training



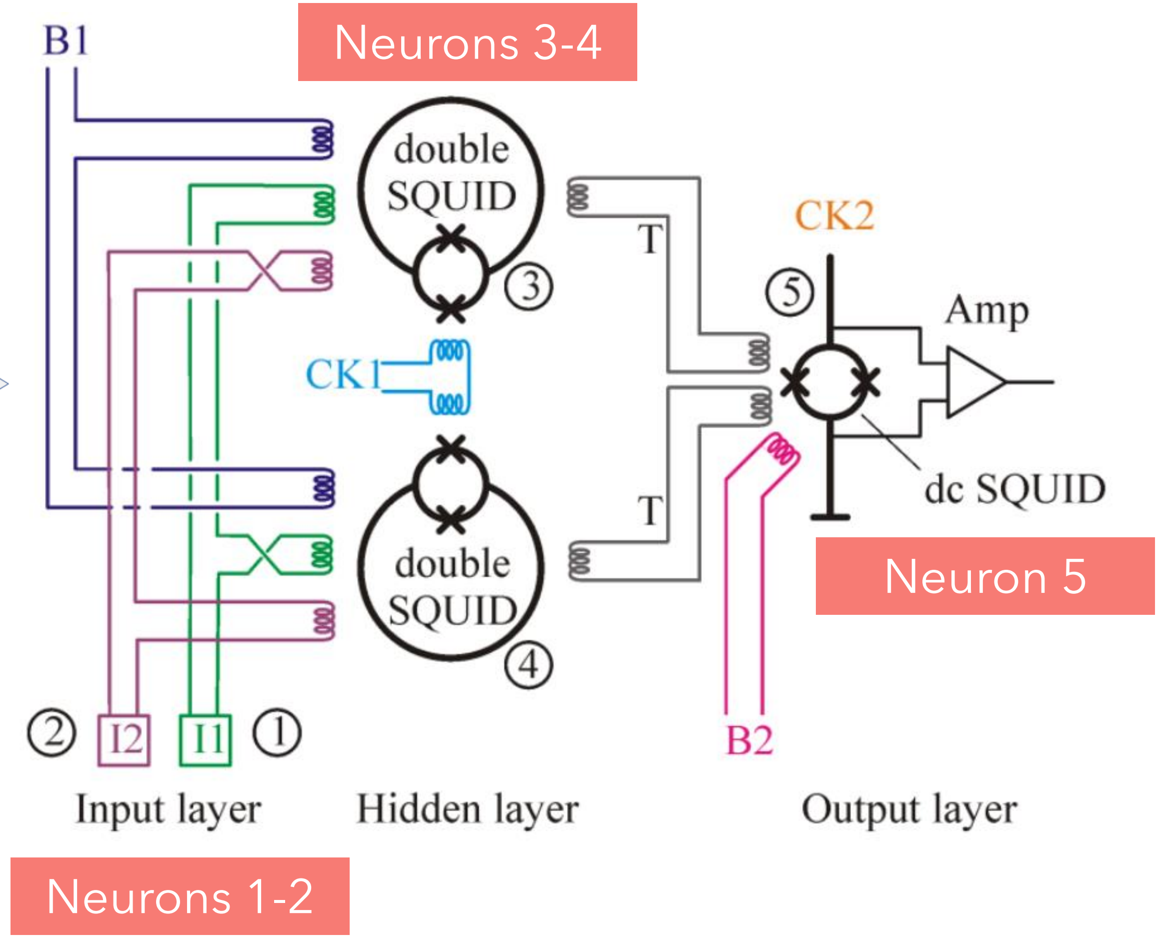
Before and After Training



Time encoded: 3 Neurons

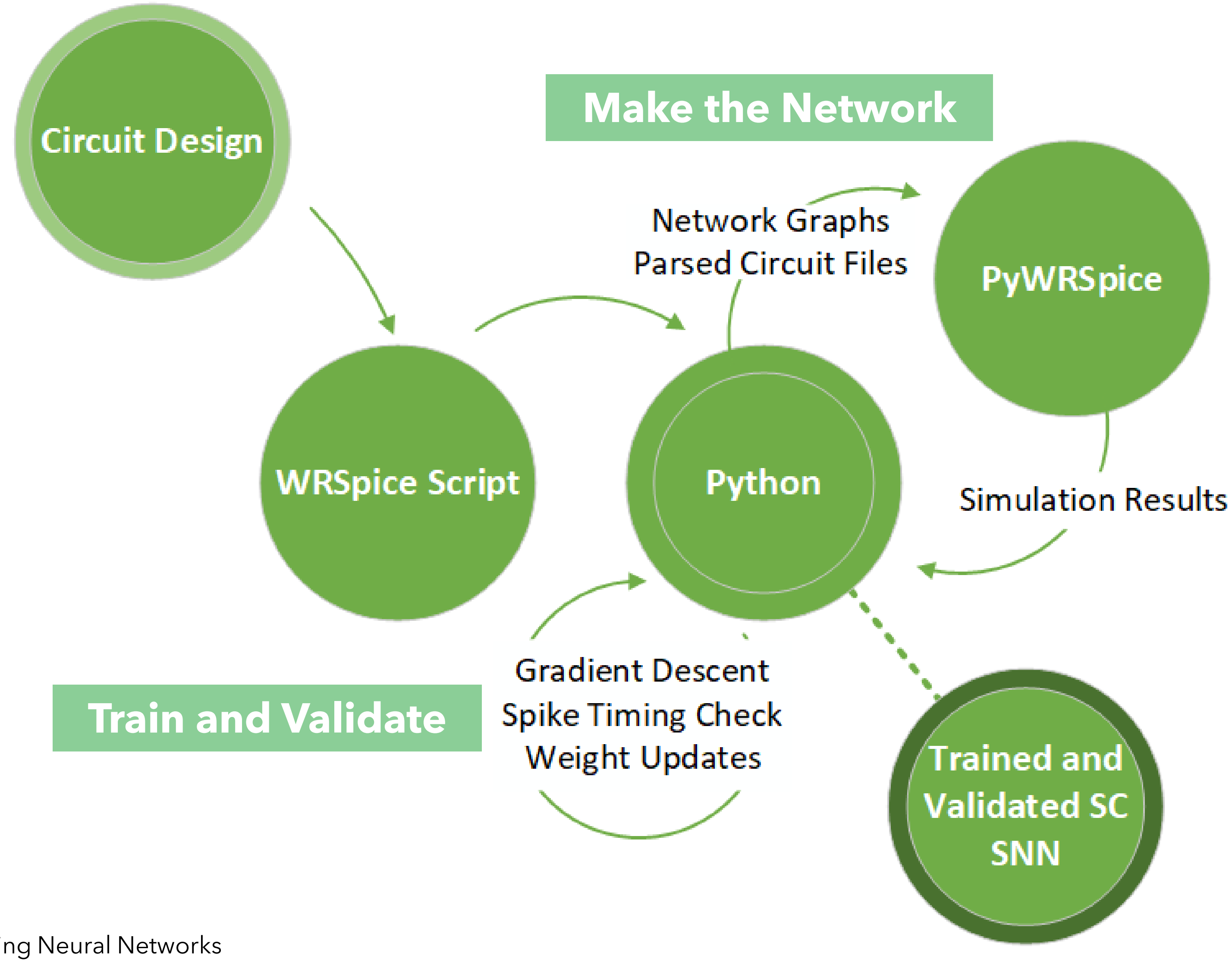


Rate encoded: 5 Neurons

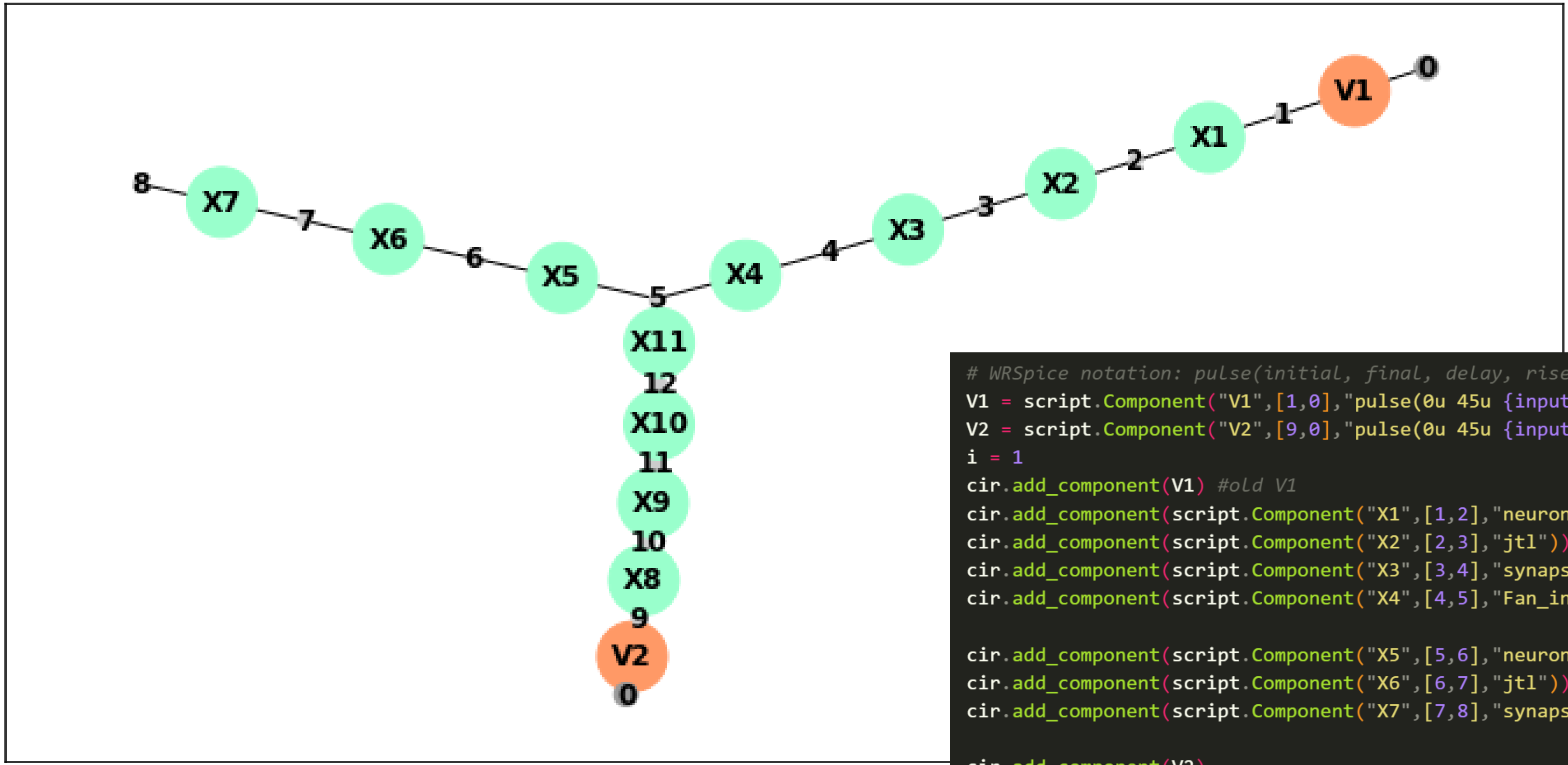


Creating the networks must have taken ages, right?

Python made it simple.



We can do any graph we want, sort of.



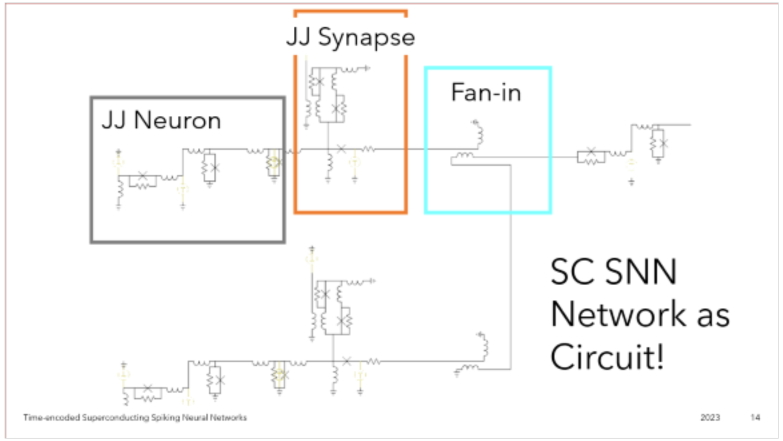

```python
# WRSpice notation: pulse(initial, final, delay, rise, fall, pulse width, period)
V1 = script.Component("V1",[1,0],"pulse(0u 45u {input0}p 0.1p 0.1p 50p 1n)", comment="Pulse source")
V2 = script.Component("V2",[9,0],"pulse(0u 45u {input1}p 0.1p 0.1p 50p 1n)", comment="Pulse source")
i = 1
cir.add_component(V1) #old V1
cir.add_component(script.Component("X1",[1,2],"neuron")) # 1, 2
cir.add_component(script.Component("X2",[2,3],"jtl")) #josesphson transission line to retain connectivity
cir.add_component(script.Component("X3",[3,4],"synapse0"))
cir.add_component(script.Component("X4",[4,5],"Fan_in")) #fan_in from neuron 1 synapse0

cir.add_component(script.Component("X5",[5,6],"neuron")) #integration part
cir.add_component(script.Component("X6",[6,7],"jtl"))
cir.add_component(script.Component("X7",[7,8],"synapse2")) # 8 is output node

cir.add_component(V2)
cir.add_component(script.Component("X8",[9,10],"neuron"))
cir.add_component(script.Component("X9",[10,11],"jtl"))
cir.add_component(script.Component("X10",[11,12],"synapse1"))
cir.add_component(script.Component("X11",[12,5],"Fan_in")) #fan_in from neuron 2
```

Research trajectory?

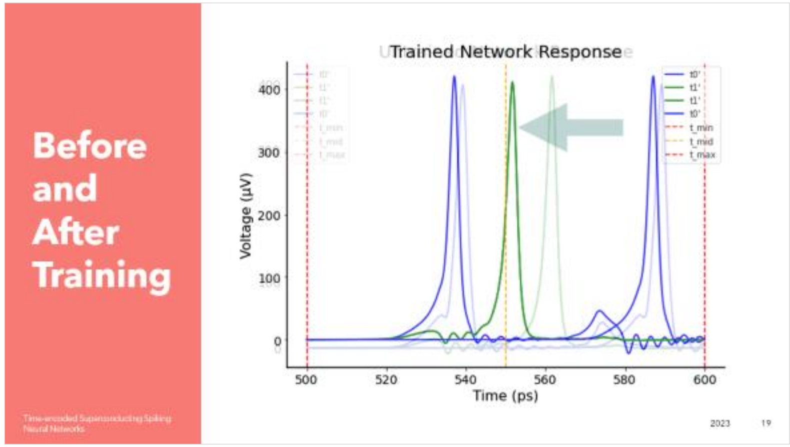
2020	2021	2022	2023	2024
Started working with Prof. Segall, looking into reservoirs and recurrent networks.	Continued reservoir work, we stumbled upon ghosting for the first time.	Hiatus, refined the Python codebase.	Solved XOR separability. Python automated design. Network sizes of 5x2 possible.	Spiking MNIST? Spike-GPT? Optimization?

Summary

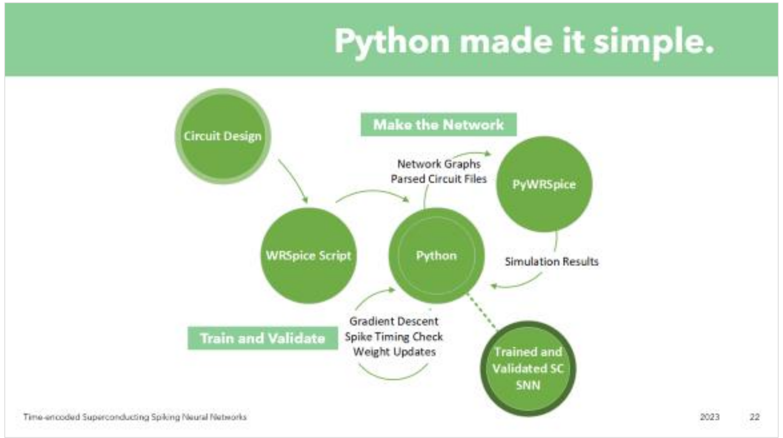
Missing piece: Time Encoding
Period vs Neuron Bias Current
Ghosting!
Period (ps)
Current (uA)
Time-encoded Superconducting Spiking Neural Networks
2023
10



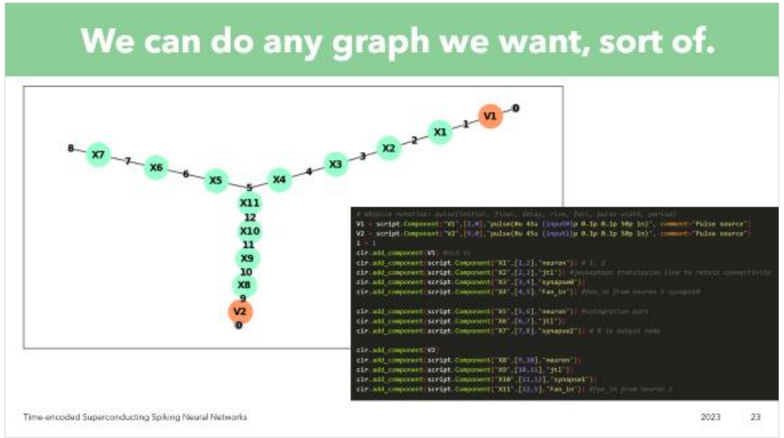
JJ Synapse
JJ Neuron
Fan-in
SC SNN Network as Circuit!
Time-encoded Superconducting Spiking Neural Networks
2023
14



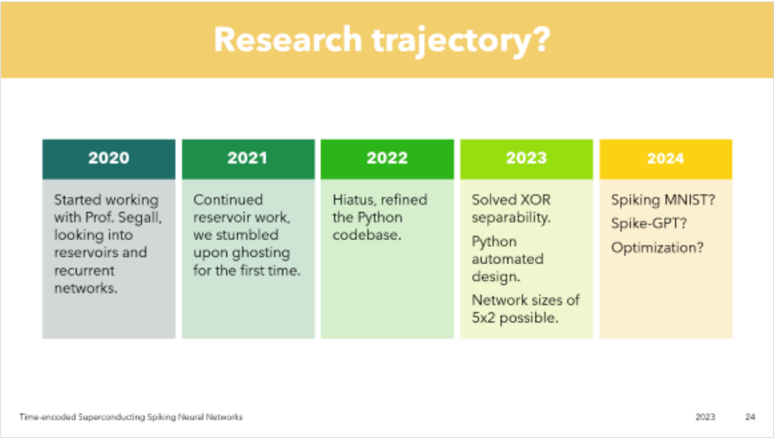
Before and After Training
Trained Network Response
Voltage (µV)
Time (ps)
2023
19



Python made it simple.
Circuit Design
Make the Network
Network Graphs
Parsed Circuit Files
PyWRSpice
WRSpice Script
Python
Simulation Results
Train and Validate
Gradient Descent
Spike Timing Check
Weight Updates
Trained and Validated SC SNN
Time-encoded Superconducting Spiking Neural Networks
2023
22



We can do any graph we want, sort of.
Time-encoded Superconducting Spiking Neural Networks
2023
23



Research trajectory?
2020
Started working with Prof. Segall, looking into reservoirs and recurrent networks.
2021
Continued reservoir work, we stumbled upon ghosting for the first time.
2022
Hiatus, refined the Python codebase.
2023
Solved XOR separability.
Python automated design.
Network sizes of 5x2 possible.
2024
Spiking MNIST?
Spike-GPT?
Optimization?
Time-encoded Superconducting Spiking Neural Networks
2023
24

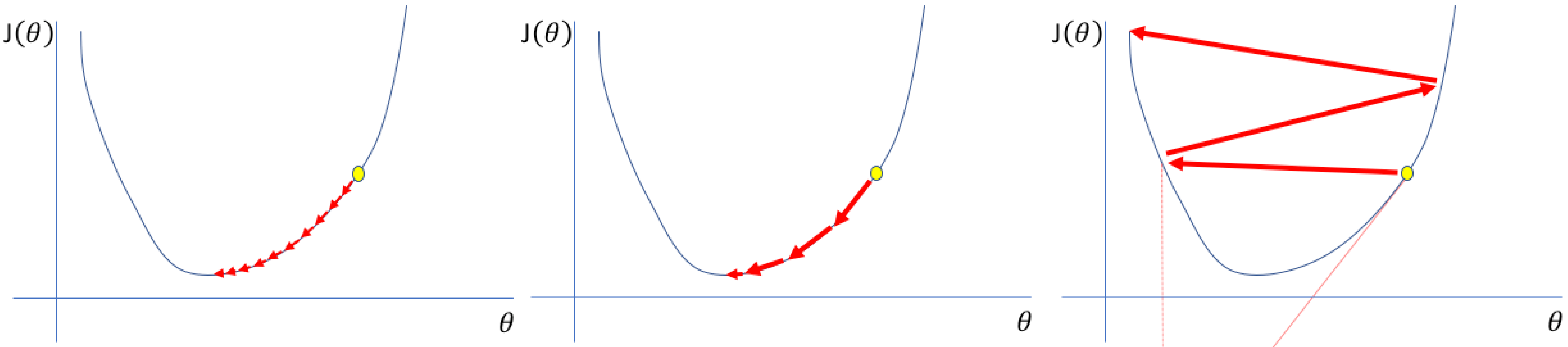
Daniel A. Espinosa Gonzalez
despinosa@colgate.edu

Thank you

References

- Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the Carbon Emissions of Machine Learning, November 2019. arXiv:1910.09700 (1)
- Espinosa, Segall. Time-encoded Superconducting Spiking Neural Networks, May 2023 (2)
- Wang Y, Wang R, Xu X. Neural Energy Supply-Consumption Properties Based on Hodgkin-Huxley Model. Neural Plast. 2017;2017:6207141. doi: 10.1155/2017/6207141. Epub 2017 Feb 16. PMID: 28316842; PMCID: PMC5337805. (3)
- Chiarello, F, P Carelli, M G Castellano, and G Torrioli. "Artificial Neural Network Based on SQUIDs: Demonstration of Network Training and Operation." *Superconductor Science and Technology* 26, no. 12 (December 1, 2013): 125009. https://doi.org/10.1088/0953-2048/26/12/125009. (4)
- Bodo Rueckauer and Shih-Chii Liu. Conversion of analog to spiking neural networks using sparse temporal coding. In 2018 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–5, Florence, 2018. IEEE.
- Rui-Jie Zhu, Qihang Zhao, and Jason K. Eshraghian. SpikeGPT: Generative Pre-trained Language Model with Spiking Neural Networks. 2023. Publisher: arXiv Version Number: 2.

Too low | Just right | Too high

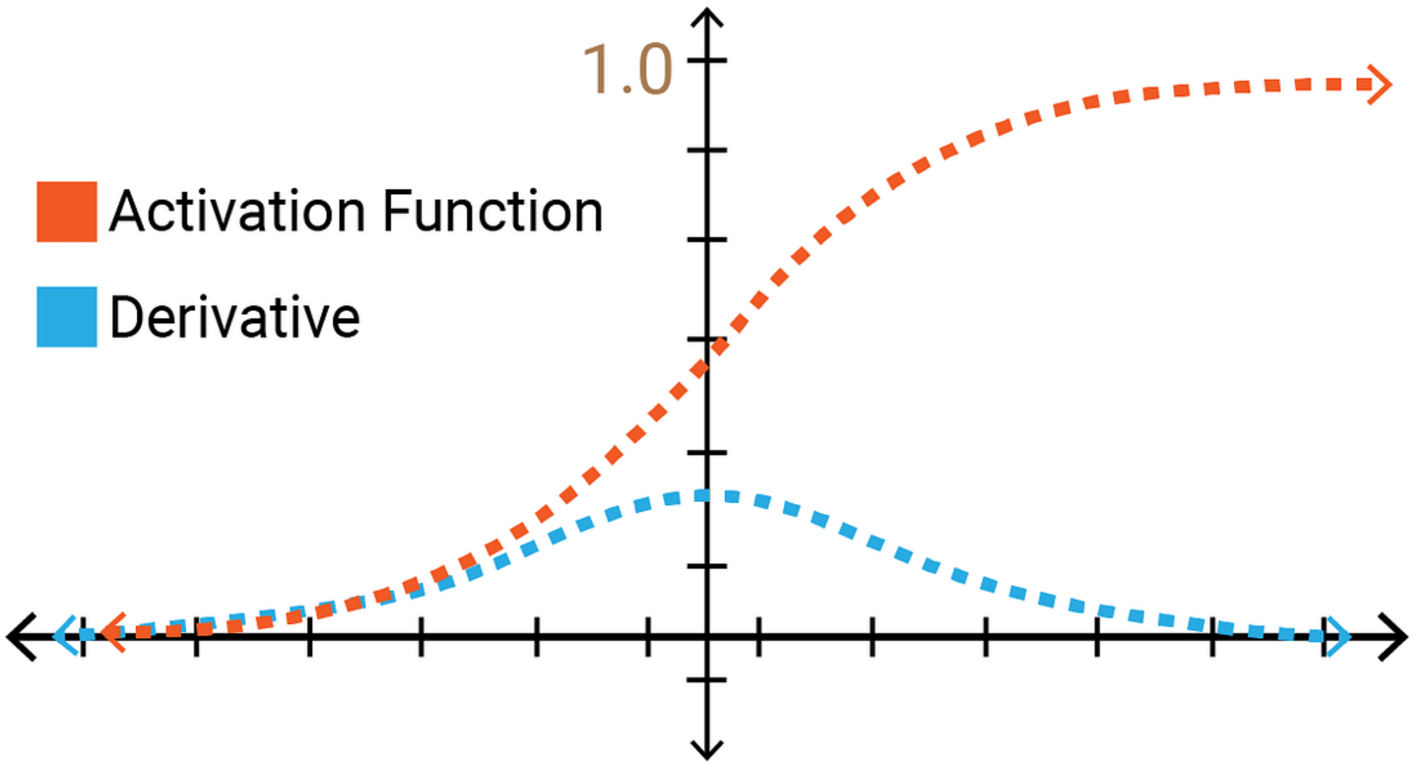


A small learning rate requires many updates before reaching the minimum point

The optimal learning rate swiftly reaches the minimum point

Too large of a learning rate causes drastic updates which lead to divergent behaviors

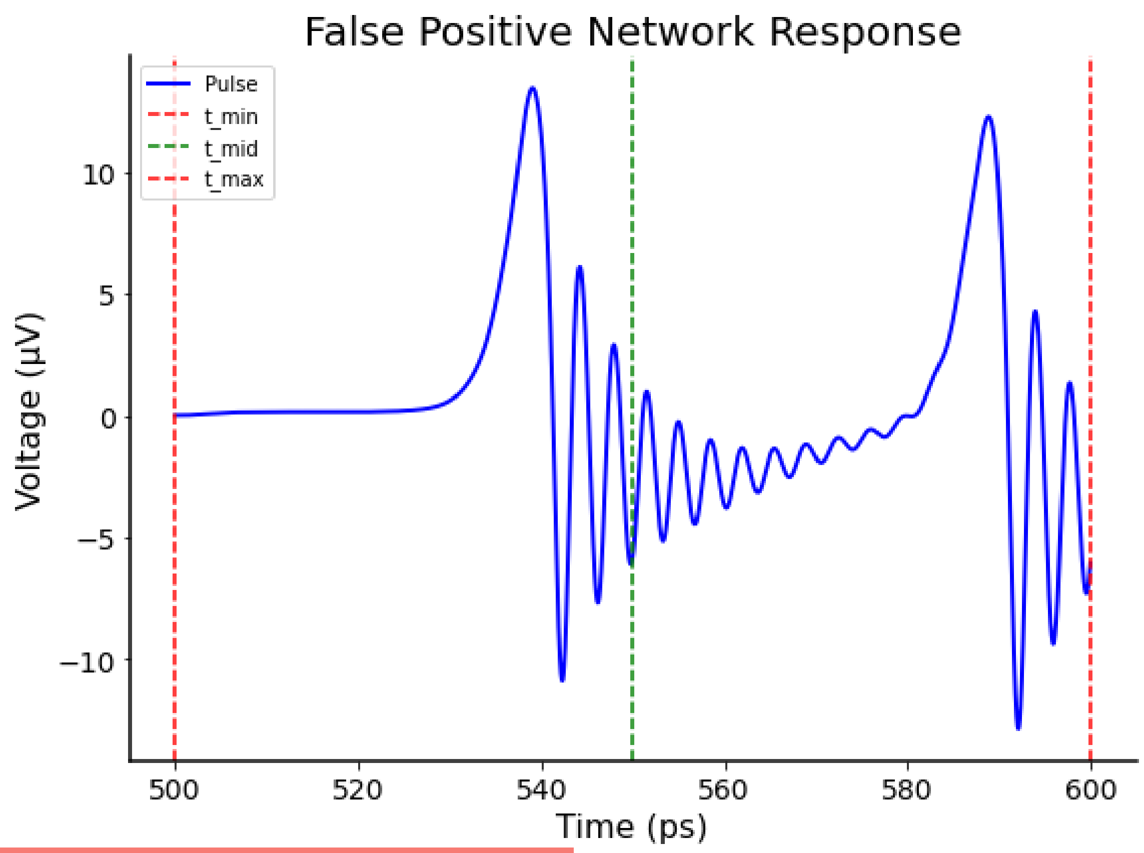
Learning Rate



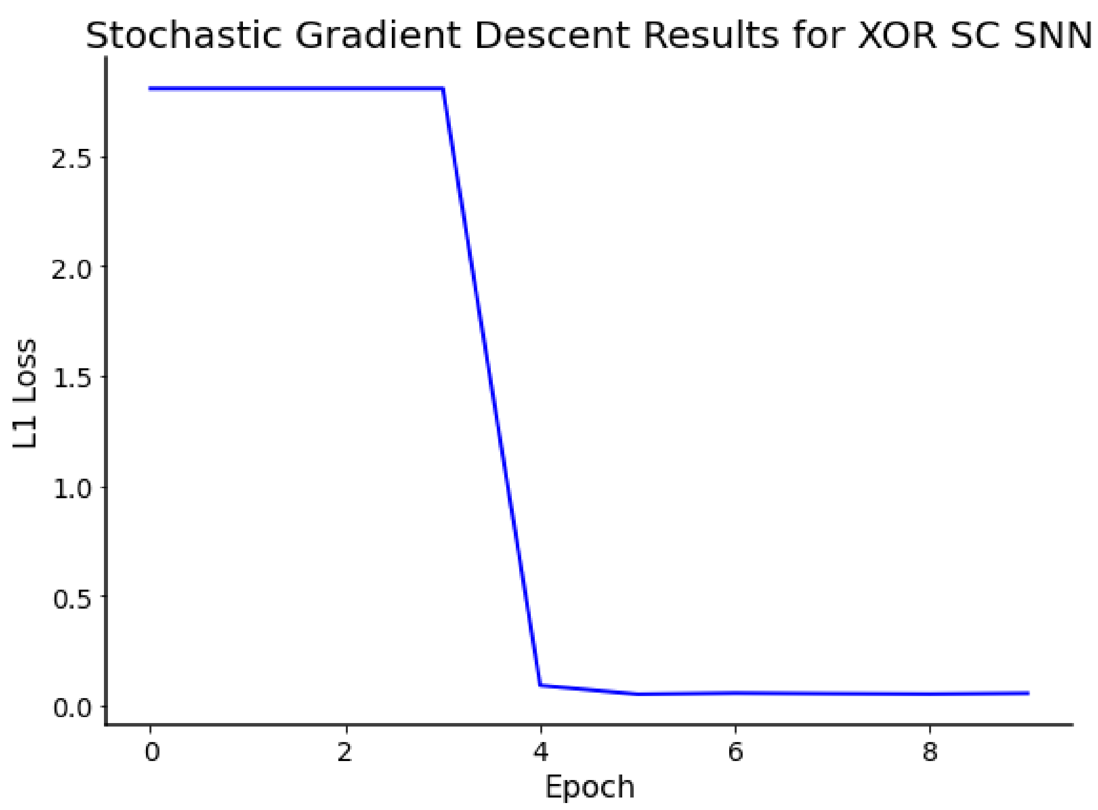
Vanishing Gradients

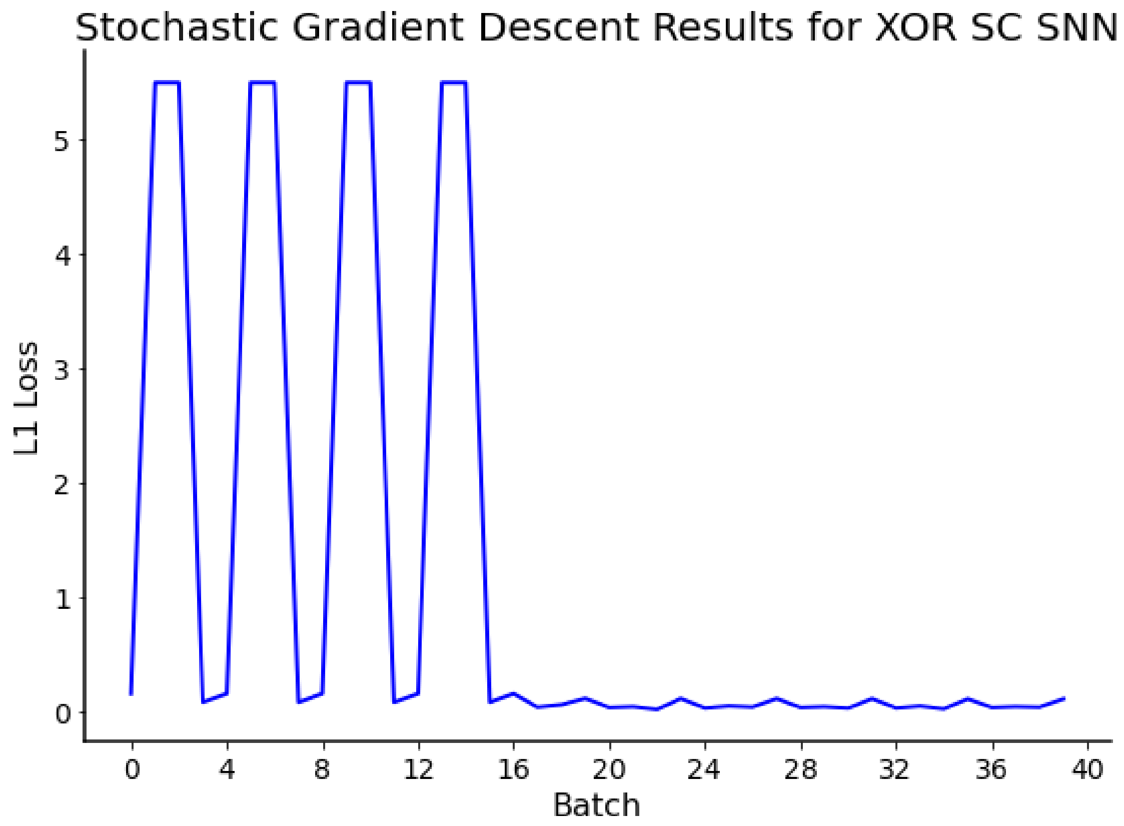
The function searches for the maximum voltage value and returns the corresponding time value as the time of the first spike.

We assume the maximum value is an action potential.



Detecting Spike Timing


Stochastic Gradient Descent Results for XOR SC SNN
L1 Loss
Epoch
0.0
0.5
1.0
1.5
2.0
2.5
0
2
4
6
8


Stochastic Gradient Descent Results for XOR SC SNN
L1 Loss
Batch
0
1
2
3
4
5
0
4
8
12
16
20
24
28
32
36
40

Loss Curves