Spiking Neural Networks Are Not Transformers (Yet):

The Architectural Problems For SNNs in Modeling Long-Range Dependencies

William Fishell

June 8, 2025

Combined with **auto-regressive** prompting, **LLMs** trained on predicting the next token transcend mere auto-complete tools

Combined with **auto-regressive** prompting, **LLMs** trained on predicting the next token transcend mere auto-complete tools

Auto-regressive
Next-Token
Prediction: A
Frozen Example

Combined with **auto-regressive** prompting, **LLMs** trained on predicting the next token transcend mere auto-complete tools

Auto-regressive Next-Token Prediction: A Frozen Example

Auto-Regressive LLM Next-Token Models Can Perform Higher Reasoning

Prompt:	1394×8618=
MLP:	(4×1+9×10+3×100+1×1000)× (8×1+1×10+6×100+8×1000)= : : **12013492**
GPT-4:	The multiplication of 1394 and 8618 equals **12**, **01** **4**, **05** **2**.
Answer:	**12013492**

Model	Acc. (exact/per-digit)
MLP-775M	96.9% / 99.5%
GPT-3.5	1.2% / 61.9%
GPT-4*	5.3% / 61.8%
Goat-7B*	96.9% / 99.2%

Combined with **auto-regressive** prompting, **LLMs** trained on predicting the next token transcend mere auto-complete tools

Auto-regressive Next-Token Prediction: A Frozen Example

Auto-Regressive LLM Next-Token Models Can Perform Higher Reasoning

Prompt:	1394×8618=
MLP:	(4×1+9×10+3×100+1×1000)× (8×1+1×10+6×100+8×1000)= : : 12013492
GPT-4:	The multiplication of 1394 and 8618 equals 12 , 01 4 , 05 2 .
Answer:	12013492

Model	Acc. (exact/per-digit)
MLP-775M	96.9% / 99.5%
GPT-3.5	1.2% / 61.9%
GPT-4*	5.3% / 61.8%
Goat-7B*	96.9% / 99.2%

- For any function F that can be computed using a Turing Machine, there exists a data set D which approximates F using next-token predictions (Malach et al.).

Combined with **auto-regressive** prompting, **LLMs** trained on predicting the next token transcend mere auto-complete tools

Auto-regressive Next-Token Prediction: A Frozen Example

Auto-Regressive LLM Next-Token Models Can Perform Higher Reasoning

Prompt:	1394×8618=
MLP:	(4×1+9×10+3×100+1×1000)× (8×1+1×10+6×100+8×1000)= : : **12013492**
GPT-4:	The multiplication of 1394 and 8618 equals **12**, **01** **4**, **05** **2**.
Answer:	**12013492**

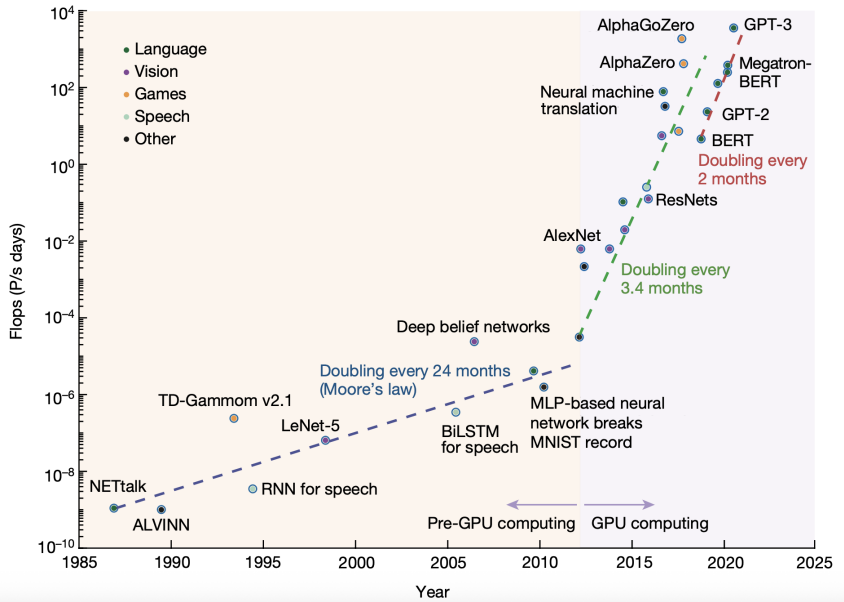
Model	Acc. (exact/per-digit)
MLP-775M	96.9% / 99.5%
GPT-3.5	1.2% / 61.9%
GPT-4*	5.3% / 61.8%
Goat-7B*	96.9% / 99.2%

- For any function F that can be computed using a Turing Machine, there exists a data set D which approximates F using next-token predictions (Malach et al.).
- The large amount of training data coupled with their focus on next token prediction makes LLMs uniquely situated for this task.

The desire to build larger and larger models-specifically Large Language Models- is driving energy consumption to **unsustainable levels**

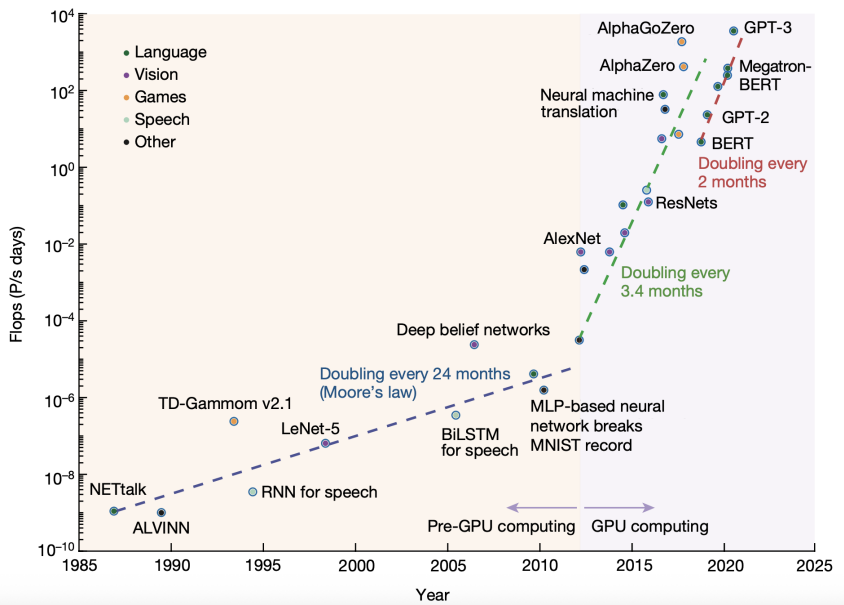
The desire to build larger and larger models-specifically Large Language Models- is driving energy consumption to **unsustainable levels**

Growing Compute Demands of AI

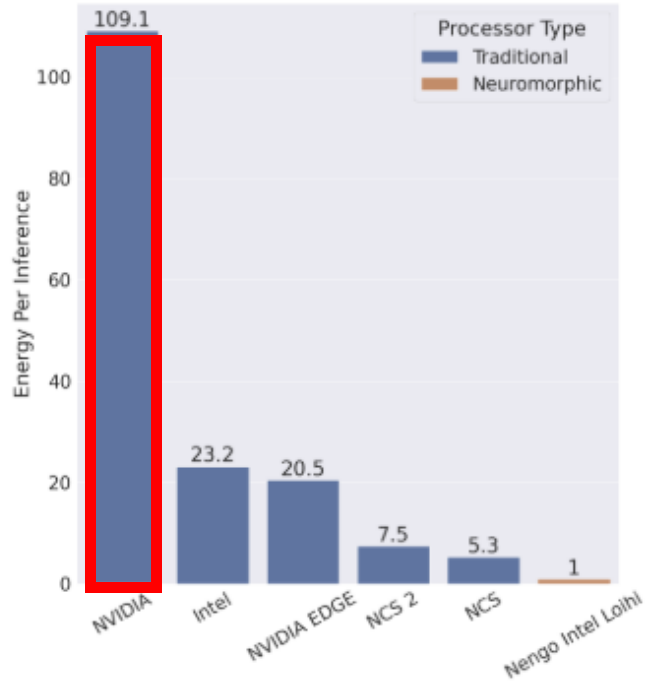


The desire to build larger and larger models-specifically Large Language Models- is driving energy consumption to **unsustainable levels**

Growing Compute Demands of AI

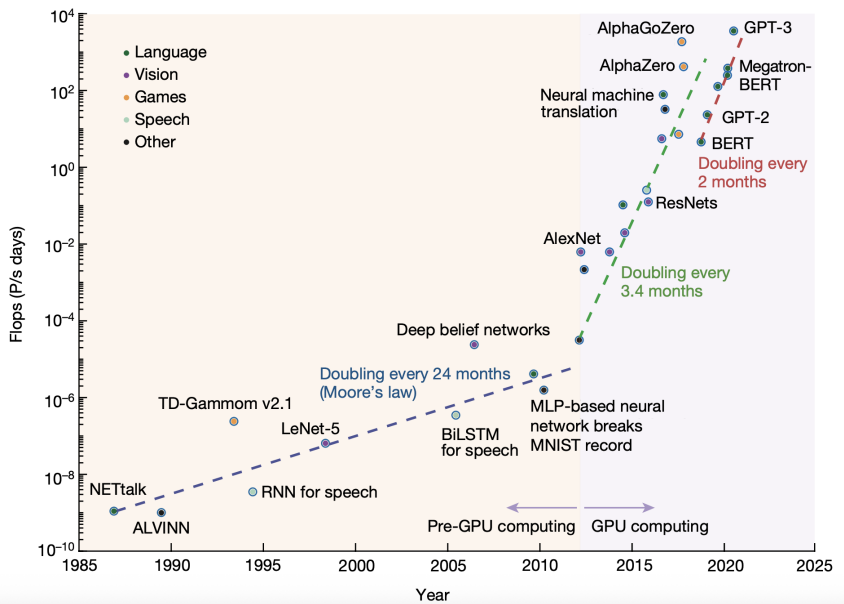


GPU vs. Neuromorphic

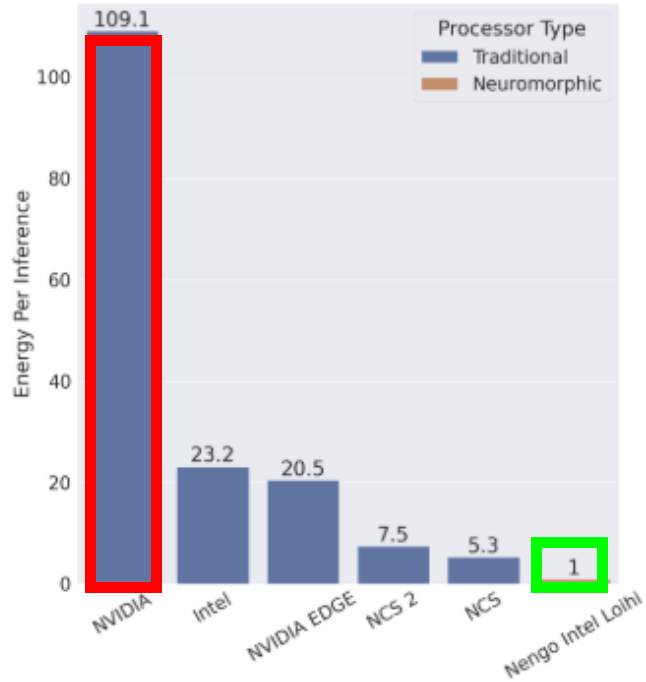


The desire to build larger and larger models-specifically Large Language Models- is driving energy consumption to **unsustainable levels**

Growing Compute Demands of AI

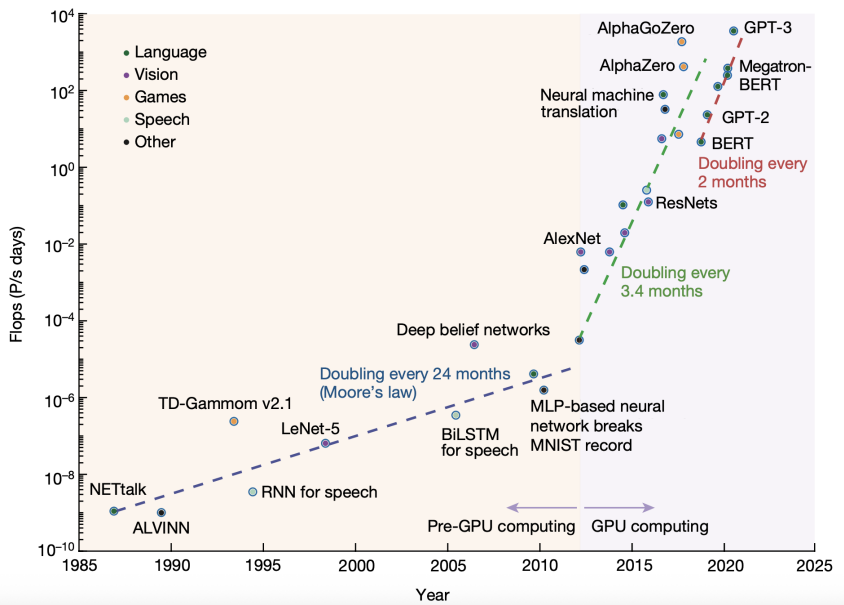


GPU vs. Neuromorphic

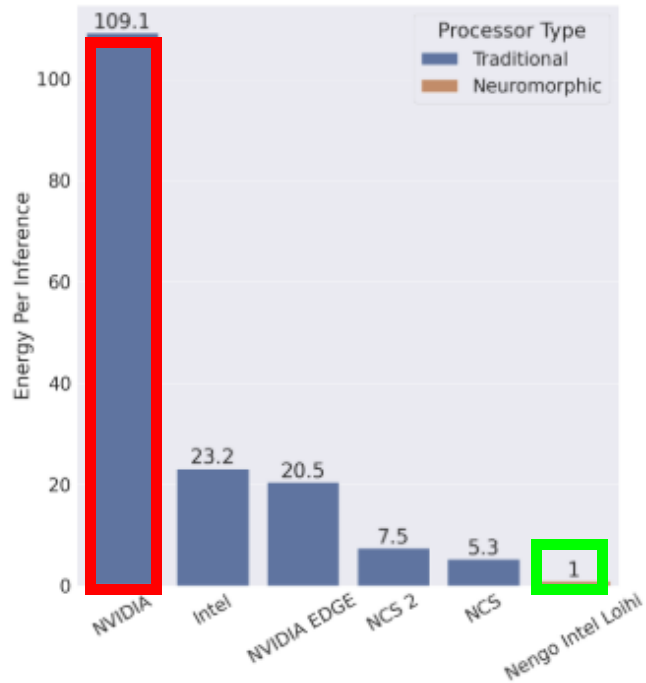


The desire to build larger and larger models-specifically Large Language Models- is driving energy consumption to **unsustainable levels**

Growing Compute Demands of AI



GPU vs. Neuromorphic

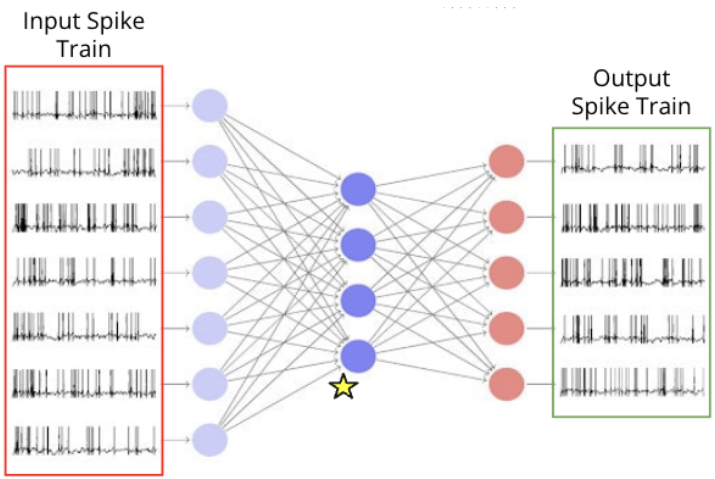


If neuromorphic hardware is so **much more power efficient while maintaining performance**, what obstacles are preventing widespread adoption of this technology?

Neuromorphic hardware uses **Spiking Neural Networks**, biologically plausible models that evolve in time and emit discrete spikes

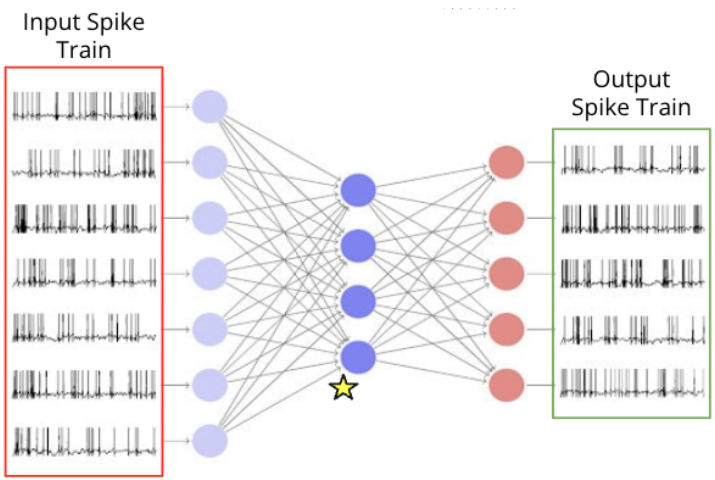
Neuromorphic hardware uses **Spiking Neural Networks**, biologically plausible models that evolve in time and emit discrete spikes

Feedforward Spiking Neural Network (SNN)

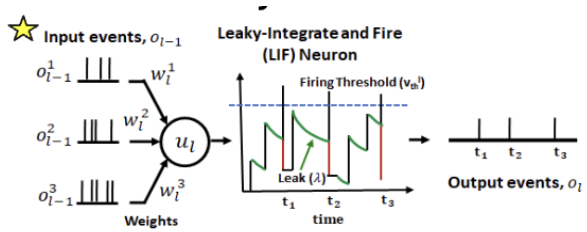


Neuromorphic hardware uses **Spiking Neural Networks**, biologically plausible models that evolve in time and emit discrete spikes

Feedforward Spiking Neural Network (SNN)



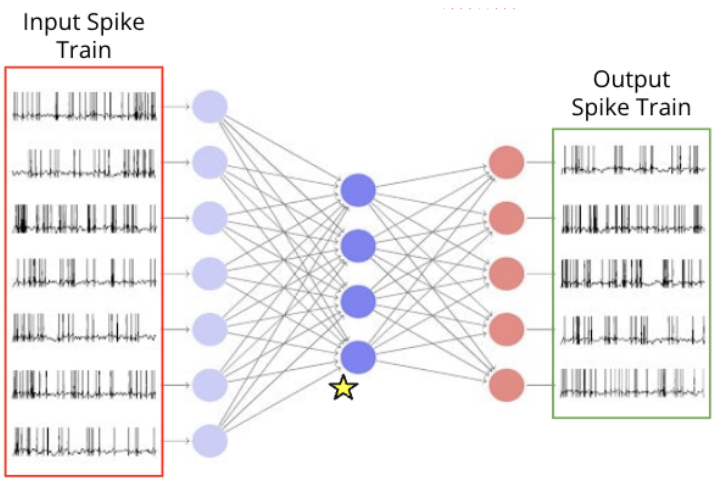
Single Neuron Spiking Dynamics



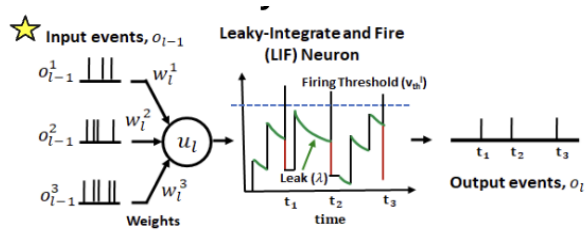
The sparse spiking dynamics in SNNs make them well suited for low powered neuromorphic hardware and more energy efficient then artificial neural networks (ANNs).

Neuromorphic hardware uses **Spiking Neural Networks**, biologically plausible models that evolve in time and emit discrete spikes

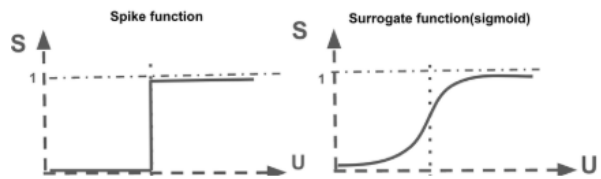
Feedforward Spiking Neural Network (SNN)



Single Neuron Spiking Dynamics



SNNs learn via surrogate gradients and backpropagation through time (BPTT)

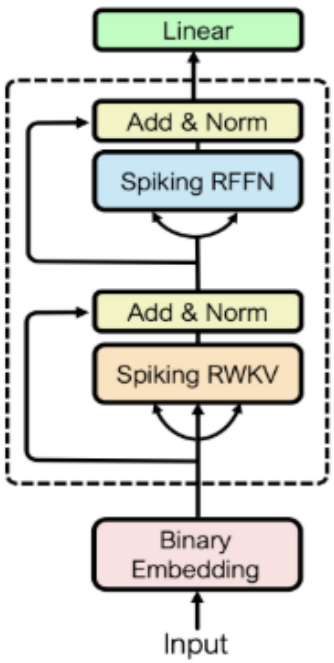


The sparse spiking dynamics in SNNs make them well suited for low powered neuromorphic hardware and more energy efficient then artificial neural networks (ANNs).

Despite their energy efficiency, **SNNs have lagged behind ANNs** in state of the art performance in language modeling

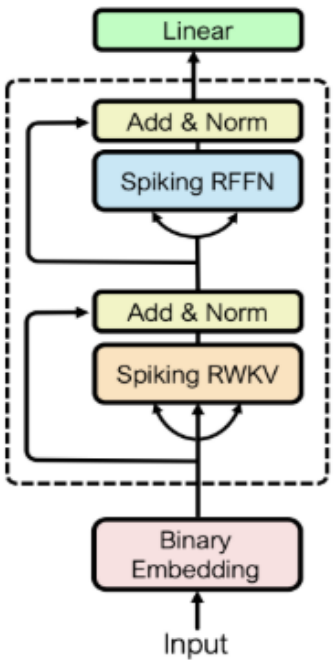
SpikeGPT Zhang & Eshraghian et al.

- SpikeGPT matched GPT-3's performance with 10× fewer parameters and 33× less power during inference



Despite their energy efficiency, **SNNs have lagged behind ANNs** in state of the art performance in language modeling

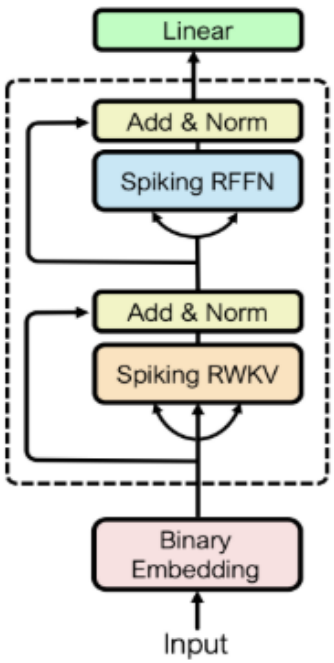
SpikeGPT Zhang & Eshraghian et al.



- SpikeGPT matched GPT-3's performance with 10× fewer parameters and 33× less power during inference
- SpikeFormer and similar SNN-based Transformer designs fail to resolve these issues because SNNs' sequential integration conflicts with self-attention.

Despite their energy efficiency, **SNNs have lagged behind ANNs** in state of the art performance in language modeling

SpikeGPT Zhang & Eshraghian et al.



- SpikeGPT matched GPT-3's performance with 10× fewer parameters and 33× less power during inference
- SpikeFormer and similar SNN-based Transformer designs fail to resolve these issues because SNNs' sequential integration conflicts with self-attention.
- SNNs have an inductive bias towards low-frequency functions, which is not optimal for modeling sparse long-range dependencies (Latham et al.)

Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

Dependency Parsing in Language Modeling

Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

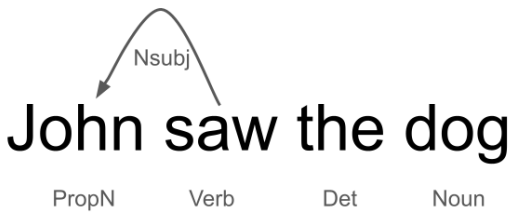
Dependency Parsing in Language Modeling

John saw the dog

PropN Verb Det Noun

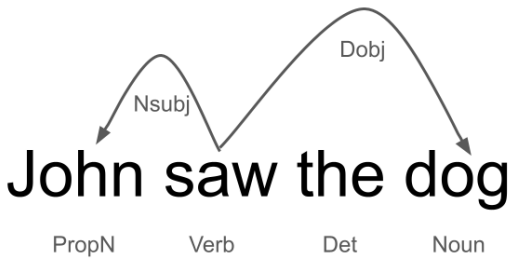
Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

Dependency Parsing in Language Modeling



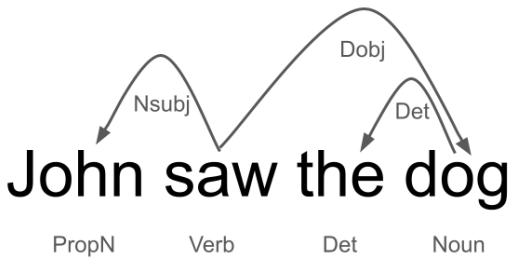
Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

Dependency Parsing in Language Modeling



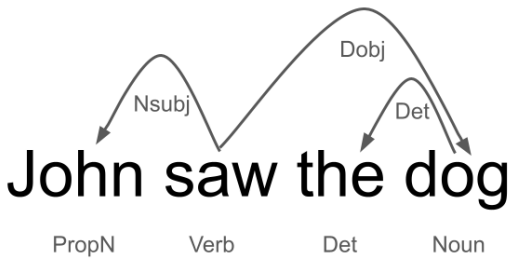
Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

Dependency Parsing in Language Modeling



Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

Dependency Parsing in Language Modeling

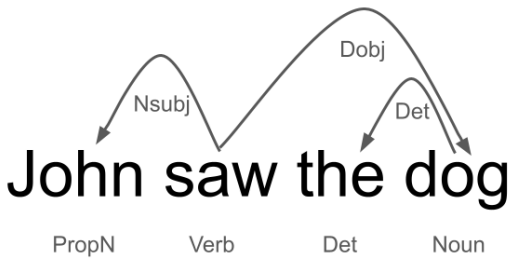


Value-weighted attention scores for attention head in MLP

<START>Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large moustache. Mrs Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbours. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs Potter was Mrs Dursley's sister, but they hadn't met for several years; in fact, Mrs Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbours would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.

Long-Range Dependencies are relationships in sequential data where predictions are **dependent** on points much earlier

Dependency Parsing in Language Modeling



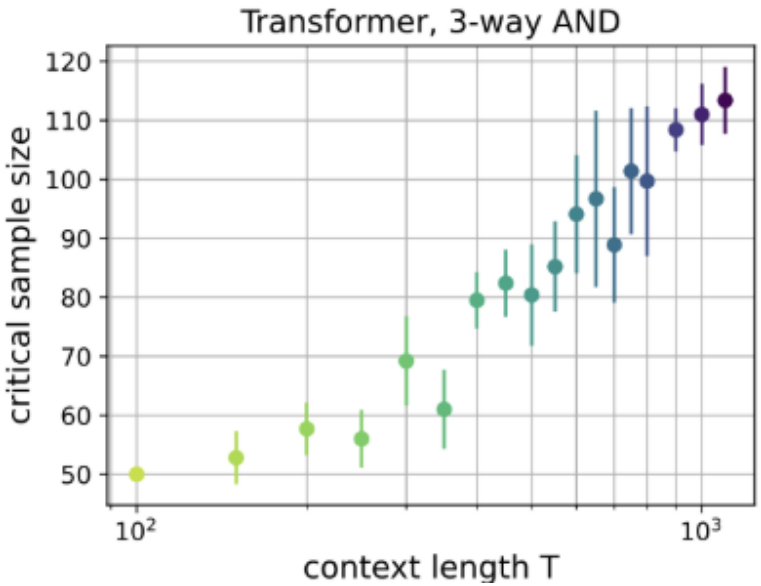
Value-weighted attention scores for attention head in MLP

<START>Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large moustache. Mrs Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbours. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs Potter was Mrs Dursley's sister, but they hadn't met for several years; in fact, Mrs Dursley pretended she didn't have a sister, because her sister and her good- for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbours would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.

Language modeling is **dependent on a sparse set of long-range dependencies**, thus strong language modeling is contingent on modeling long-range dependencies well.

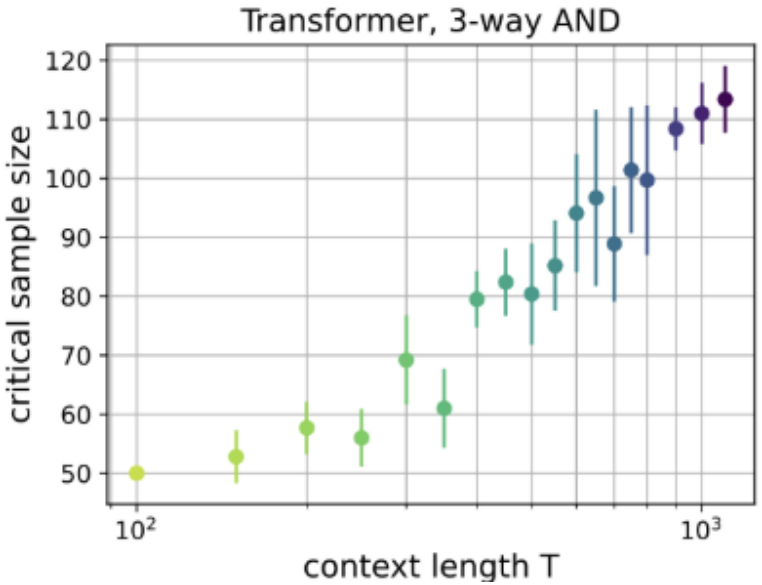
Transformers can **learn long-range dependencies** modeling sequences of length T given $O(\log(T))$ **samples**

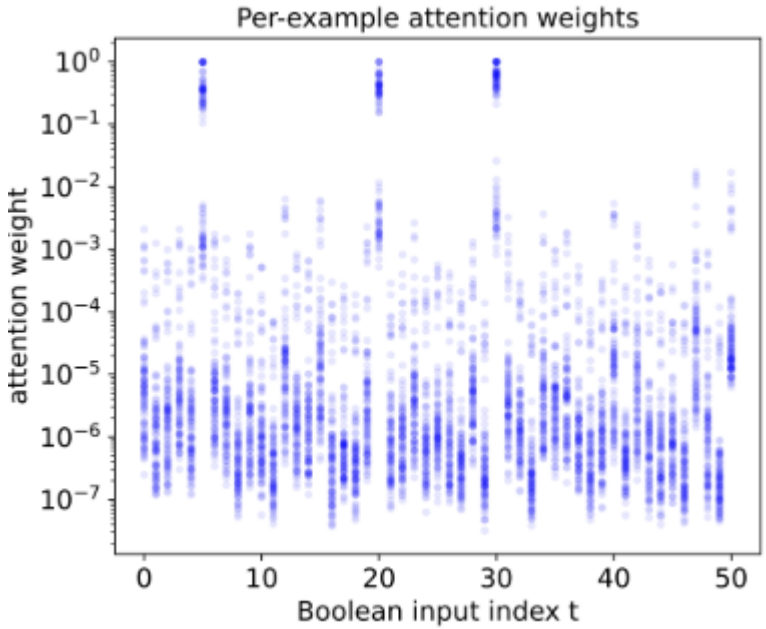
Transformers can **learn long-range dependencies** modeling sequences of length T given $O(\log(T))$ **samples**



- 3-way AND: Given a Boolean sequence of length T with a hidden rule, the model must identify the three elements whose $\wedge$ conjunction determines the output with perfect accuracy on the sample set

Transformers can **learn long-range dependencies** modeling sequences of length T given $O(\log(T))$ **samples**

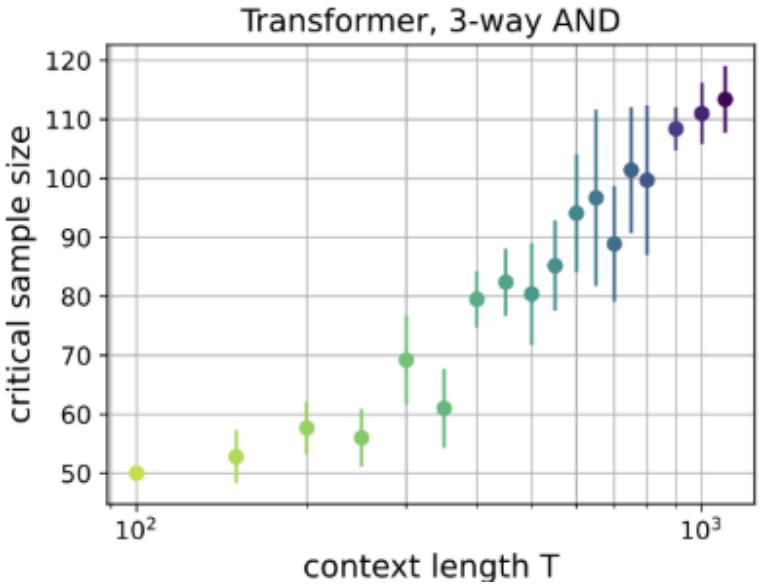


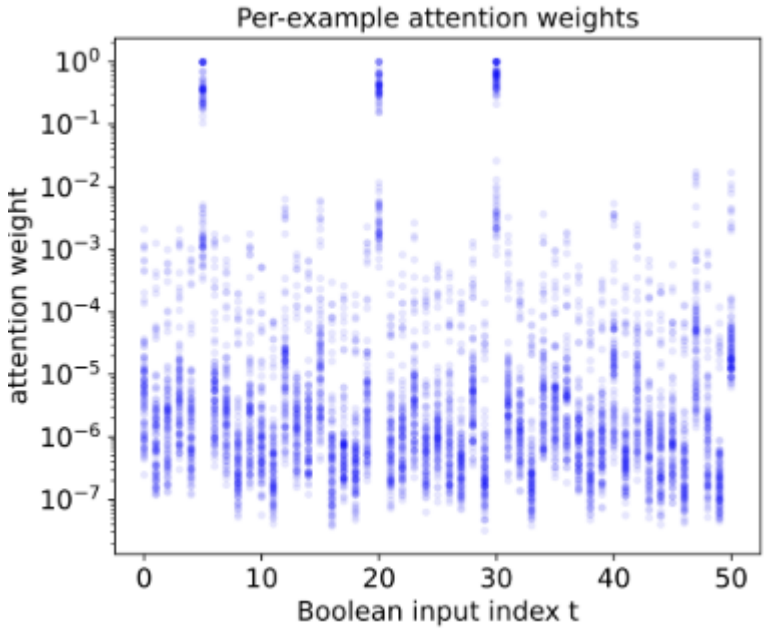


- 3-way AND: Given a Boolean sequence of length T with a hidden rule, the model must identify the three elements whose $\wedge$ conjunction determines the output with perfect accuracy on the sample set

- Per-index attention weights over 300 length-50 boolean vectors with important indices 5, 20, 30.

Transformers can **learn long-range dependencies** modeling sequences of length T given $O(\log(T))$ **samples**





- 3-way AND: Given a Boolean sequence of length T with a hidden rule, the model must identify the three elements whose $\wedge$ conjunction determines the output with perfect accuracy on the sample set

- Per-index attention weights over 300 length-50 boolean vectors with important indices 5, 20, 30.

Self-attention's sparsity bias lets Transformers capture long-range dependencies in lengthy contexts, making them effective for language modeling (Edelman et al.).

Learning-theory research in SNN architectures lags behind ANNs; understanding SNNs is crucial to improving them

Learning-theory research in SNN architectures lags behind ANNs; understanding SNNs is crucial to improving them

Unlike Transformers and RNNs, there is no analysis of how SNNs model long-range dependencies

Learning-theory research in SNN architectures lags behind ANNs; understanding SNNs is crucial to improving them

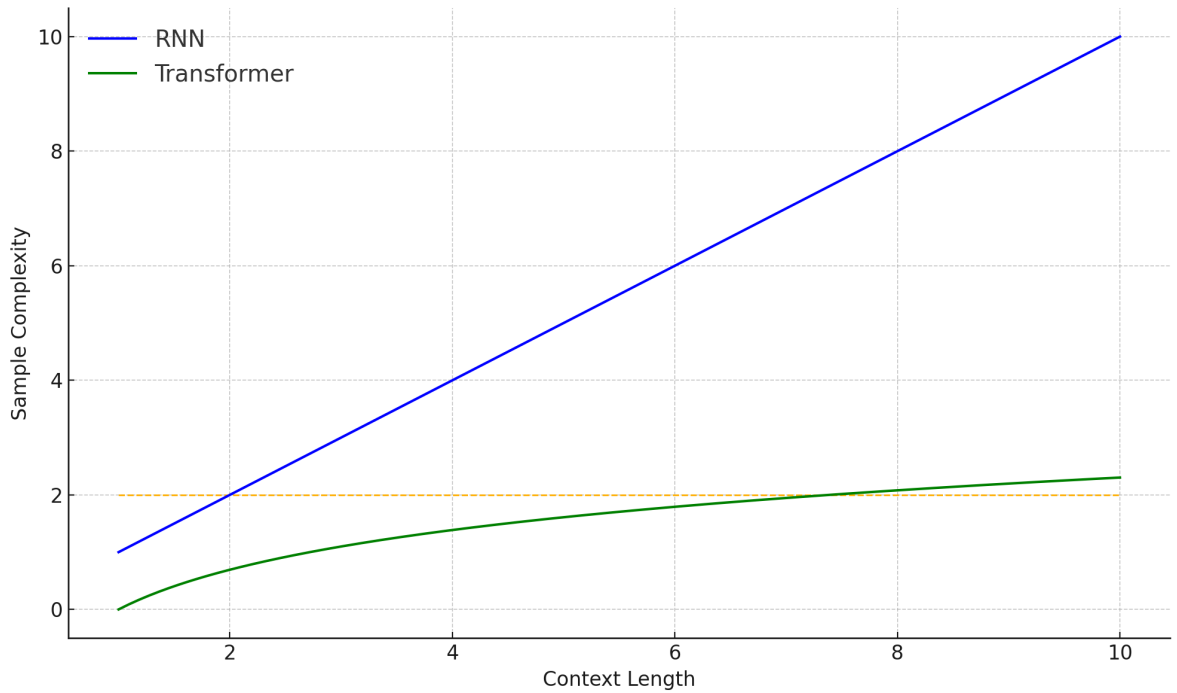
- Unlike Transformers and RNNs, there is no analysis of how SNNs model long-range dependencies
- The discontinuity in these networks makes many classical approaches unusable

Learning-theory research in SNN architectures lags behind ANNs; understanding SNNs is crucial to improving them

Unlike Transformers and RNNs, there is no analysis of how SNNs model long-range dependencies

The discontinuity in these networks makes many classical approaches unusable

RNN & Transformer Sample Complexity With Respect to Context Length



This work analyzes SNNs to understand how sample complexity varies with respect to context length in SNNs **illustrating how well these architectures model long-range dependencies**

Covering-number bounds can be used to measure how sample complexity of a class varies with input length

$$\log N\big(\varepsilon; F_d, \|\cdot\|_\infty\big) \;=\; \Theta\big(d\,\log(L/\varepsilon)\big).$$

$$\Downarrow$$

$$n(\varepsilon,\delta) \;=\; O\Big(\frac{d\,\log(L/\varepsilon)}{\varepsilon^2} \;+\; \frac{\log(1/\delta)}{\varepsilon^2}\Big).$$

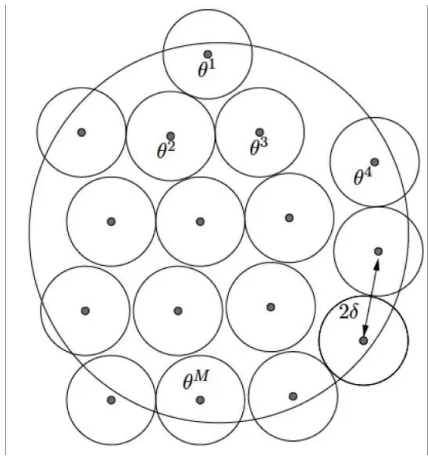
Covering-number bounds can be used to measure how sample complexity of a class varies with input length

$$\log N\big(\varepsilon; F_d, \|\cdot\|_\infty\big) \;=\; \Theta\big(d\,\log(L/\varepsilon)\big).$$

$$\Downarrow$$

$$n(\varepsilon,\delta) \;=\; O\Big(\frac{d\,\log(L/\varepsilon)}{\varepsilon^2} \;+\; \frac{\log(1/\delta)}{\varepsilon^2}\Big).$$

Covering a Function Class With ϵ balls



Covering-number arguments place a collection of small ϵ-balls (in the visual they use δ) so that every function lies within one of these ϵ balls

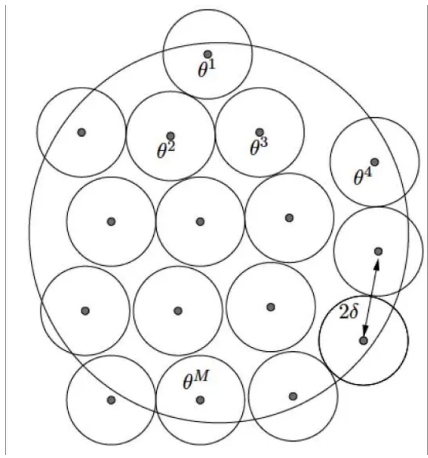
Covering-number bounds can be used to measure how sample complexity of a class varies with input length

$$\log N\big(\varepsilon; F_d, \|\cdot\|_\infty\big) \;=\; \Theta\big(d\,\log(L/\varepsilon)\big).$$

$$\Downarrow$$

$$n(\varepsilon,\delta) \;=\; O\Big(\frac{d\,\log(L/\varepsilon)}{\varepsilon^2} \;+\; \frac{\log(1/\delta)}{\varepsilon^2}\Big).$$

Covering a Function Class With ϵ balls



Covering-number arguments place a collection of small ϵ-balls (in the visual they use δ) so that every function lies within one of these ϵ balls

Analyzing an SNN via covering-number techniques reveals how effectively it learns increasingly long contexts.

Non Leaky-Integrate-&-Fire networks (nLIF)–a type of SNN–are **locally Lipschitz continuous** between pre & post synaptic layers

Important Terminology

Non Leaky-Integrate-&-Fire networks (nLIF)–a type of SNN–are **locally Lipschitz continuous** between pre & post synaptic layers

Important Terminology

Non Leaky-Integrate-&-Fire: This is a simplification of SNN neuronal dynamics where the built up weight in a neuron does not leak away over time

Non Leaky-Integrate-&-Fire networks (nLIF)–a type of SNN–are **locally Lipschitz continuous** between pre & post synaptic layers

Important Terminology

Non Leaky-Integrate-&-Fire: This is a simplification of SNN neuronal dynamics where the built up weight in a neuron does not leak away over time

Causal Set: The set of spike times and neurons denoted C_i^ℓ in layer $\ell - 1$ that contributed to neuron n_ℓ^i's spiking

Non Leaky-Integrate-&-Fire networks (nLIF)–a type of SNN–are **locally Lipschitz continuous** between pre & post synaptic layers

Important Terminology

Non Leaky-Integrate-&-Fire: This is a simplification of SNN neuronal dynamics where the built up weight in a neuron does not leak away over time

Causal Set: The set of spike times and neurons denoted C_i^ℓ in layer $\ell - 1$ that contributed to neuron n_ℓ^i's spiking

$$\left\| t_i^{(\ell)}(a) \; - \; t_i^{(\ell)}(b) \right\|_{L_\infty\left(P[C_i^{(\ell)}]\right)} \; \leq \; 2\,|C_i^{(\ell)}|\, \max\!\left(\frac{\bar{W}}{\delta},\, \frac{\tau_s}{\delta}\right) \left\| a - b \right\|_{L_\infty\left(P[C_i^{(\ell)}]\right)}$$

For an SNN, local Lipschitz continuity in spike times is **equivalent** to local Lipschitz continuity of the function measured by those spike times (Dold et al.)

Non Leaky-Integrate-&-Fire networks (nLIF)–a type of SNN–are **locally Lipschitz continuous** between pre & post synaptic layers

Important Terminology

Non Leaky-Integrate-&-Fire: This is a simplification of SNN neuronal dynamics where the built up weight in a neuron does not leak away over time

Causal Set: The set of spike times and neurons denoted C_i^ℓ in layer $\ell - 1$ that contributed to neuron n_ℓ^i's spiking

$$\left\| t_i^{(\ell)}(a) \; - \; t_i^{(\ell)}(b) \right\|_{L_\infty\left(P[C_i^{(\ell)}]\right)} \; \leq \; 2\,|C_i^{(\ell)}|\,\max\!\left(\frac{\bar{W}}{\delta},\,\frac{\tau_s}{\delta}\right)\,\left\| a - b \right\|_{L_\infty\left(P[C_i^{(\ell)}]\right)}$$

For an SNN, local Lipschitz continuity in spike times is **equivalent** to local Lipschitz continuity of the function measured by those spike times (Dold et al.)

Importantly this equation expresses a relationship between Lipschitz continuity and spike time integration

Problem Motivation

If we can extend the local Lipschitz continuity between pre and post synpatic layers to be globally Lipschitz across the entire network, then we can say something about the relationship between spiking dynamics and long-range dependencies

A **globally Lipschitz SNN** can be constructed and illustrates how the covering number changes as a function of the spiking dynamics

Architecture Assumptions

Let

A **globally Lipschitz SNN** can be constructed and illustrates how the covering number changes as a function of the spiking dynamics

Architecture Assumptions

Let

1. the network be a feedforward nLIF SNN

A **globally Lipschitz SNN** can be constructed and illustrates how the covering number changes as a function of the spiking dynamics

Architecture Assumptions

Let

1. the network be a feedforward nLIF SNN
2. M be a set of training examples s.t.

$$\forall\, n \in \mathrm{nLIF},\ \exists\, m_i \in M :\ F(m_i) \implies \text{neuron } n \text{ spikes}$$

A **globally Lipschitz SNN** can be constructed and illustrates how the covering number changes as a function of the spiking dynamics

Architecture Assumptions

Let

1. the network be a feedforward nLIF SNN
2. M be a set of training examples s.t.

$$\forall\, n \in \mathrm{nLIF},\ \exists\, m_i \in M :\ F(m_i) \implies \text{neuron } n \text{ spikes}$$

3. all synaptic weights w_{ij}^{ℓ} are positive and, for every causal piece P,

$$\sum_{(i,j)\in P} w_{ij}^{L} \ - \ \vartheta \ > \ \delta > 0.$$

A **globally Lipschitz SNN** can be constructed and illustrates how the covering number changes as a function of the spiking dynamics

Architecture Assumptions

Let

1. the network be a feedforward nLIF SNN
2. M be a set of training examples s.t.

$$\forall\, n \in \mathrm{nLIF},\ \exists\, m_i \in M :\ F(m_i) \implies \text{neuron } n \text{ spikes}$$

3. all synaptic weights w_{ij}^{ℓ} are positive and, for every causal piece P,

$$\sum_{(i,j)\in P} w_{ij}^{L} \ - \ \vartheta \ > \ \delta > 0.$$

4. the reset for a neuron's membrane after spiking is instantaneous

Theorem 1

Theorem (Global Lipschitz Continous)

Under Assumptions, there exists a constant L_{global} *s.t. for any two input-spike patterns* a, b *with* $|a - b|_{L_\infty} \leq \gamma$,

$$\left| T_{\text{output}}(a) - T_{\text{output}}(b) \right|_{L_\infty} \leq L_{\text{global}} \, |a - b|_{L_\infty}$$

Proof Sketch: Global Lipschitz Bound for nLIF SNN

Consider the simple case where each input neuron receives at most a single input spike

Proof.

Proof Sketch: Global Lipschitz Bound for nLIF SNN

Consider the simple case where each input neuron receives at most a single input spike

Proof.

1. Each neuron n_{out}^{i} in the output layer is locally Lipschitz continuous with the set of pre synaptic neurons that contributed to its spiking

Proof Sketch: Global Lipschitz Bound for nLIF SNN

Consider the simple case where each input neuron receives at most a single input spike

Proof.

1. Each neuron n_{out}^{i} in the output layer is locally Lipschitz continuous with the set of pre synaptic neurons that contributed to its spiking
2. $\exists$ a δ margin across all the thresholds and the causal set of each of these output neurons is non empty we can construct a Layer Lipschitz bound across this layer by taking the

$$L_{\text{outlayer}} = 2 * \max_{i}\Big(\Big|C_i^{\ell}\Big| \max\Big(\frac{\bar{W}}{\delta}, \frac{\tau_s}{\delta}\Big)\Big).$$

Proof Sketch: Global Lipschitz Bound for nLIF SNN

Consider the simple case where each input neuron receives at most a single input spike

Proof.

1. Each neuron n_{out}^{i} in the output layer is locally Lipschitz continuous with the set of pre synaptic neurons that contributed to its spiking
2. $\exists$ a δ margin across all the thresholds and the causal set of each of these output neurons is non empty we can construct a Layer Lipschitz bound across this layer by taking the

$$L_{\text{outlayer}} = 2 * \max_{i}\Big(\Big|C_i^{\ell}\Big| \ \max\Big(\frac{\bar{W}}{\delta}, \frac{\tau_s}{\delta}\Big)\Big).$$

3. any presynaptic inputs a,b $\in P(C_i^{output})$ s.t $|a-b|_{L_\infty} \leq \delta$ are locally Lipschitz continuous for neuron n_i^{ℓ} s.t.

$$|T_i^{\ell}(a) - T_i^{\ell}(b)|_{L_\infty} \leq L_{outlayer}|a-b|_{L_\infty}$$

This can be traced recursively through the layers towards the input by telescoping and taking the max for each layer s.t.

$$L_{\text{Global}} = \prod_{\ell=1}^{D} \max_{i}\Big(\Big|C_i^{\ell}\Big| \ \max\Big(\frac{\bar{W}}{\delta}, \frac{\tau_s}{\delta}\Big)\Big). \text{ covers the entire network}$$

Proof Sketch: Global Lipschitz Bound for nLIF SNN

Consider the simple case where each input neuron receives at most a single input spike

Proof.

1. Each neuron n_{out}^{i} in the output layer is locally Lipschitz continuous with the set of pre synaptic neurons that contributed to its spiking
2. $\exists$ a δ margin across all the thresholds and the causal set of each of these output neurons is non empty we can construct a Layer Lipschitz bound across this layer by taking the

$$L_{\text{outlayer}} = 2 * \max_{i}\Big(\Big|C_i^{\ell}\Big| \ \max\Big(\frac{\bar{W}}{\delta}, \frac{\tau_s}{\delta}\Big)\Big).$$

3. any presynaptic inputs a,b $\in P(C_i^{output})$ s.t $|a-b|_{L_\infty} \leq \delta$ are locally Lipschitz continuous for neuron n_i^{ℓ} s.t.

$$|T_i^{\ell}(a) - T_i^{\ell}(b)|_{L_\infty} \leq L_{outlayer}|a-b|_{L_\infty}$$

This can be traced recursively through the layers towards the input by telescoping and taking the max for each layer s.t.

$$L_{\text{Global}} = \prod_{\ell=1}^{D} \max_{i}\Big(\Big|C_i^{\ell}\Big| \ \max\Big(\frac{\bar{W}}{\delta}, \frac{\tau_s}{\delta}\Big)\Big). \text{ covers the entire network}$$

4. Since δ is a free term, it can be arbitrarily defined s.t. any γ space of interest is covered by a global lipschitz constant

Growth in the context length T increases the global Lipschitz constant **exponentially** causing a $O(T^2)$ growth in **sample complexity**

Growth in the context length T increases the global Lipschitz constant **exponentially** causing a $O(T^2)$ growth in **sample complexity**

Theorem (Global Lipschitz constant for long-range spike trains)

Under the assumptions, there exists a constant L_{globalS} s.t. for any two multi-spike input patterns a, b of length T,

$$\left| T_{\mathrm{output}}(a) - T_{\mathrm{output}}(b) \right|_{L_\infty} \leq L_{\mathrm{globalS}} \, |a - b|_{L_\infty}$$

Here, L_{globalT} is the global Lipschitz constant for the sequence-based encoding of length $|S|$, and it satisfies

$$L_{\mathrm{globalT}} = O(L_{\mathrm{global}}^{|T|})$$

where L_{global} is the global Lipschitz constant established for a context length of 1.

Growth in the context length T increases the global Lipschitz constant **exponentially** causing a $O(T^2)$ growth in **sample complexity**

Theorem (Global Lipschitz constant for long-range spike trains)

Under the assumptions, there exists a constant L_{globalS} s.t. for any two multi-spike input patterns a, b of length T,

$$\left| T_{\text{output}}(a) - T_{\text{output}}(b) \right|_{L_\infty} \leq L_{\text{globalS}} \, |a - b|_{L_\infty}$$

Here, L_{globalT} is the global Lipschitz constant for the sequence-based encoding of length $|S|$, and it satisfies

$$L_{\text{globalT}} = O(L_{\text{global}}^{|T|})$$

where L_{global} is the global Lipschitz constant established for a context length of 1.

Recall this relationship between sample complexity and Lipschitz constant

$$n(\varepsilon, \delta) = O\Big(\frac{d \, \log(L/\varepsilon)}{\varepsilon^2} + \frac{\log(1/\delta)}{\varepsilon^2}\Big).$$

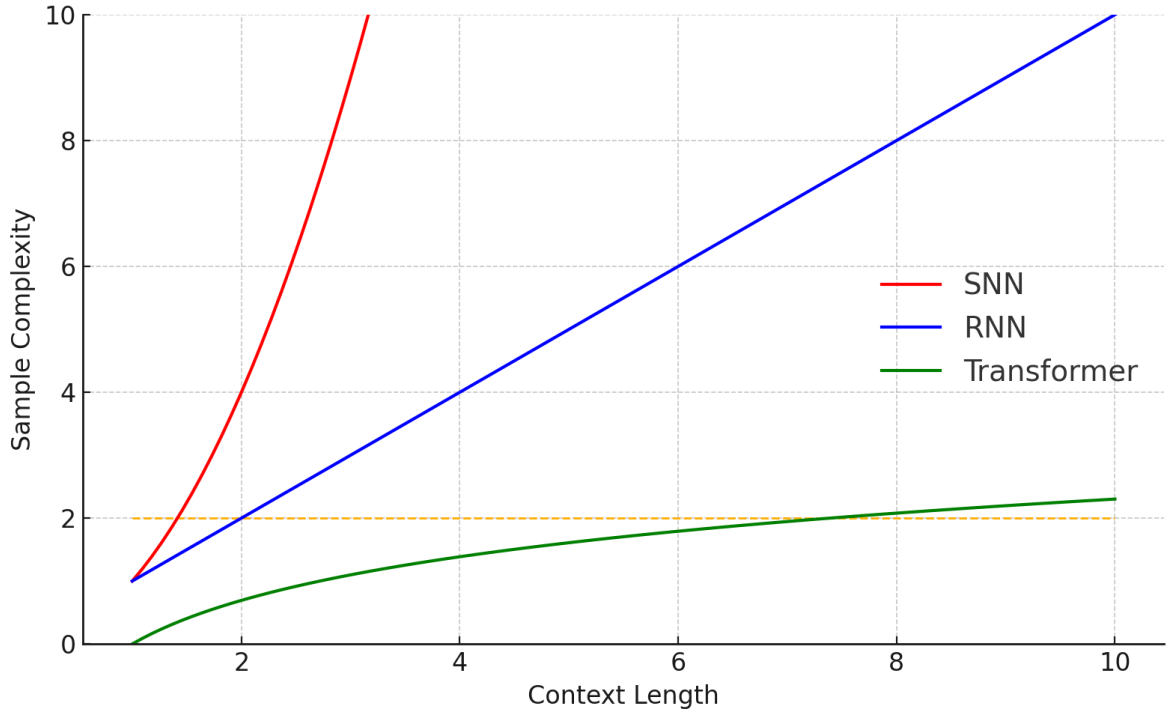
Thus the sample complexity for this problem can be rewritten as

$$M = O(D * |T| \log(L_{Global}^{|T|}) + \frac{\log(1/\delta)}{\varepsilon^2}\Big)$$

$$M \approx O(D * |T|^2 \log(L_{Global})$$

Growth in the context length T increases the global Lipschitz constant **exponentially** causing a $O(T^2)$ growth in **sample complexity**

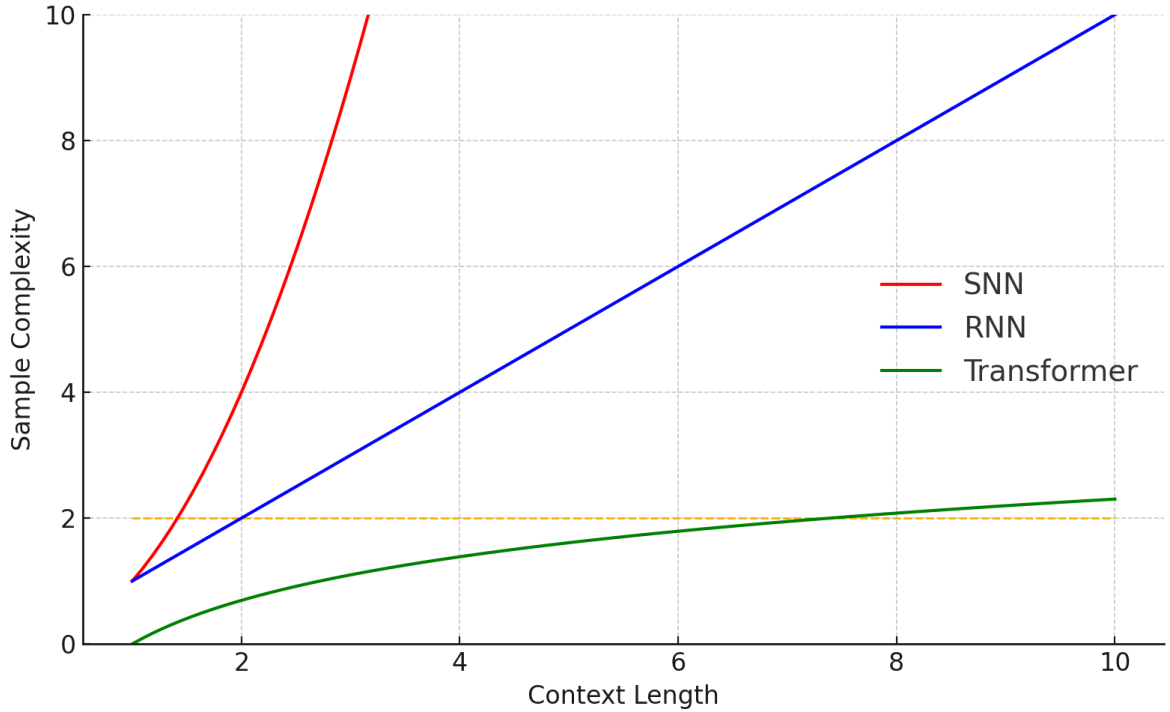
Visualization of the impact of the difference in sample complexity of SNNs VS. RNNs VS. Transformers



- The driver of quadratic sample complexity is the uncontrolled excitation in neurons in the nLIF architecture

Growth in the context length T increases the global Lipschitz constant **exponentially** causing a $O(T^2)$ growth in **sample complexity**

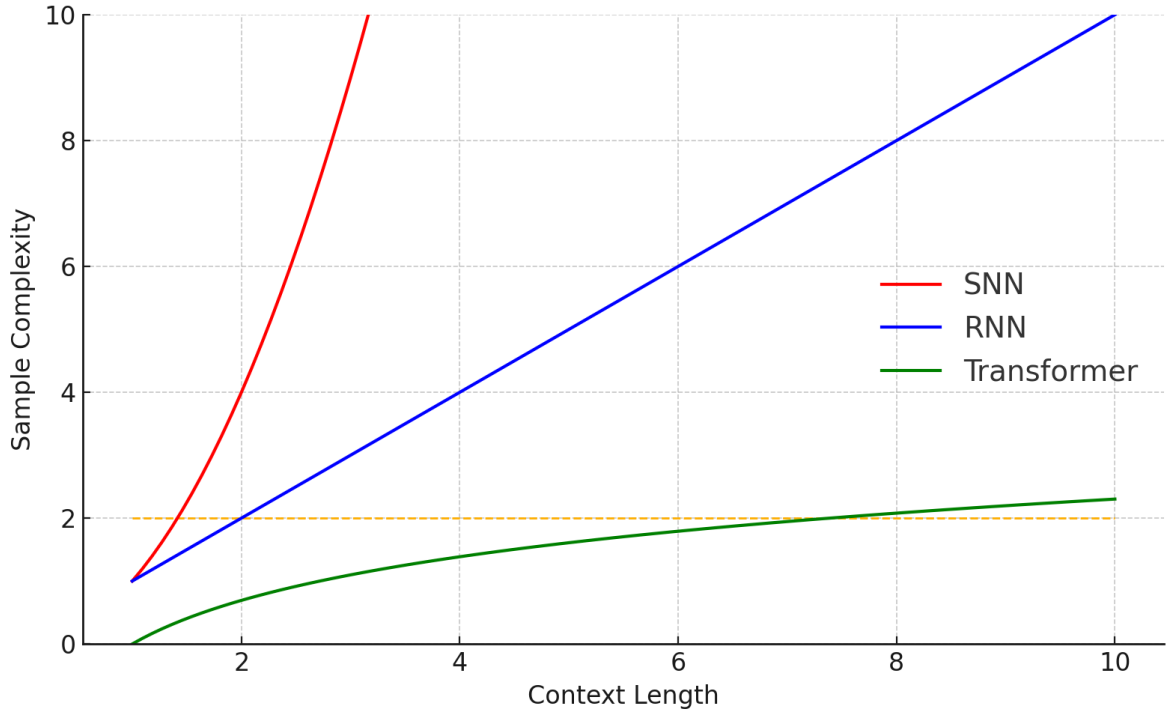
Visualization of the impact of the difference in sample complexity of SNNs VS. RNNs VS. Transformers



- The driver of quadratic sample complexity is the uncontrolled excitation in neurons in the nLIF architecture
- This excitation is a problem in all SNNs not only our idealized SNN

Growth in the context length T increases the global Lipschitz constant **exponentially** causing a $O(T^2)$ growth in **sample complexity**

Visualization of the impact of the difference in sample complexity of SNNs VS. RNNs VS. Transformers



- The driver of quadratic sample complexity is the uncontrolled excitation in neurons in the nLIF architecture
- This excitation is a problem in all SNNs not only our idealized SNN
- Specialized models which try and incorporate attention or other sparsity inducing measures cannot solve this fundemental problem of sequential integration in SNNs

Unbounded excitation in spiking neurons is the driver of poor sample complexity bounds in learning from long context windows

Unbounded excitation in spiking neurons is the driver of poor sample complexity bounds in learning from long context windows

Lateral Inhibitory processes in real neurons are used to bound endless excitation

Unbounded excitation in spiking neurons is the driver of poor sample complexity bounds in learning from long context windows

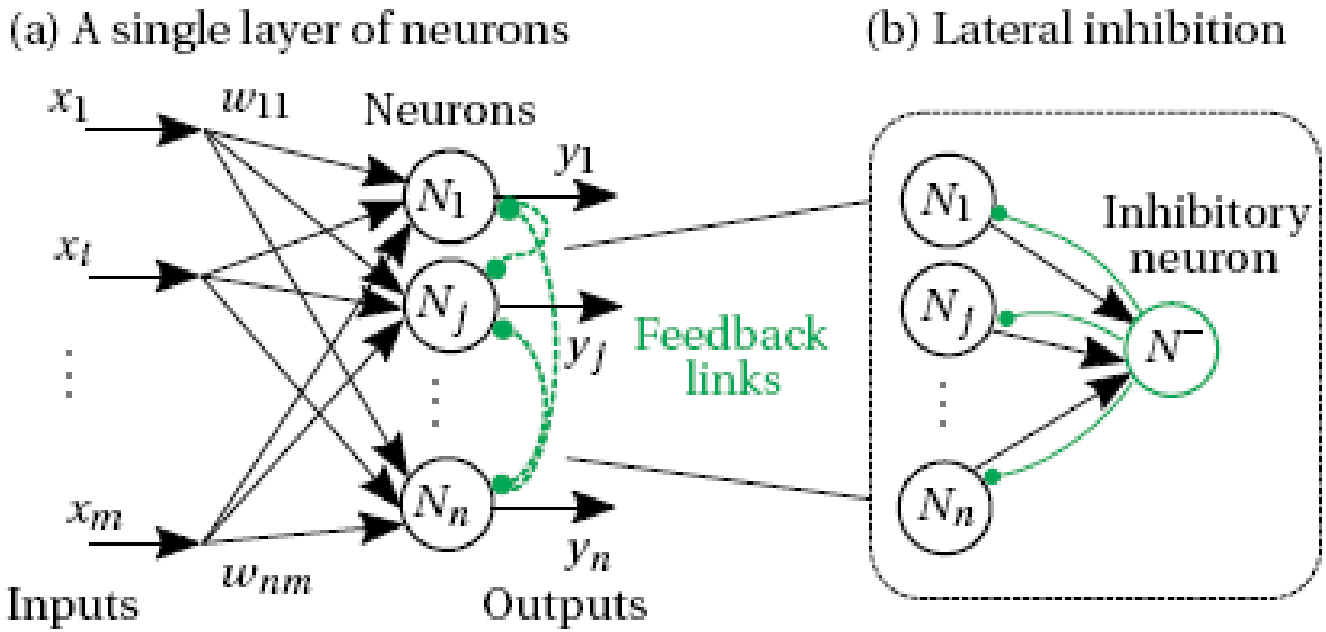
Lateral Inhibitory processes in real neurons are used to bound endless excitation

Due to its inducing of sparse networks we believe this process is crucial to how biological systems models long range dependencies

Unbounded excitation in spiking neurons is the driver of poor sample complexity bounds in learning from long context windows

Lateral Inhibitory processes in real neurons are used to bound endless excitation

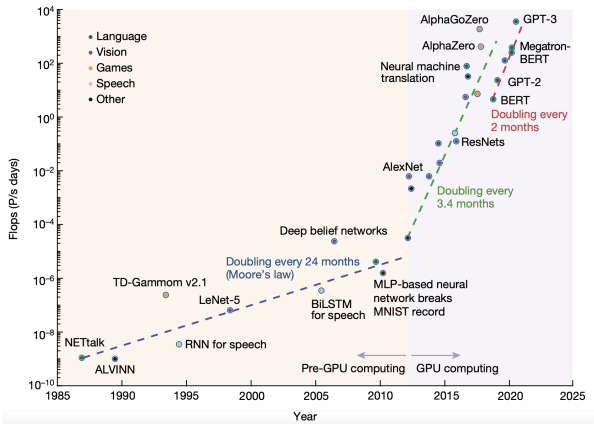
Due to its inducing of sparse networks we believe this process is crucial to how biological systems models long range dependencies



1. When a neuron spikes it sends inhibitory signals to other neurons in the same layer
2. Inhibitory layers can be used to bound the causal set of the neurons in a network, and thus bound the global Lipchitz constant

Next Steps & Discussion

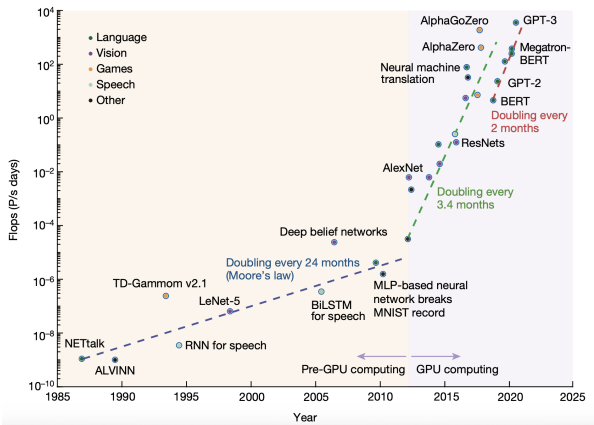
Energy Consumption of ML Models



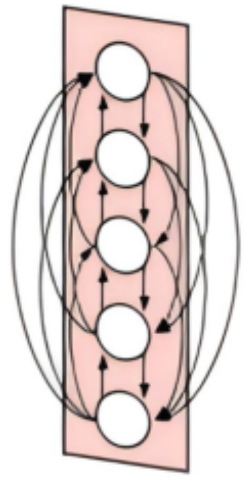
- Wider adoption of neuromorphic hardware is necessary as energy use becomes a bottleneck in using machine learning models
- SNNs are key to unlocking neuromorphic potential

Next Steps & Discussion

Energy Consumption of ML Models



Lateral Inhibition Layer Design



- Wider adoption of neuromorphic hardware is necessary as energy use becomes a bottleneck in using machine learning models
- SNNs are key to unlocking neuromorphic potential
- Lateral inhibitory processes have been used in SNNs to achieve higher accuracy on classic computer vision problems (Liu et al.).
- Building off this work, we aim to build inhibitory layers for sequence modeling tasks to test whether this is an effective way of improving Long Range dependencies