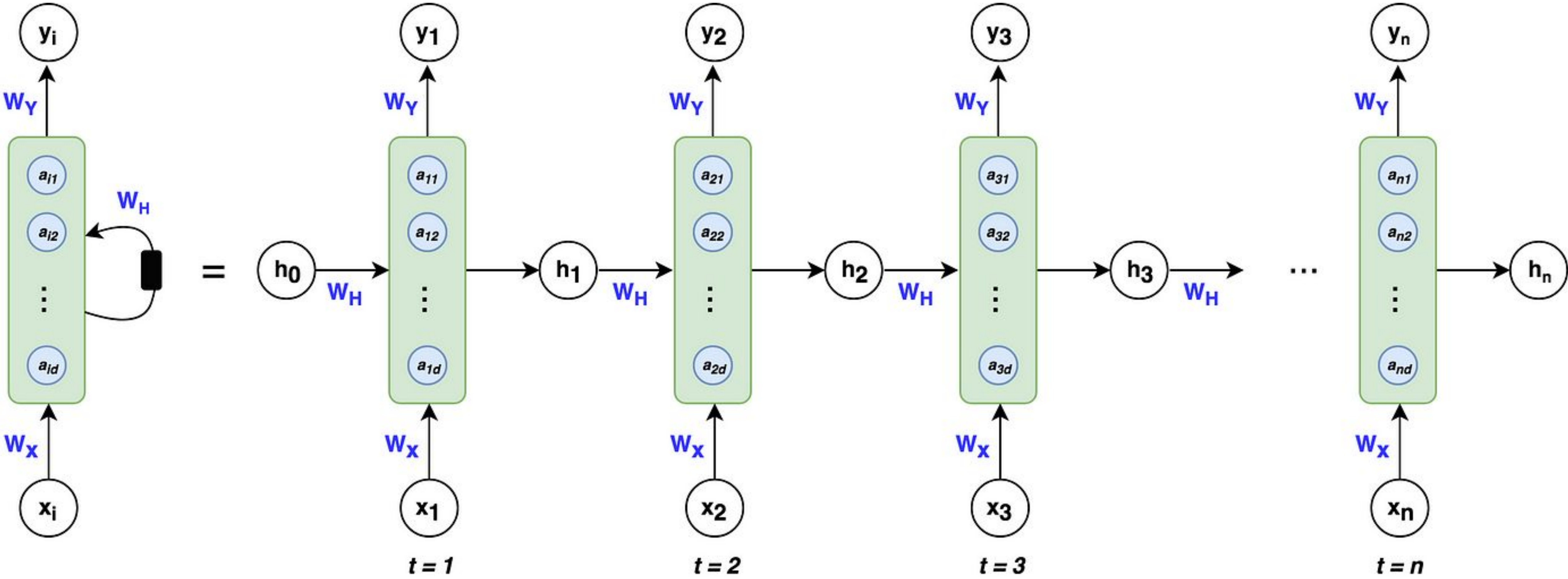
Recurrent Neural Networks (RNNs)

Naeemullah Khan

naeemullah.khan@kaust.edu.sa

KAUST Academy

King Abdullah University of Science and Technology



yi
Wy
Wh
Wx
xi
ai1
ai2
aid
=
h0
h1
h2
h3
hn
y1
y2
y3
yn
x1
x2
x3
xn
a11
a12
a1d
a21
a22
a2d
a31
a32
a3d
an1
an2
and
t = 1
t = 2
t = 3
t = n

Table of Contents

Motivation

أكاديمية كاوست
KAUST ACADEMY

Why Do We Need RNNs?

- Traditional feedforward networks don't handle **sequential data** effectively.
- Many applications (**language modeling, time-series prediction, speech recognition**) require memory of past inputs.
- RNNs enable **temporal dynamic behavior** by maintaining hidden states.

Examples Where Order Matters:

- Translating "I am happy" vs "Happy I am"
- Predicting next stock price based on past trends
- Understanding a sentence word-by-word

Key Idea

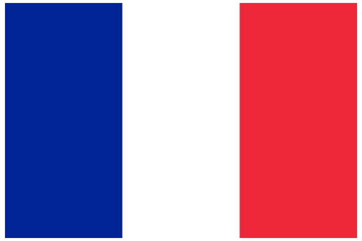
Add a feedback loop to remember previous computations — introducing memory into neural networks.

Learning Outcomes

By the end of this session, you should be able to:

- Understand the structure and working of Recurrent Neural Networks (RNNs)
- Recognize various RNN architectures (One-to-Many, Many-to-One, Many-to-Many)
- Explain the concept of shared parameters in RNNs
- Evaluate RNN limitations and how advanced models improve them
- Explore potential future directions of sequential models

Traditional Language Models



J'ai vu le match de foot

Sequence	$P(\text{Sequence})$
I saw the game of soccer	4.5 e-5
$\boxed{\text{I saw the soccer game}}$	6.0 e-5
I saw the soccer match	4.6 e-5
Saw I the game of soccer	2.6 e-9

N-grams

KAUST ACADEMY أكاديمية كاوست

$$P(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$ → Bigrams

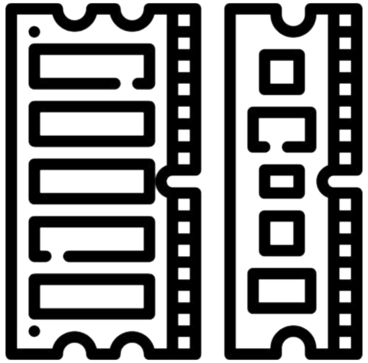
$$P(w_3|w_1, w_2) = \frac{\text{count}(w_1, w_2, w_3)}{\text{count}(w_1, w_2)}$$ Trigrams

$$\boxed{P(w_1, w_2, w_3) = P(w_1) \times P(w_2|w_1) \times P(w_3|w_2)}$$

- Large N-grams to capture dependencies between distant words
- Need a lot of space and RAM

Summary

- N-grams consume a lot of memory
- Different types of RNNs are the preferred alternative



What is an RNN?

A neural network with loops — allowing information to persist.

Core Elements:

- **Hidden State** h_t: captures memory of previous inputs
- **Input** x_t, **Output** y_t
- Same weights used across time steps (parameter sharing)

Mathematical Formulation:

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$$
$$y_t = W_{hy}h_t + b_y$$

This recurrence allows information to propagate through time.

Advantages of RNNs

Nour was supposed to study with me. I called her but she did not answer

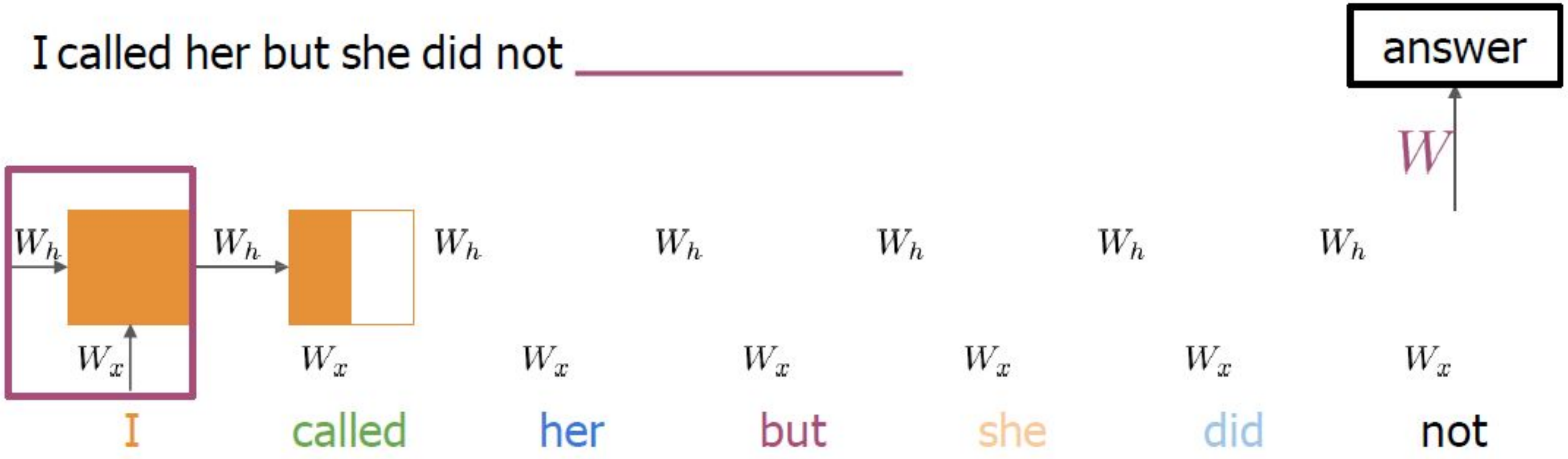
want
respond
choose
want
have
ask
attempt
answer
know

RNNs look at every previous word

Similar probabilities with trigram

RNNs Basic Structure

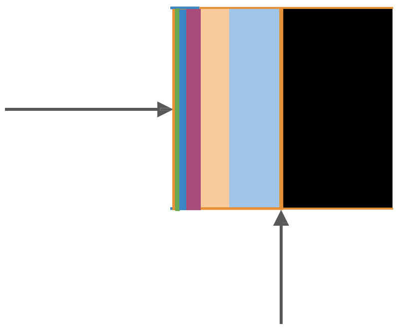
I called her but she did not ___________



Learnable parameters

Summary

- RNNs model relationships among distant words
- In RNNs a lot of computations share parameters

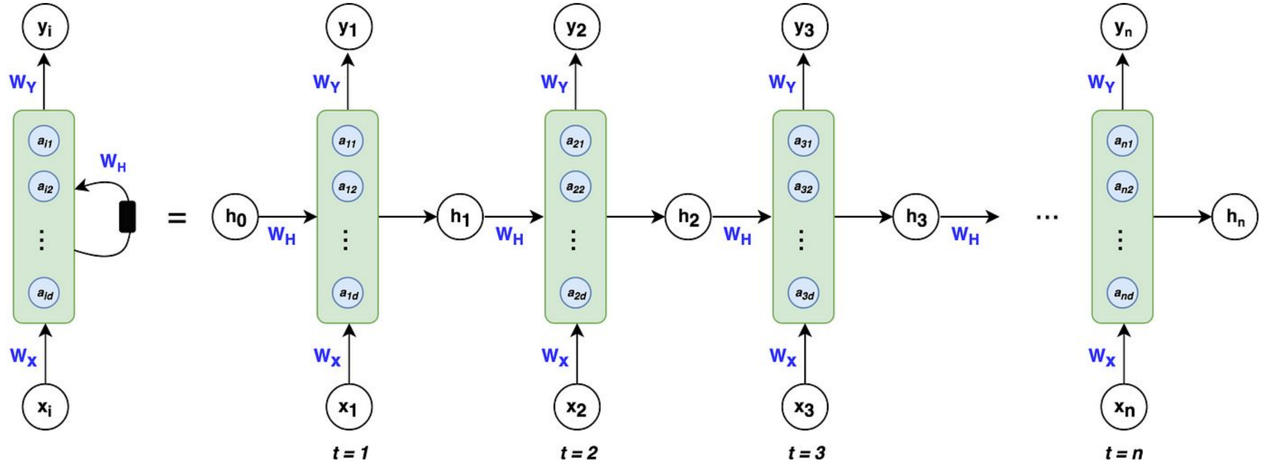


RNN Architecture: Unrolled View

Unrolled RNN:
An RNN is essentially a chain of repeating neural network modules, one for each time step.

Each time step shares the same parameters:

- Input x_t
- Hidden state h_t
- Output y_t



Key Idea

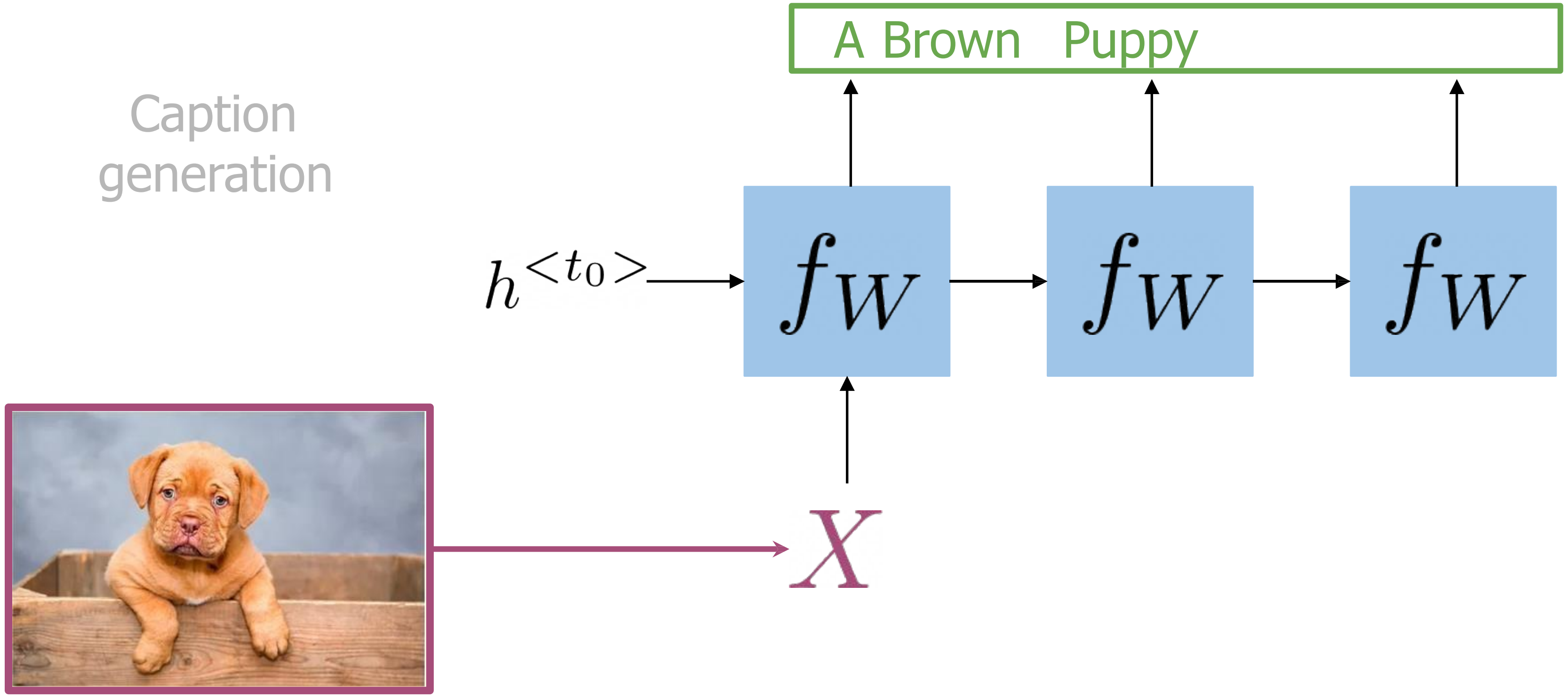
Temporal representation without increasing parameter count!

One to One

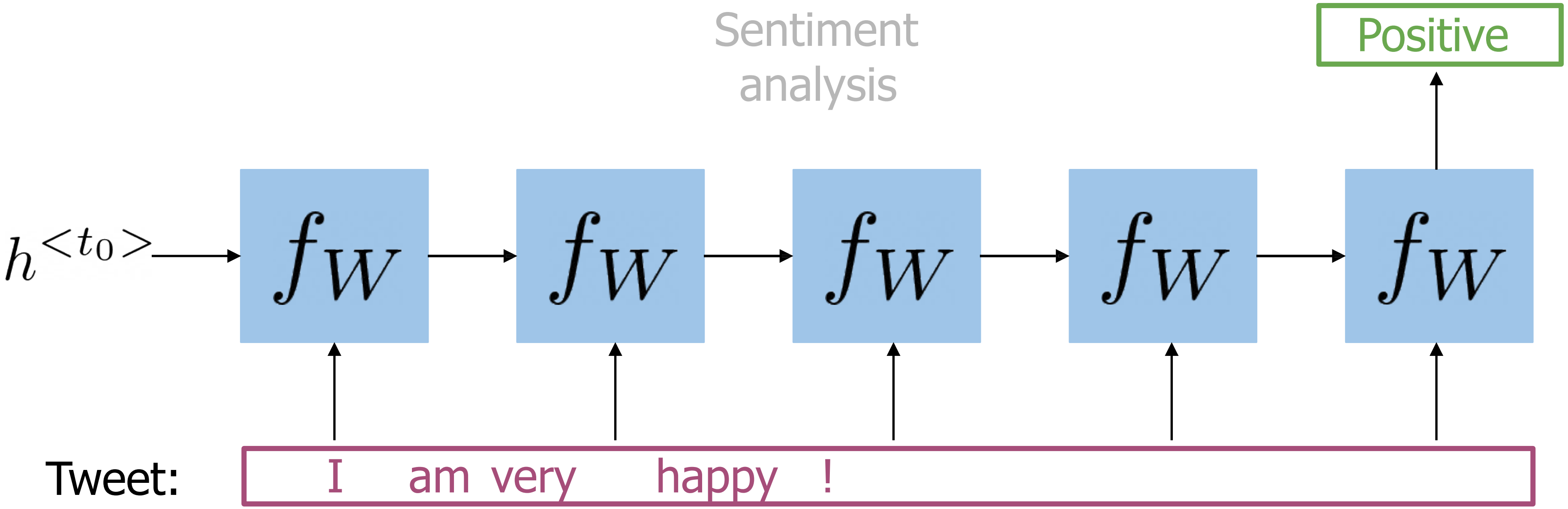

LaLiga
Santander
Real Valladolid
0
1
Real Madrid
Real Zaragoza
0
4
Real Madrid
Atletico Madrid
0
1
Real Madrid
$h^{<t_0>}$
f_W
X

One to Many

Caption generation

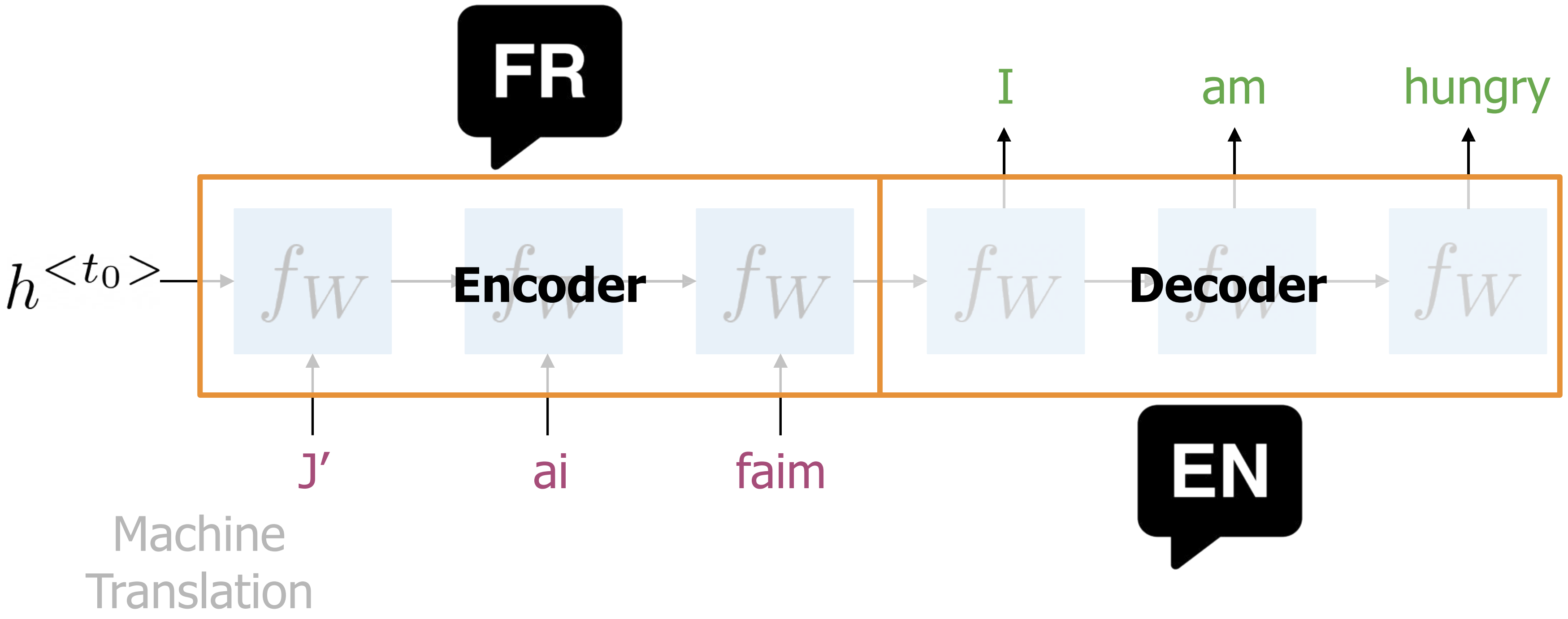


Many to One



Sentiment
analysis
Positive
$h^{<t_0>}$
f_W
f_W
f_W
f_W
f_W
Tweet:
I am very happy !

Many to Many

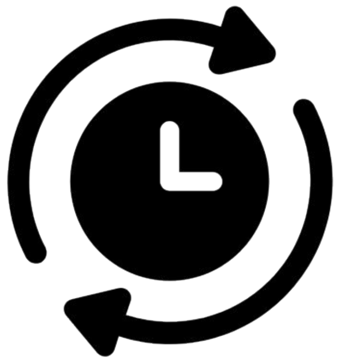

FR
I
am
hungry
$h^{<t_0>}$
f_W
Encoder
f_W
f_W
f_W
Decoder
f_W
f_W
J'
ai
faim
EN
Machine
Translation

Summary

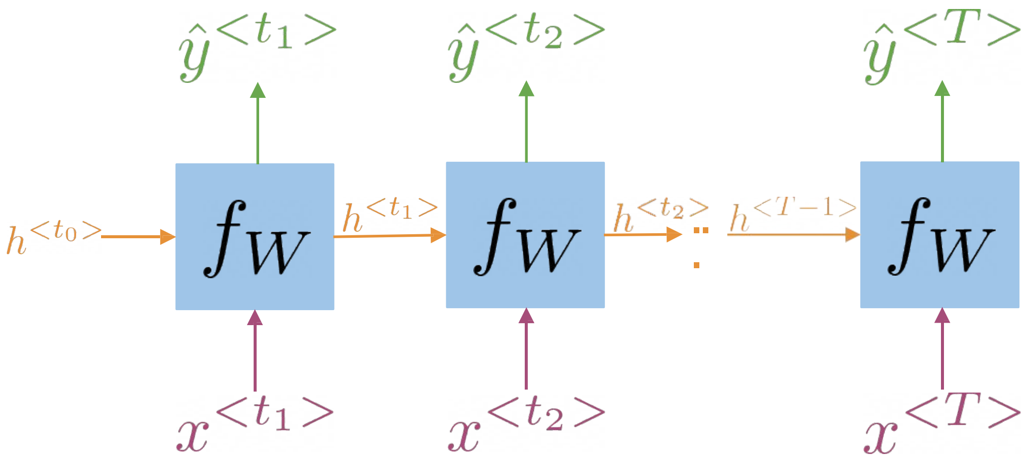
- RNNs can be implemented for a variety of NLP tasks
- Applications include Machine translation and caption generation

Math in Simple RNNs

- How RNNs propagate information (Through time!)
- How RNNs make predictions



A Vanilla RNN

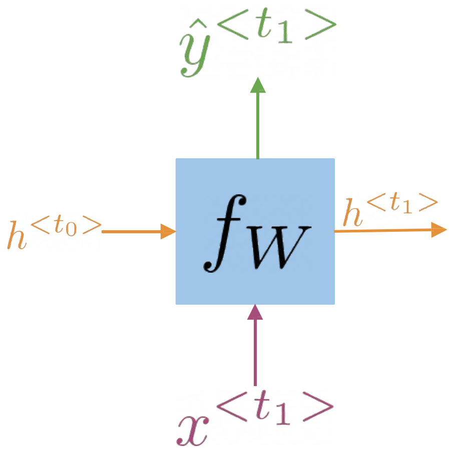


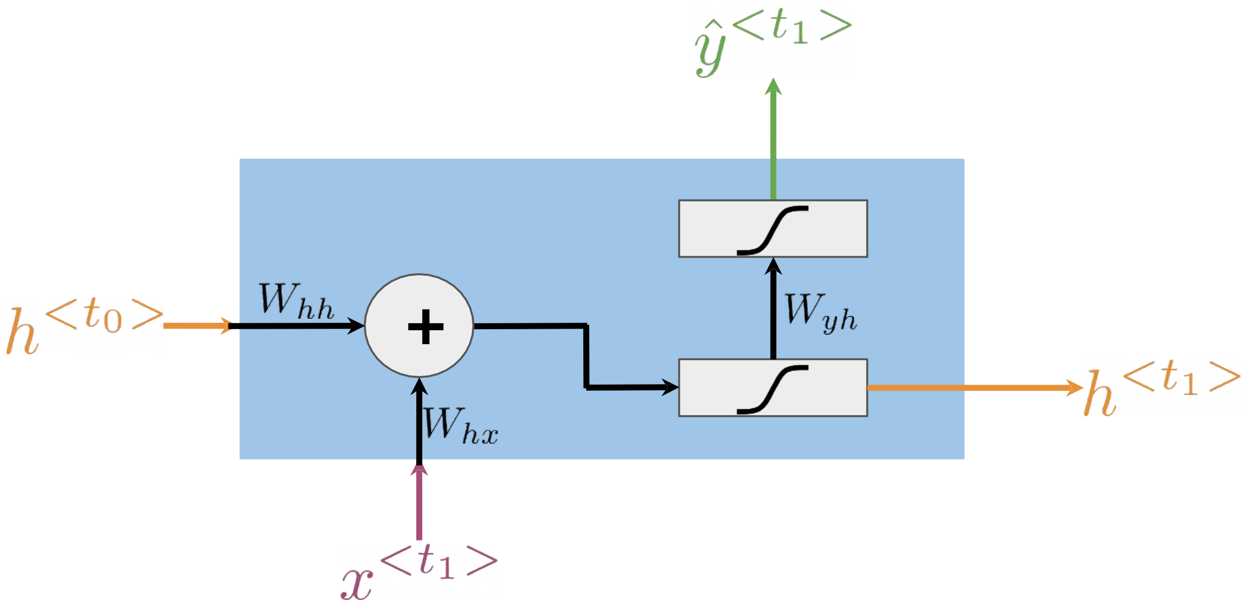
$$h^{<t>} = g(\boxed{W_h[h^{<t-1>}, x^{<t>}]} \boxed{+ b_h})$$

$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$

$$\boxed{h^{<t>} = g(W_{hh}h^{<t-1>} + W_{hx}x^{<t>} + b_h)}$$

A Vanilla RNN





$$h^{<t>} = g(\boxed{W_{hh}h^{<t-1>} + W_{hx}x^{<t>}} + b_h)$$

$$\hat{y}^{<t>} = g(W_{yh}h^{<t>} + b_y)$$

RNN Training Challenges

Backpropagation Through Time (BPTT):

- Training RNNs involves **unfolding** the network across time steps.
- Standard backpropagation is applied through this unrolled structure.

RNN Training Challenges

Problems:

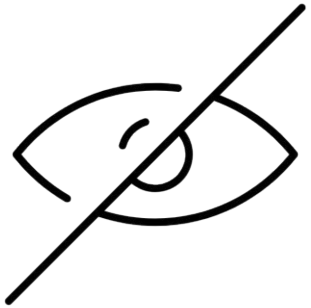
- **Vanishing Gradients:** Gradients shrink as they are propagated back, making it hard to learn long-term dependencies.
- **Exploding Gradients:** Gradients grow exponentially, leading to unstable updates.

Solutions Preview (covered in future modules):

- Use of LSTM and GRU architectures
- Gradient Clipping

Summary

- Hidden states propagate information through time
- Basic recurrent units have two inputs at each time: $h^{<t-1>}$ $x^{<t>}$



Applications of RNNs

أكاديمية كاوست
KAUST ACADEMY

Natural Language Processing

- Language modeling
- Named Entity Recognition
- Machine Translation

Time-Series Forecasting

- Stock prediction
- Weather forecasting

Audio Processing

- Speech recognition
- Music generation

Cognitive Modeling

- Simulating memory in brain-like systems

Limitations of RNNs

- Sequential computation — hard to parallelize
- Forget long-term dependencies
- Slow training due to sequential nature
- Struggle with varying-length sequences

Key Developments

- **LSTM and GRU:** Designed to address memory and gradient issues
- **Transformers:** Non-recurrent, highly parallelizable models

Future Directions: What's Beyond Vanilla RNNs?

- **LSTM (Long Short-Term Memory):** Overcomes vanishing gradients
- **GRU (Gated Recurrent Unit):** Simpler than LSTM, efficient gating
- **Attention Mechanisms:** Focus on relevant parts of the input sequence
- **Transformers & Self-Attention:** Replace recurrence with parallelizable attention
- **Neural ODEs:** Model continuously evolving hidden states

Hybrid Models:

- Combine RNNs, CNNs, and Attention for complex tasks (e.g., video, multimodal text)

Summary

- RNNs introduce memory into neural nets for sequence modeling
- Use shared weights across time steps
- Architectures like One-to-Many, Many-to-One fit various tasks
- RNNs face training and memory limitations
- Advances like LSTM, GRU, and Transformers push beyond RNNs

References

These slides have been adapted from

- Younes Mourri & Lukasz Kaiser, Natural Language Processcing Specialization, DeepLearning.Ai

References

Core Papers:

- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*.
- Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., & Khudanpur, S. (2010). Recurrent neural network based language model. *Interspeech*.
- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*.
- Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to Sequence Learning with Neural Networks. *NeurIPS*.