RECURRENT NEURAL NETWORKS

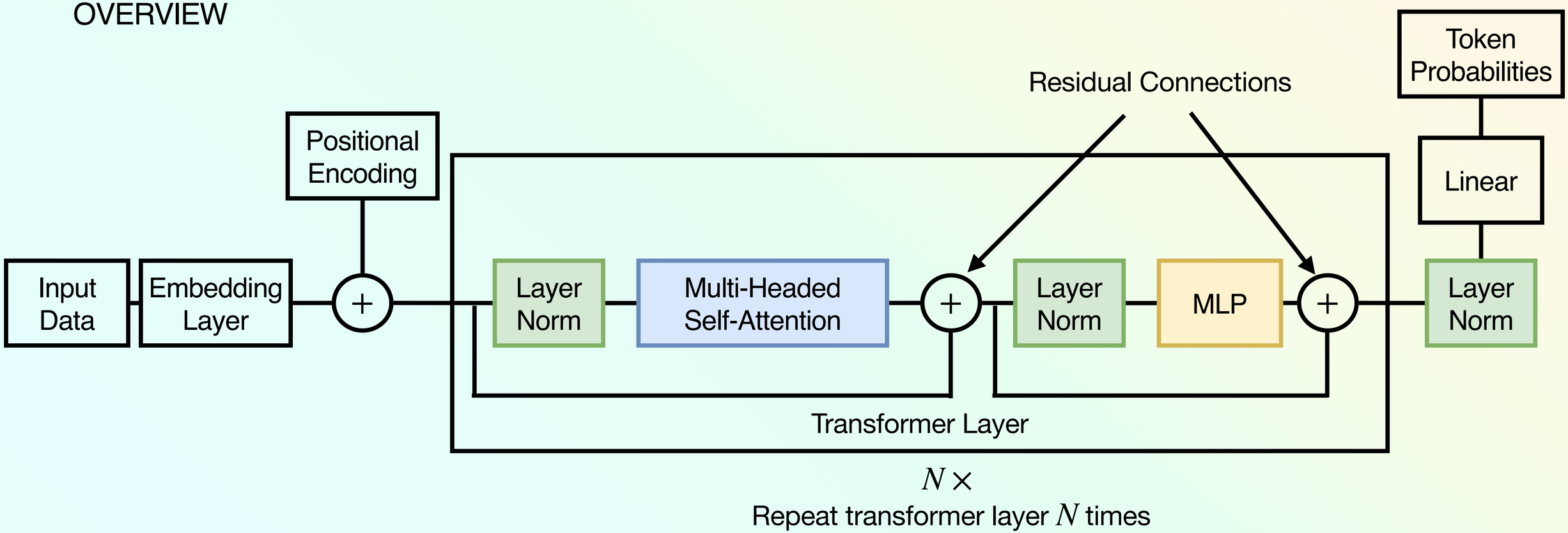
TODAY’S CLASS

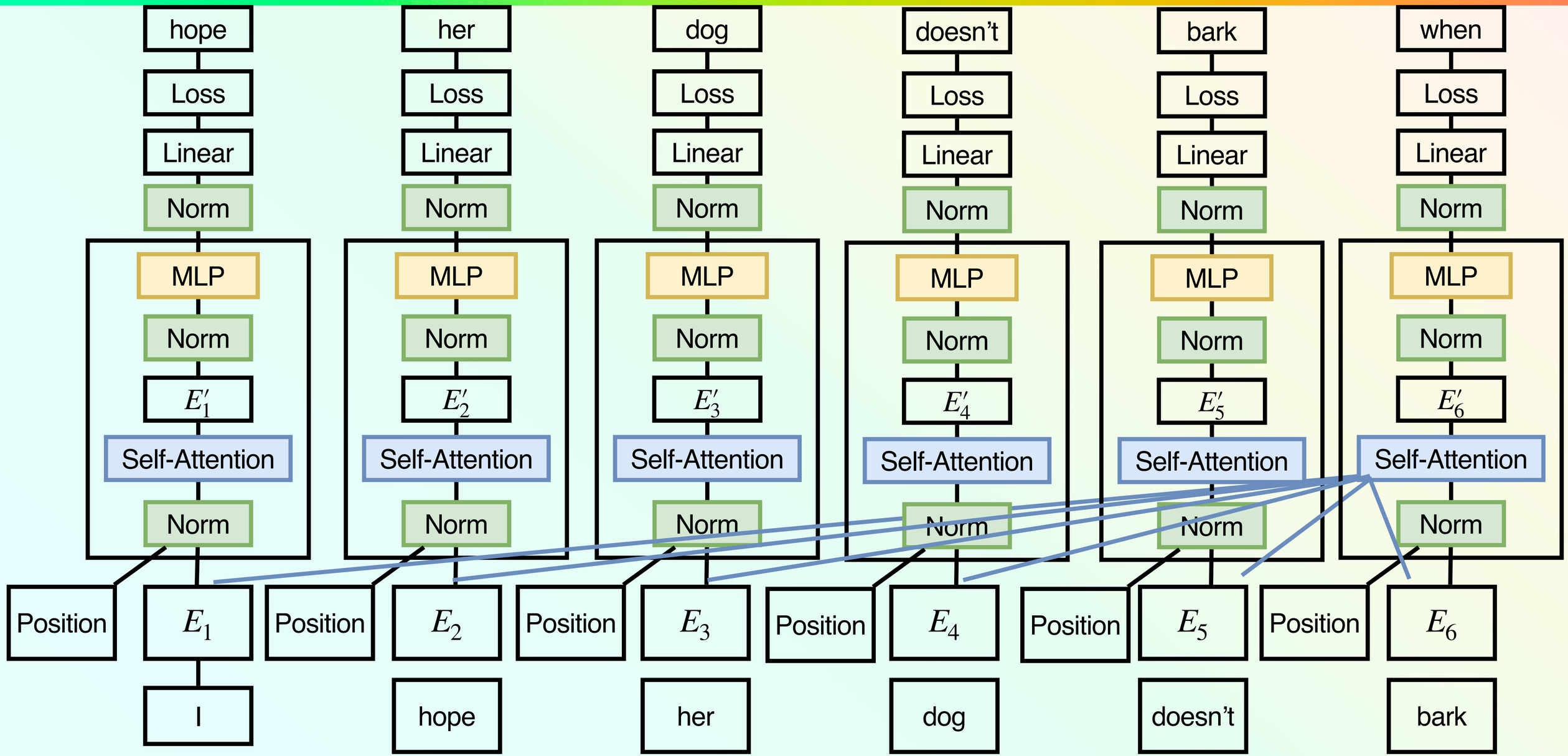
GOALS

1. Review transformers and discuss a few properties
2. Motivate RNNs,Define them, give the basics, and their problems
3. Cover LSTMs and what they do better than RNNs

TRANSFORMERS

OVERVIEW




hope
Loss
Linear
Norm
MLP
Norm
E'_1
Self-Attention
Norm
Position
E_1
I
her
Loss
Linear
Norm
MLP
Norm
E'_2
Self-Attention
Norm
Position
E_2
hope
dog
Loss
Linear
Norm
MLP
Norm
E'_3
Self-Attention
Norm
Position
E_3
her
doesn't
Loss
Linear
Norm
MLP
Norm
E'_4
Self-Attention
Norm
Position
E_4
dog
bark
Loss
Linear
Norm
MLP
Norm
E'_5
Self-Attention
Norm
Position
E_5
doesn't
when
Loss
Linear
Norm
MLP
Norm
E'_6
Self-Attention
Norm
Position
E_6
bark

GPT

DEFINITION

Generative Pretrained Transformer

Train on a MASSIVE dataset

Model predictions are “generally” “good”

- Fine-tune on new tasks
- Pretraining is a smart weight initialization for the new task

Chatbots have extra steps

QUIZ

TRANFORMERS

HOW THEY PROCESS INFORMATION

- Self-attention — brings in information from other points in the sequence
- MLP — performs nonlinear operations to combine and transform this information

- More attention heads —> more information from the sequence can be brought into each layer
- Bigger MLP —> more processing that can happen at each layer

- Multiple transformer layers — sequentially process more information about the sequence
- The last layer must contain all relevant information from the sequence to predict the next token

TRANFORMERS

HOW THEY PROCESS INFORMATION

A transformer with T layers can perform the same computations as a finite Autonoma on a T-length sequence.

Short-cut solutions exist and are easily learned with standard training paradigms

- $O(\log T)$ depth solution always exists
- $O(1)$ solutions are common

These shortcut solutions are brittle to out-of-distribution sequences:

- Training on math operations for one and two-digit numbers, then trying for three-digit numbers

QUIZ

TRANFORMERS

HOW THEY PROCESS INFORMATION

Some operations are hard to perform “3+4x10+6=”

Predicting 49 as the next two tokens is hard.

The network has to compute the answer and determine that the right thing to output is just the number.

TRANFORMERS

HOW THEY PROCESS INFORMATION

Some operations are hard to perform “3+4x10+6=”

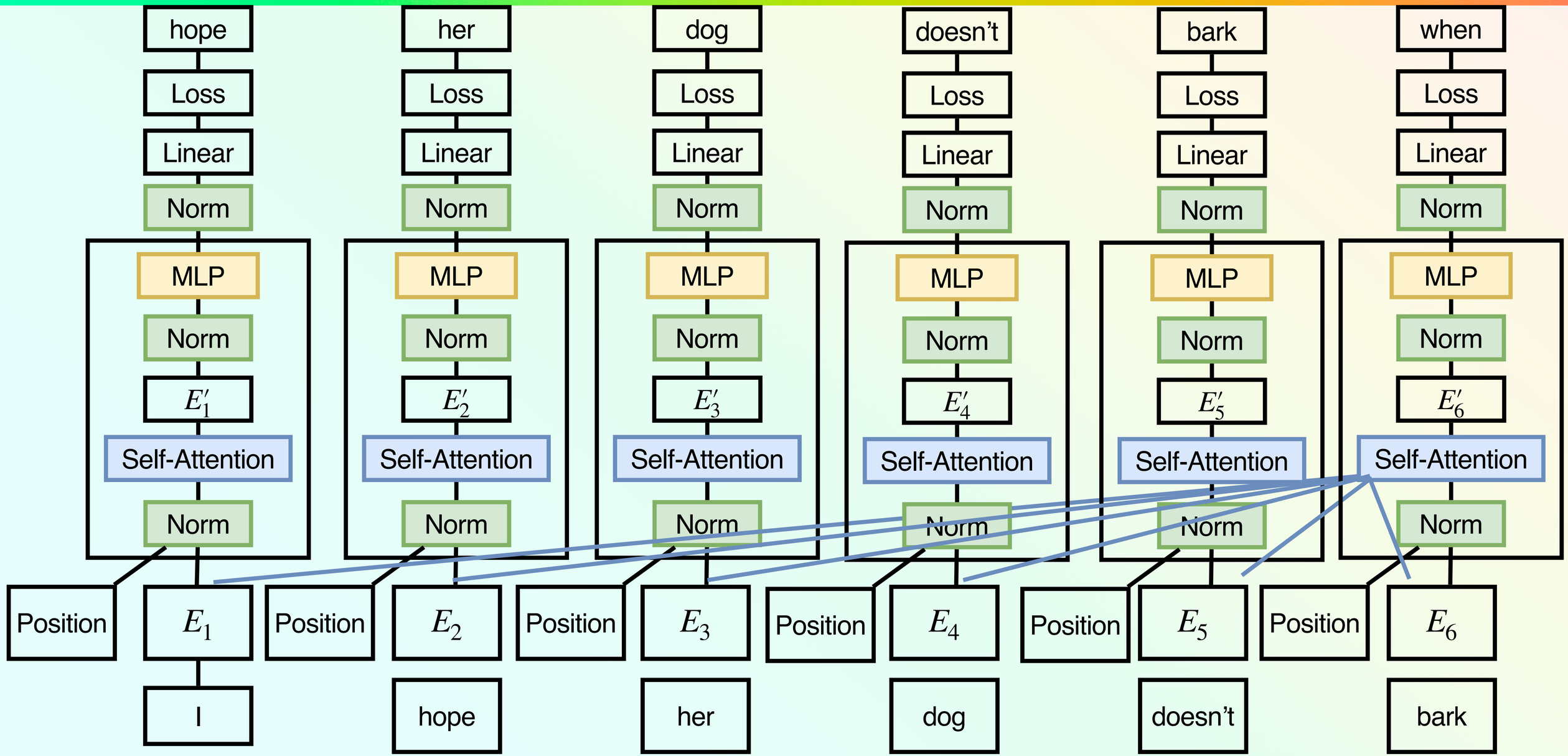
Predicting 49 as the next two tokens is hard.

The network has to compute the answer and determine that the right thing to output is just the number.

Let the network show its work so the computation is easier:

“What is 3+4x10+6? Show your work?” —> ok so we first multiply 4 with 10, which equals 40…

Allows the network to use its output as working memory. — Scratch-pad training


hope
Loss
Linear
Norm
MLP
Norm
E'_1
Self-Attention
Norm
Position
E_1
I
her
Loss
Linear
Norm
MLP
Norm
E'_2
Self-Attention
Norm
Position
E_2
hope
dog
Loss
Linear
Norm
MLP
Norm
E'_3
Self-Attention
Norm
Position
E_3
her
doesn't
Loss
Linear
Norm
MLP
Norm
E'_4
Self-Attention
Norm
Position
E_4
dog
bark
Loss
Linear
Norm
MLP
Norm
E'_5
Self-Attention
Norm
Position
E_5
doesn't
when
Loss
Linear
Norm
MLP
Norm
E'_6
Self-Attention
Norm
Position
E_6
bark

TRANSFORMERS

COMPUTATIONAL COST

Self-attention requires $O(T^2)$ computation for sequence length T ← Expensive

Long context is important for considering huge amounts of information or staying on topic

$O(T^2)$ requires too much computation and memory

TRANSFORMERS

COMPUTATIONAL COST

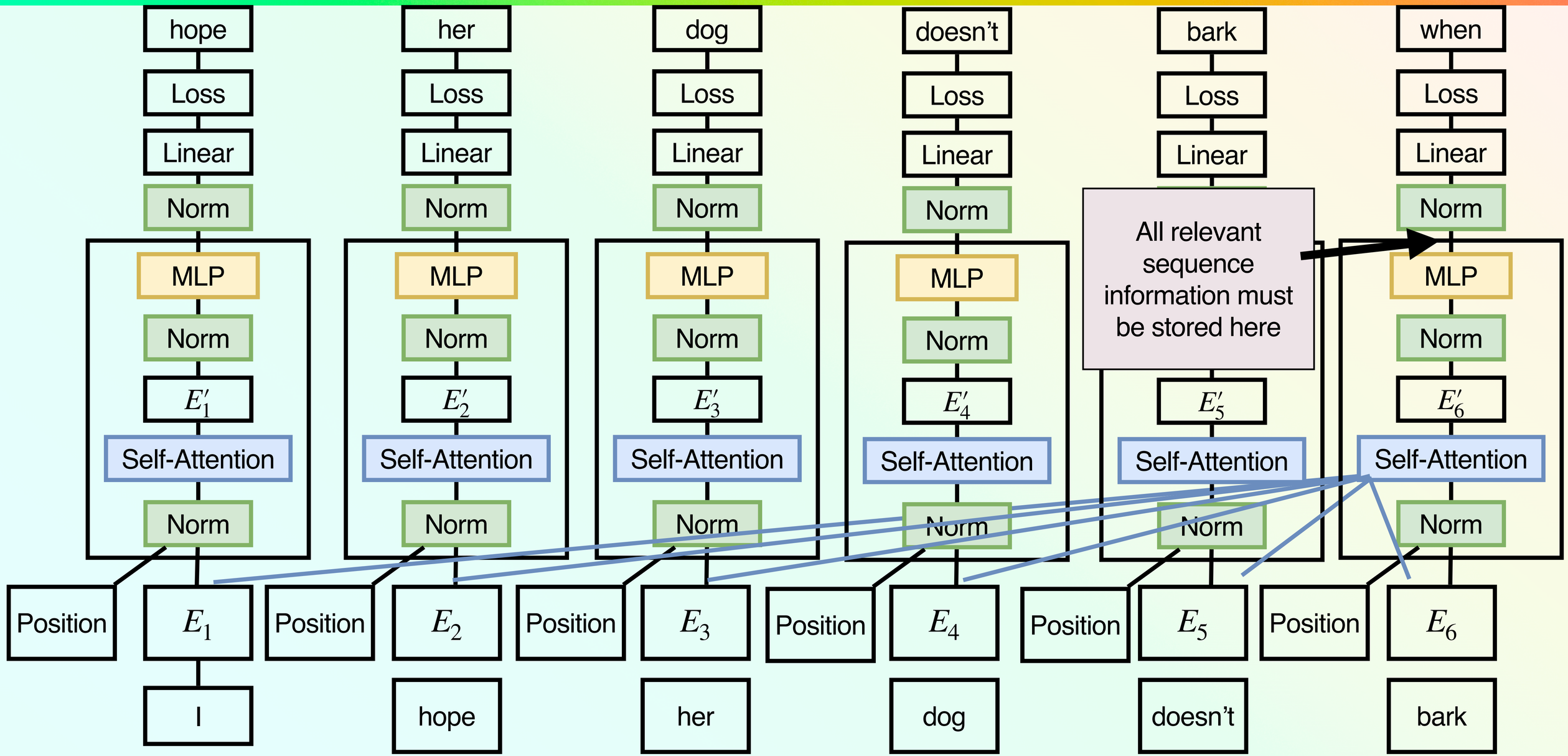
Self-attention requires $O(T^2)$ computation for sequence length T

Long context is important for considering huge amounts of information or staying on topic

$O(T^2)$ requires too much computation and memory

Solution:

- Attention mechanisms that scale better with T, e.g., $T \log(T)$
- Different kinds of neural networks that have constant $O(1)$ memory and computation


hope
Loss
Linear
Norm
MLP
Norm
E'_1
Self-Attention
Norm
Position
E_1
I
her
Loss
Linear
Norm
MLP
Norm
E'_2
Self-Attention
Norm
Position
E_2
hope
dog
Loss
Linear
Norm
MLP
Norm
E'_3
Self-Attention
Norm
Position
E_3
her
doesn't
Loss
Linear
Norm
MLP
Norm
E'_4
Self-Attention
Norm
Position
E_4
dog
bark
Loss
Linear
Norm
E'_5
Self-Attention
Norm
Position
E_5
doesn't
All relevant sequence information must be stored here
when
Loss
Linear
Norm
MLP
Norm
E'_6
Self-Attention
Norm
Position
E_6
bark

RECURRENT NEURAL NETWORKS

IDEA

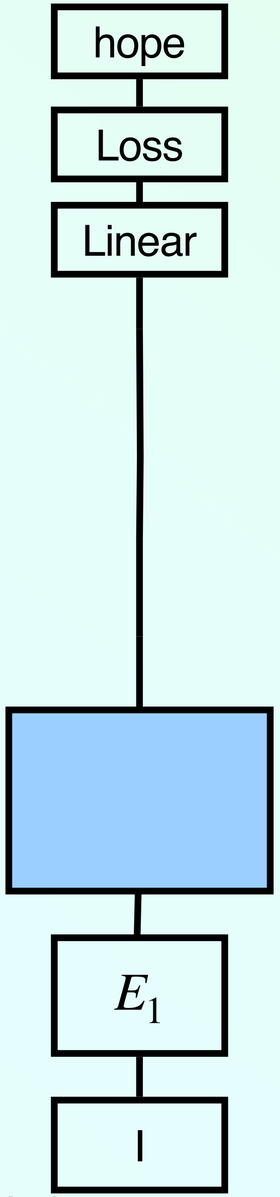
Transformer:

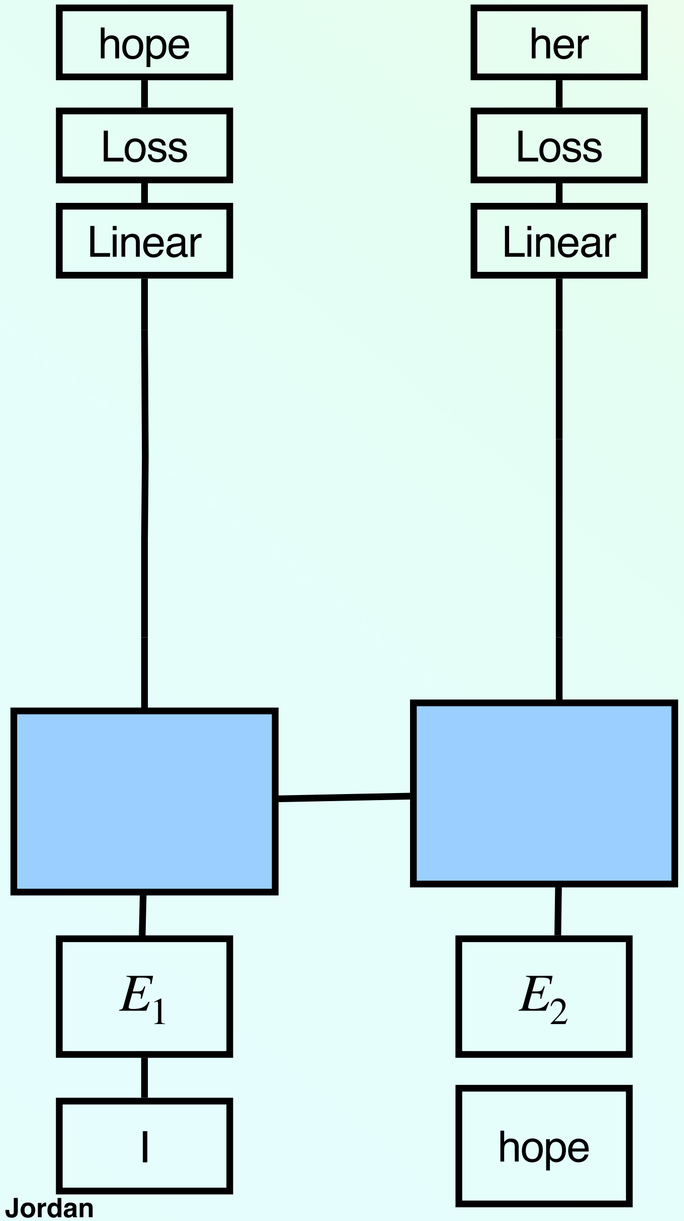
— sequentially include more information with each layer

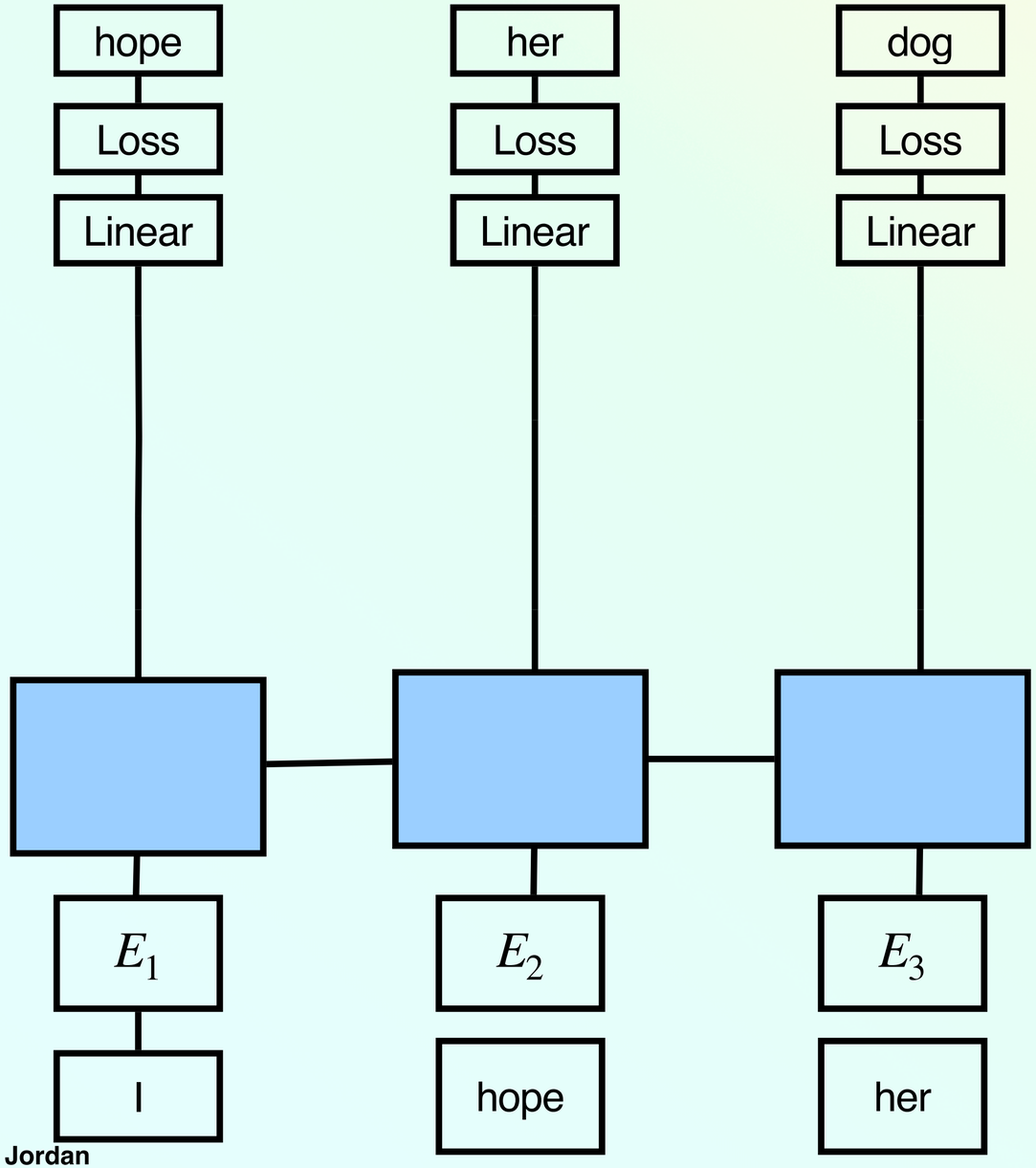
— last layer includes all relevant information for **just** the next token prediction

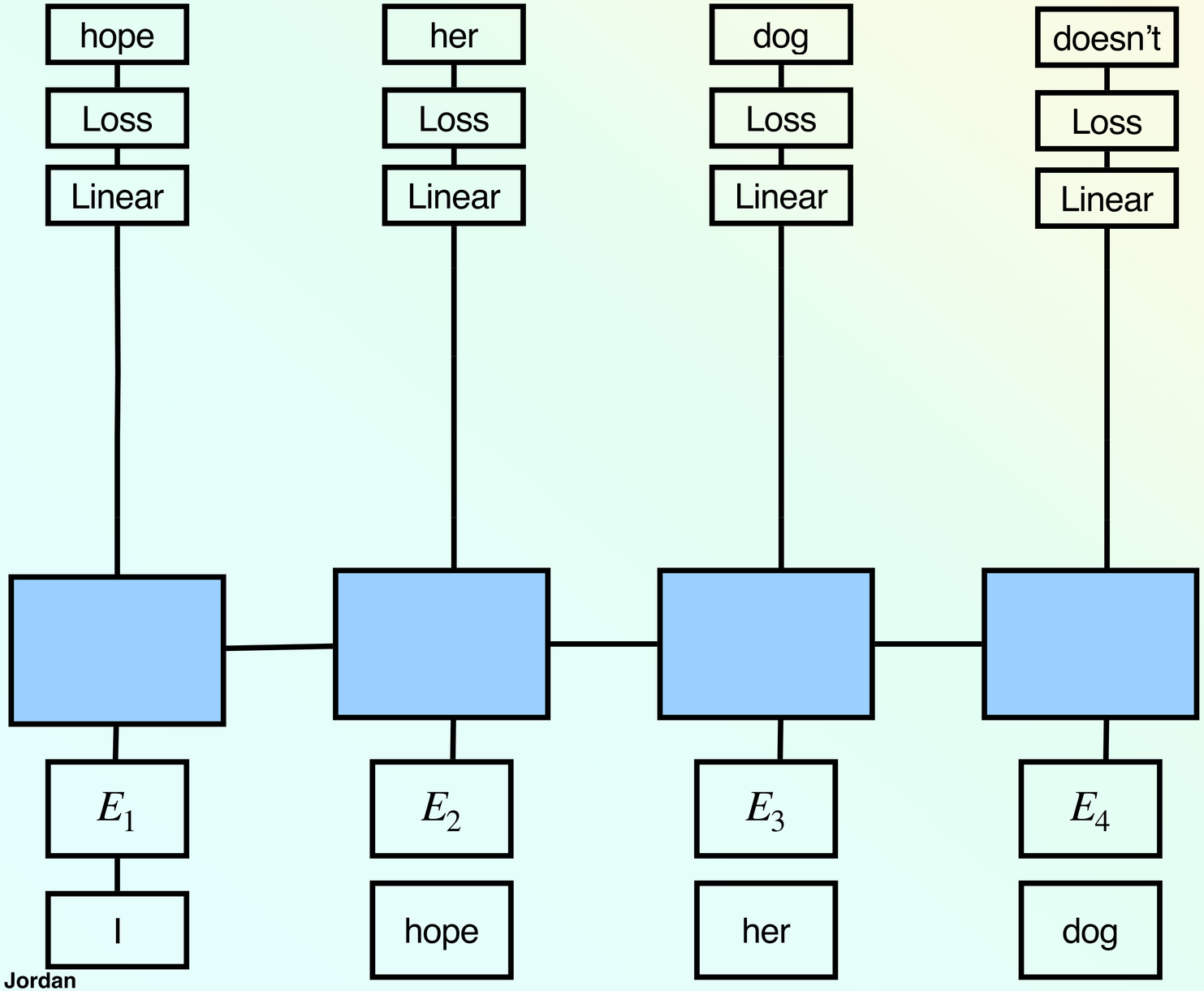
Idea:

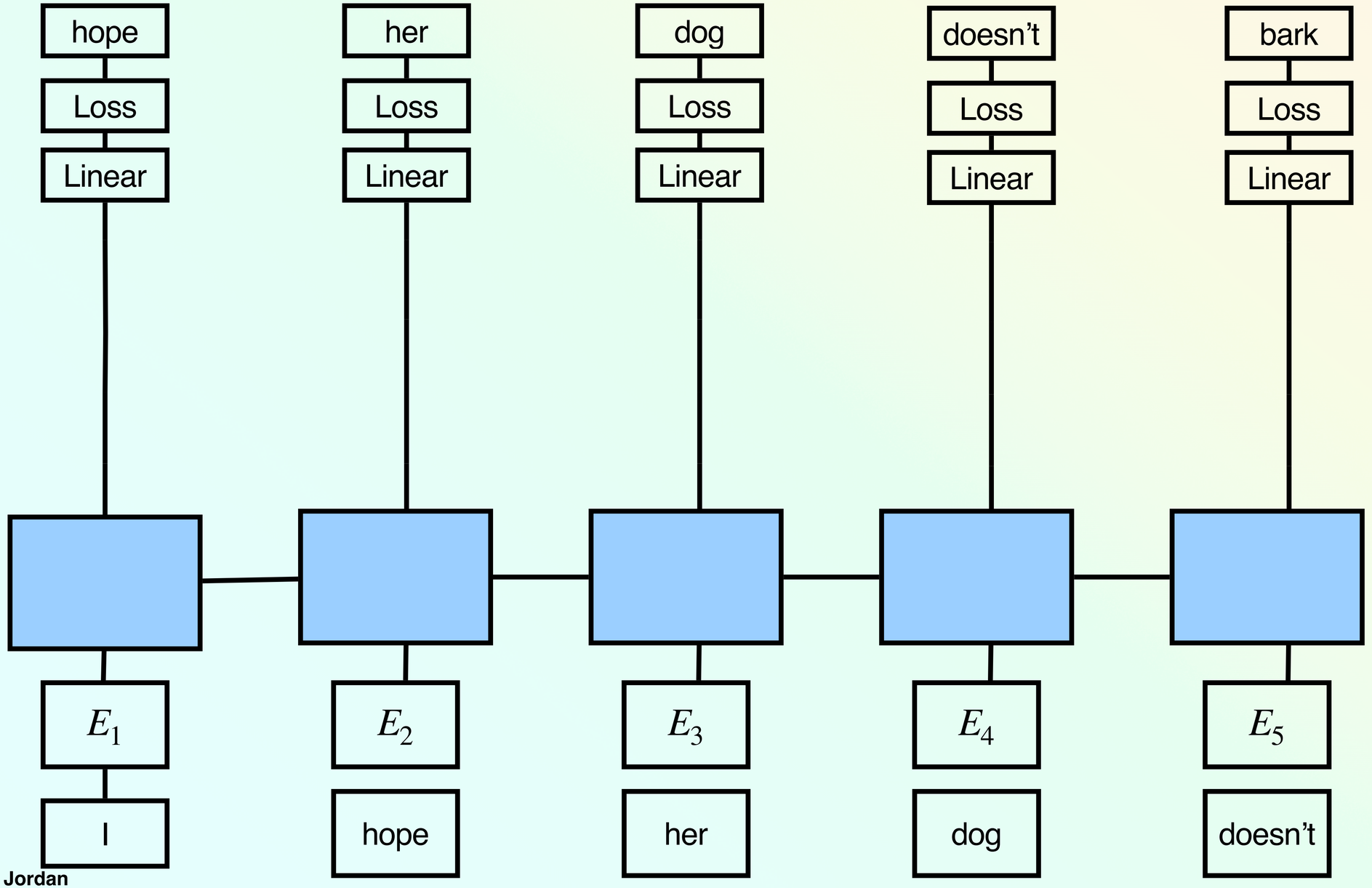
Have a layer that remembers all relevant information for **all** the next tokens

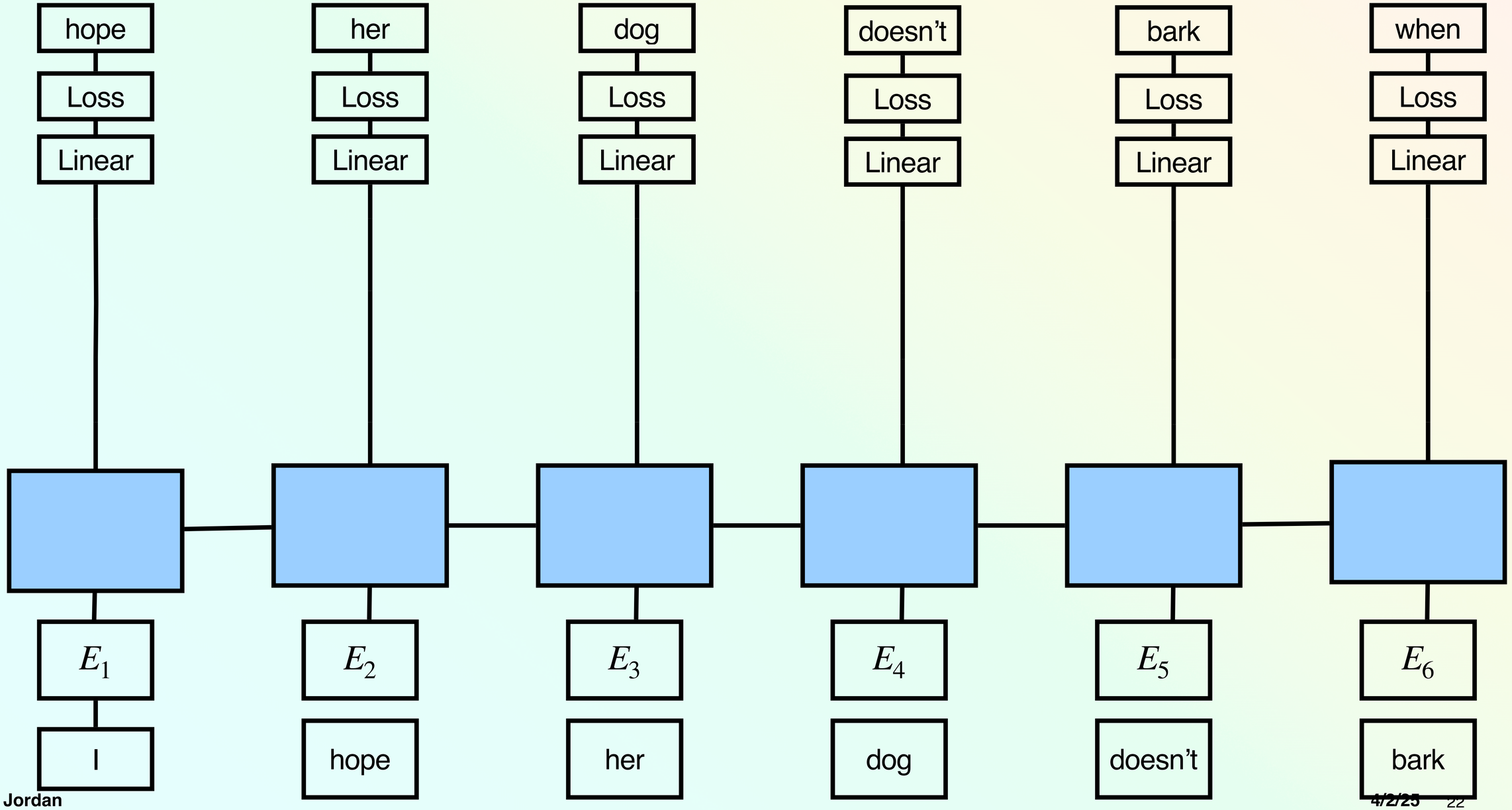

hope
Loss
Linear
E_1
I

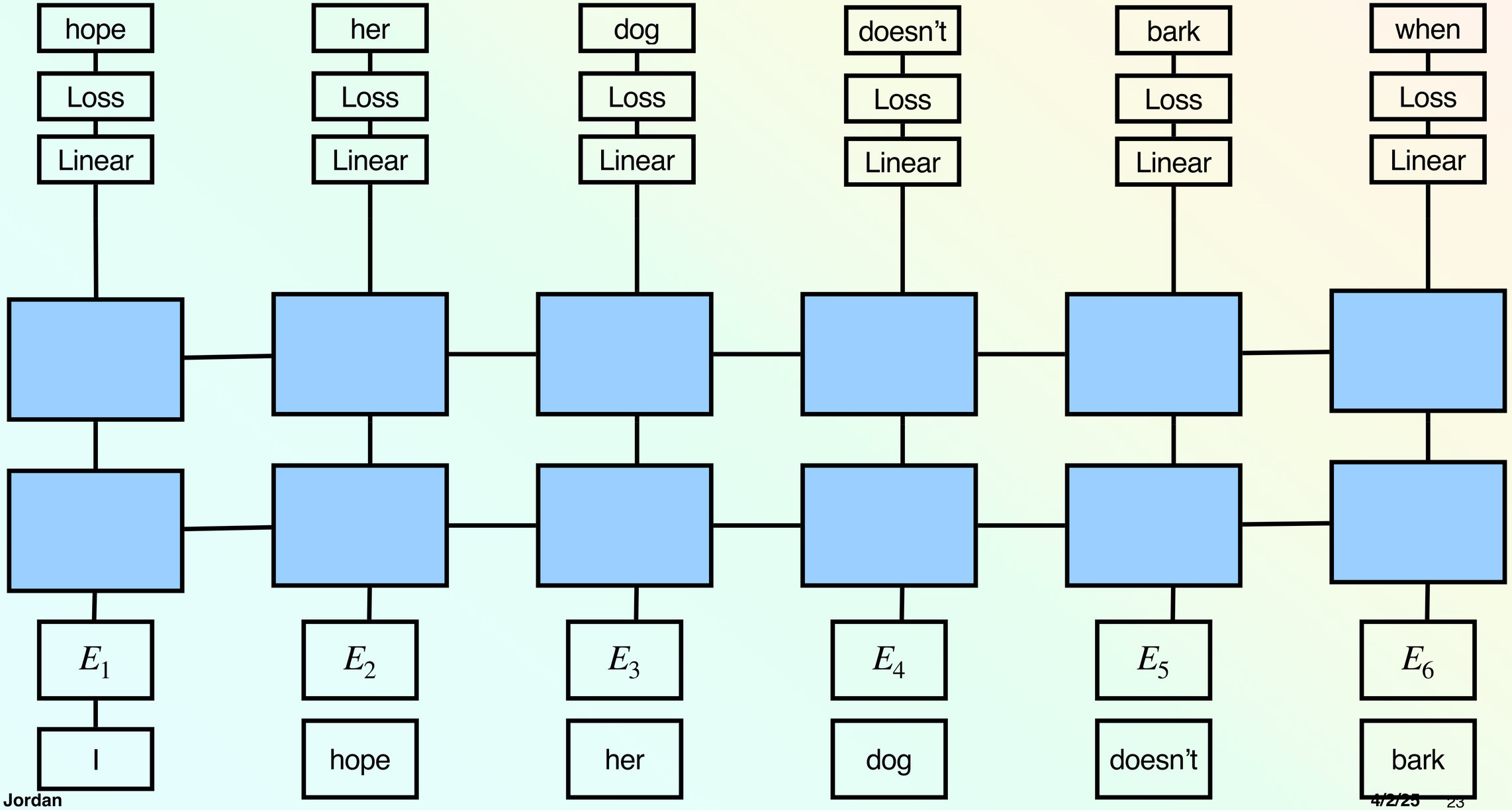

hope
Loss
Linear
her
Loss
Linear
E_1
E_2
I
hope

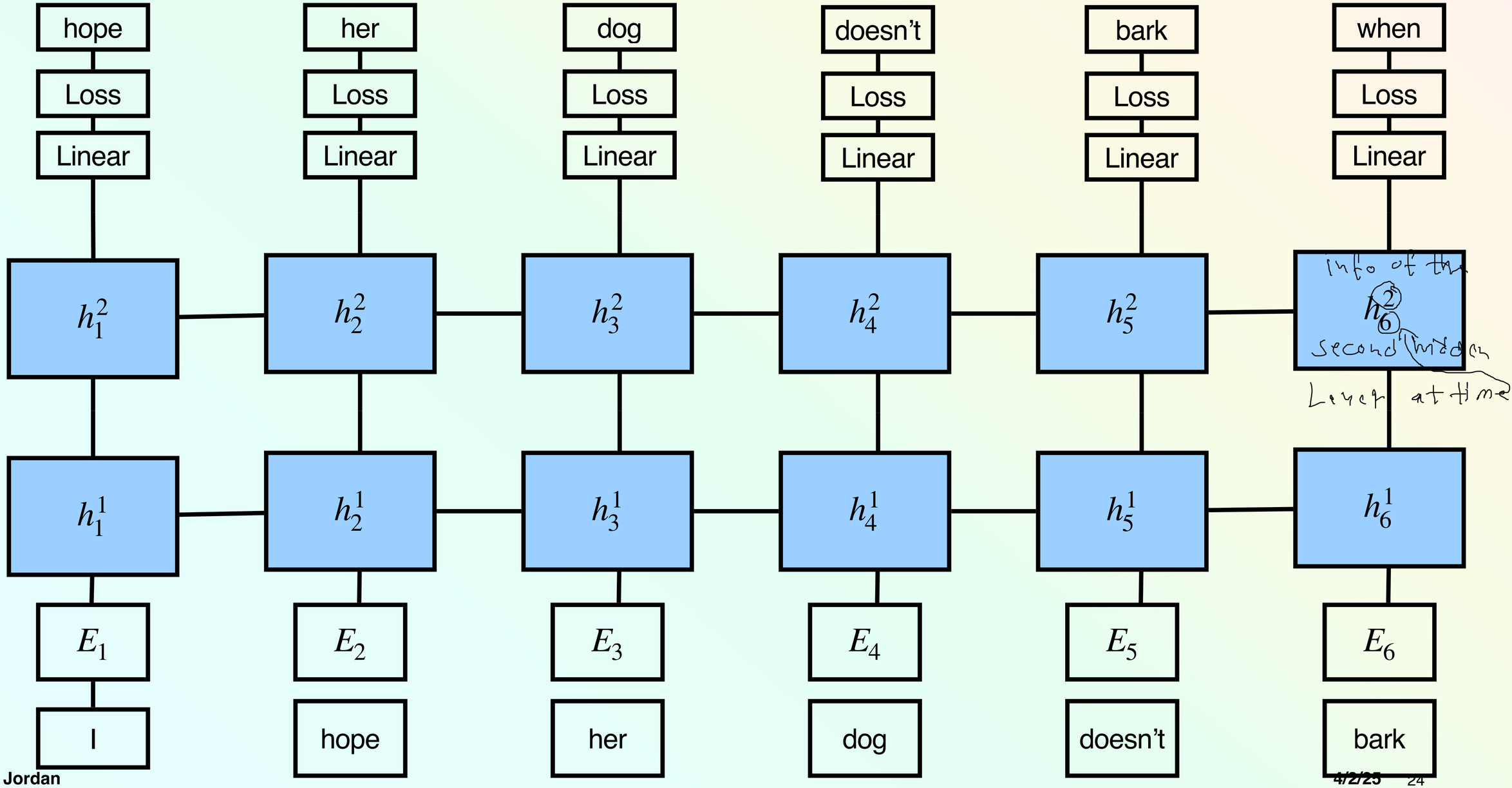

hope
Loss
Linear
her
Loss
Linear
dog
Loss
Linear
E_1
E_2
E_3
I
hope
her


hope
her
dog
doesn’t
Loss
Loss
Loss
Loss
Linear
Linear
Linear
Linear
E_1
E_2
E_3
E_4
I
hope
her
dog


hope
Loss
Linear
her
Loss
Linear
dog
Loss
Linear
doesn’t
Loss
Linear
bark
Loss
Linear
E_1
E_2
E_3
E_4
E_5
I
hope
her
dog
doesn’t

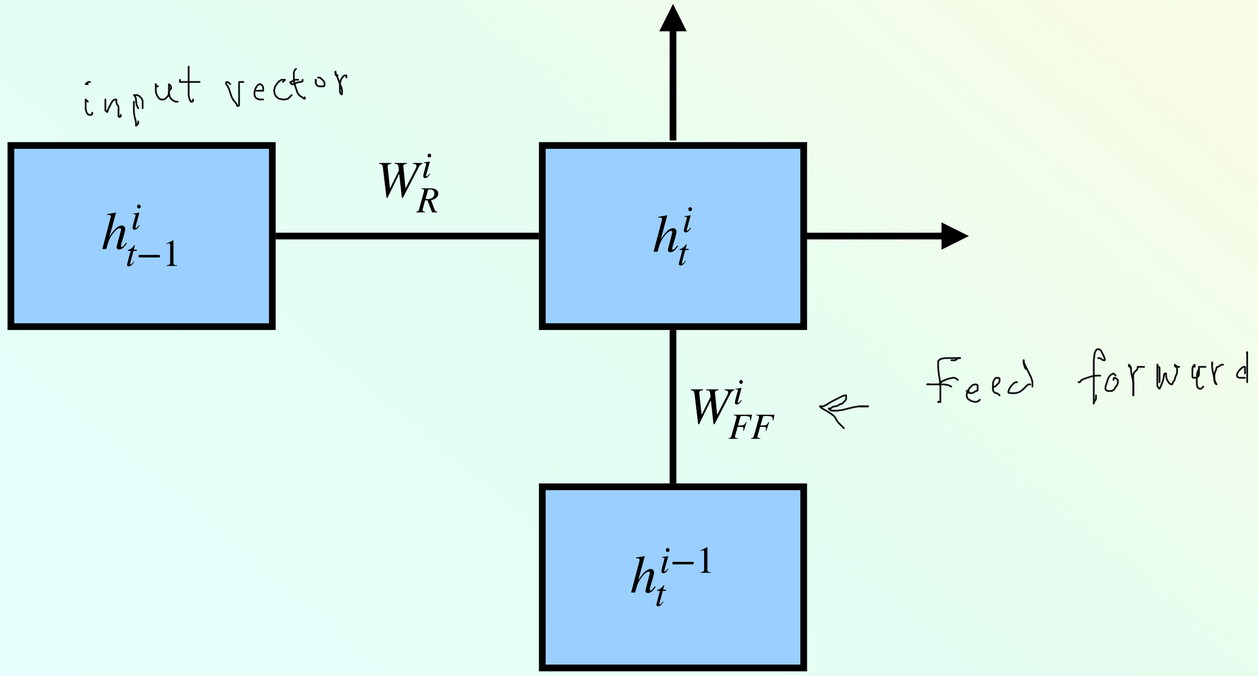

hope
Loss
Linear
her
Loss
Linear
dog
Loss
Linear
doesn’t
Loss
Linear
bark
Loss
Linear
when
Loss
Linear
E_1
E_2
E_3
E_4
E_5
E_6
I
hope
her
dog
doesn’t
bark


hope
her
dog
doesn’t
bark
when
Loss
Loss
Loss
Loss
Loss
Loss
Linear
Linear
Linear
Linear
Linear
Linear
E_1
E_2
E_3
E_4
E_5
E_6
I
hope
her
dog
doesn’t
bark


hope
her
dog
doesn't
bark
when
Loss
Loss
Loss
Loss
Loss
Loss
Linear
Linear
Linear
Linear
Linear
Linear
h_1^2
h_2^2
h_3^2
h_4^2
h_5^2
h_6^2
info of the
second word
Layer at time
h_1^1
h_2^1
h_3^1
h_4^1
h_5^1
h_6^1
E_1
E_2
E_3
E_4
E_5
E_6
I
hope
her
dog
doesn't
bark

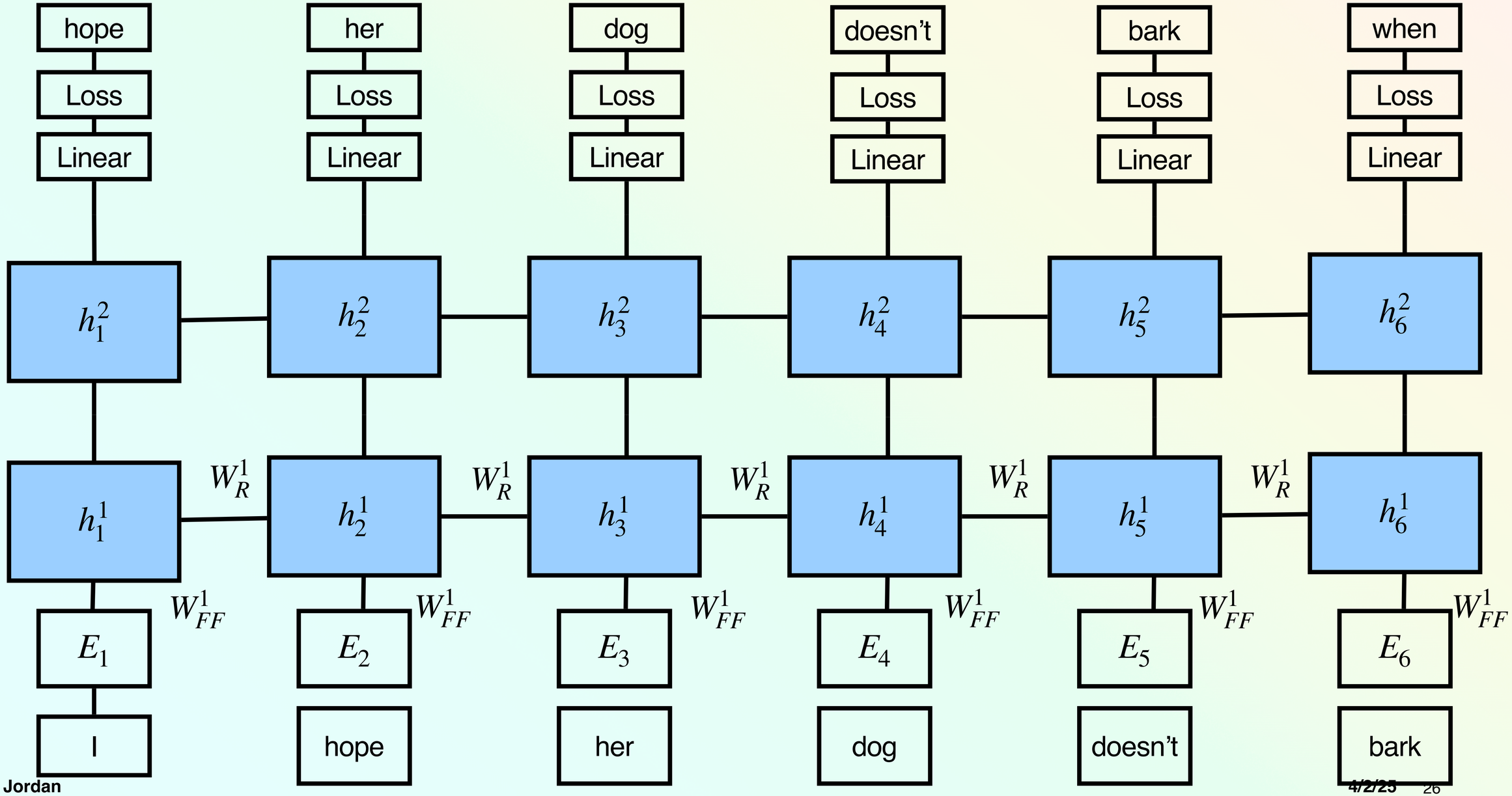
RNNS

LAYER

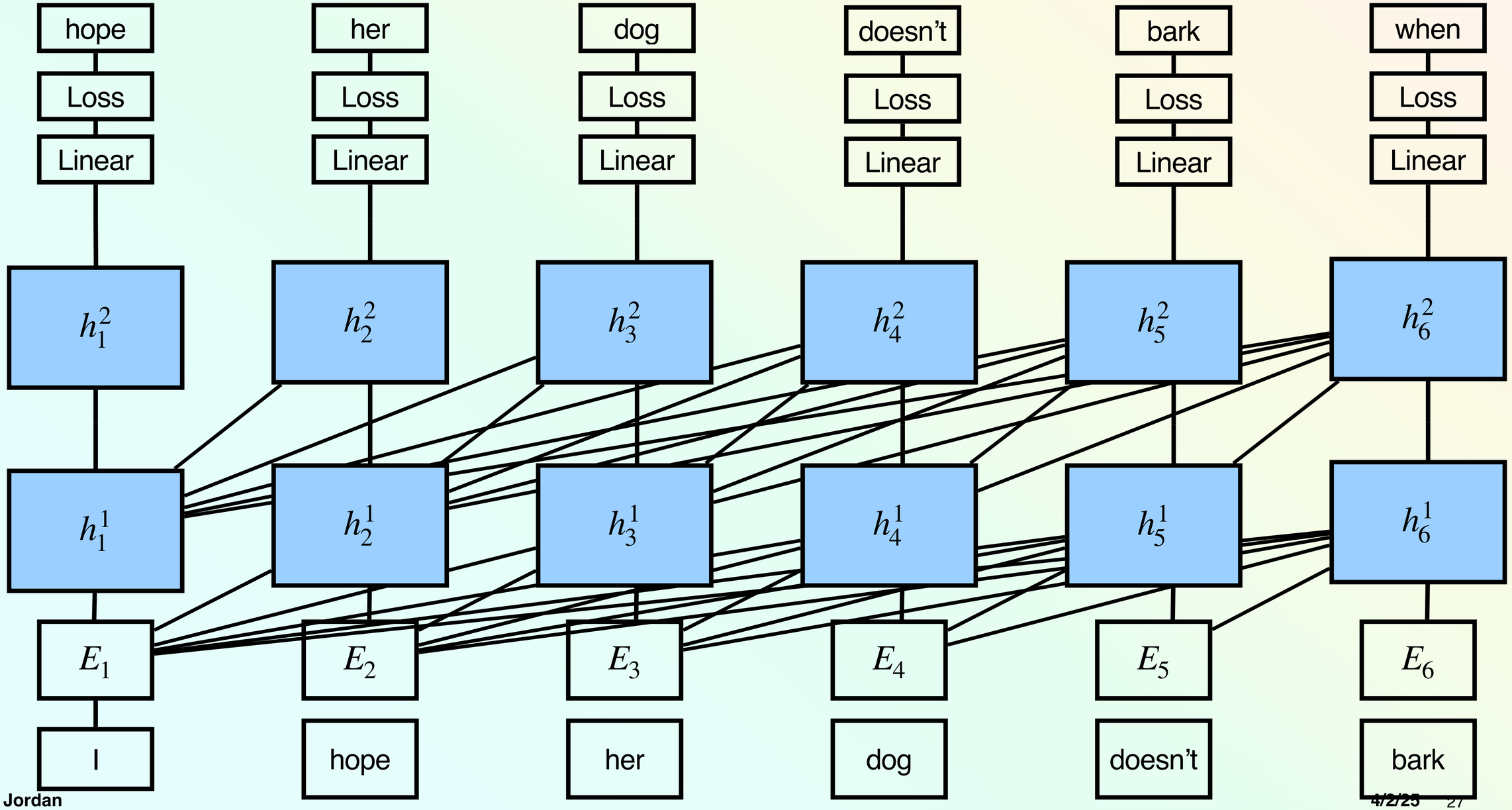


$$h_t^i = f^i(h_t^{i-1}, h_{t-1}^i, W_{FF}^i, W_R^i) = \sigma\left(h_t^{i-1} {W_{FF}^i}^\top + h_{t-1}^i {W_R^i}^\top\right)$$

RNN Block Diagram



Transformer Block Diagram



RNNS

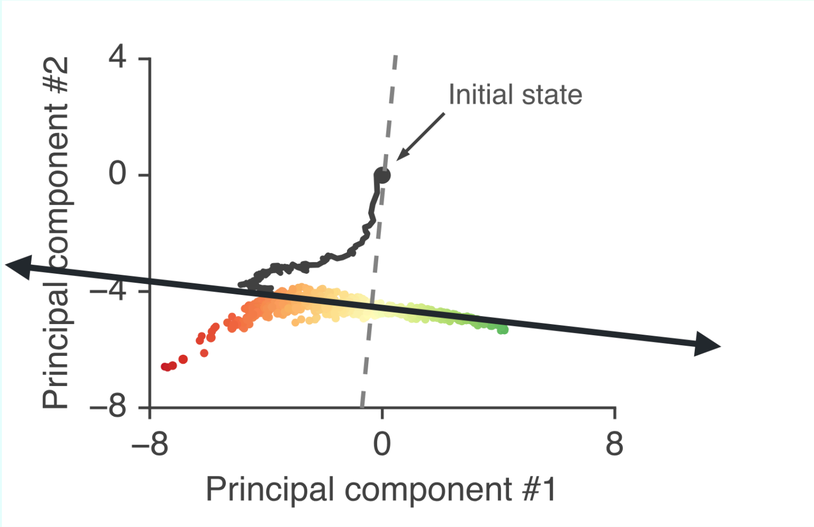
PROPERTIES

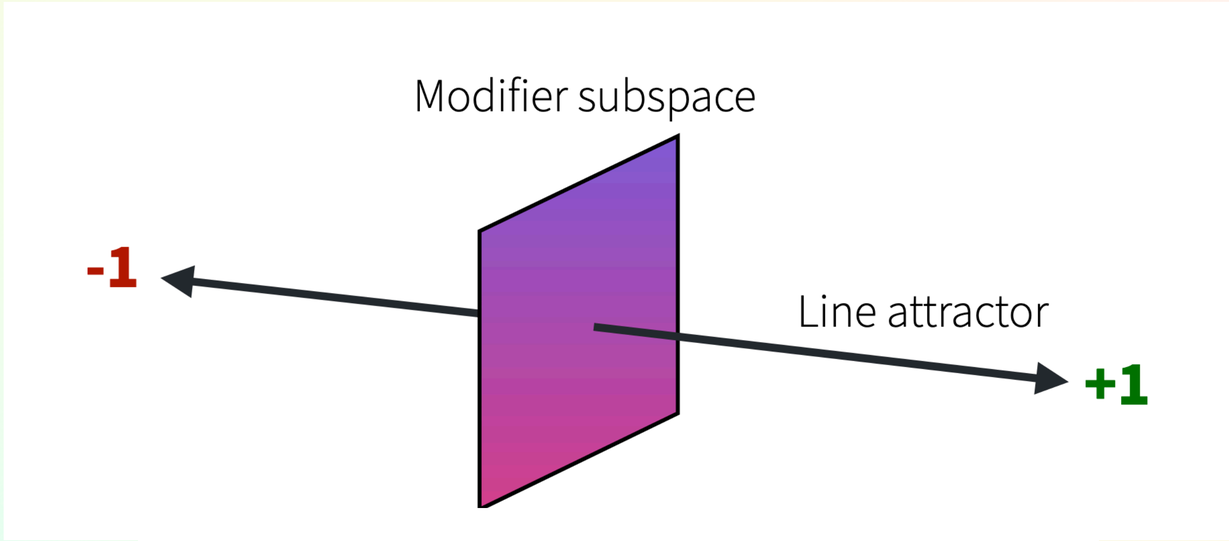
- Constant memory and computation requirements at inference time
  - It only needs to know h_{t-1} not all $h_{k<t}$
- Essentially, an MLP layer that is repeated at every time steps
- Need to learn some h_0^i (can be set to 0) ← could be an extra parameter to fine tune
- The hidden layer has to learn to remember all relevant information for the future
  - This is hard
  - “thinking” about all the information needed in the future all the time

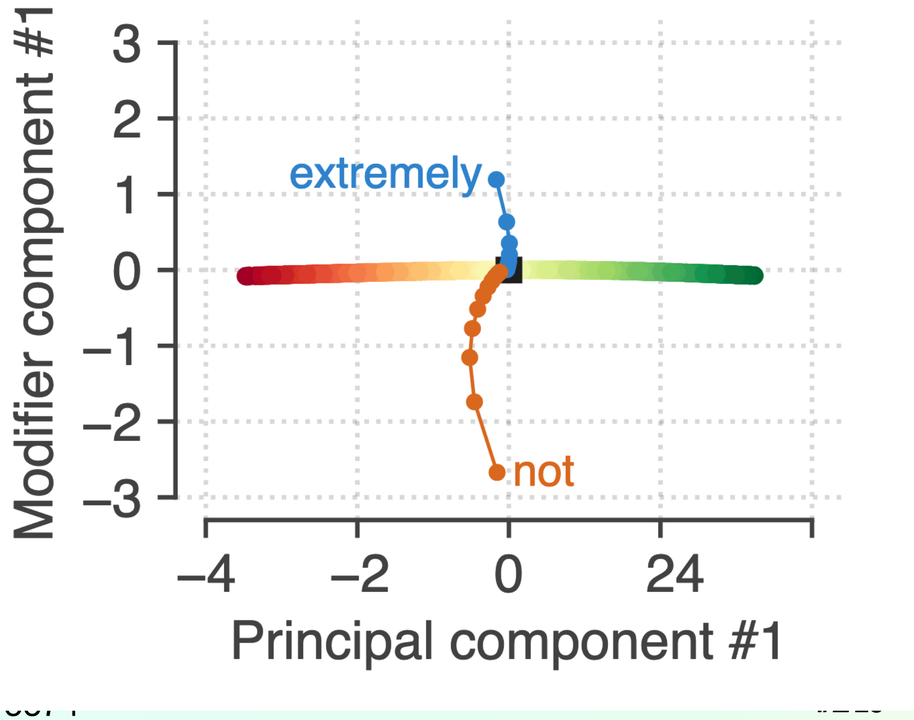
RNNS

WHAT THEY LEARN

Trained for sentiment analysis







RNNS

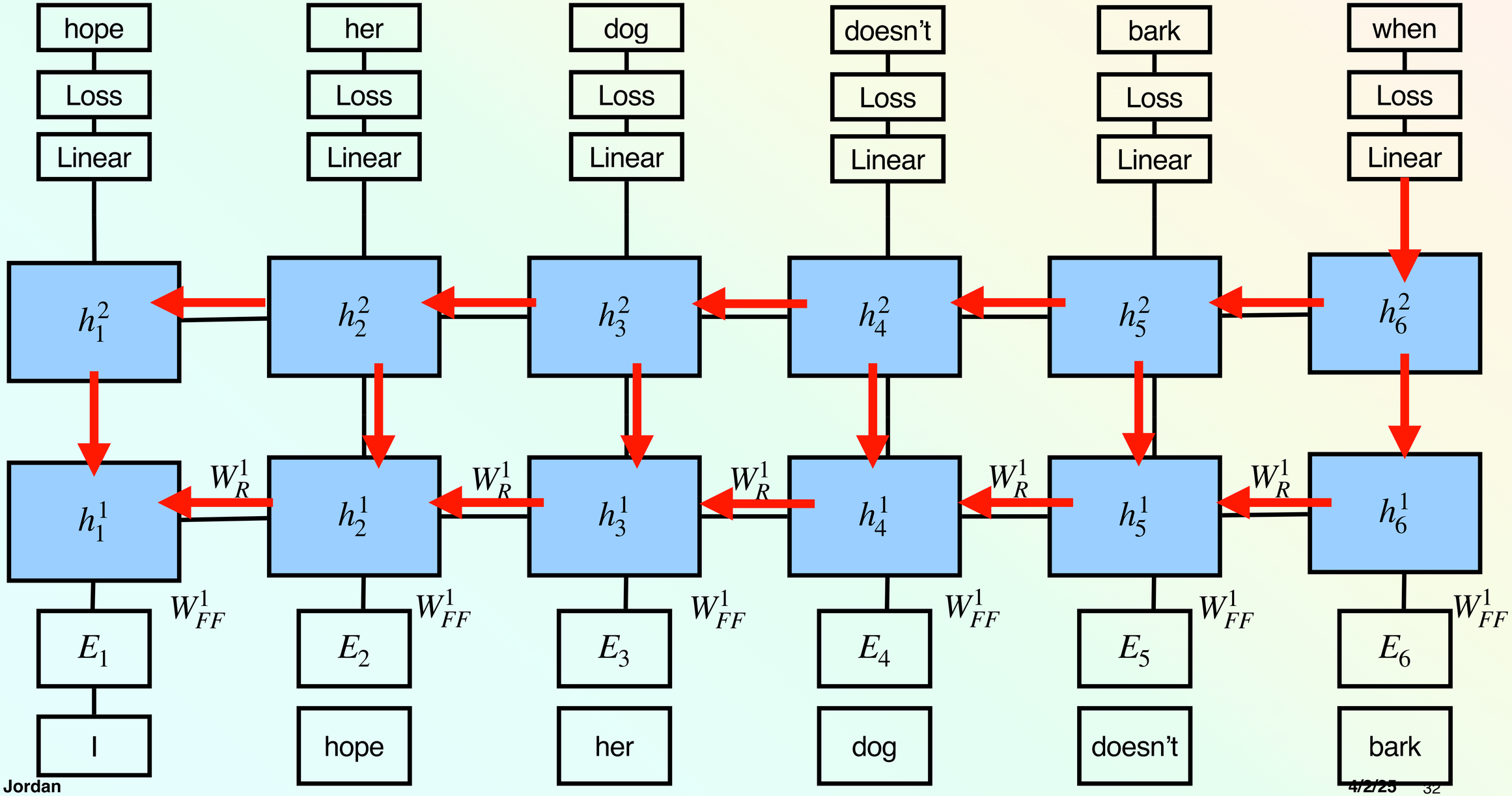
WHAT THEY LEARN

- Learns a context vector
- Context vectors get modified based on new inputs
- Context decays to some baseline representation
  - Slowly forgetting past inputs

RNNS

TRAINING

- Need to identify long-term relationships
- Each time step is like having another layer in an MLP
  - 500 length sequence is a 500 layer MLP
- **Vanishing and Exploding Gradients!**


hope
her
dog
doesn't
bark
when
Loss
Linear
h_1^2
h_2^2
h_3^2
h_4^2
h_5^2
h_6^2
h_1^1
h_2^1
h_3^1
h_4^1
h_5^1
h_6^1
W_R^1
W_{FF}^1
E_1
E_2
E_3
E_4
E_5
E_6
I
hope
her
dog
doesn't
bark

LONG SHORT-TERM MEMORY

LSTM RNN

- Problem: the derivative for h_1^i can vanish when coming from h_T^i
- Idea: create path through the network that has less shrinking on derivatives over time

Save some value for the hidden unit

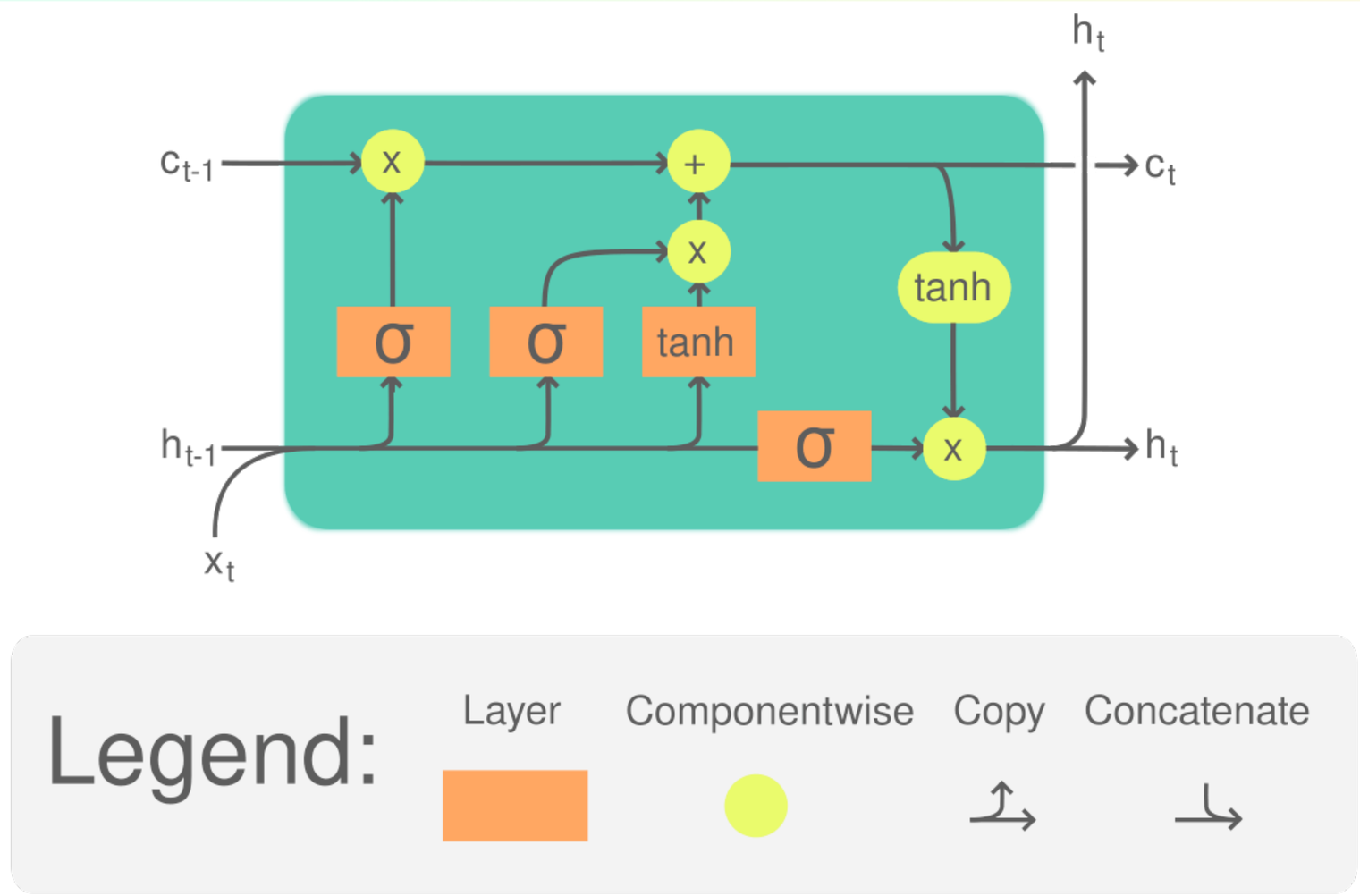
Add to this value with new information

When previous history becomes irrelevant, forget it (set it to saved value to 0)

Conditionally output relevant parts of the hidden information.

LONG SHORT-TERM MEMORY

LSTM RNN



LONG SHORT-TERM MEMORY

LSTM RNN

c_t — contains the value that has less shrinkage over time

h_t — outputs the hidden state of the recurrent layer

$$c_t = c_{t-1} \odot F_t + I_t \odot \tilde{C}_t$$

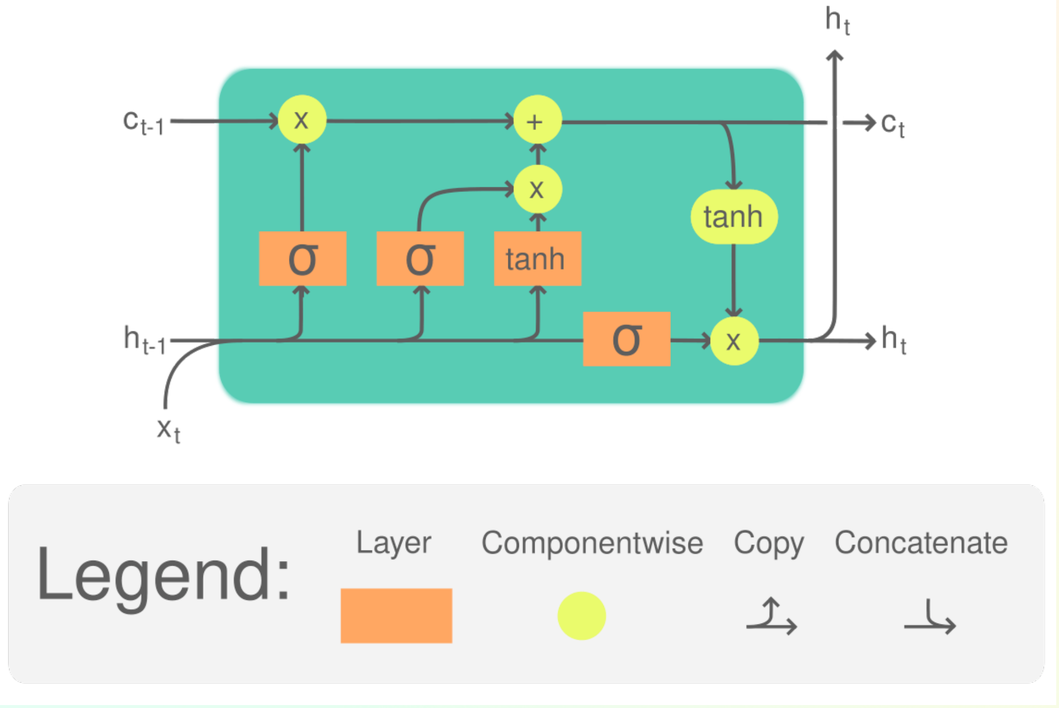
$F_t = \text{sigmoid}([h_{t-1}, x_t]W_F^\top)$ — forget gate

$I_t = \text{sigmoid}([h_{t-1}, x_t]W_I^\top)$ — increment gate

$\tilde{C}_t = \tanh([h_{t-1}, x_t]W_C^\top)$ — increment value

$h_t = \tanh(c_t) \odot O_t$

$O_t = \text{sigmoid}([h_{t-1}, x_t]W_O^\top)$ — output gate



LONG SHORT-TERM MEMORY

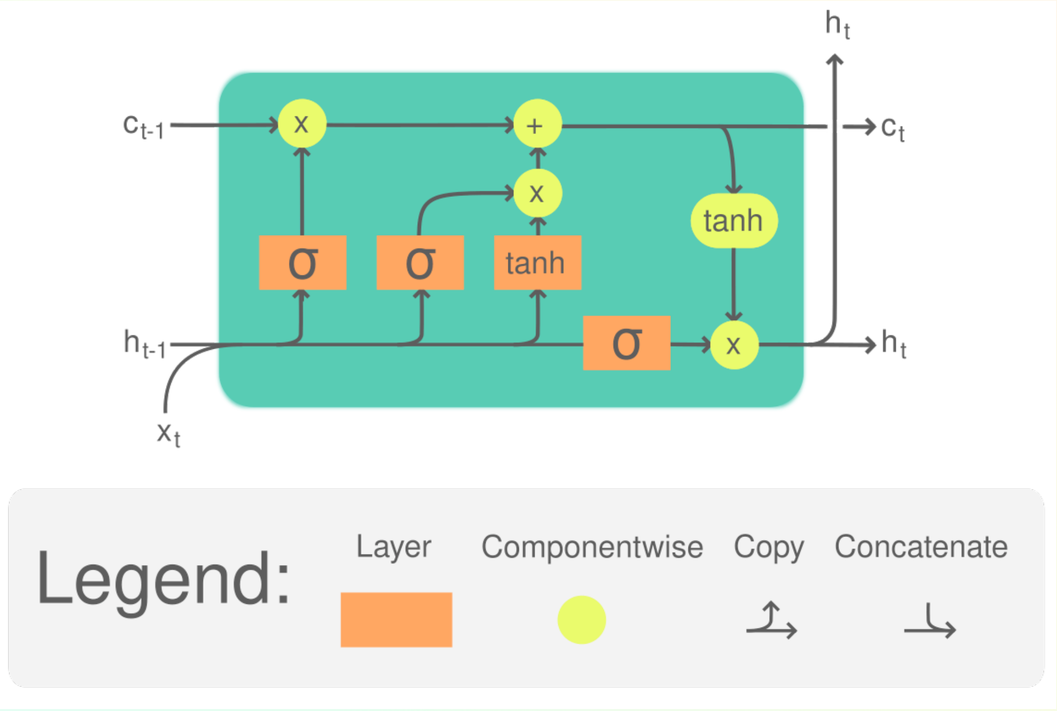
LSTM RNN

$$c_t = c_{t-1} \odot F_t + I_t \odot \tilde{C}_t$$

If $F \approx 1$ the derivative does not shrink much

$F \approx 0$ means prior information was not useful, and the derivative can be small.

LSTMs can learn relationships over longer sequences more easily than RNNs.



RECCURENT NEURAL NETWORKS

SUMMARY

- Constant run time and memory at inference
- Stores all important information of the past in a single vector
- Optimizing over long sequences is difficult (vanishing/exploding gradients)
- LSTMs help with vanishing gradients

Transformer models can learn longer sequences.

Transformer models are parallelizable

RNN-type layers have to be computed sequentially over the sequence (slow to train)

NEXT CLASS

Reinforcement Learning