Theoretical Limitations of Self-Attention in Neural Sequence Models

Michael Hahn
Stanford University
ACL 2020

The Stanford Natural Language Processing Group

Transformers are at the heart of the state-of-the-art in NLP.

What is their computational power?

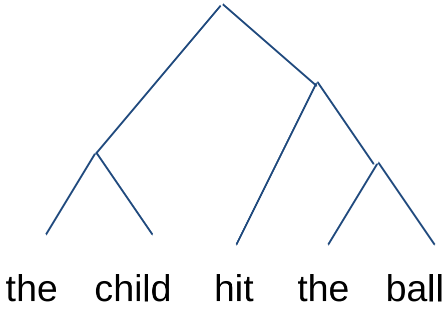
Understanding this might help us

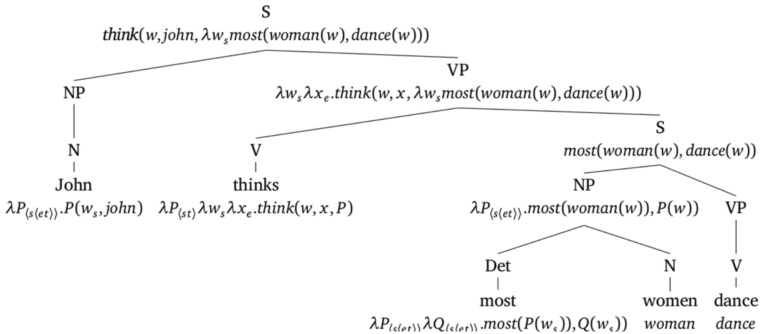
- Design better models
- Learn something about the nature of language

What is the computational power of transformers?

Can they model hierarchical structure?

Thought to be essential to modeling syntax and meaning of natural language.





What is the computational power of transformers?

I’ll investigate this question theoretically.

Can transformers theoretically model hierarchical structure with unbounded recursion?

1. Can they correctly close brackets?
2. Can they evaluate iterated negation?

PARITY

Set of bit strings with an **even number of 1s**

- Extremely simple regular language, recognized by automaton with two states
- Prerequisite to evaluating Boolean formulas
- RNNs, LSTMs, GRUs,... can do this (at least in theory)
- If transformers can't model this, they can't model any non-quasi-aperiodic language

✓	✗
11	1
101	01
0000101011	000001011

DYCK_2

- **Correctly bracketed** words over (, [,],)
- All context-free languages arise from some DYCK_n through intersection with regular language + letter substitution (Chomsky and Schutzenberger, 1963)
- LSTMs are capable of solving this perfectly, at least in theory, using infinite precision

✓	✗
()	(]
([])()	((((
[[()]()]	(()]

What is the computational power of transformers?

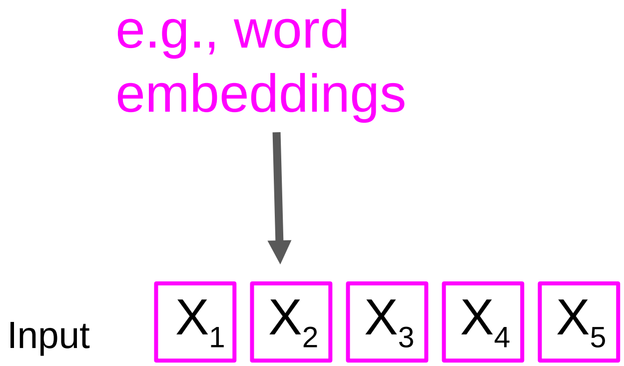
I'll investigate this question theoretically.

Can transformers theoretically model hierarchical structure with unbounded recursion?

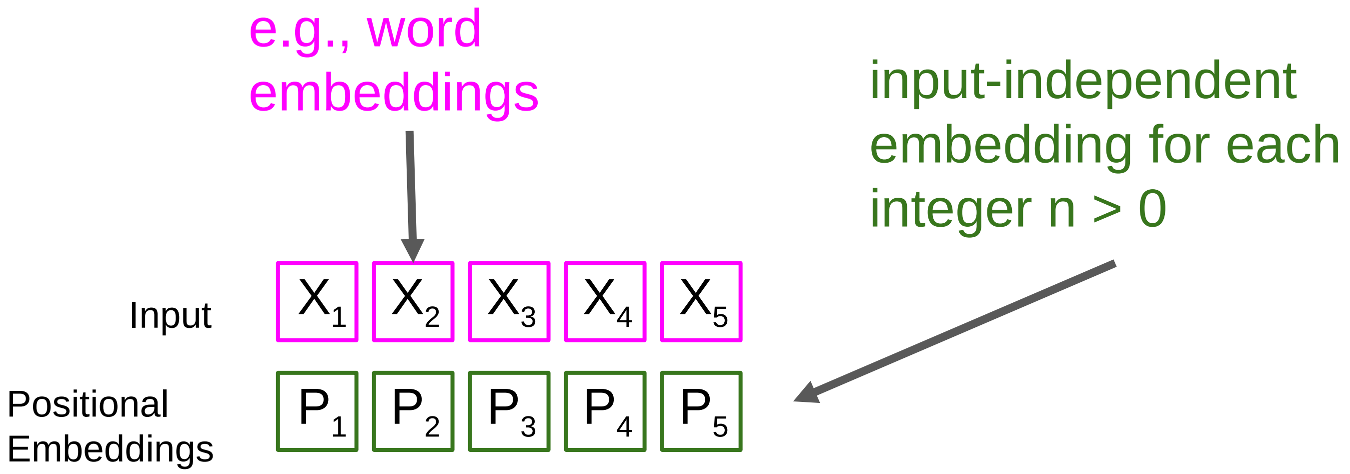
1. Can they correctly close brackets? -- Dyck_2
2. Can they evaluate iterated negation? -- Parity

Can a transformer correctly classify inputs as (not) belonging to these languages?

Transformer Architecture (Vaswani et al., 2017)

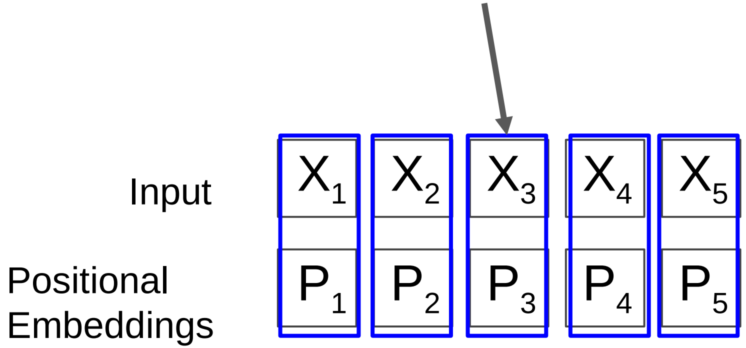


Transformer Architecture (Vaswani et al., 2017)

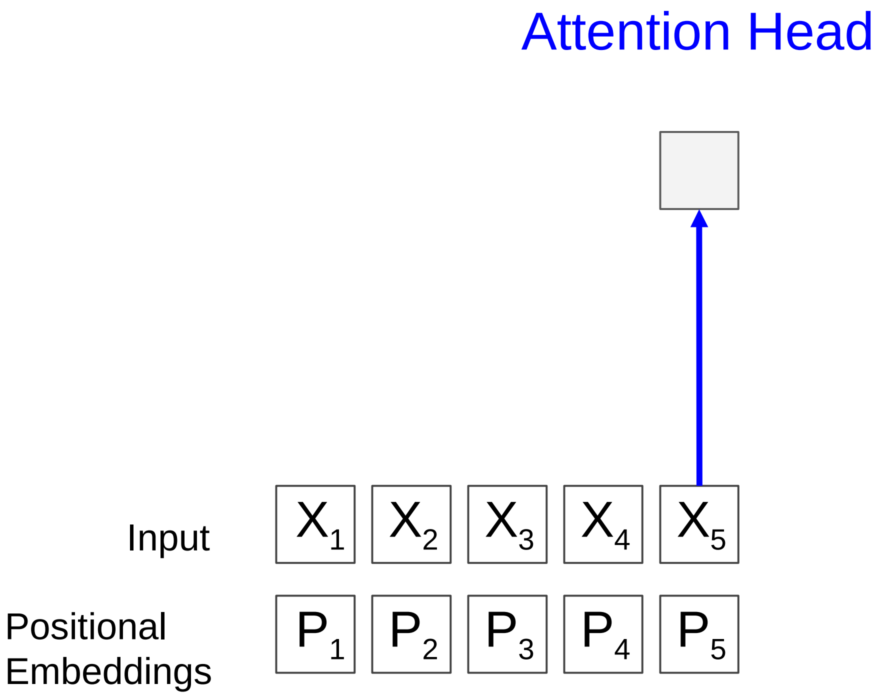


Transformer Architecture (Vaswani et al., 2017)

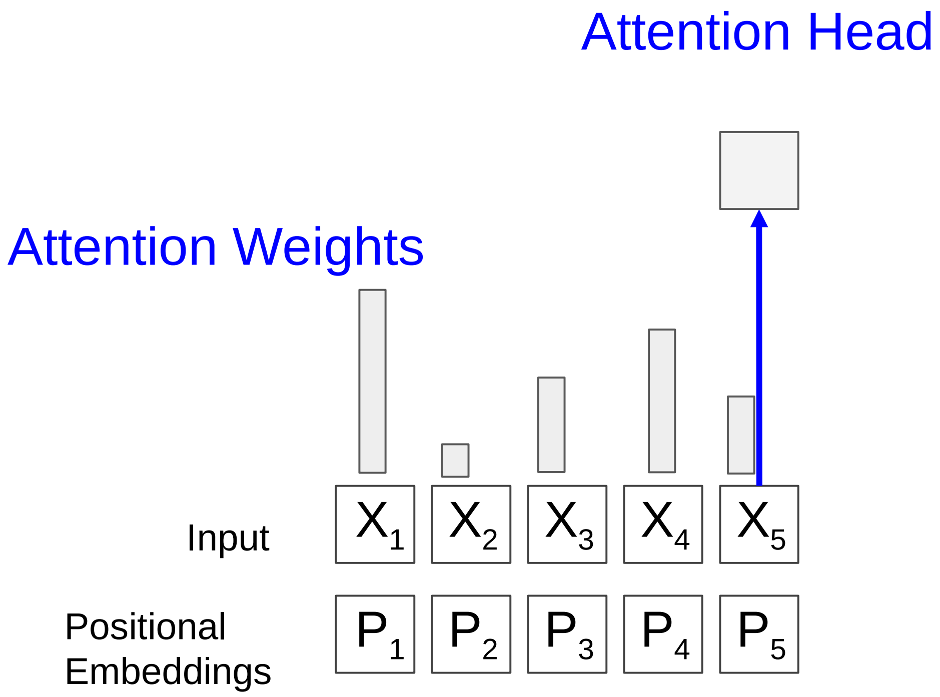
e.g., Addition or Concatenation



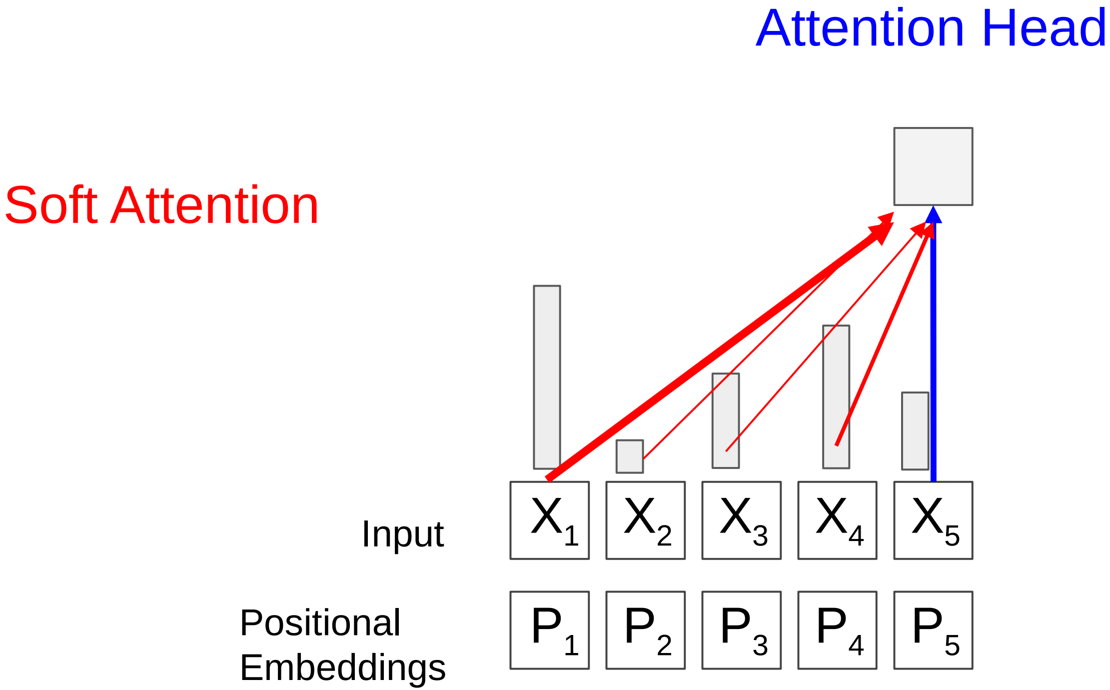
Transformer Architecture (Vaswani et al., 2017)



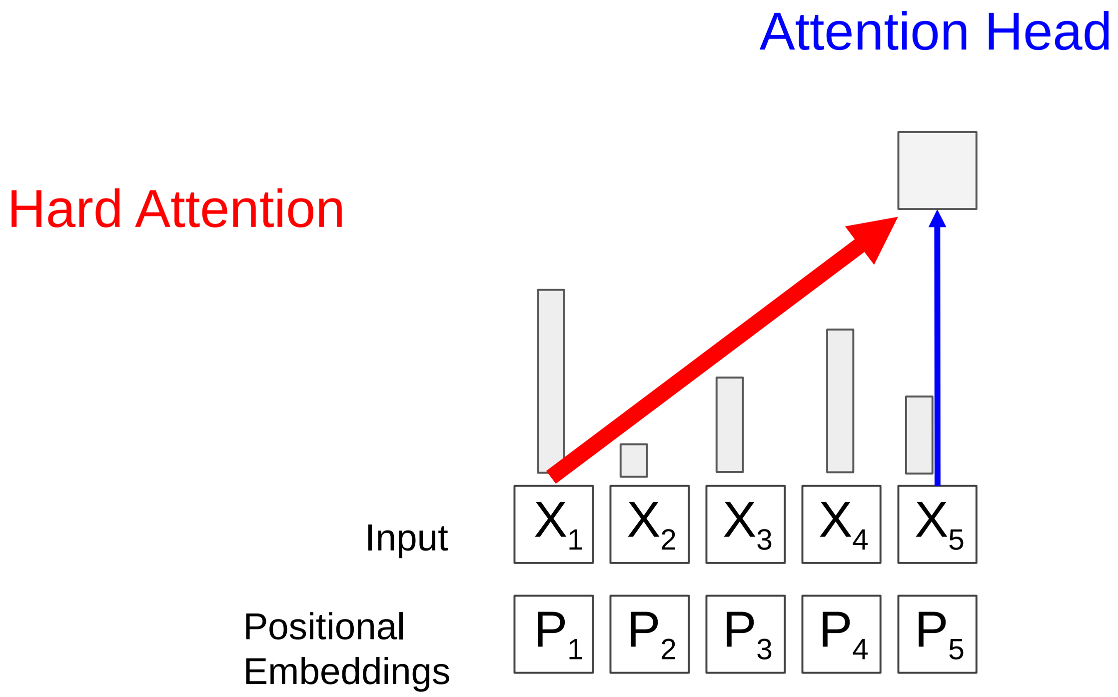
Transformer Architecture (Vaswani et al., 2017)



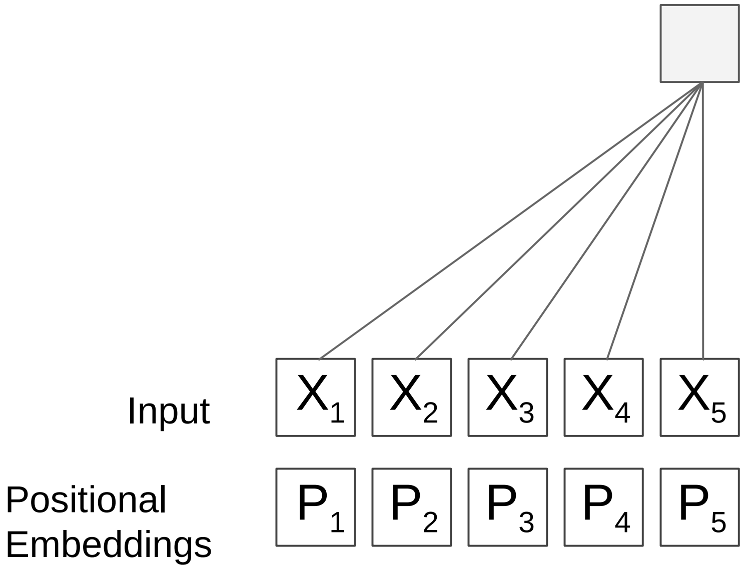
Transformer Architecture (Vaswani et al., 2017)



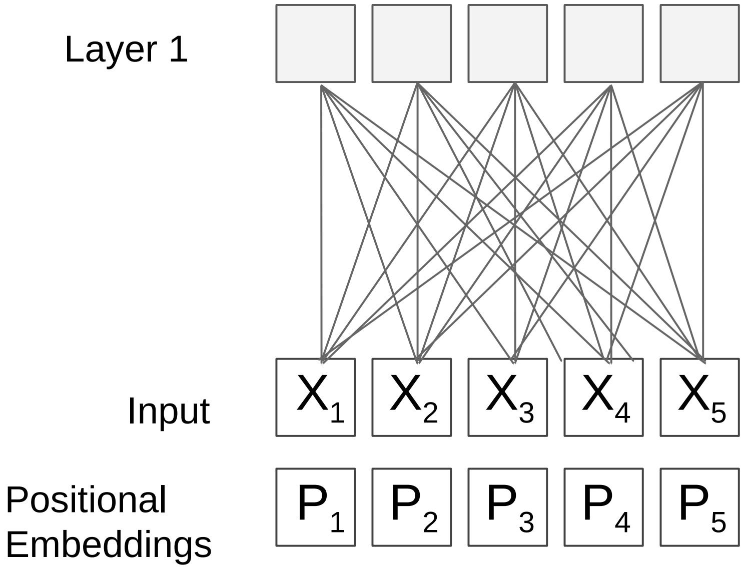
Transformer Architecture (Vaswani et al., 2017)



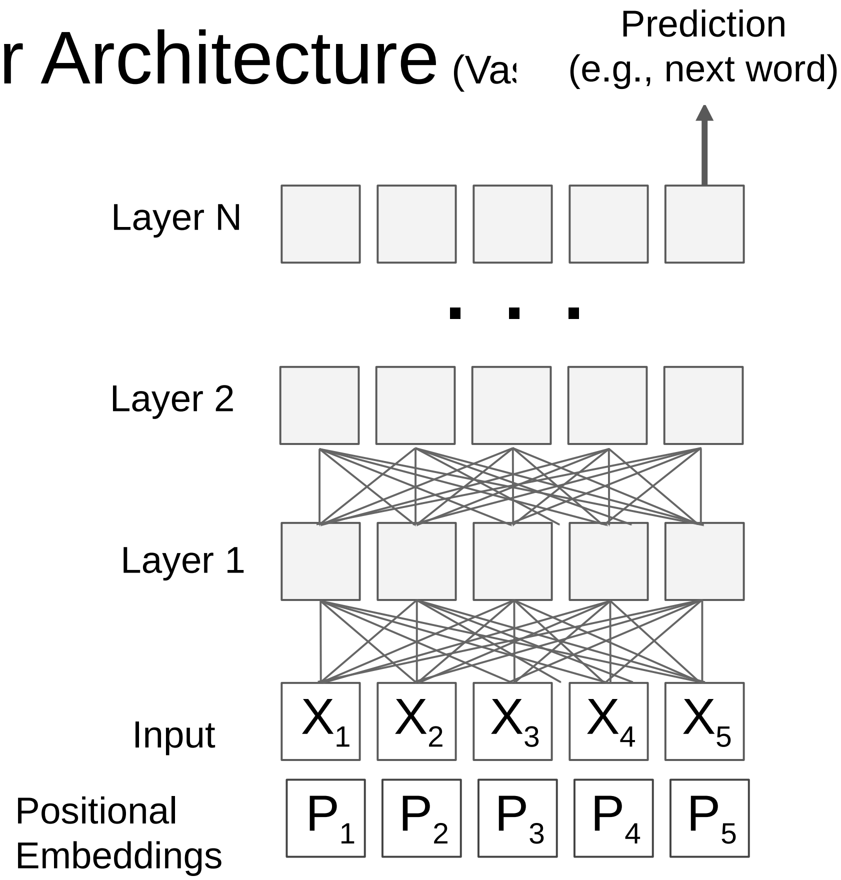
Transformer Architecture (Vaswani et al., 2017)



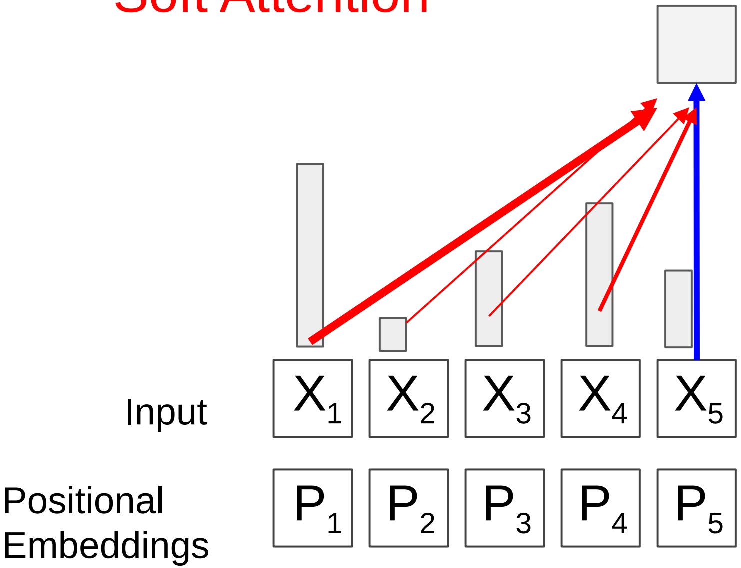
Transformer Architecture (Vaswani et al., 2017)



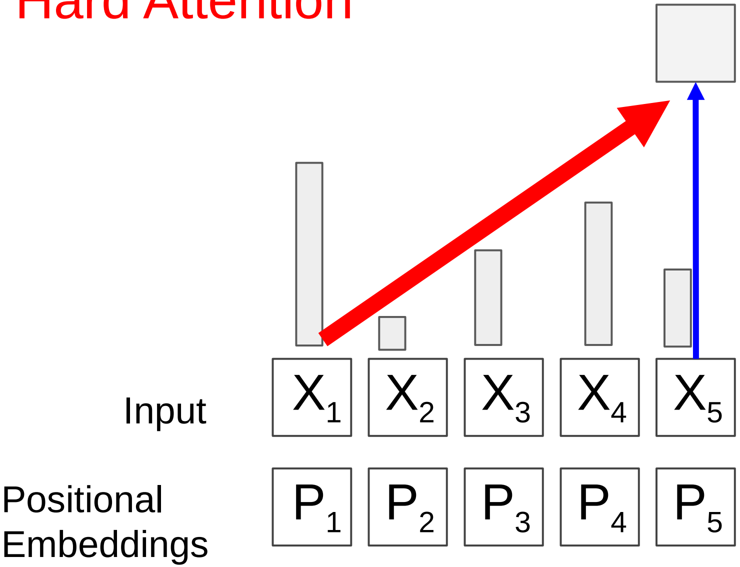
Transformer Architecture (Vas



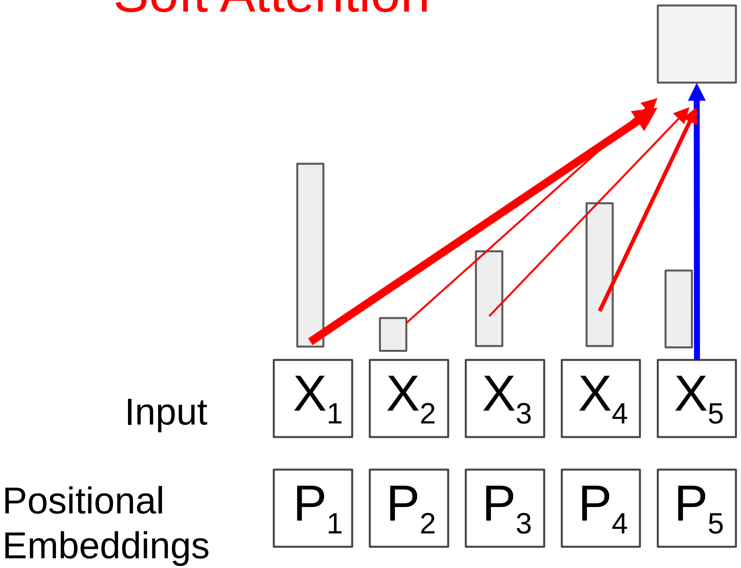
Soft Attention



Hard Attention

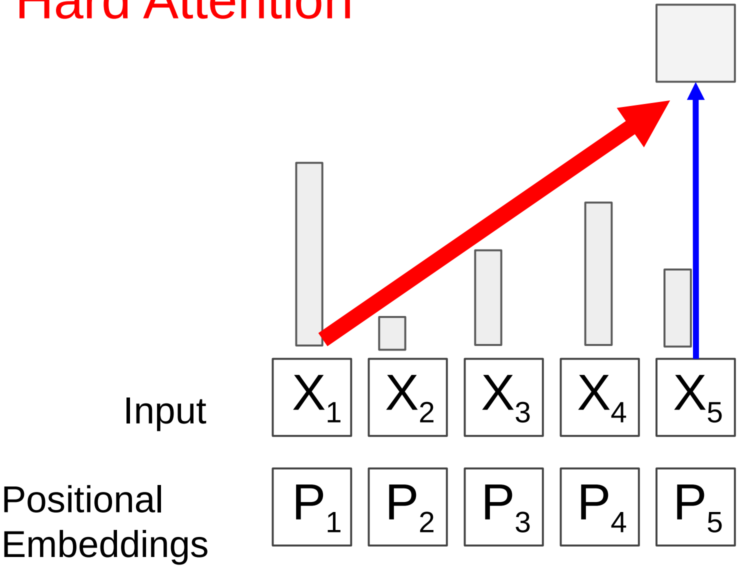


Soft Attention



- standard choice in practice
- easier to train with SGD

Hard Attention



- less easy to train from scratch (Shen et al 2018)
- But, heads in trained NLP models tend to concentrate their attention on few positions
  (Voita et al., 2019; Clark et al., 2019)
  ⇒ may be a reasonable theoretical model

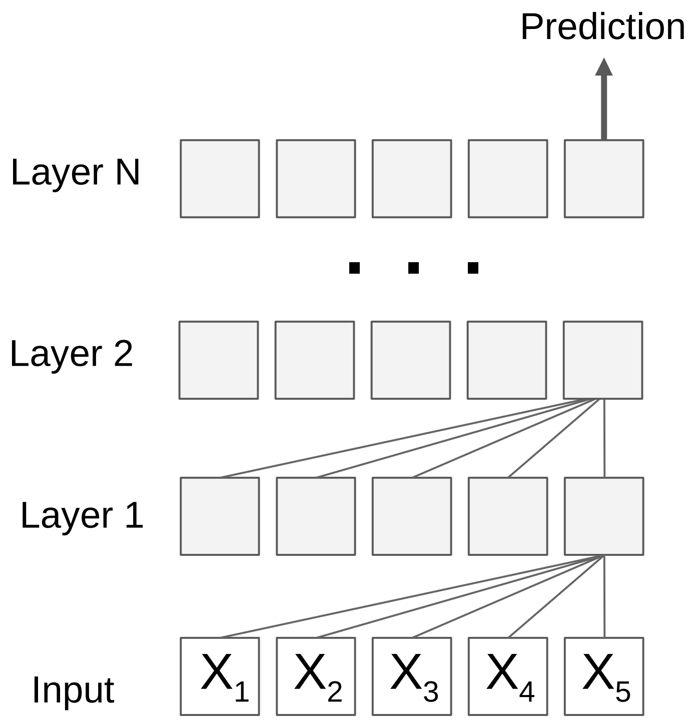
Part 1: Hard Attention

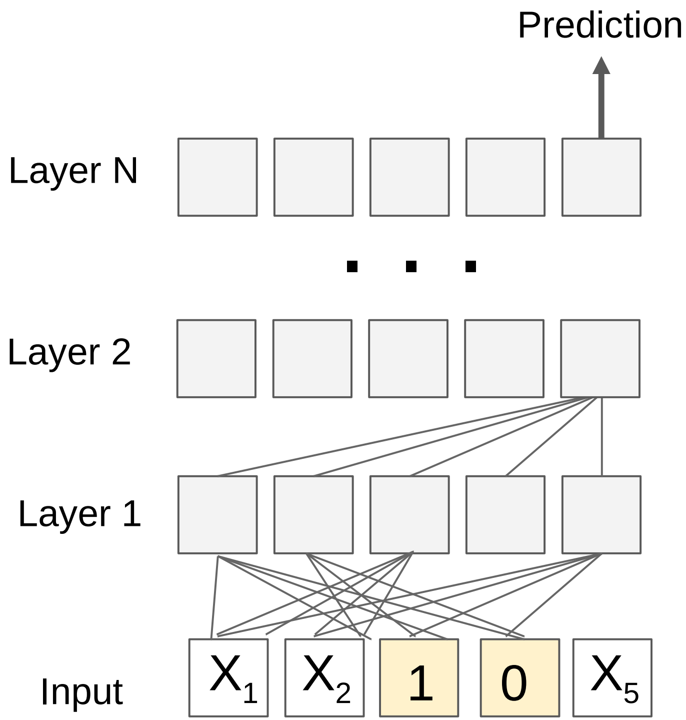
Proof Idea:

Assume we have a candidate transformer given.

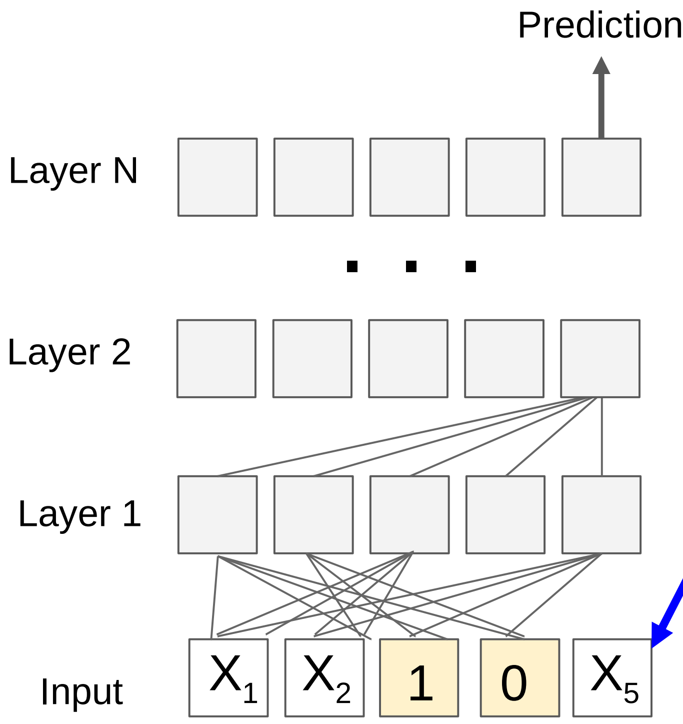
We construct a pair of inputs that are classified the same, even though one is in Parity and the other is not (same for Dyck_2).

Method: We fix a few input bits to ‘distract’ the transformer, so that it ignores most input bits.


Prediction
Layer N
Layer 2
Layer 1
Input
X_1
X_2
X_3
X_4
X_5



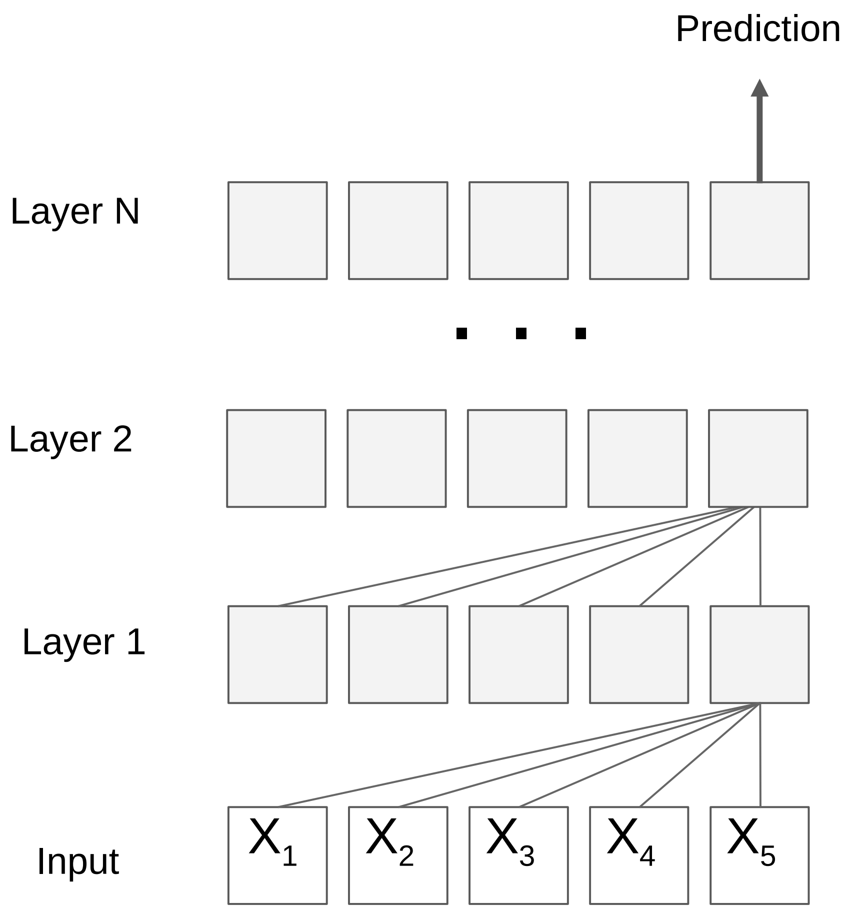
Idea: Some input bits will be ignored, since their attention weights are always smaller than those of the fixed bits.

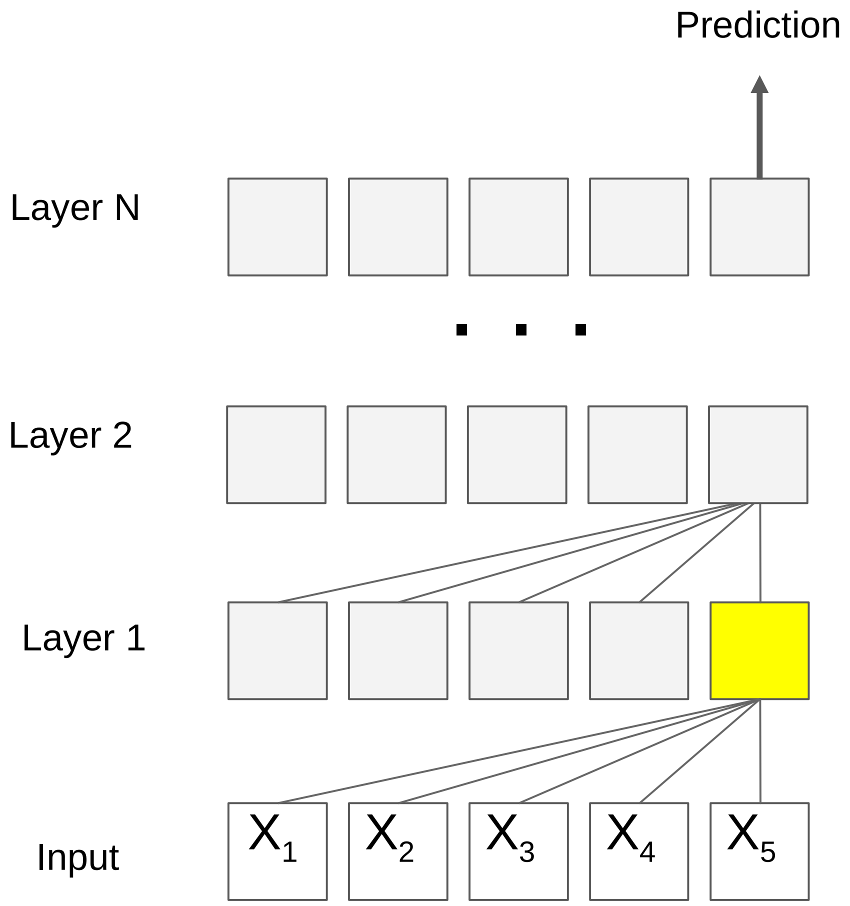


The entire network ignores this bit!

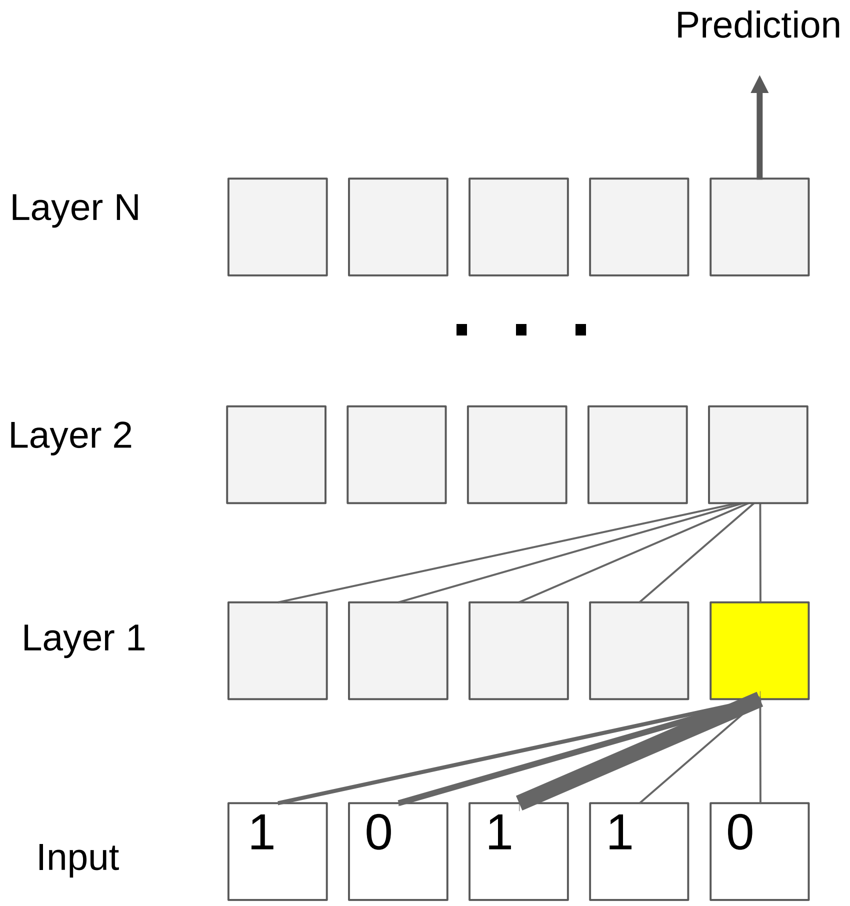
Consequence: It could not have modeled PARITY, since every bit matters for PARITY.

(Similar proof works for DYCK_2)

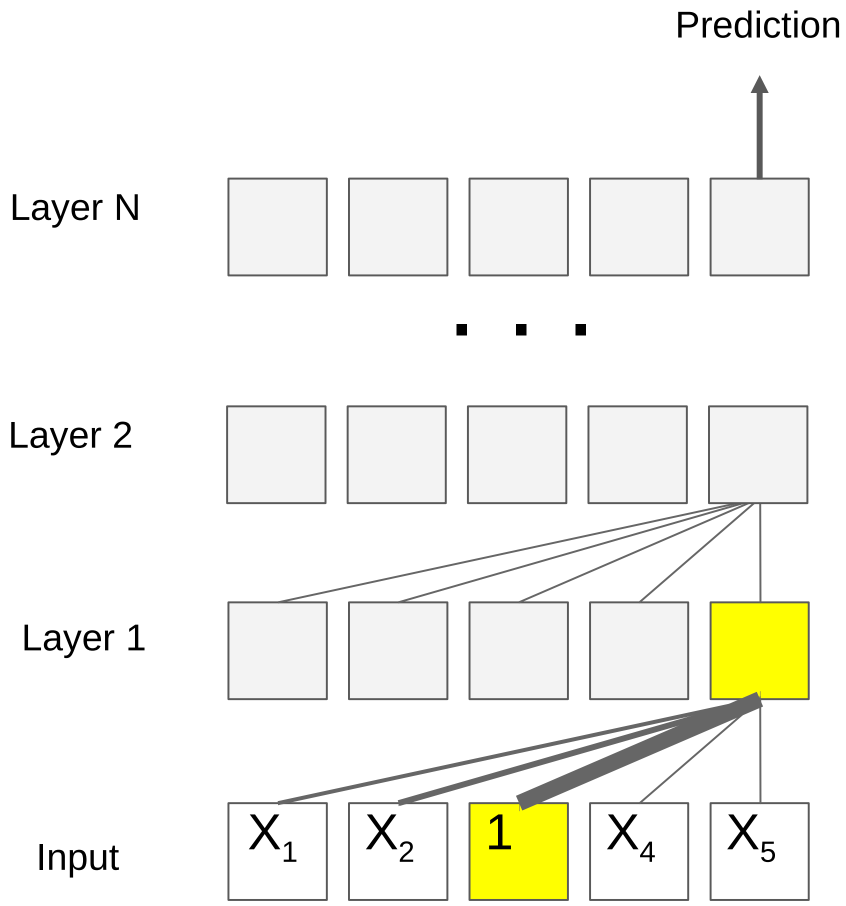

Prediction
Layer N
Layer 2
Layer 1
Input
X1
X2
X3
X4
X5



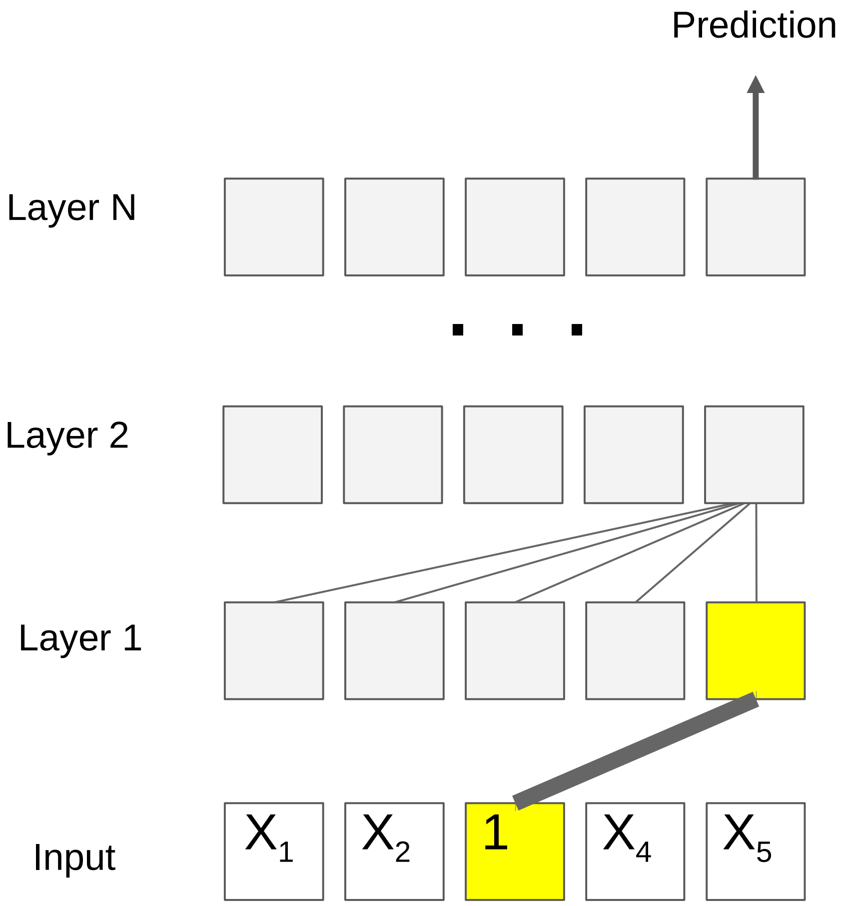
Prediction
Layer N
. . .
Layer 2
Layer 1
Input
X_1
X_2
X_3
X_4
X_5



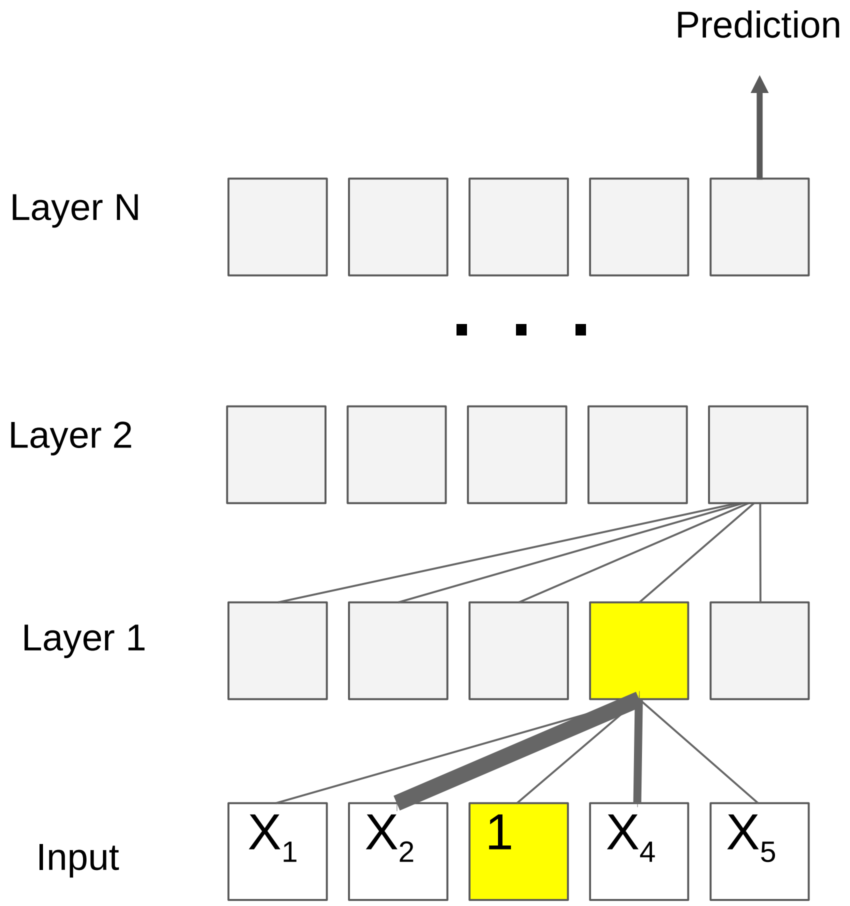
For each input bit, imagine the highest possible attention value.



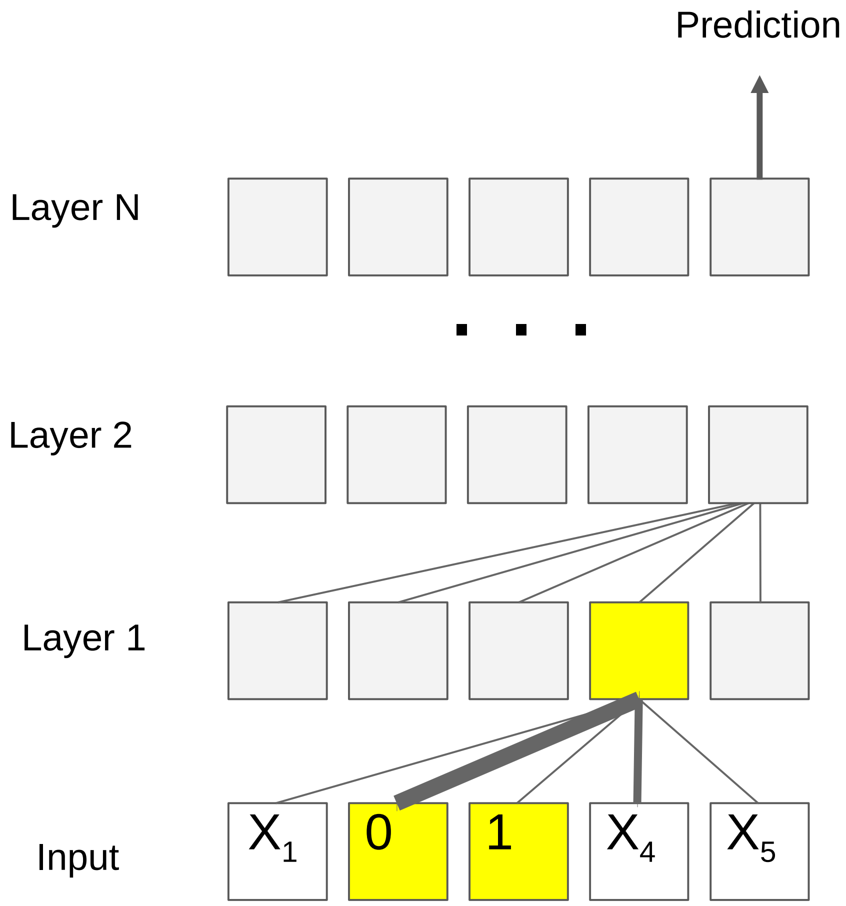
By fixing one input, we can make the head ignore all remaining input bits.



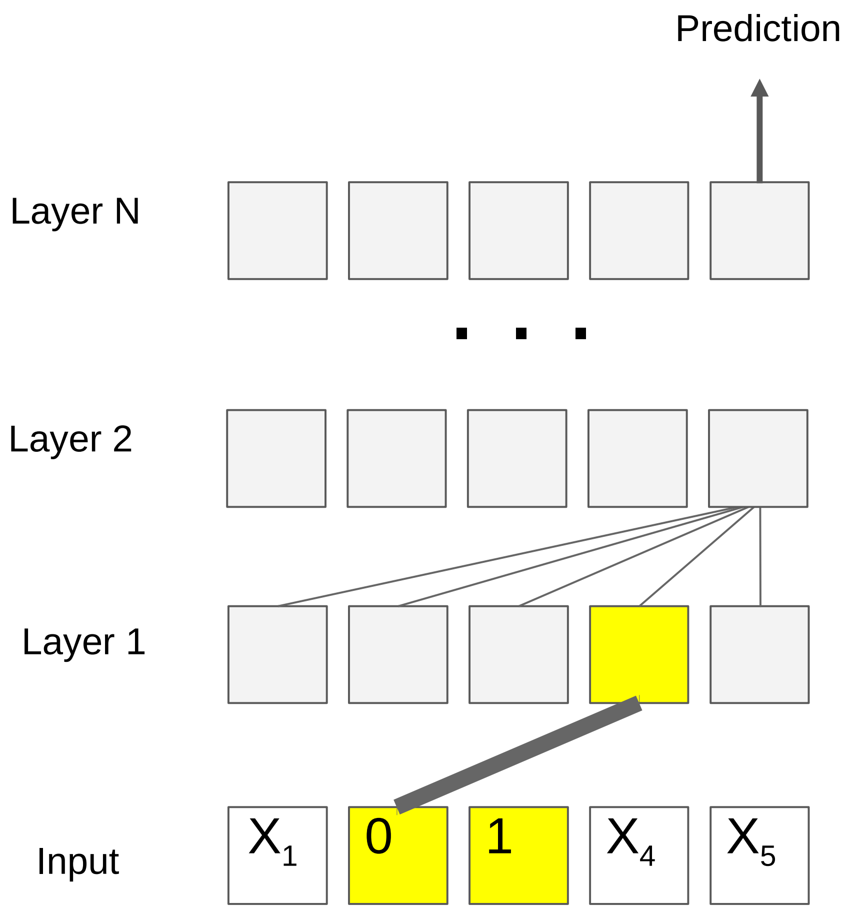
By fixing one input, we can make the head ignore all remaining input bits.



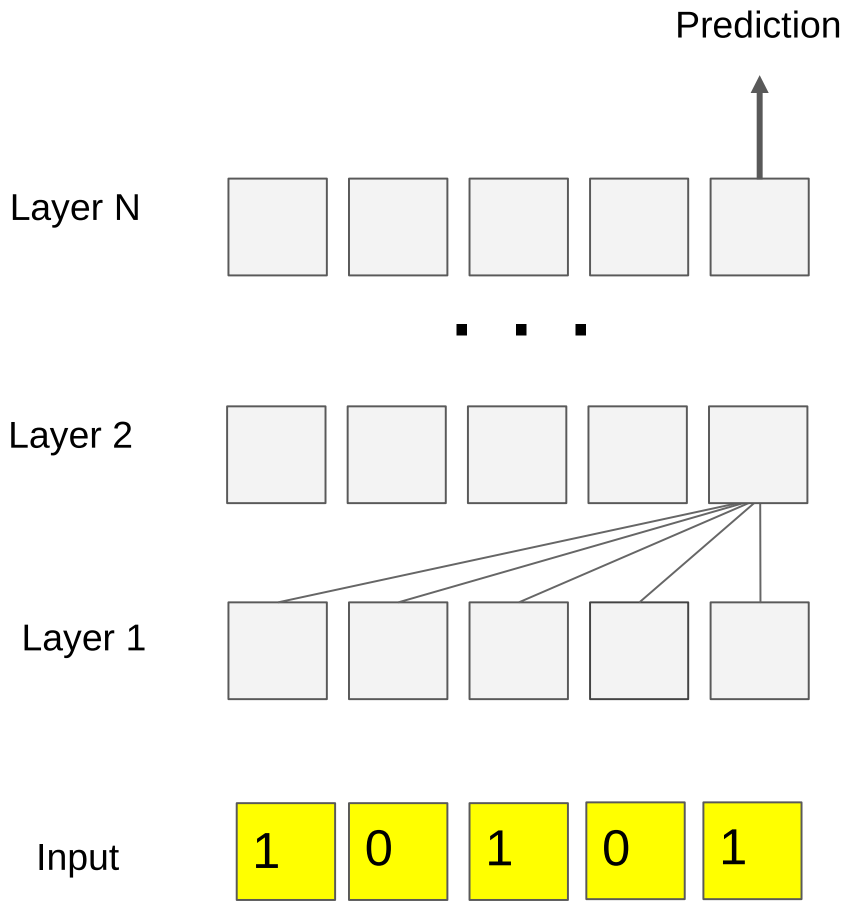
Now let's repeat this for every Layer 1 head.



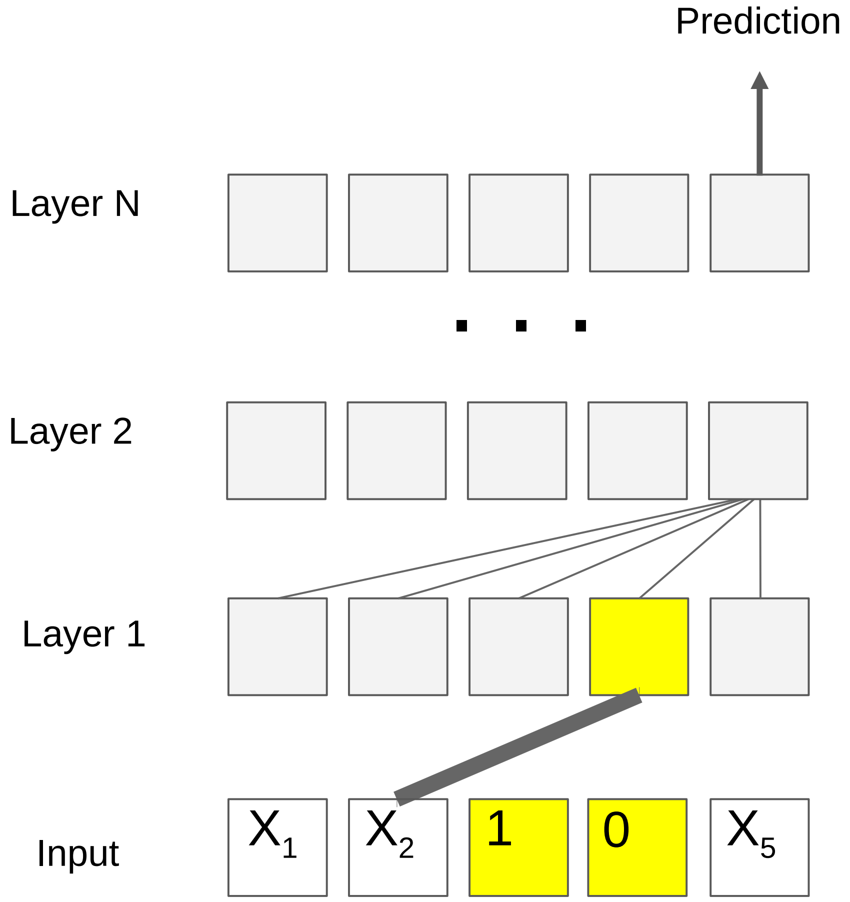
Now let's repeat this for every Layer 1 head.



Now let's repeat this for every Layer 1 head.

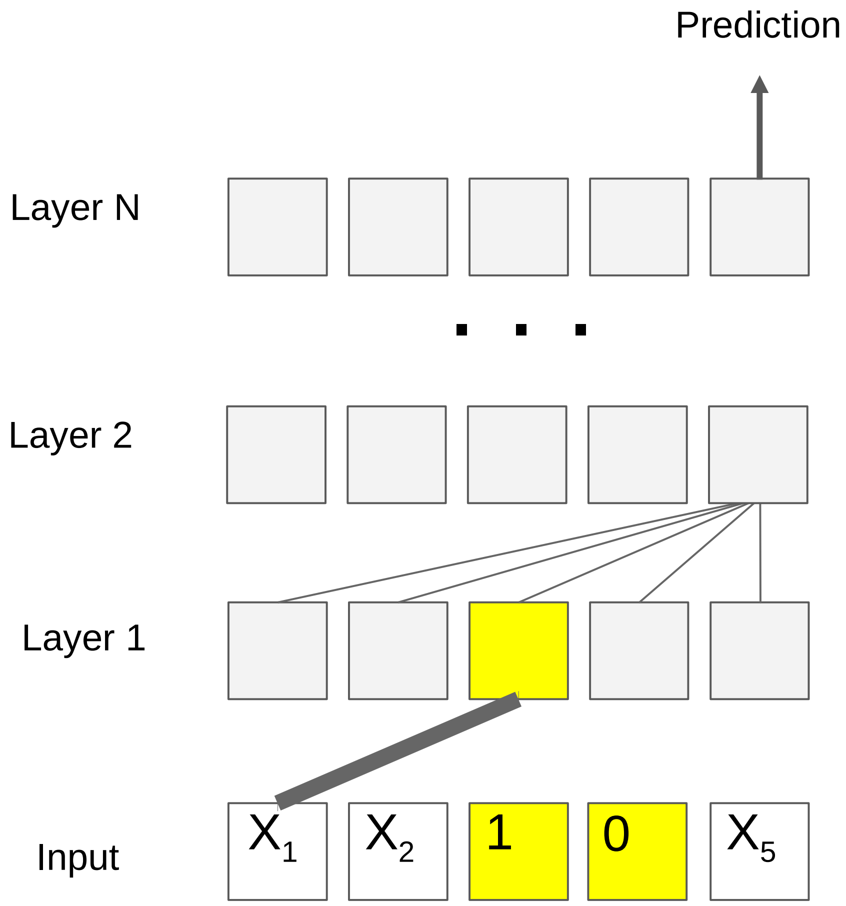


Problem: We might end up fixing all inputs.



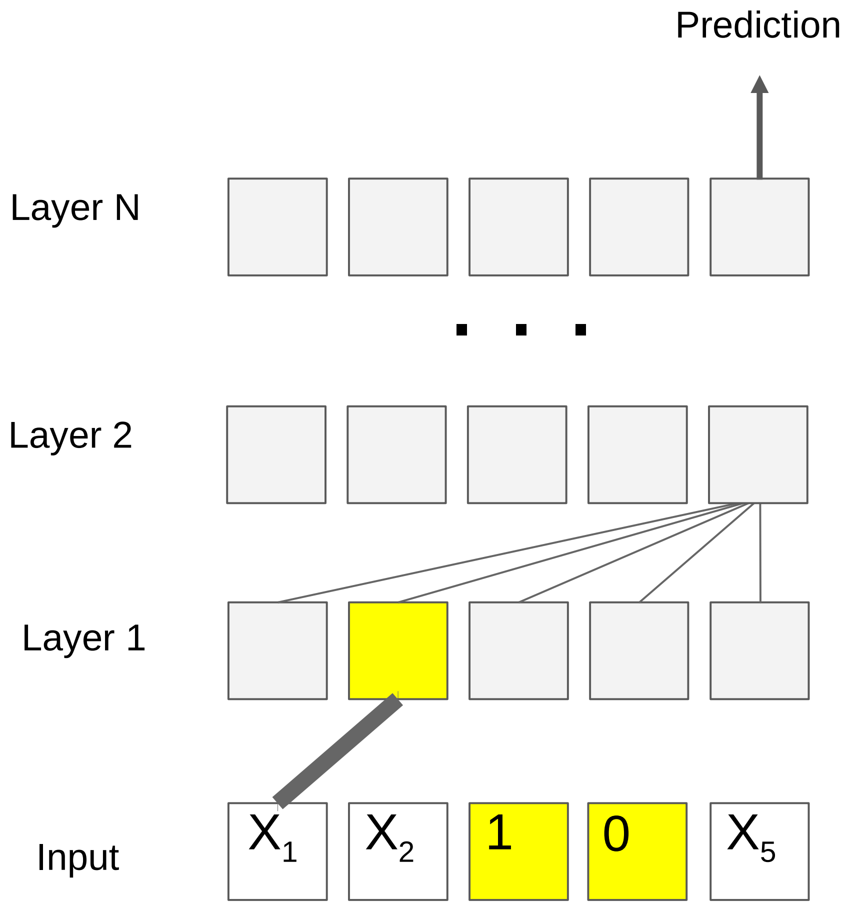
Problem: We might end up fixing all inputs.

Solution: Fix bits in such a way that each head ignores all but *k* input bits (for some constant *k*)



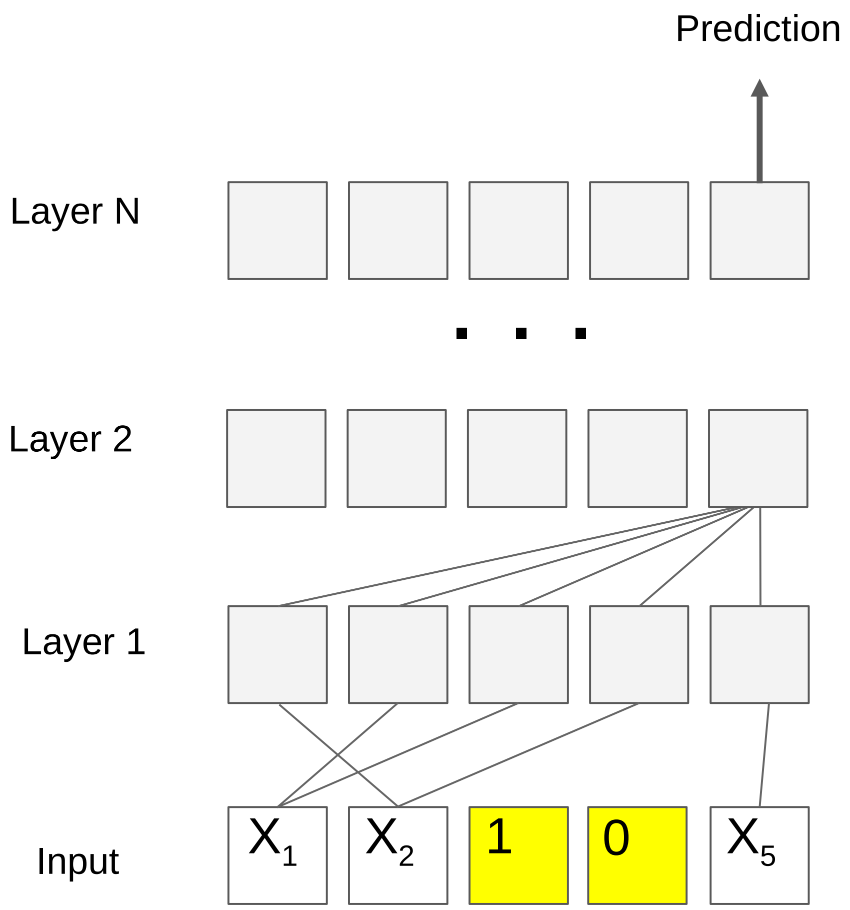
Problem: We might end up fixing all inputs.

Solution: Fix bits in such a way that each head ignores all but *k* input bits (for some constant *k*)



Problem: We might end up fixing all inputs.

Solution: Fix bits in such a way that each head ignores all but *k* input bits (for some constant *k*)

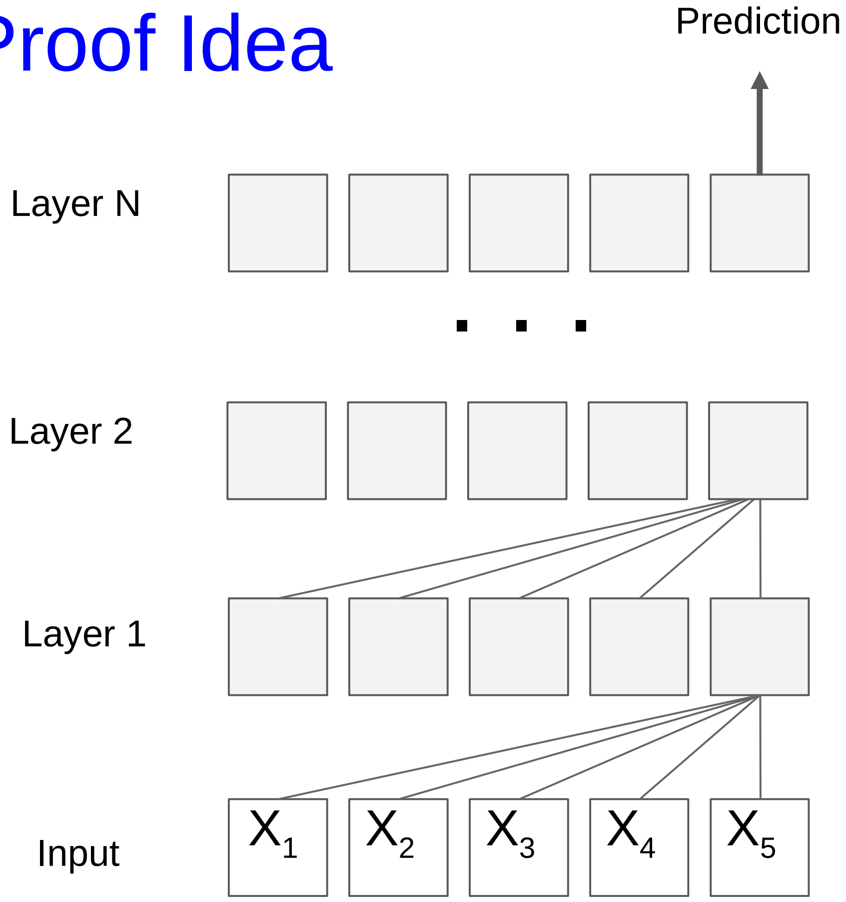


Problem: We might end up fixing all inputs.

Solution: Fix bits in such a way that each head ignores all but *k* input bits (for some constant *k*).

Can guarantee that this fixes only < 10% of bits.

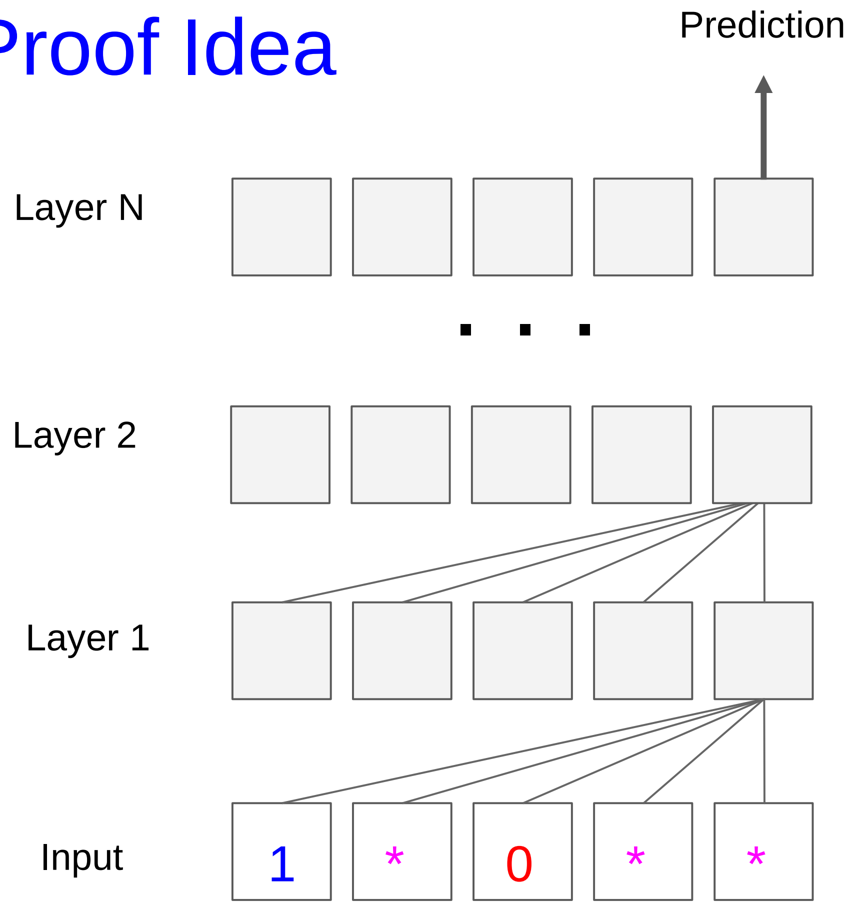
Proof Idea



Set each input i.i.d. to

* with p=95%
0 with p=2.5%
1 with p=2.5%

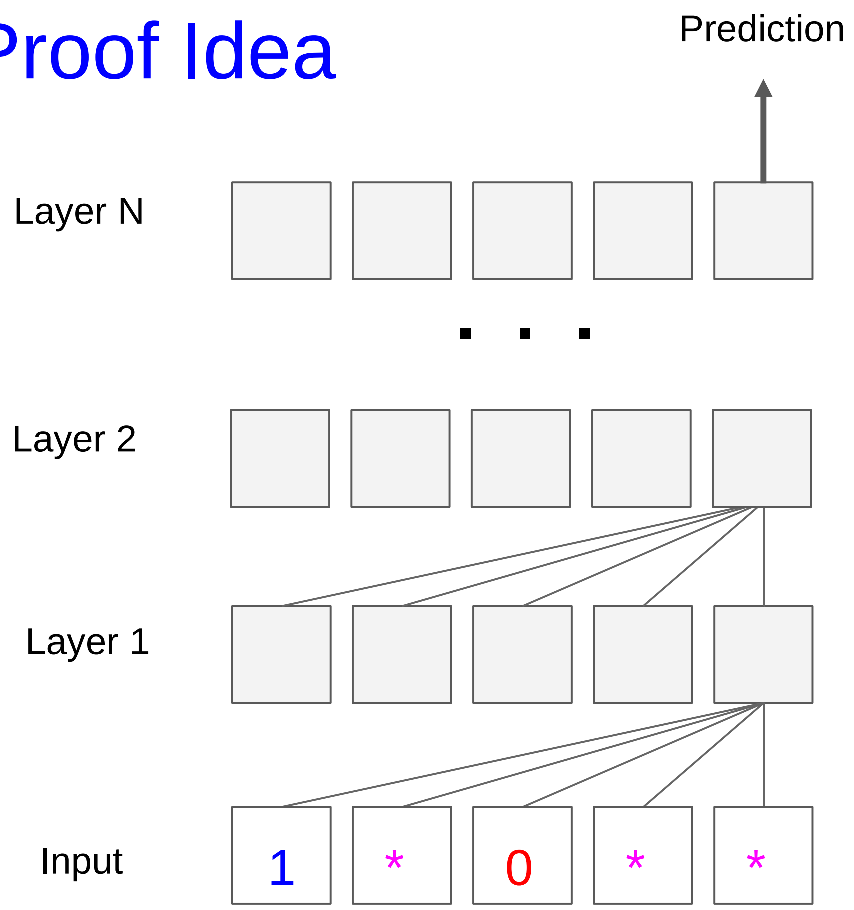
Proof Idea



Set each input i.i.d. to

* with p=95%
0 with p=2.5%
1 with p=2.5%

Proof Idea

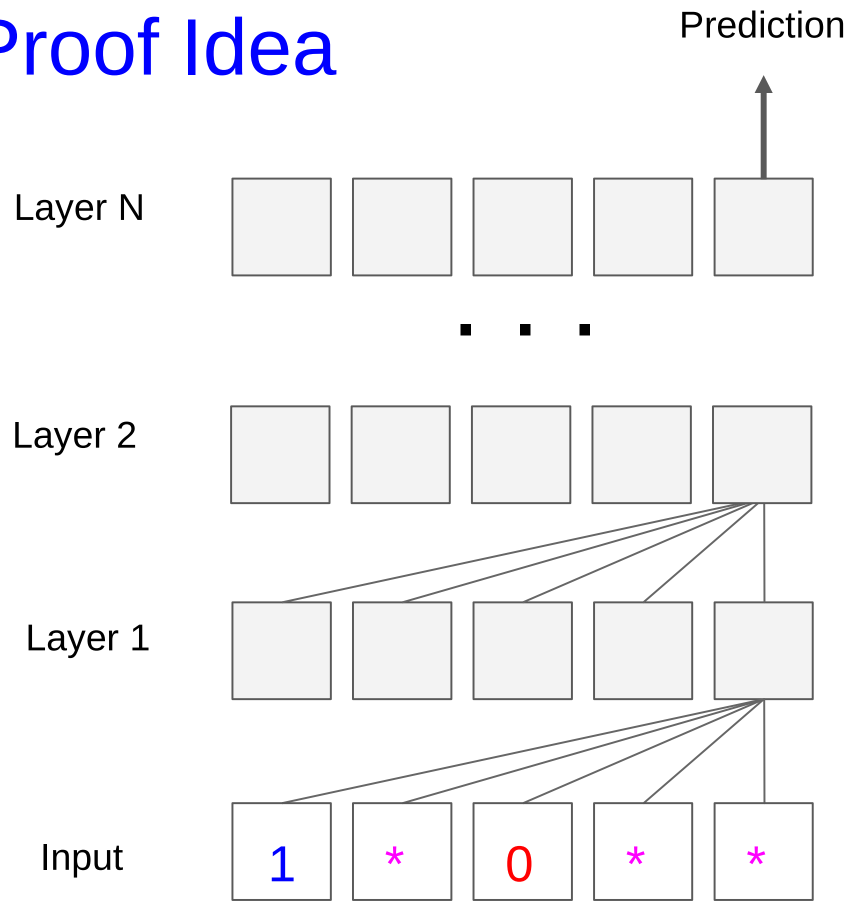


What is Probability that

1) each head depends on only k inputs, and
2) only < 10% of bits are fixed?

Enough to show that this is > 0!

Proof Idea

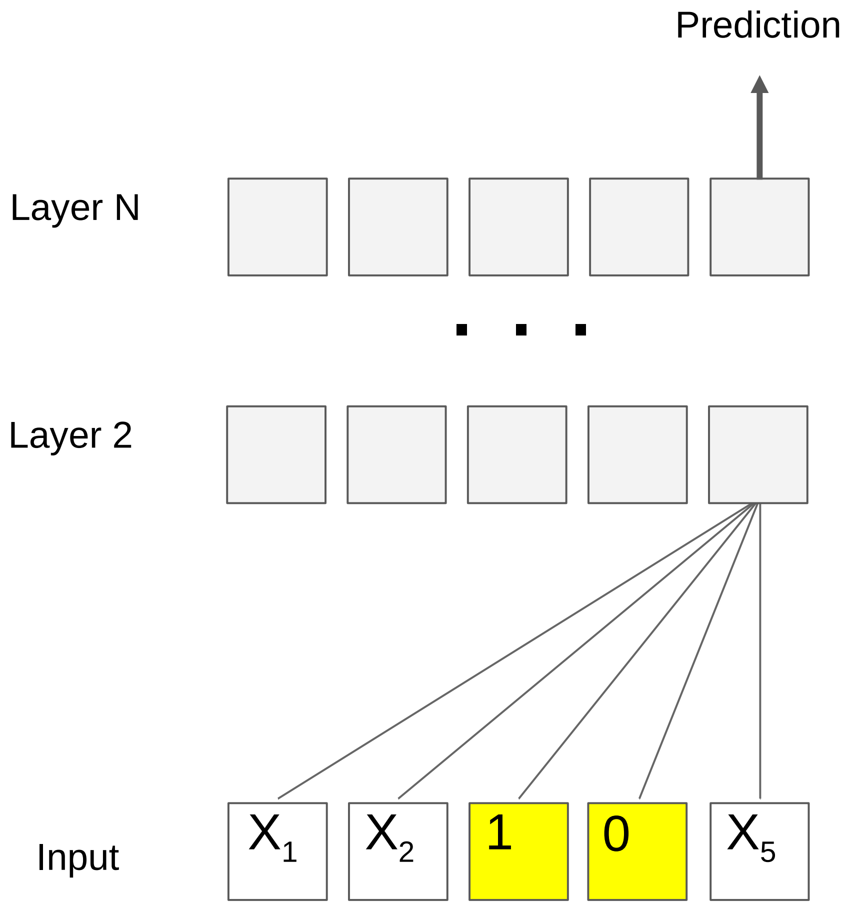


What is Probability that

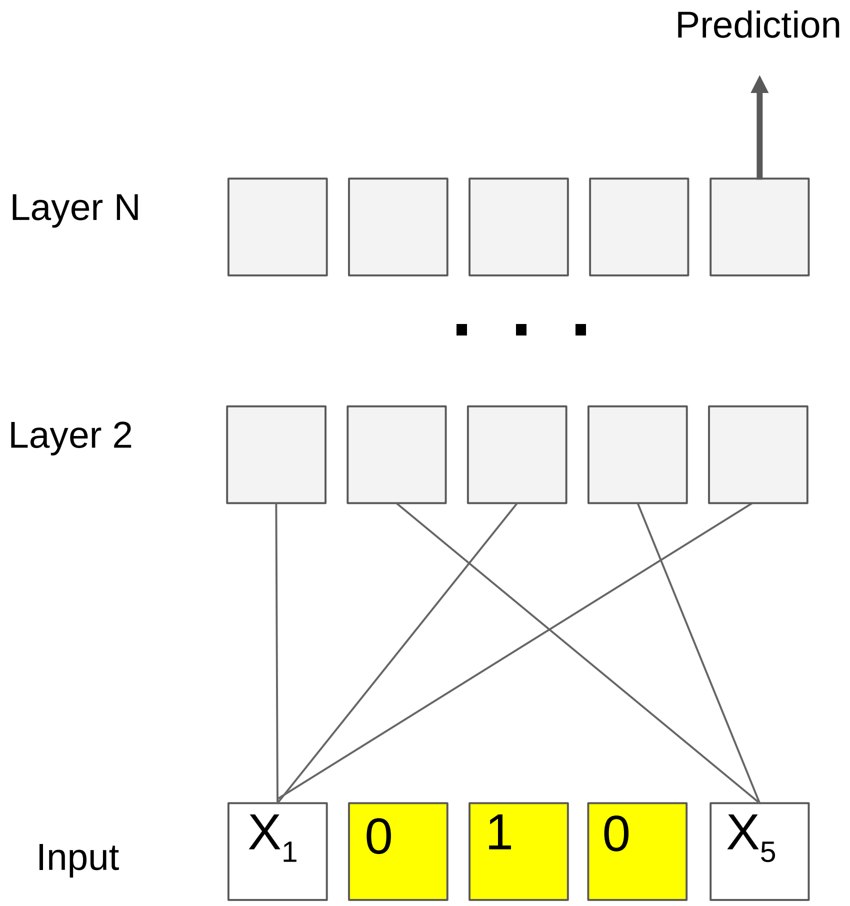
1) each head depends on only k inputs, and
2) only < 10% of bits are fixed?

Enough to show that this is > 0!

Show this by calculating for each head and combining via Lovasz Local Lemma.

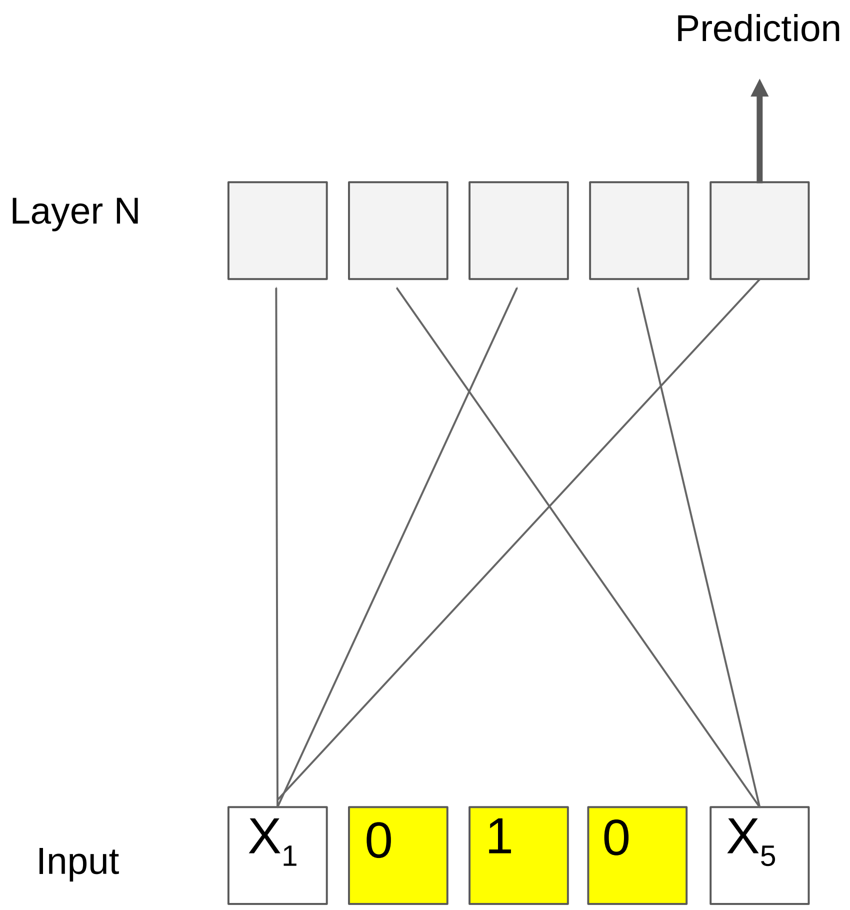


We can now fold Layer 1 into Layer 2….



We can now fold Layer 1 into Layer 2….

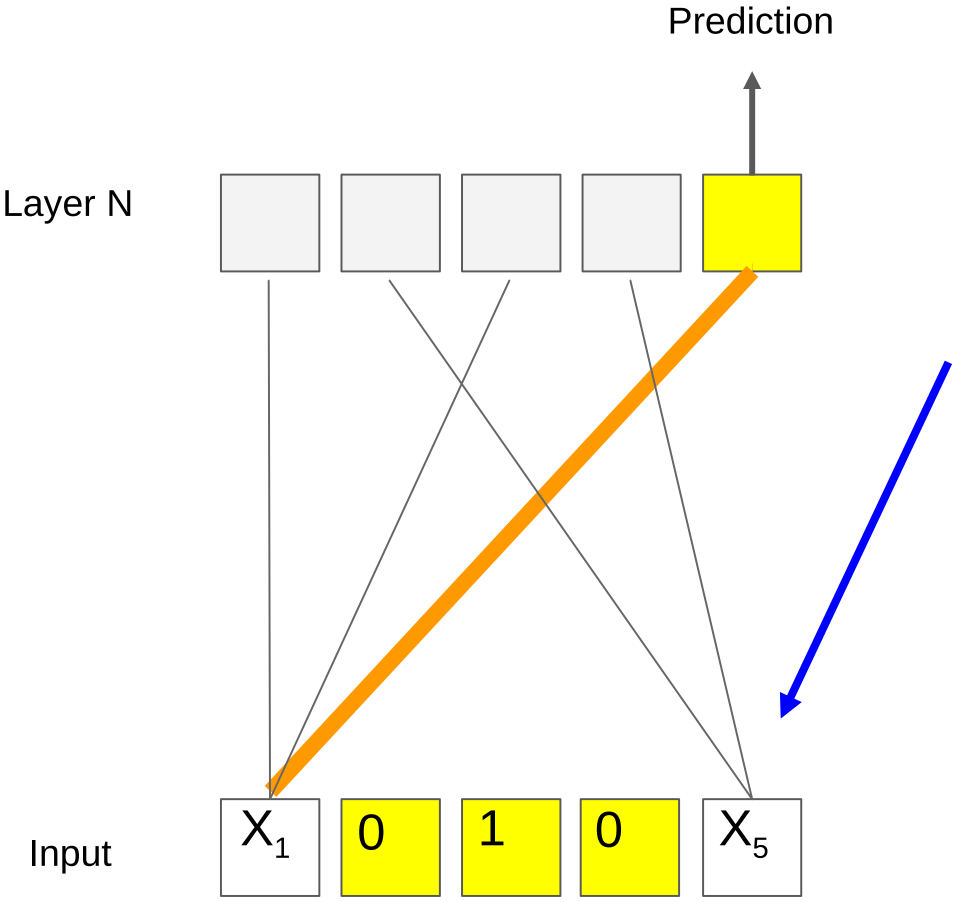
...and repeat the construction...



We can now fold Layer 1 into Layer 2….

...and repeat the construction…

...until only the final layer remains!



The prediction ignores bit X_5!

Thus, the transformer could never have modeled Parity (or Dyck$_2$).

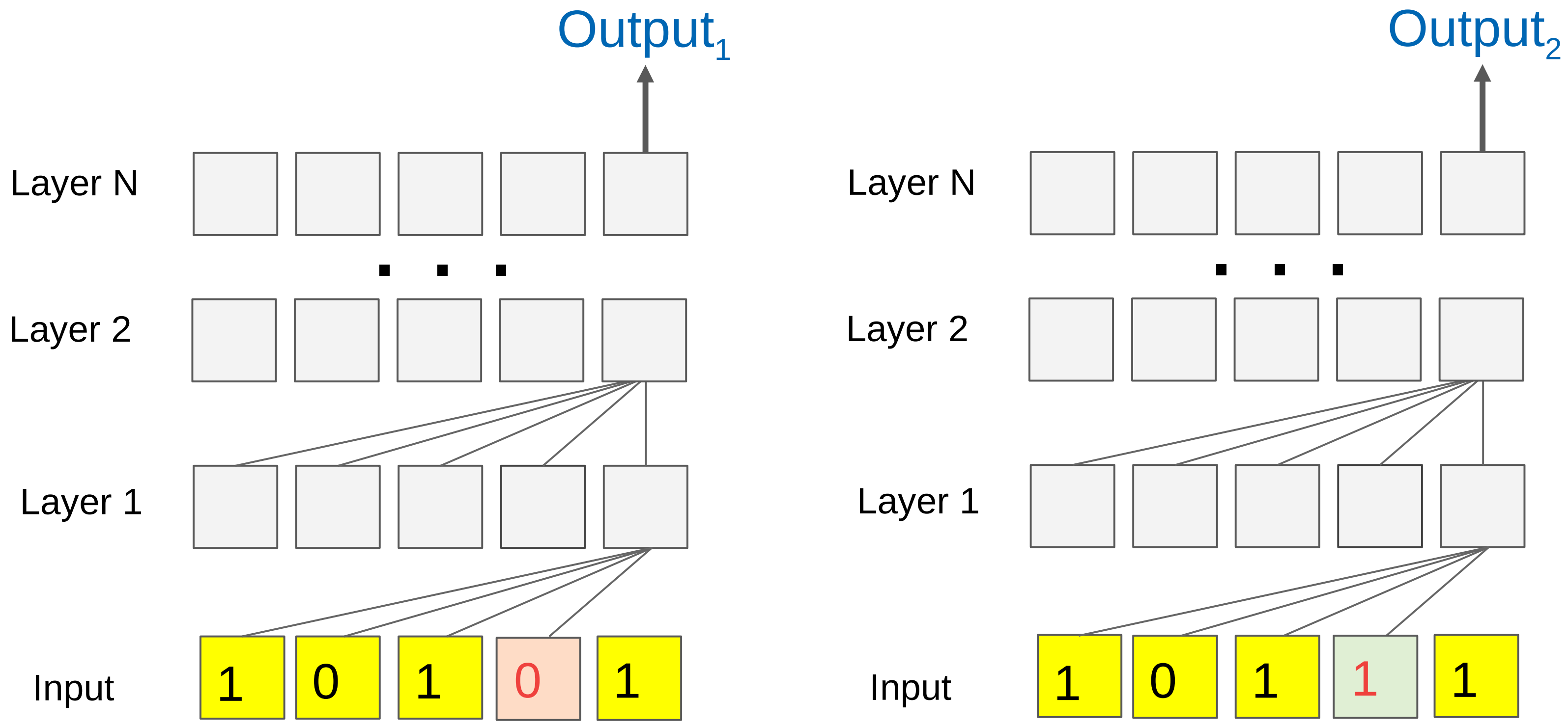
Part 2: Soft Attention

Results as strong as for hard attention would settle outstanding problem in computational complexity

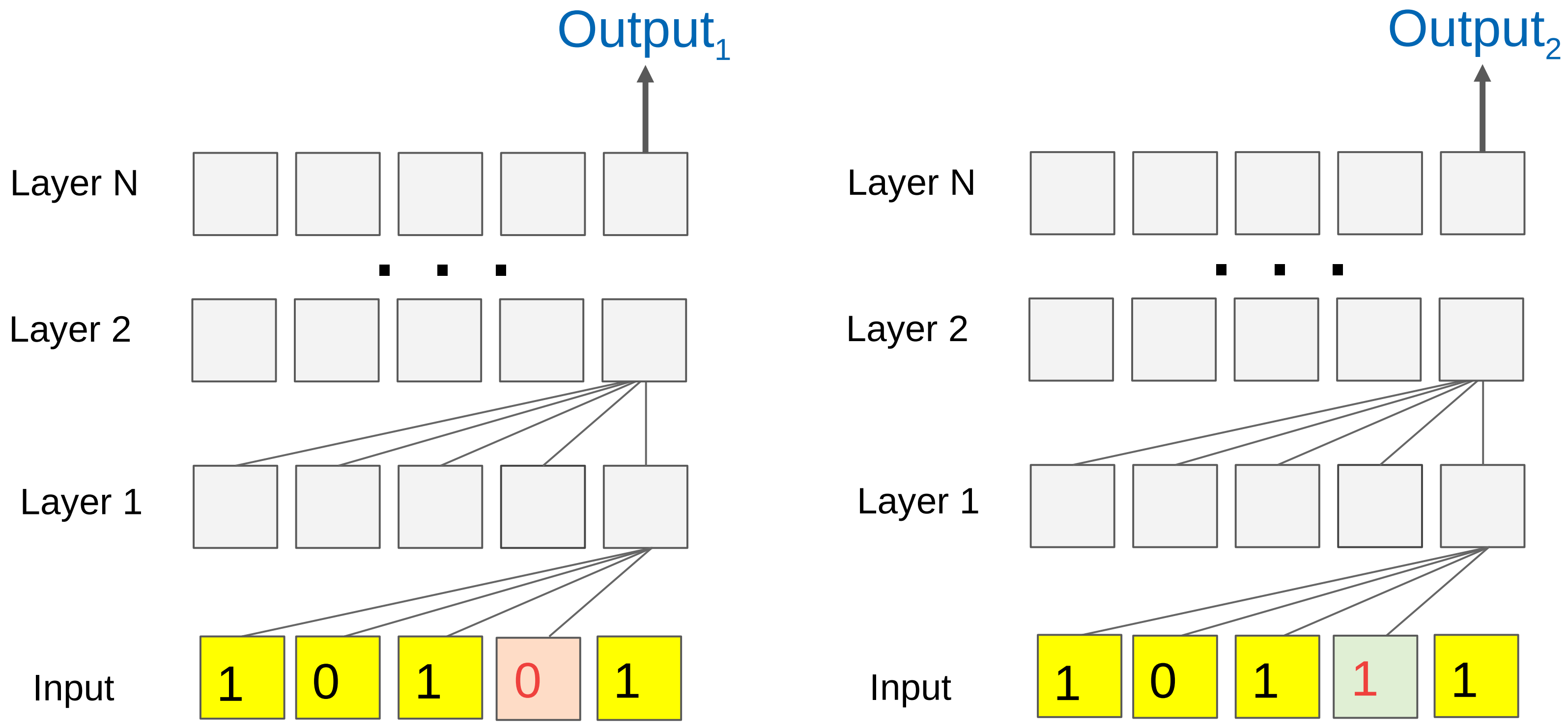
→ probably very hard to attain with currently available methods!

Instead:

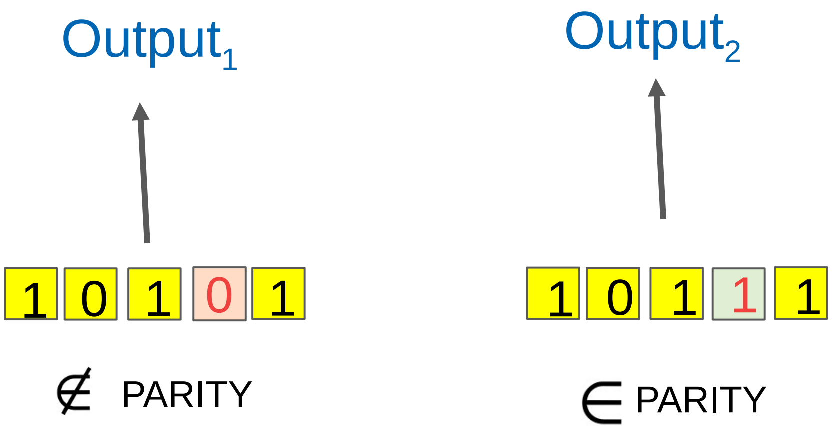
- Prove bounds on cross-entropy in prediction
- Focus on smooth activation functions as found in practice



Imagine we change one input symbol.
How much can this change the prediction?

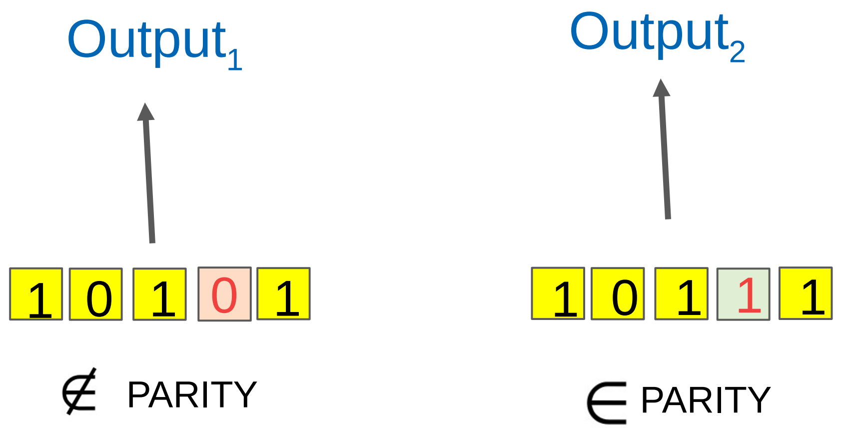


As input length n → ∞: **$|Output_1 - Output_2| = O(1/n)$**



As input length n $\rightarrow \infty$: $|\text{Output}_1 - \text{Output}_2| = O(1/n)$

But the strings should receive opposite labels.



As input length n → ∞: $|\text{Output}_1 - \text{Output}_2| = O(1/n)$

But the strings should receive opposite labels.

For a continuous prediction function, cross-entropy goes to chance level as n → ∞

Similar argument for DYCK_2.

Discussion

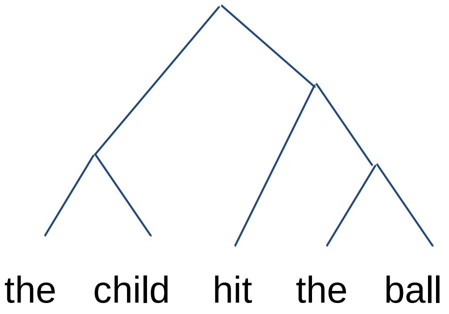
Transformers cannot model PARITY or DYCK_2 as inputs get longer.

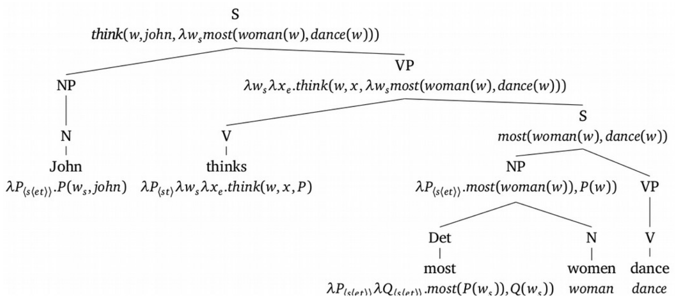
⇒ they cannot model wide subclasses of regular and context-free languages

Contrast with LSTMs, which can model these perfectly (at least when precision is unbounded).

What does this mean for language and NLP?

- Chomsky (1956): Language not regular, maybe not even context-free
- Montague (1972): Meaning described by logical formulas transduced from context-free parse trees.
- Shieber (1985): Swiss German not even context-free





… while transformers cannot emulate stacks.

What does this mean for language and NLP?

- Natural language can be approximated well with a model far too weak for the formal languages typically assumed in theoretical linguistics.

What does this mean for language and NLP?

- Natural language can be approximated well with a model far too weak for the formal languages typically assumed in theoretical linguistics.

- How to reconcile with success in NLP?

What does this mean for language and NLP?

- Natural language can be approximated well with a model that is far too weak for the formal languages typically assumed in theoretical linguistics.
- How to reconcile with success in NLP?
  - Practical models circumvent asymptotic limitations by having many layers and parameters?

What does this mean for language and NLP?

- Natural language can be approximated well with a model that is far too weak for the formal languages typically assumed in theoretical linguistics.
- How to reconcile with success in NLP?
  - Practical models circumvent asymptotic limitations by having many layers and parameters?
  - Transformers cannot fully understand language like humans do?

    Low Perplexity != True Understanding?

What does this mean for language and NLP?

- Natural language can be approximated well with a model that is far too weak for the formal languages typically assumed in theoretical linguistics.
- How to reconcile with success in NLP?
    - Practical models circumvent asymptotic limitations by having many layers and parameters?
    - Transformers cannot fully understand language like humans do?

      Low Perplexity != True Understanding?
    - Humans also have trouble doing these things accurately?

      Understanding naturalistic input might not require fully solving these problems?

Thank you!

Conclusion

- Investigated computational power of transformers to model hierarchical structure
- Focus on Parity and $Dyck_2$
- Negative results for both hard and soft attention

Directions for Future Work

1. What statistical properties of language make transformers work well on language? Can this give us new insights about the nature of language?

Directions for Future Work

1. What statistical properties of language make transformers work well on language? Can this give us new insights about the nature of language?
2. When do transformers work better than LSTMs?

Directions for Future Work

1. What statistical properties of language make transformers work well on language? Can this give us new insights about the nature of language?
2. When do transformers work better than LSTMs?
3. Link self-attention to psycholinguistic models that predict that humans have trouble with recursion?

Directions for Future Work

1. What statistical properties of language make transformers work well on language? Can this give us new insights about the nature of language?
2. When do transformers work better than LSTMs?
3. Link self-attention to psycholinguistic models that predict that humans have trouble with recursion?
4. Can transformers help understand why humans can do some recursion but struggle with center-embedding?

Questions

1. Can we characterize more exactly the formal languages that transformers capture? E.g., which regular languages?
2. What theoretical properties explain why transformers work better than LSTMs?
3. Link self-attention to psycholinguistic models that predict that humans have trouble with recursion?
4. Can transformers help understand why humans can do some recursion but struggle with center-embedding?
5. What do you think?

Discussion

Examples that hard-attention transformers ***can*** solve easily:

- 1^* (also a two-state regular language)
- $a^n b^n$ (also a basic context-free language)

Difference from Parity and Dyck_2: Fixing a few bits can easily force word to be outside of language.

What is the computational power of transformers?

Tran et al (EMNLP 2018):

LSTMs appear to work better than transformers at

- English agreement
- evaluating logical formulas

Similar result: Evans et al. (ICLR 2018)

The Importance of Being Recurrent for Modeling Hierarchical Structure

Ke Tran[1] **Arianna Bisazza**[2] **Christof Monz**[1]
[1] Informatics Institute, University of Amsterdam
[2] Leiden Institute of Advanced Computer Science, Leiden University
{m.k.tran,c.monz}@uva.nl a.bisazza@liacs.leidenuniv.nl