CS231N Review Session: RNNs and Transformers

Emily Jin

May 2nd, 2025

Agenda

- Motivation
- RNNs
- Transformers
- RNNs vs Transformers
- Colab Notebook

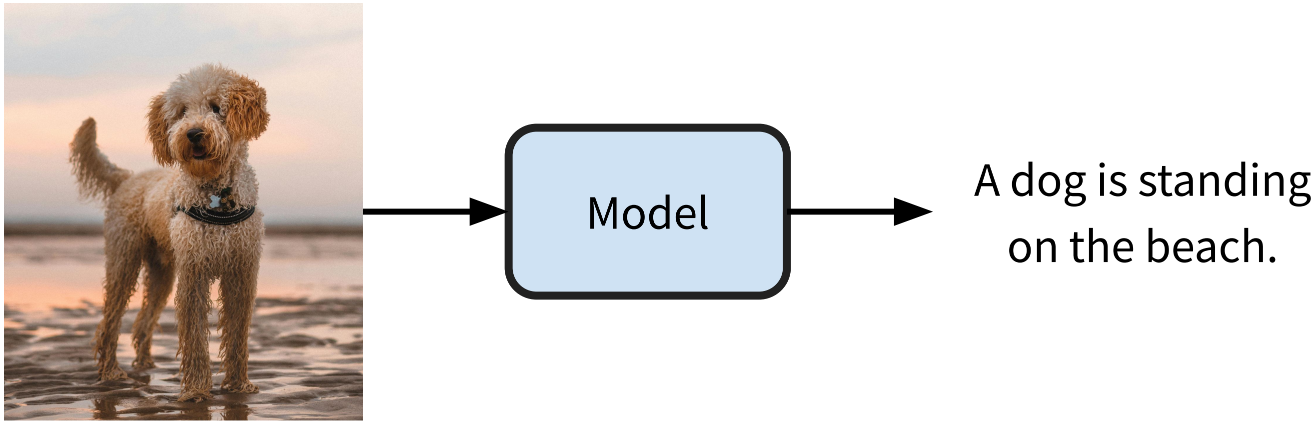
Motivation

Before RNNs and Transformers, we assume fixed-size inputs and outputs.

But many vision tasks require sequential processing.

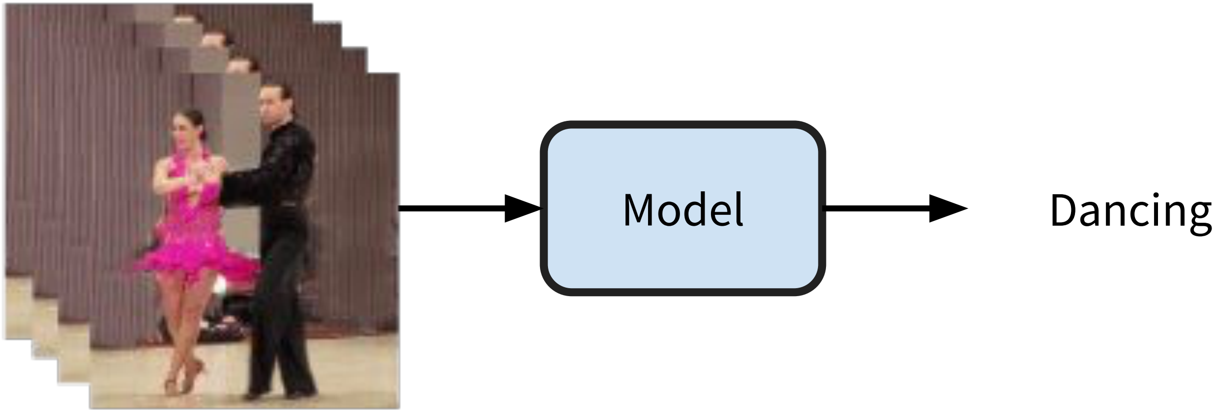
Motivation

Example: Image Captioning (one to many)



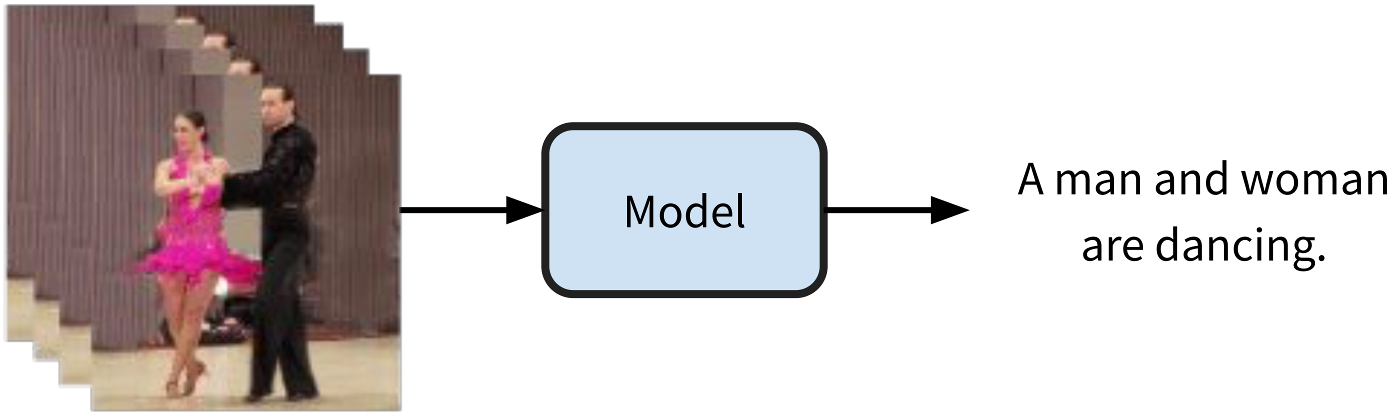
Motivation

Example: Activity Recognition (many to one)



Motivation

Example: Video Captioning (many to many)



Motivation

To solve these kinds of tasks, we need models that can:

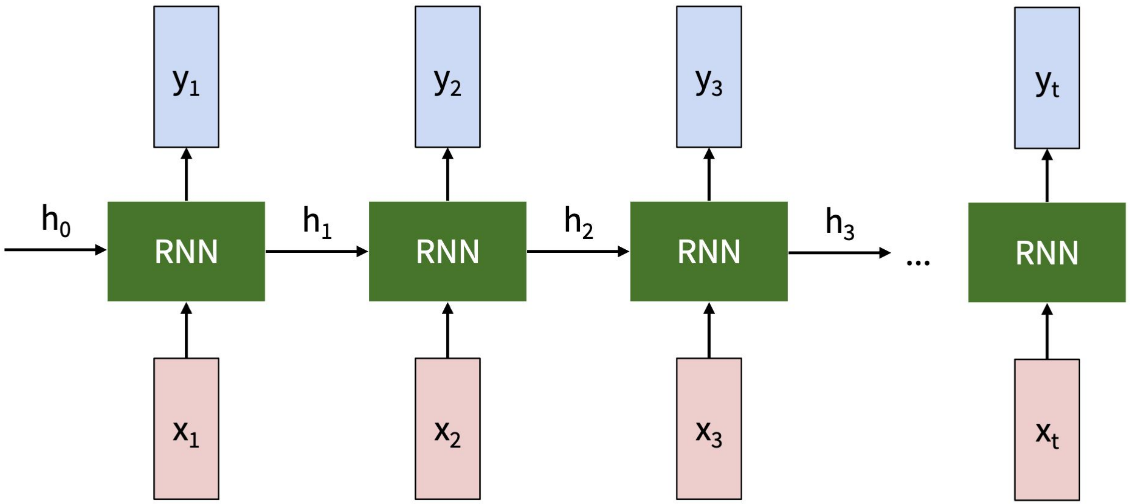
- Handle variable-length input and output sequences
- Preserve temporal structure and order
- Capture long-range dependencies

Some considerations include:

- Long-Range Dependencies: How do models learn which past inputs are relevant?
- Parallelizability: Can the model be parallelized across time steps?
- Compute & Memory Use: How do compute/memory scale with sequence length?
- Inductive Bias: How well do models capture temporal/locality structure?

RNNs

Key Idea: RNNs process sequences one step at a time, maintaining a "internal state" that summarizes past inputs & is updated as the sequence is processed



RNNs

At every time step, we use the same function / parameters to update the hidden state, which allows us to process input sequences of arbitrary length.

$$\underbrace{\boxed{h_t}}_{\text{new state}} = \underbrace{\boxed{f_W}}_{\substack{\text{some function} \\ \text{with parameters W}}} (\underbrace{\boxed{h_{t-1}}}_{\text{old state}}, \underbrace{\boxed{x_t}}_{\substack{\text{input vector at} \\ \text{some time step}}})$$

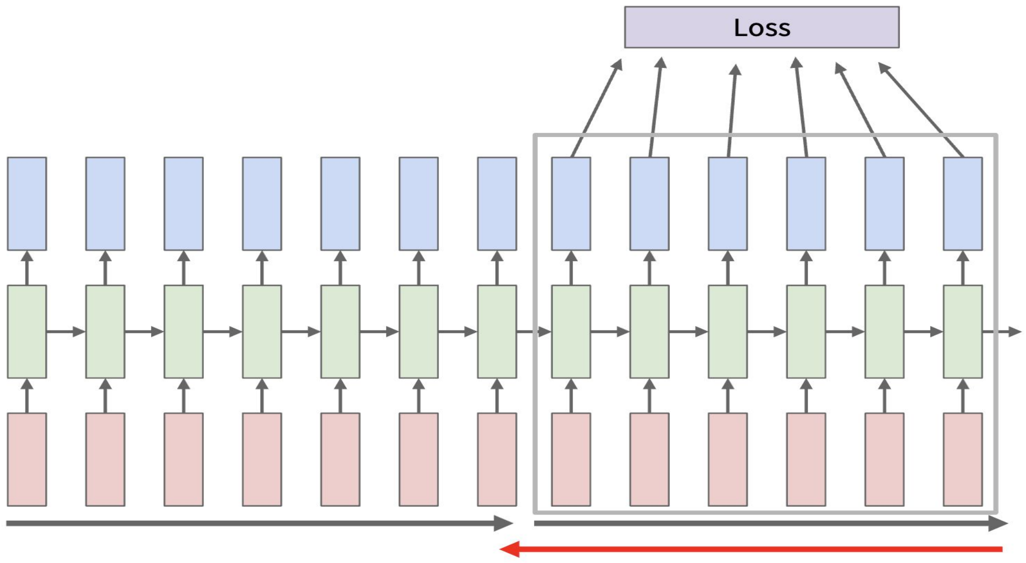
We use another function / parameters to decode the hidden state into an output, to generate output sequences.

$$\underbrace{\boxed{y_t}}_{\text{output}} = \underbrace{\boxed{f_{W_{hy}}}}_{\substack{\text{another function} \\ \text{with parameters } W_{hy}}} (\underbrace{\boxed{h_t}}_{\text{new state}})$$

RNNs

(Truncated) Backpropagation Through Time

Key Idea: Instead of backpropping through the entire sequence, we carry hidden states forward in time forever, but only backpropagate for a chunk



RNNs

Advantages

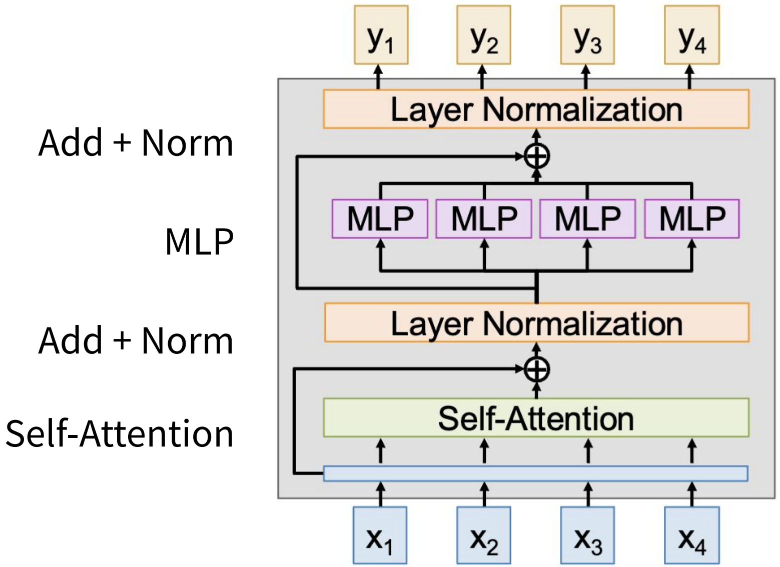
- Can process inputs of any length
- Each step can use information from previous steps (in theory)
- Model size is fixed, regardless of sequence length
- Shared weights across time → enforces temporal consistency

Disadvantages

- Slow training due to sequential / recurrent computation
- Hard to capture long-term dependencies
- Vanishing/exploding gradients
  - Gradient clipping (clip norm of gradient to a threshold)
  - LSTM / GRU (gating mechanisms help preserve / regulate flow of info over time)

Transformers

Key Idea: use self-attention to process all elements in parallel and let the model attend to most relevant parts of the input



Vaswani et al, “Attention is All You Need” NeurIPS 2017

Transformers

Self-Attention

Input Vectors: X

Queries: Q – what each token is looking for

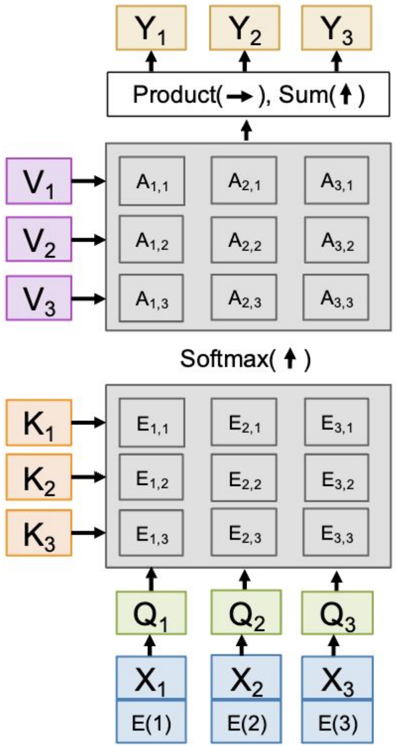
Keys: K – what each token offers

Values: V – information of each token

Compute attention scores by computing dot product between each query and the keys of all tokens + passing through softmax

Attention scores determine how much each token should pay attention to other tokens' values

Final Output: weighted sum of all values, based on attention

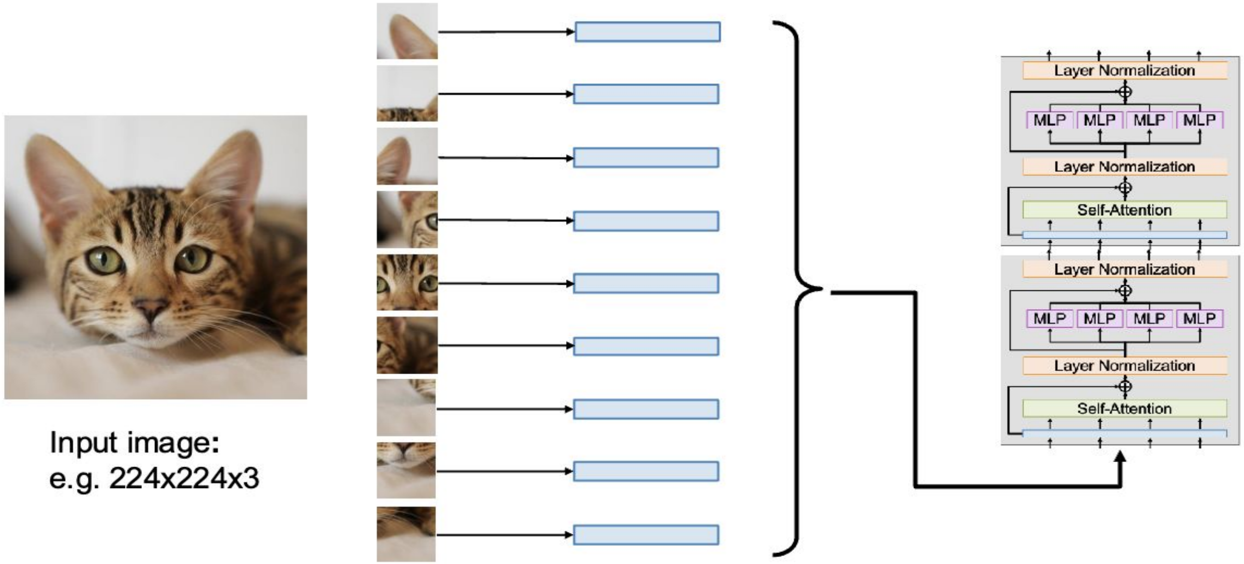


RNNs vs Transformers

	RNNs	**Transformers**
Long-Range Dependencies	Good in theory, but hard in practice	Good in practice, through self-attention over full input
Parallelizability	No – sequential computation across timesteps	Yes – process tokens in parallel
Compute & Memory Use	O(N), O(N)	O(N^2), O(N)
Inductive Bias	Strong – inherent temporal structure	Weak – needs to learn from data

Vision Transformers

Key Idea: treat images like sequences of patches, and apply the Transformer directly to those patches, using self-attention to model relationships between parts of the image.



Dosovitskiy et al, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”, ICLR 2021

Colab Notebook

https://colab.research.google.com/drive/1mC5CWwekbZ2NrYv6Zfpuv55z8DuOZXVP?usp=sharing