DDA3020 Machine Learning: L11 Recurrent Neural Networks and Transformer

Jicong Fan
SDS, CUHK-SZ

10/28/2025

Outline

1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

1. Sequential Data Analysis

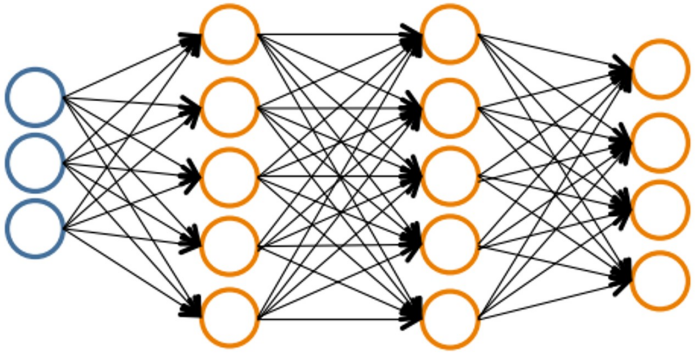
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants

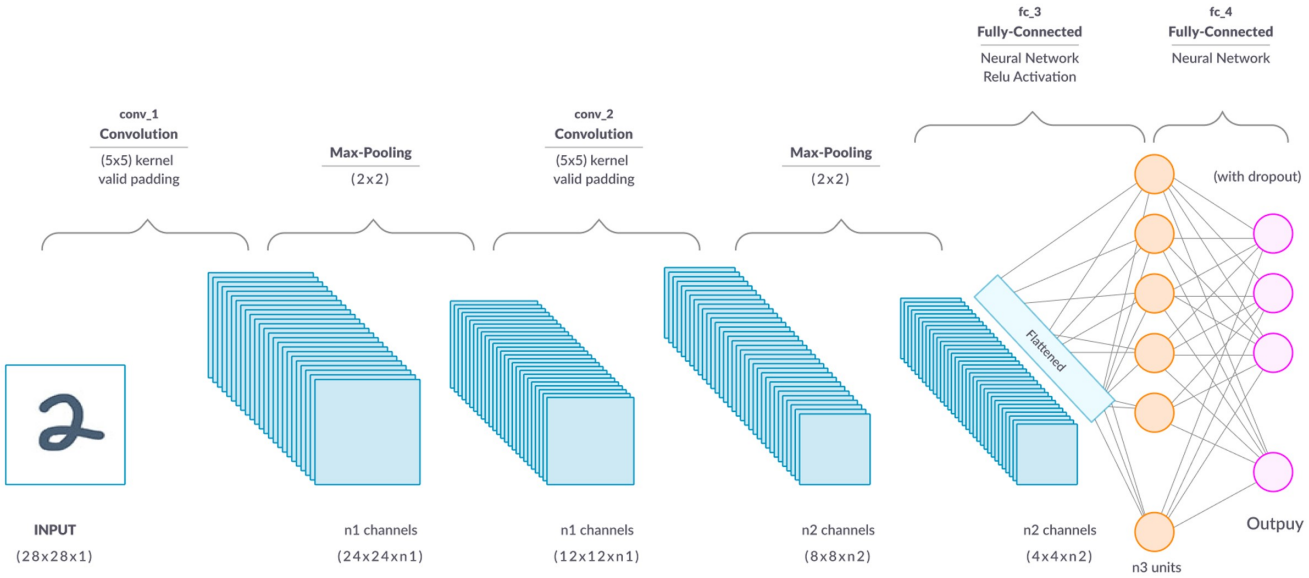
3. Transformer
   - Overview of Transformer
   - Attention Mechanism

4. Summary

Motivation

- Recall: MLP and CNN are for tabular data and images.





- What about sequential data?
  - Sequential data: data arranged in sequences where order matters.

Tasks Involving Sequential Data

Time series prediction

- Time series to time series

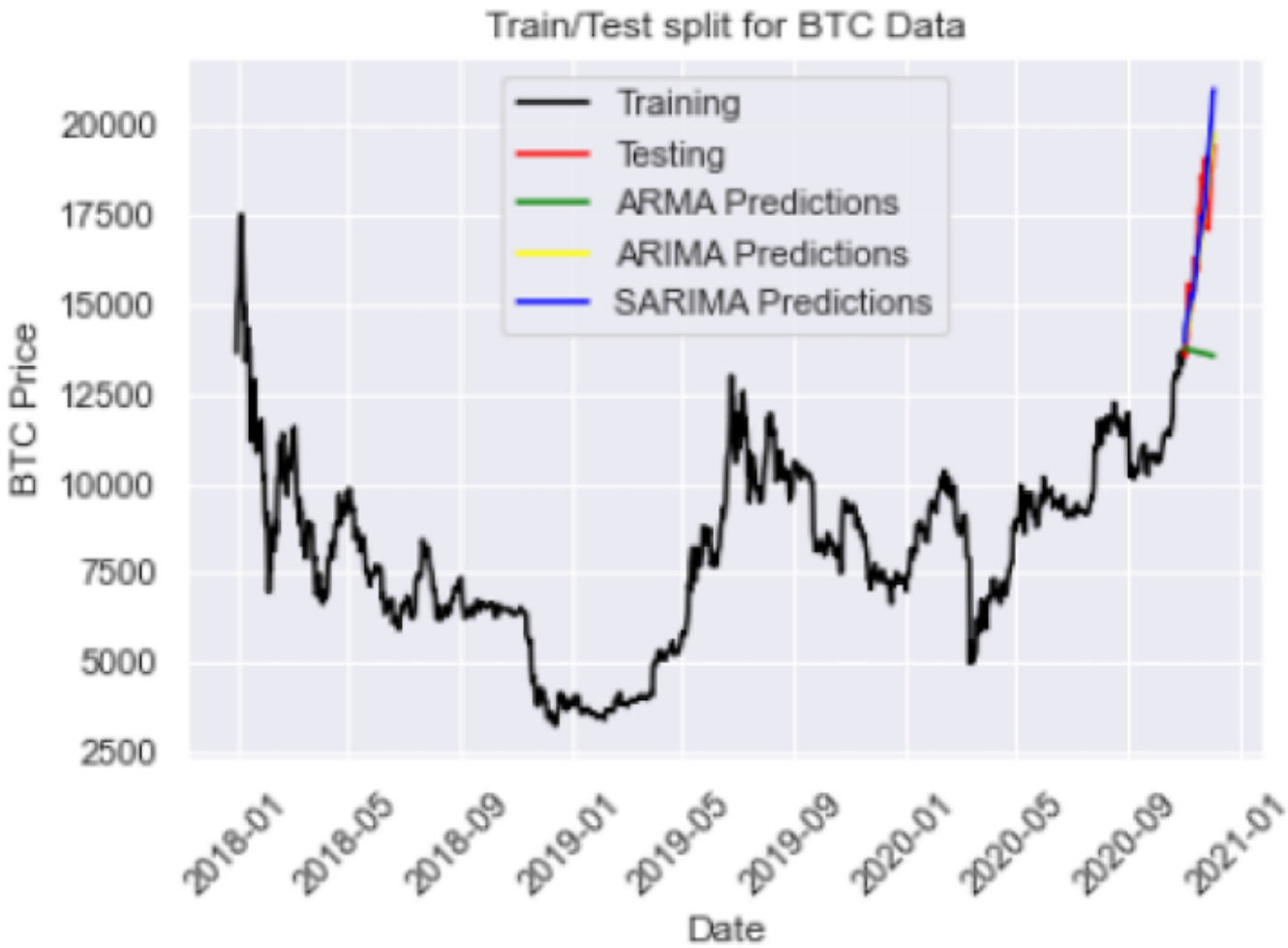
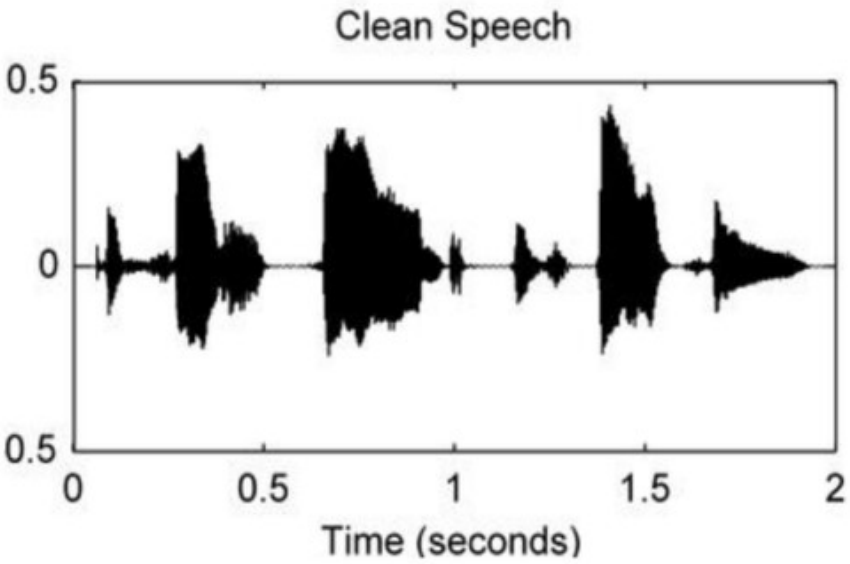


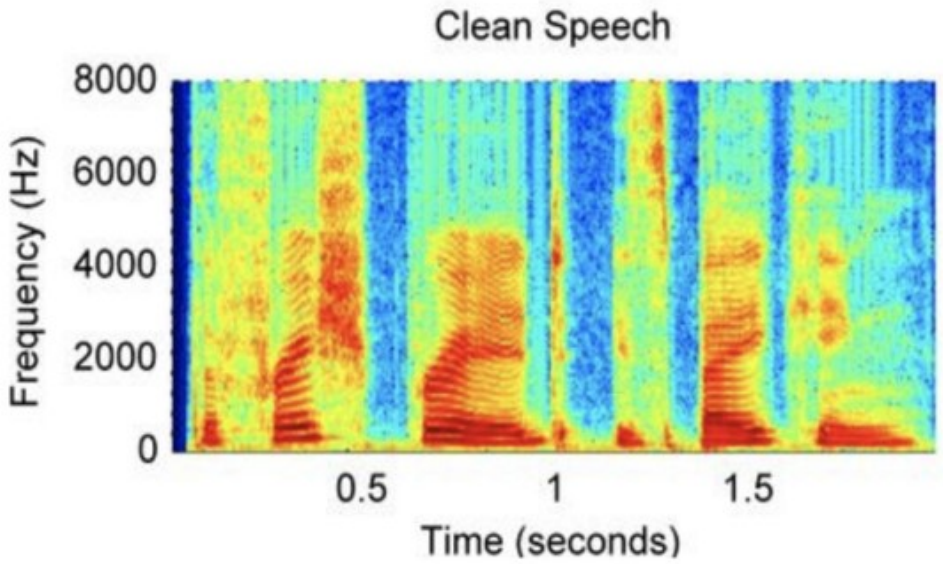
Image source: https://builtin.com/data-science/time-series-forecasting-python

Tasks Involving Sequential Data

Speech recognition

- Audio to text

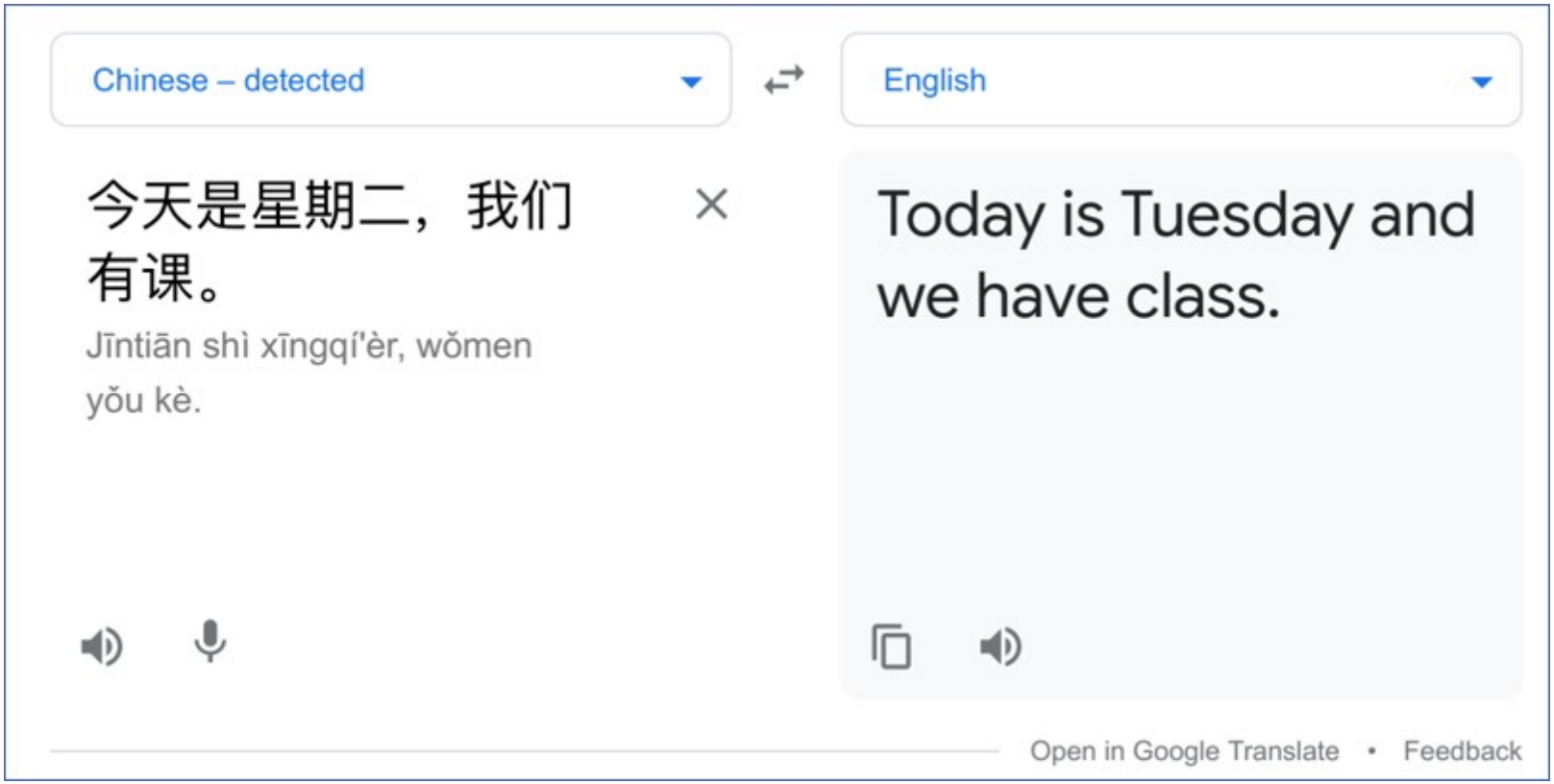




Tasks Involving Sequential Data

Machine translation

- Text to text



Tasks Involving Sequential Data

Image captioning

- Image to text

A woman is throwing a frisbee in a park.

A stop sign is on a road with a mountain in the background.

A group of people sitting on a boat in the water.

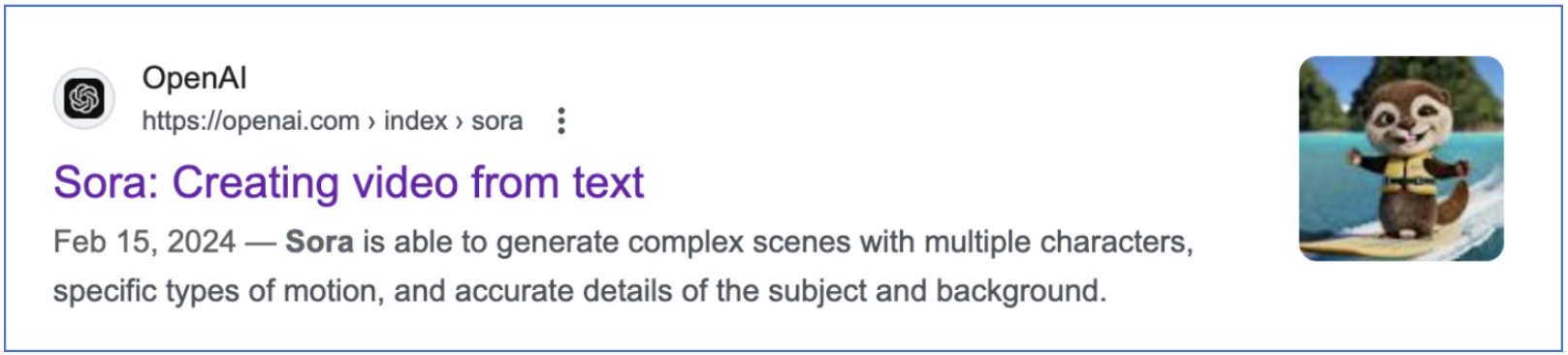
A giraffe standing in a forest with trees in the background.

Image credit: Nan Rosemary Ke

Tasks Involving Sequential Data

More examples of sequential data

- Text (language modeling, e.g. ChatGPT)
- Video (video generation/understanding, e.g. SORA)
- Biological data
  - Medical imaging
  - DNA sequences



1 Sequential Data Analysis

2 Recurrent Neural Network
- Basic RNN Architecture
- Training of Basic RNN
- LSTM
- Other Extensions of RNN (optional)
- Limitations of Basic RNN and Its Variants

3 Transformer
- Overview of Transformer
- Attention Mechanism

4 Summary

1. Sequential Data Analysis

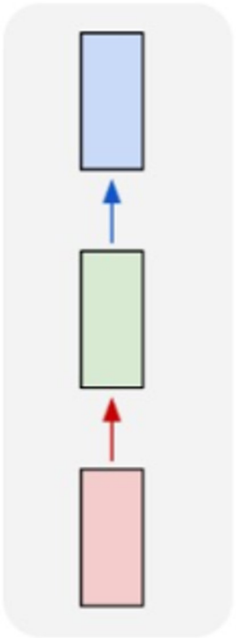
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants

3. Transformer
   - Overview of Transformer
   - Attention Mechanism

4. Summary

Neural Networks

one to one

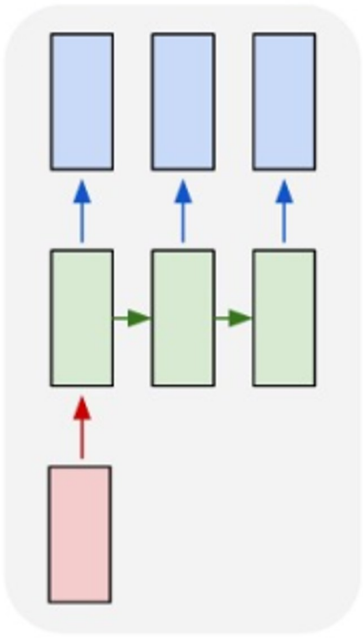


lion

tiger

e.g. **Image classification**
Image -> Label

one to many



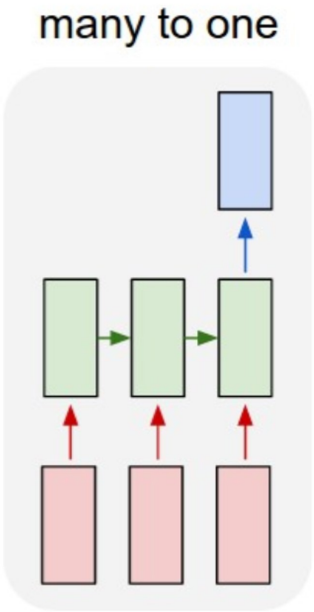
Two dogs play in the grass.

A person riding a motorcycle on a dirt road.

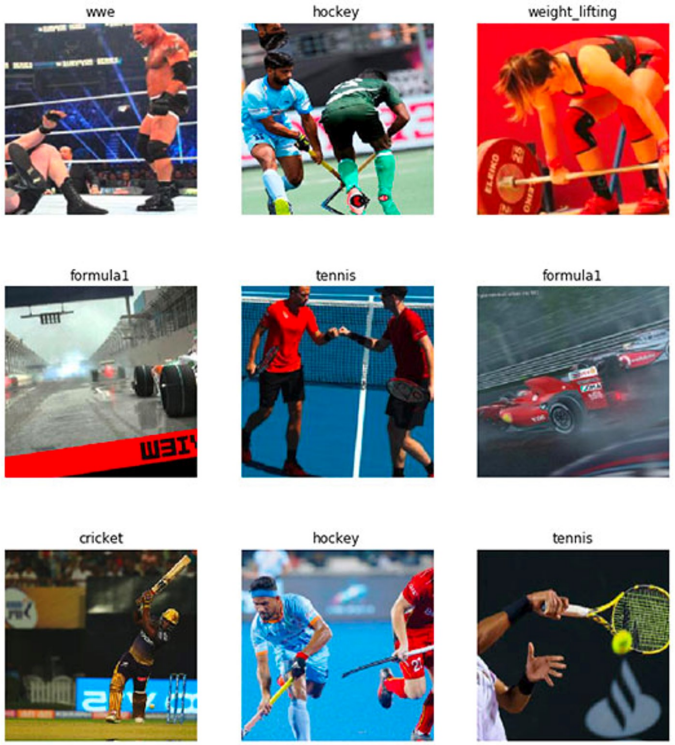
e.g. **Image Captioning**:
Image -> sequence of words

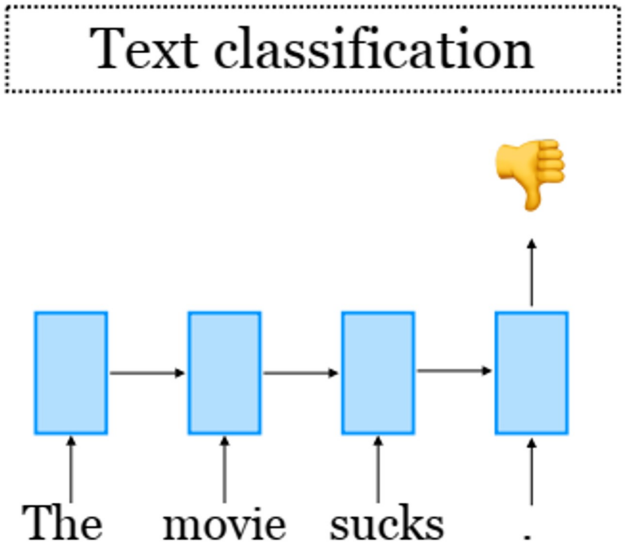
Slides adapted from Justin Johnson

Neural Networks



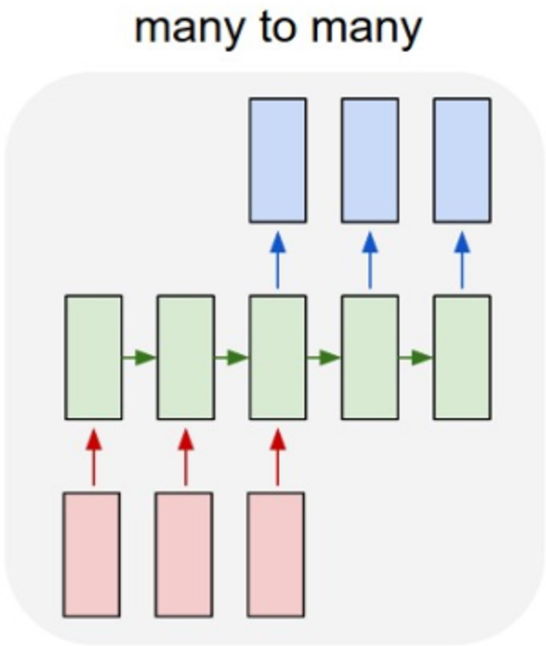
e.g. **Video classification:**
Sequence of images -> label

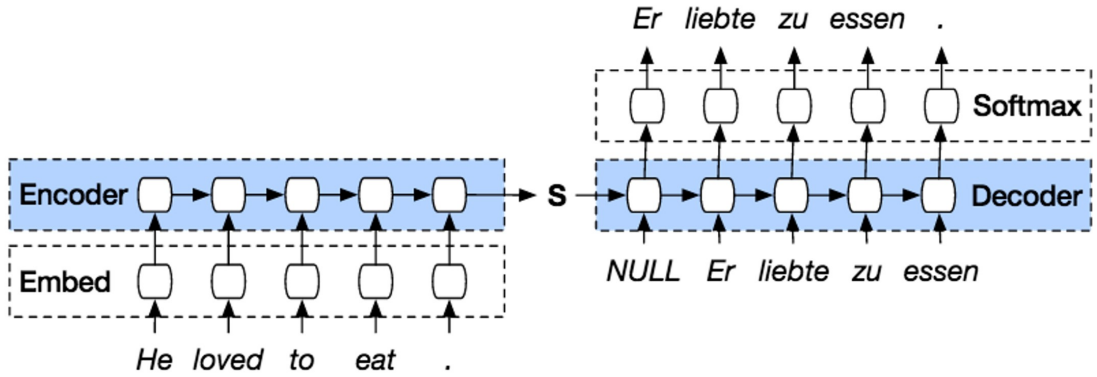




Slides adapted from Justin Johnson

Neural Networks

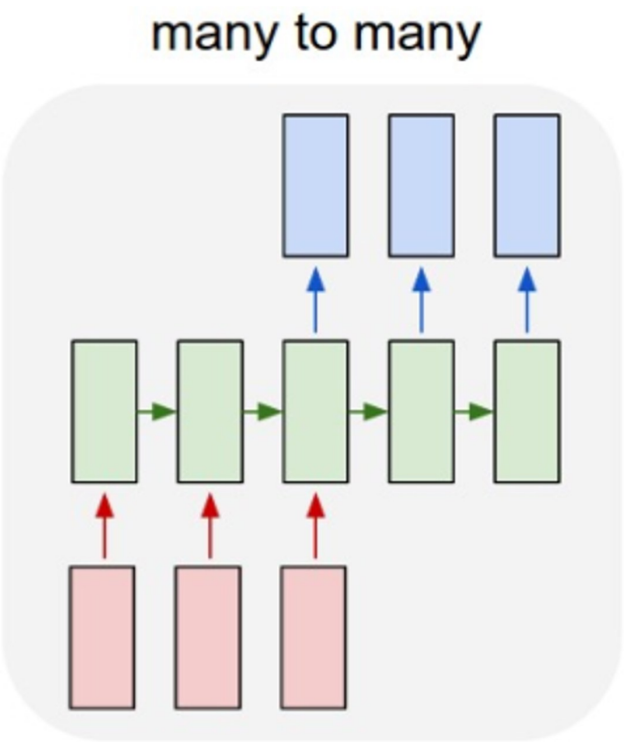




e.g. **Machine Translation:**

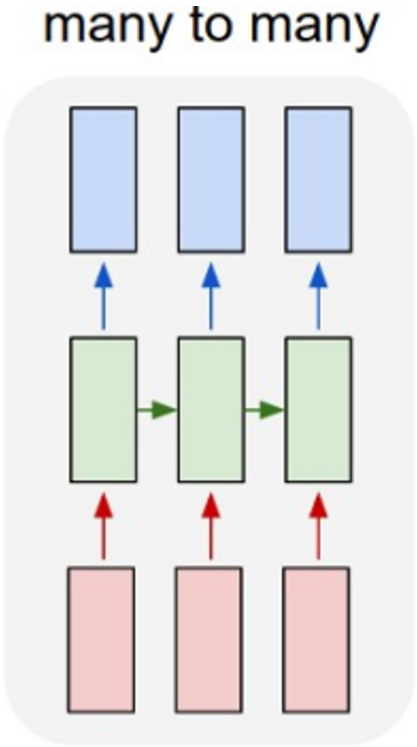
Sequence of words -> Sequence of words

Neural Networks



e.g. **Machine Translation:**

Sequence of words -> Sequence of words



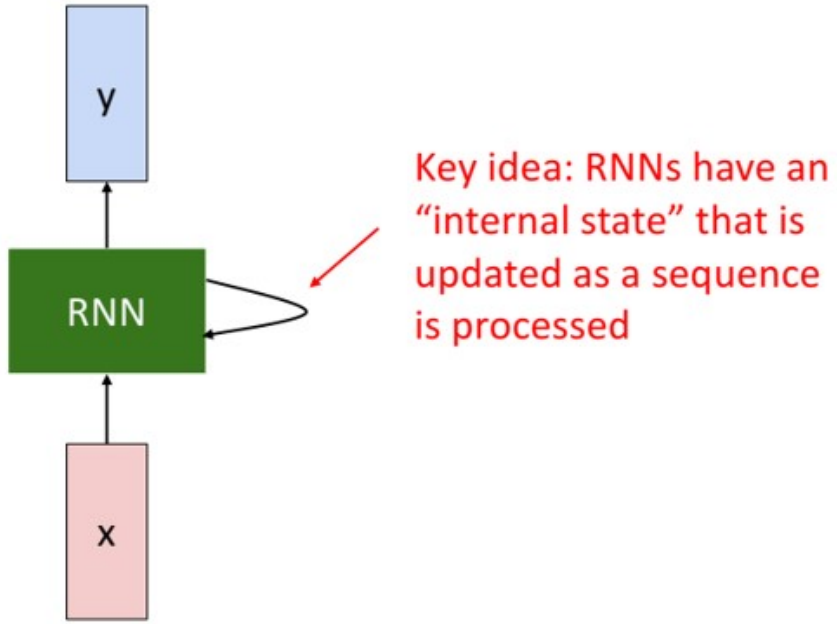
e.g. **Per-frame video classification:**

Sequence of images -> Sequence of labels

Slides adapted from Justin Johnson

Recurrent Neural Networks

- For handling sequential data
  - Variable length input
  - Variable order
    - "I visited Paris in 2014"
    - "In 2014, I visited Paris"
- Use shared parameters across time



Recurrent Neural Networks

- RNN model

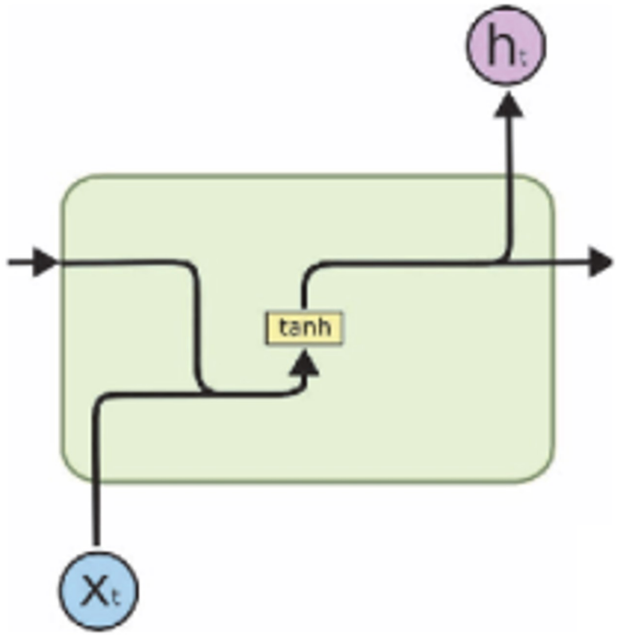
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

some function with parameters W

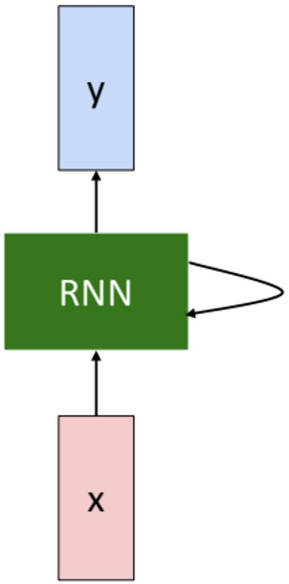


Connection and difference between RNN and feed-forward NN?

Slides adapted from Justin Johnson

Recurrent Neural Networks

- RNN model



The state consists of a single *"hidden"* vector **h**:

$$h_t = f_W(h_{t-1}, x_t)$$

(also bias term)

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

$$y_t = W_{hy}h_t$$

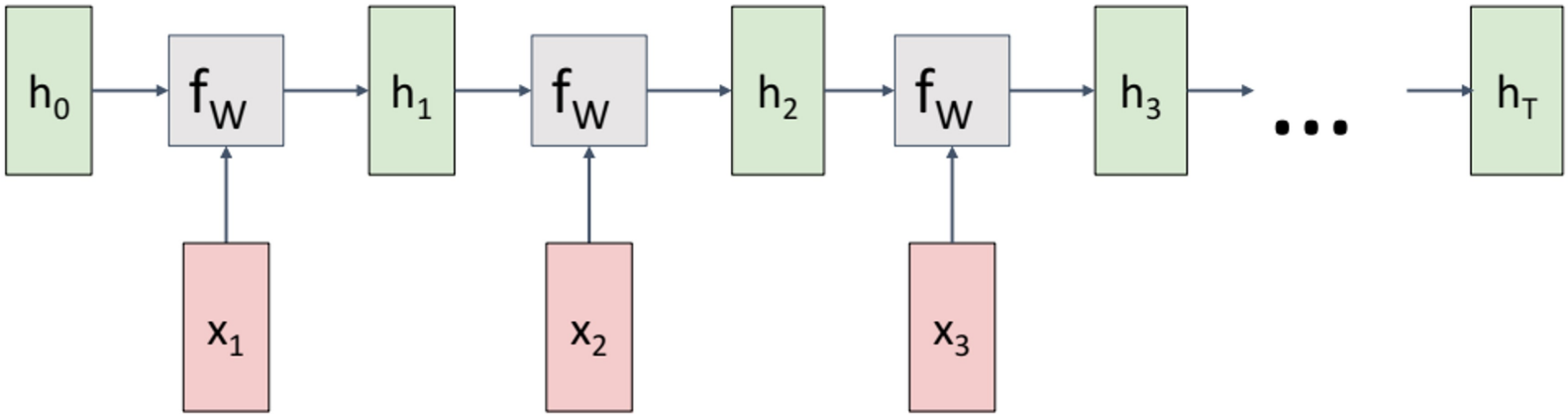
Recurrent Neural Networks

- RNN computational graph

$$h_t = f_W(h_{t-1}, x_t)$$

new state | old state | input vector at some time step

some function with parameters W



Slides adapted from Justin Johnson

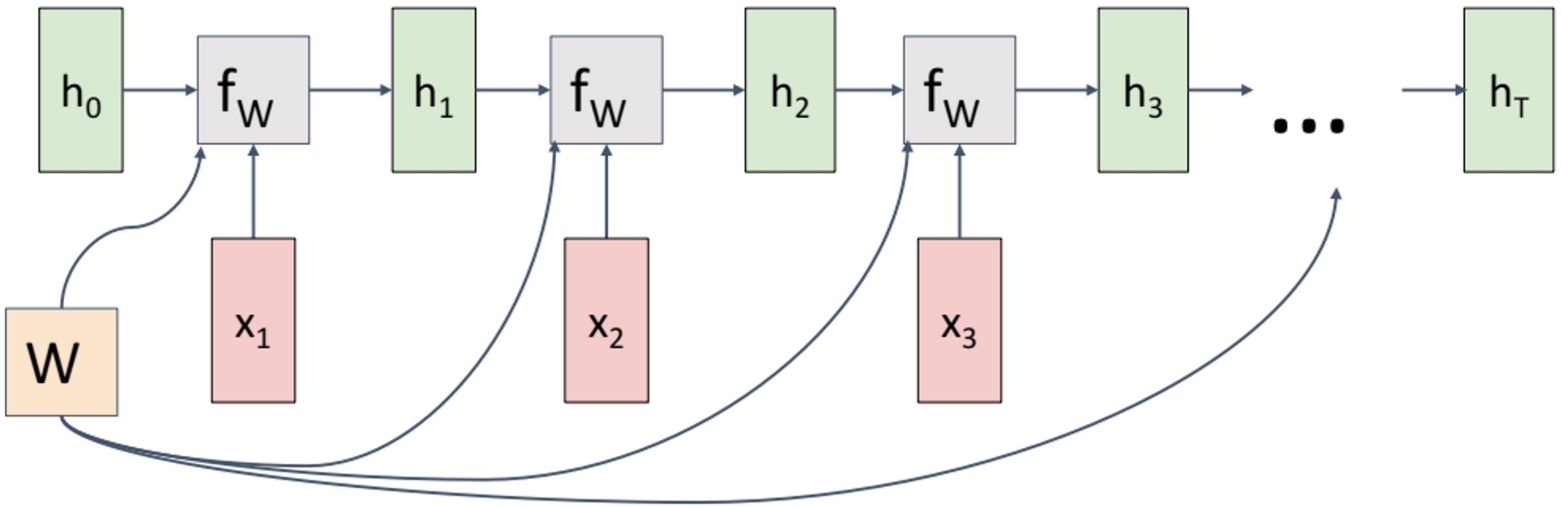
Recurrent Neural Networks

- RNN computational graph

$$h_t = f_W(h_{t-1}, x_t)$$

new state | old state | input vector at some time step

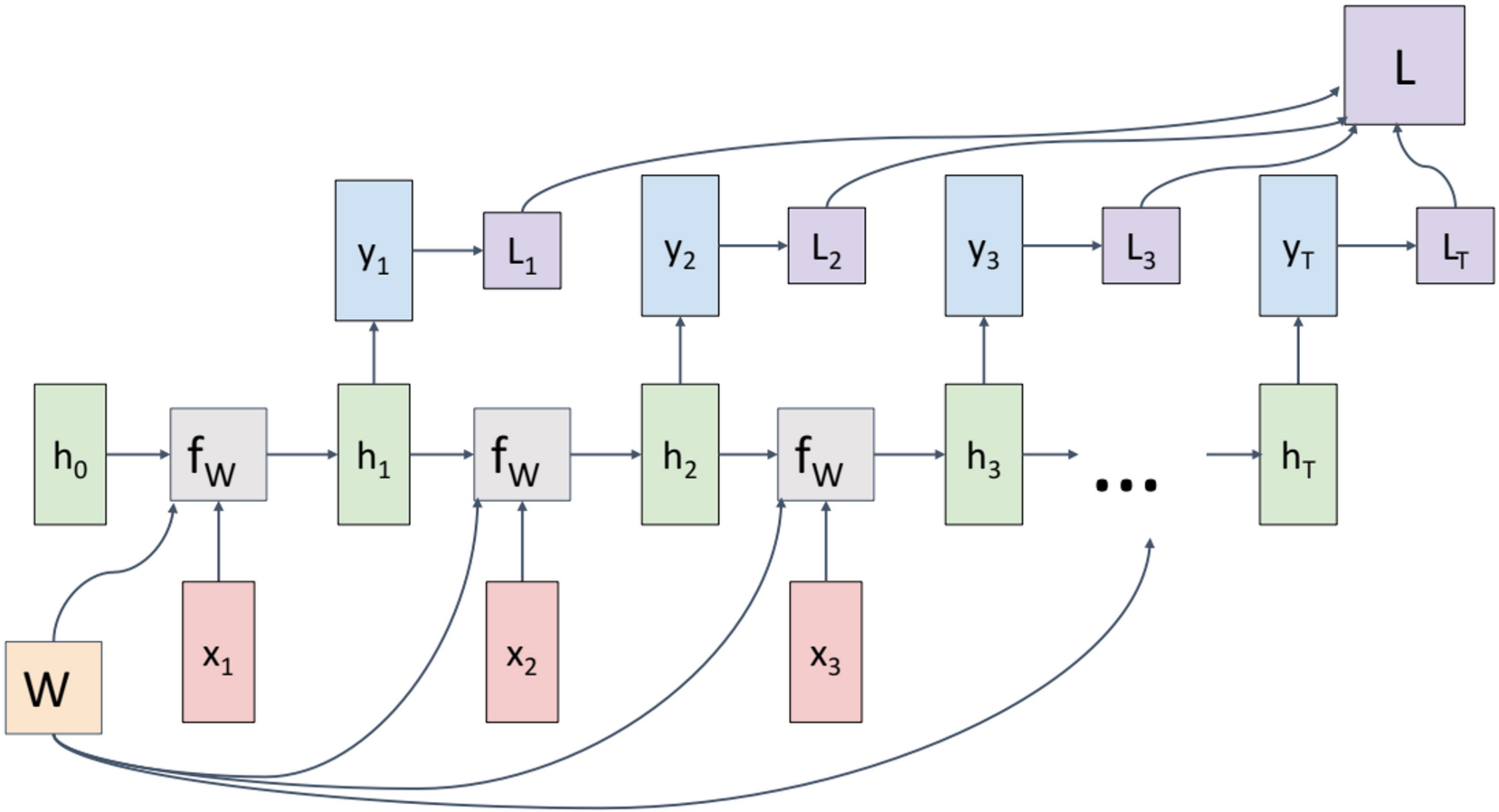
some function with parameters W



Re-use the same weight matrix at every time-step

Recurrent Neural Networks

- RNN computational graph (Many to Many)



L_t : loss at time point t

L: total loss

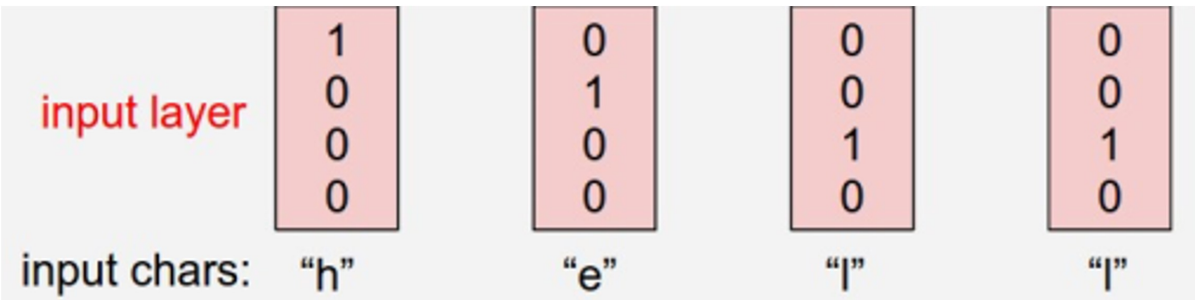
Recurrent Neural Networks

An example of language modeling: Given a few characters of a word, you want to predict other characters. Your training data is the set of all English words.

Given characters 1, 2, ..., t, model predicts character t

Training sequence: "hello"

Vocabulary: [h, e, l, o]



Note: The embedding of each character is a one-hot vector here, while there are many other embedding methods.

Slides adapted from Justin Johnson

Recurrent Neural Networks

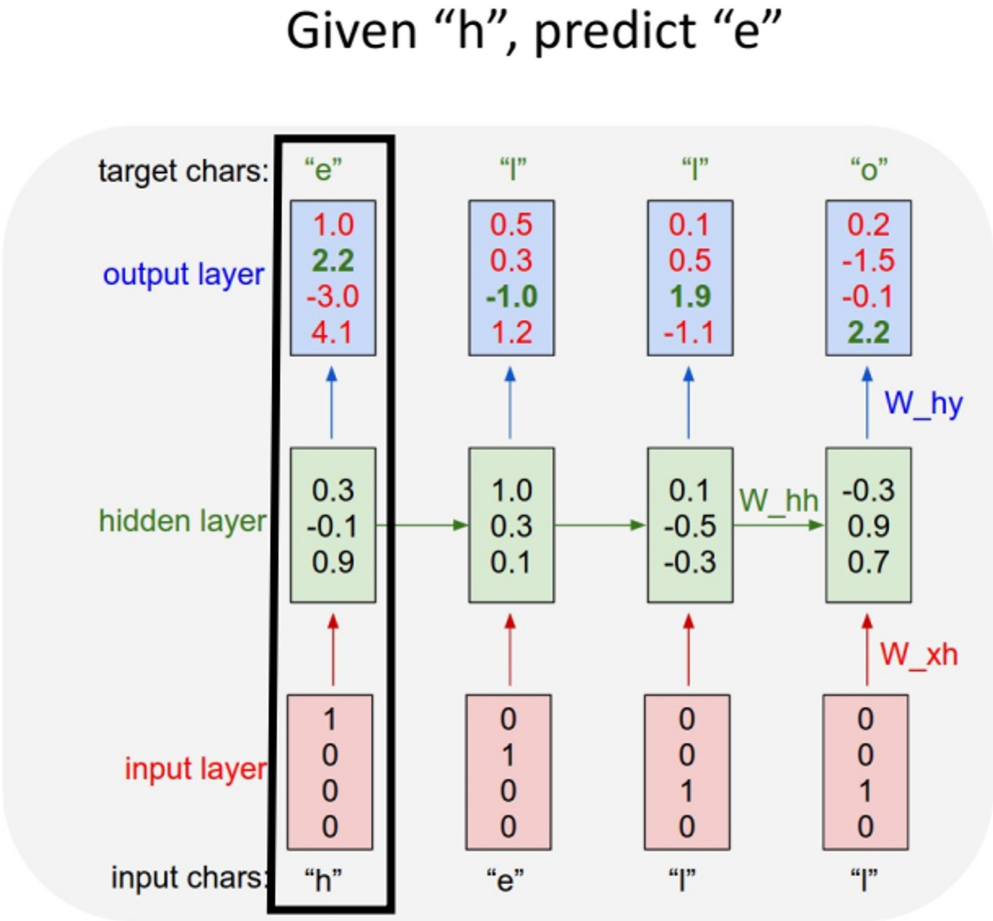
Example: Language Modeling

Given characters 1, 2, ..., t,
model predicts character t

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

Training sequence: "hello"

Vocabulary: [h, e, l, o]



Recurrent Neural Networks

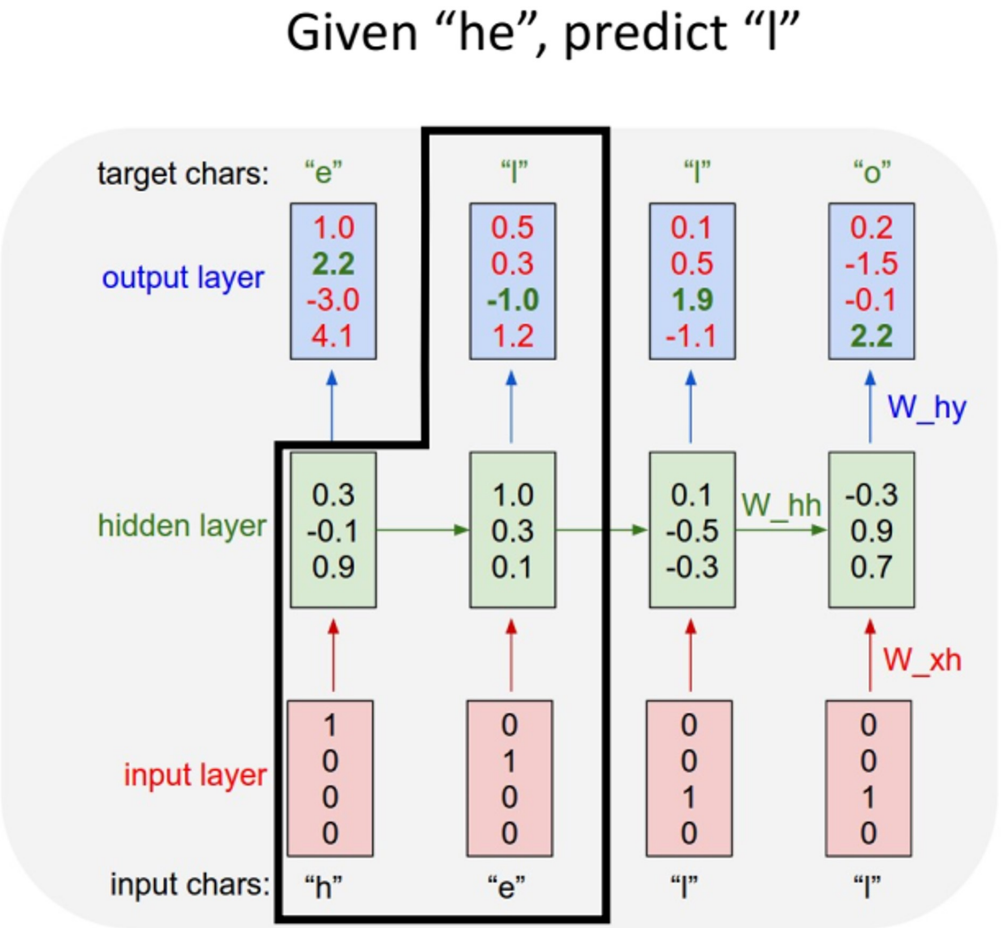
Example: Language Modeling

Given characters 1, 2, ..., t,
model predicts character t

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

Training sequence: "hello"

Vocabulary: [h, e, l, o]



Recurrent Neural Networks

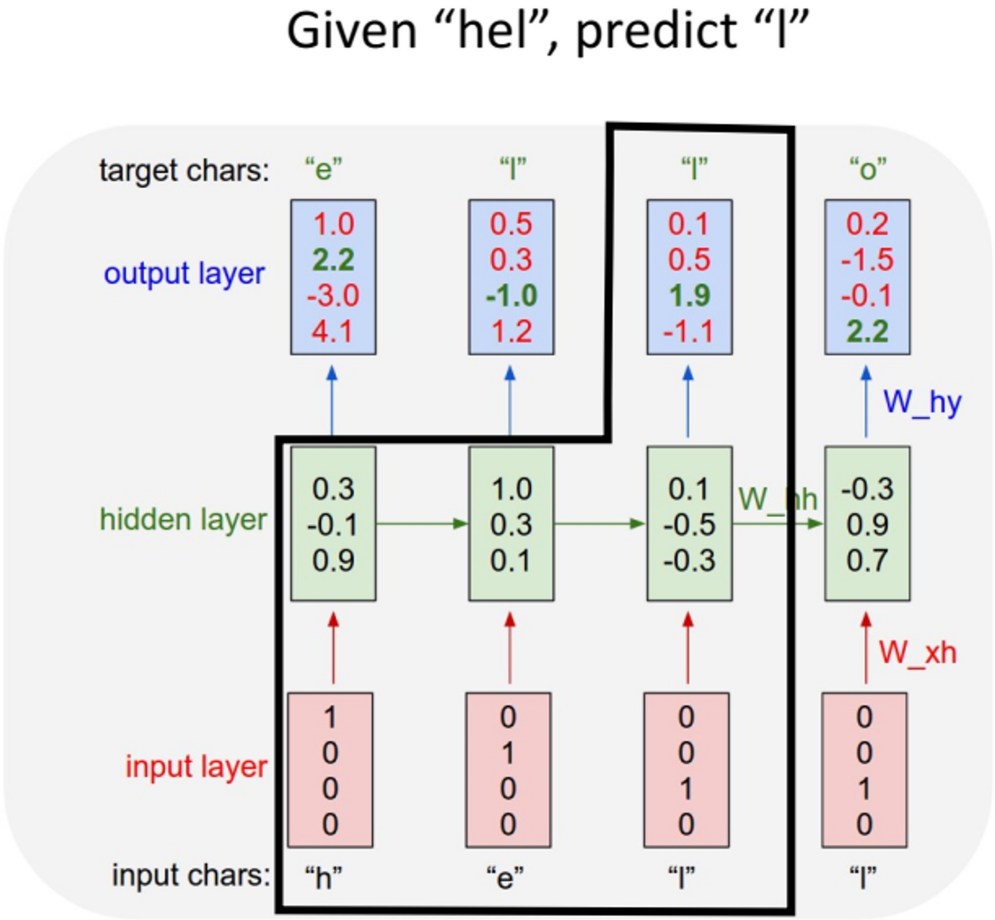
Example: Language Modeling

Given characters 1, 2, ..., t, model predicts character t

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

Training sequence: "hello"

Vocabulary: [h, e, l, o]



Recurrent Neural Networks

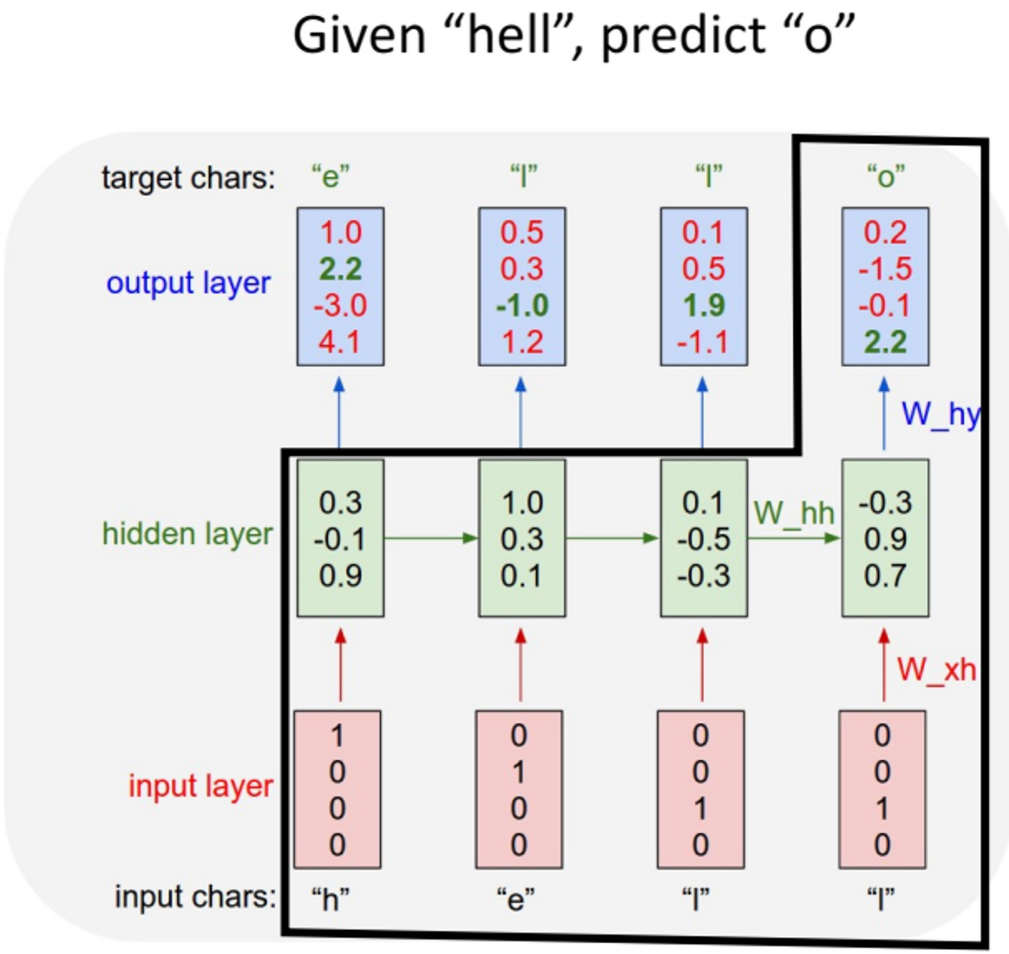
Example: Language Modeling

Given characters 1, 2, ..., t,
model predicts character t

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

Training sequence: "hello"

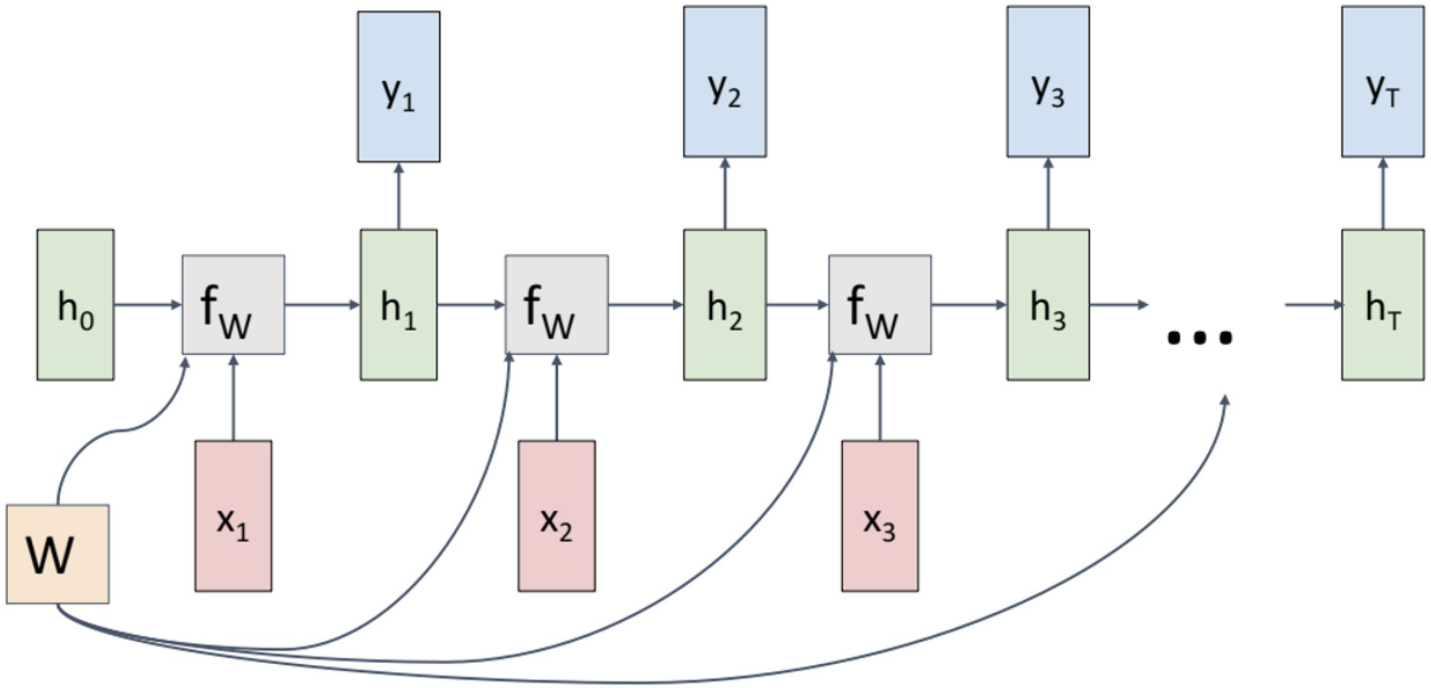
Vocabulary: [h, e, l, o]



1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

Recurrent Neural Networks

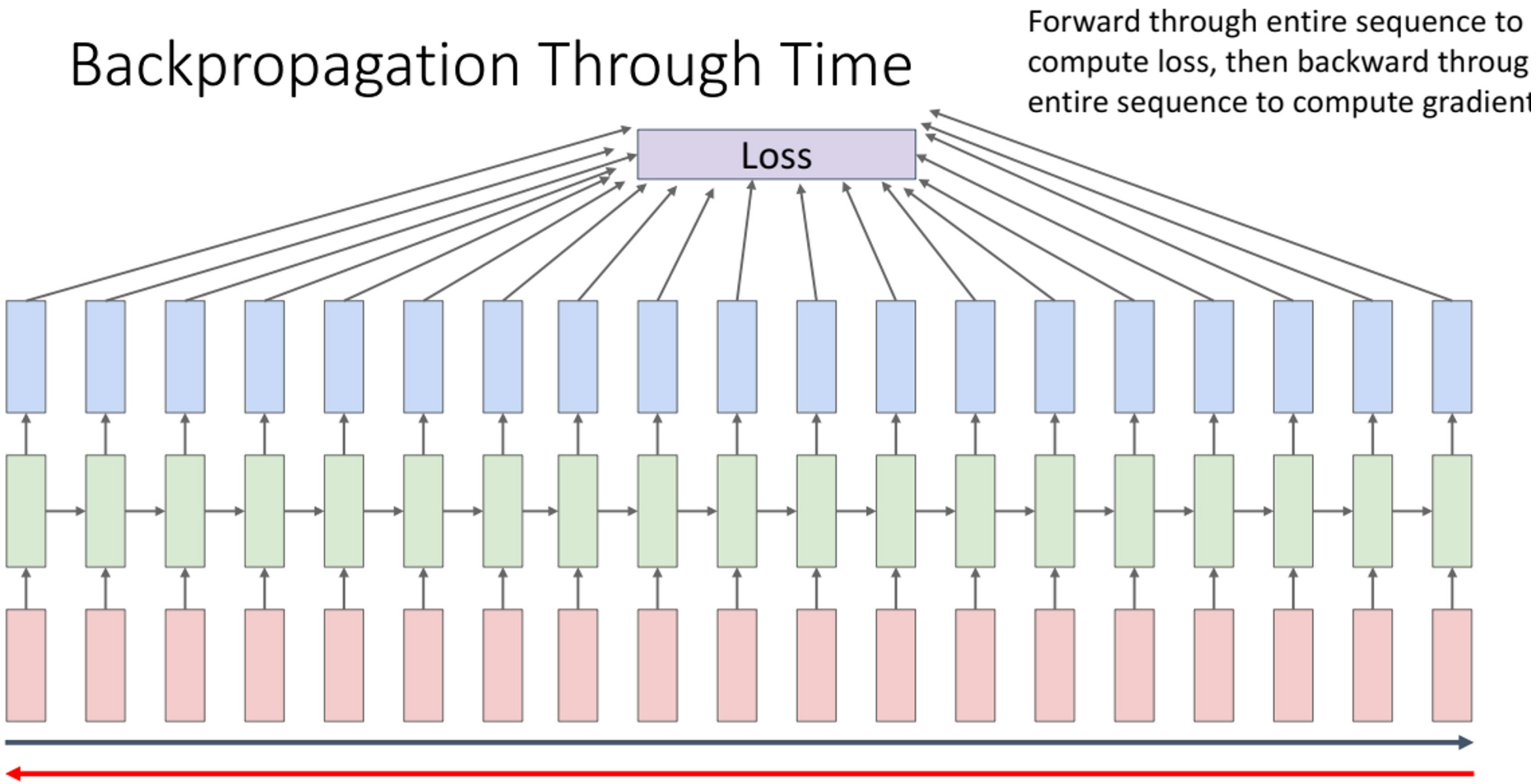
Model: $h_t = f_W(h_{t-1}, x_t), \quad \hat{y}_t = g_{W'}(h_t)$



Cost/error function: $E(\theta) = \frac{1}{T}\sum_{t=1}^{T} \mathcal{L}(y_t, \hat{y}_t)$, where $\theta = \{W, W'\}$

Optimization: backpropagation and gradient descent

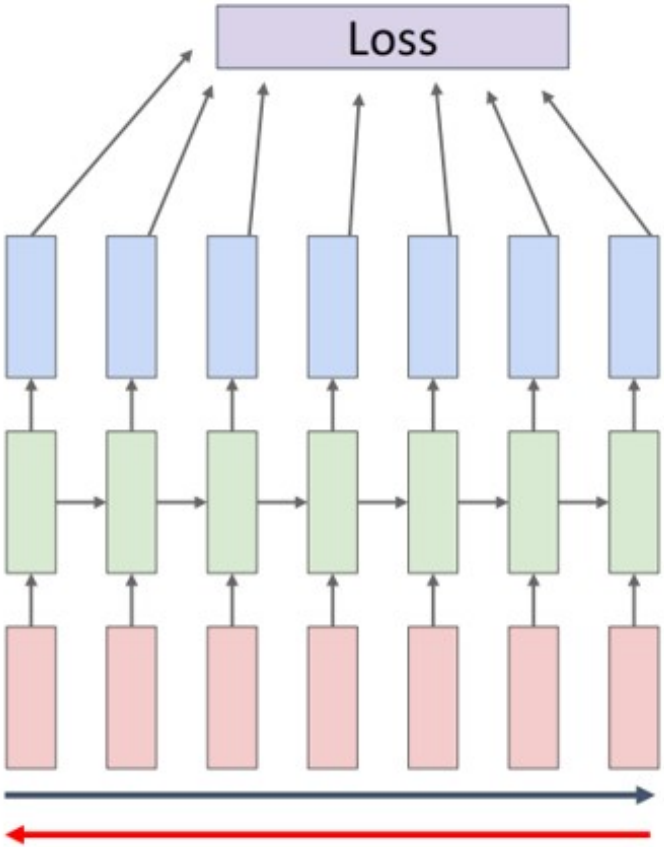
Recurrent Neural Networks



Problem: Takes a lot of memory for long sequence!

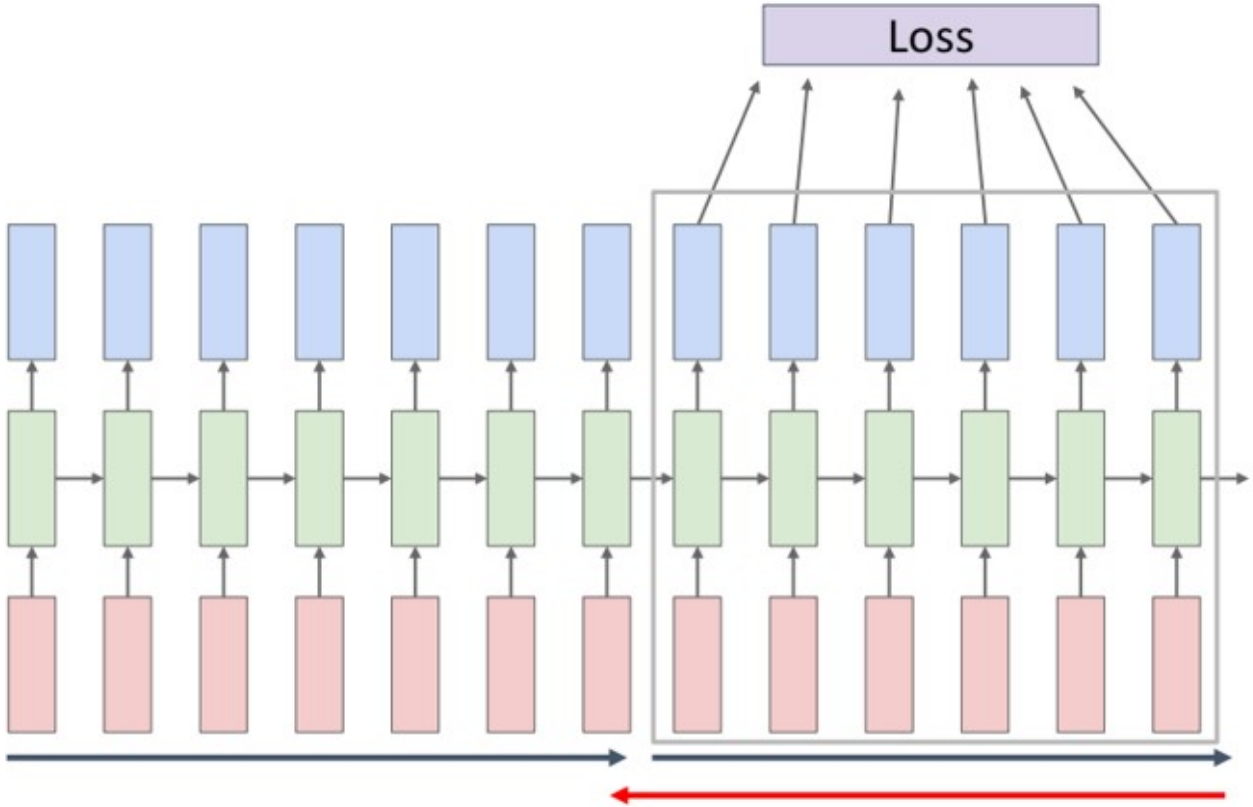
Similar to the batch optimization for feed-forward NN

Truncated Backpropagation Through Time



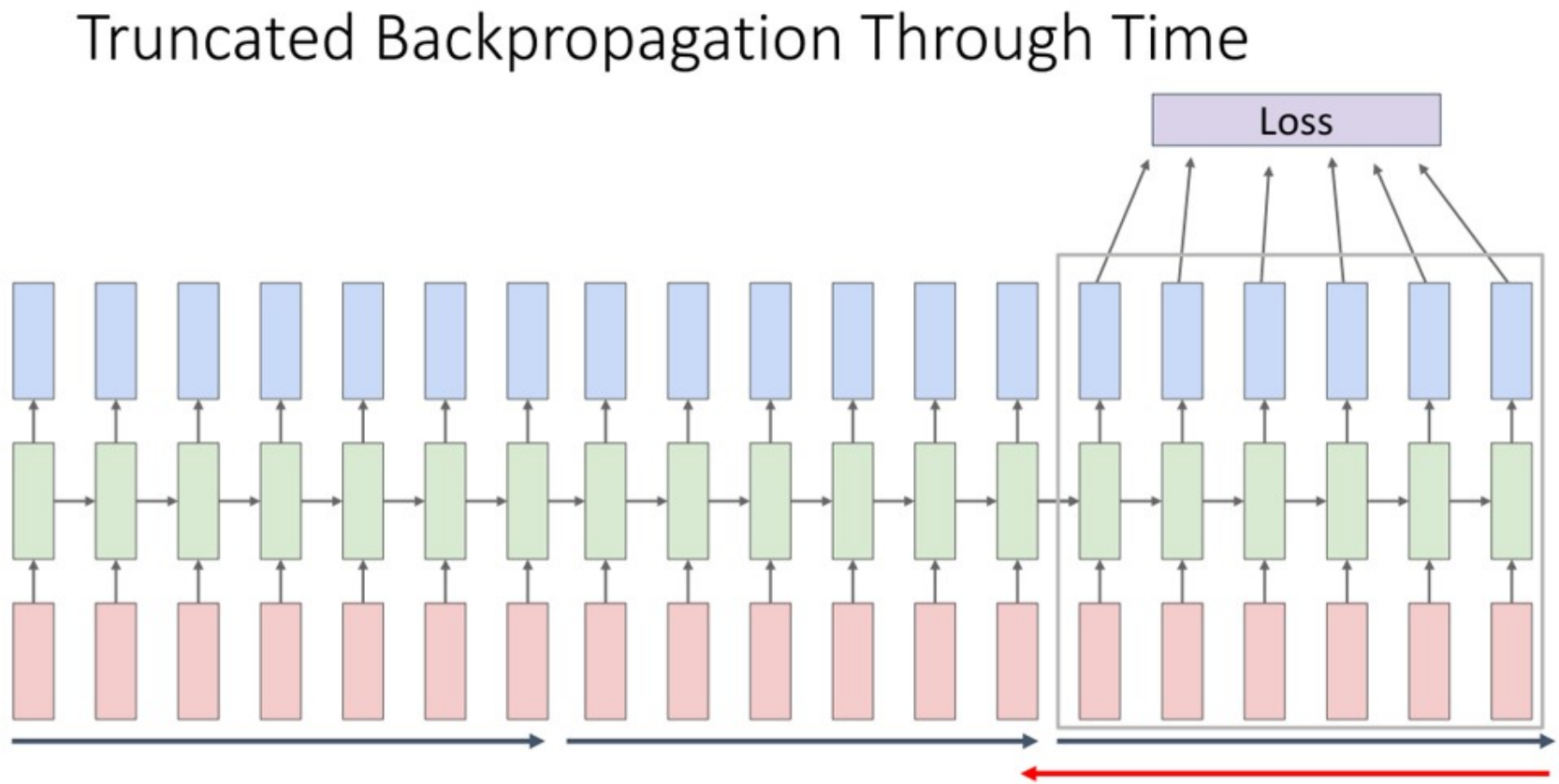
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated Backpropagation Through Time



Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Recurrent Neural Networks



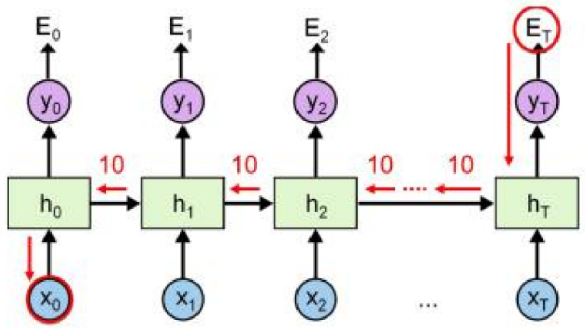
- Carry hidden states forward in time forever.
- But only backpropagate for some smaller number of steps.
- Similar to the mini-batch optimization for feed-forward NN.

Gradient Exploding and Vanishing

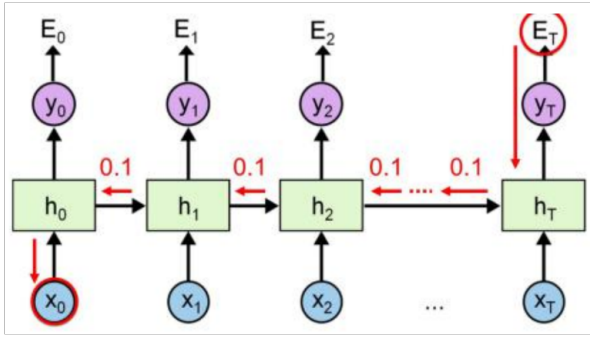
- Gradient Exploding: Gradients may become exceptionally large during training, causing numerical instability.
- Gradient vanishing: Gradients during training may become extremely small as they are backpropagated through time. This limits the network's ability to capture long-range dependencies.

$$\frac{\partial y_T}{\partial x_0} = \frac{\partial y_T}{\partial h_T}\frac{\partial h_T}{\partial h_{T-1}}\cdots\frac{\partial h_3}{\partial h_2}\frac{\partial h_2}{\partial h_1}\frac{\partial h_1}{\partial h_0}\frac{\partial h_0}{\partial x_0}$$

- Gradients can become unstable



→ exploding gradient!



→ vanishing gradient!

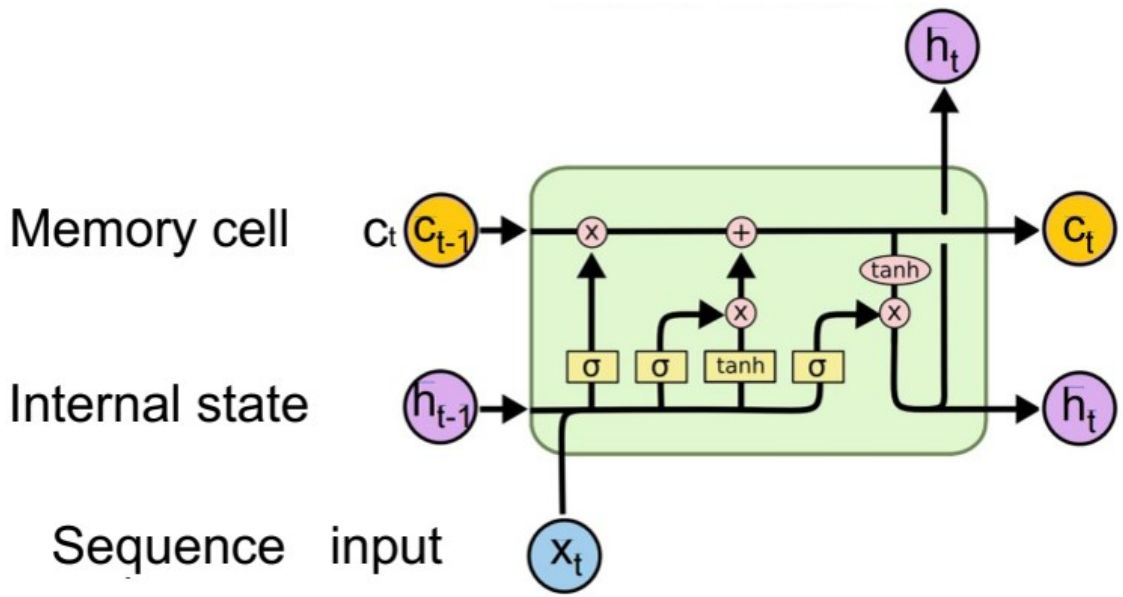
Can not learn long-term dependencies

They can be (partially) solved by gradient clipping, i.e., setting upper and lower thresholds.

1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

LSTM (Long Short-Term Memory)

- LSTM model
  - A type of RNN proposed by Hochreiter and Schmidhuber in 1997.
  - A solution to the vanishing gradients problem and long-distance dependencies.



LSTM (Long Short-Term Memory) (optional)

Comparison between basic RNN and LSTM

Vanilla RNN

$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

LSTM

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

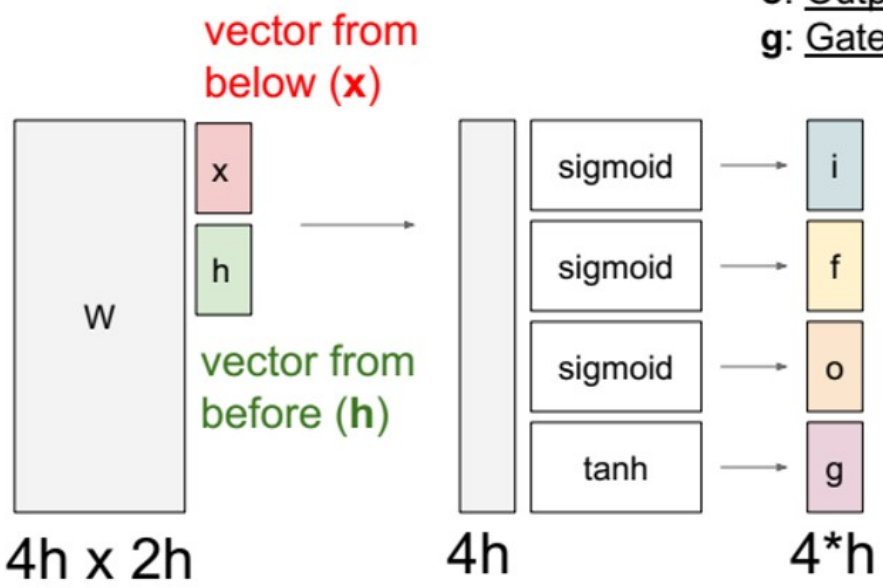
$$h_t = o \odot \tanh(c_t)$$

LSTM (Long Short-Term Memory) (optional)

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

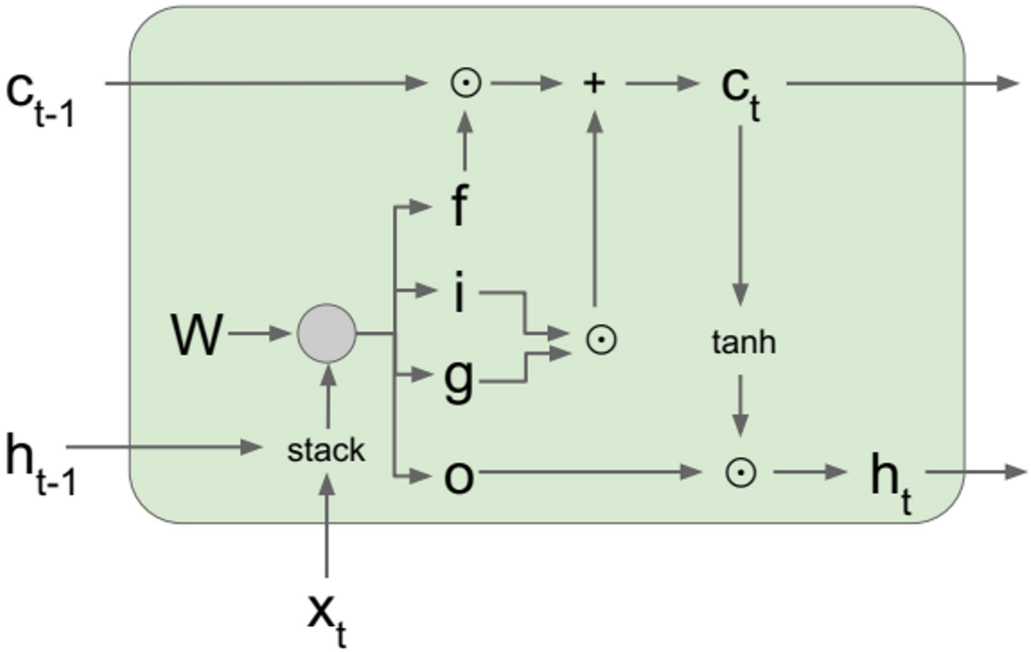
$$h_t = o \odot \tanh(c_t)$$

LSTM (Long Short-Term Memory) (optional)

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

i: Input gate, whether to write to cell
f: Forget gate, Whether to erase cell
o: Output gate, How much to reveal cell
g: Gate gate (?), How much to write to cell



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

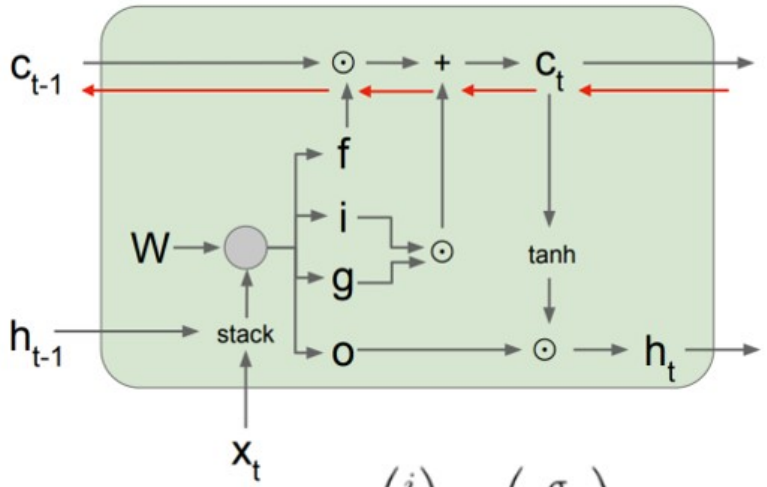
$$h_t = o \odot \tanh(c_t)$$

LSTM (Long Short-Term Memory) (optional)

There are 4 gates:

- Input gate (how much to write):
  $\mathbf{i}_t = \sigma(\mathbf{W}^{(i)}\mathbf{h}_{t-1} + \mathbf{U}^{(i)}\mathbf{x}_t + \mathbf{b}^{(i)}) \in \mathbb{R}^d$
- Forget gate (how much to erase):
  $\mathbf{f}_t = \sigma(\mathbf{W}^{(f)}\mathbf{h}_{t-1} + \mathbf{U}^{(f)}\mathbf{x}_t + \mathbf{b}^{(f)}) \in \mathbb{R}^d$
- Output gate (how much to reveal):
  $\mathbf{o}_t = \sigma(\mathbf{W}^{(o)}\mathbf{h}_{t-1} + \mathbf{U}^{(o)}\mathbf{x}_t + \mathbf{b}^{(o)}) \in \mathbb{R}^d$
- New memory cell (what to write):
  $\mathbf{g}_t = \tanh(\mathbf{W}^{(c)}\mathbf{h}_{t-1} + \mathbf{U}^{(c)}\mathbf{x}_t + \mathbf{b}^{(c)}) \in \mathbb{R}^d$
- Final memory cell: $\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$ (element-wise product)
- Final hidden cell: $\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$

Backpropagation from c_t to c_{t-1} only element wise multiplication by f, no matrix multiply by W

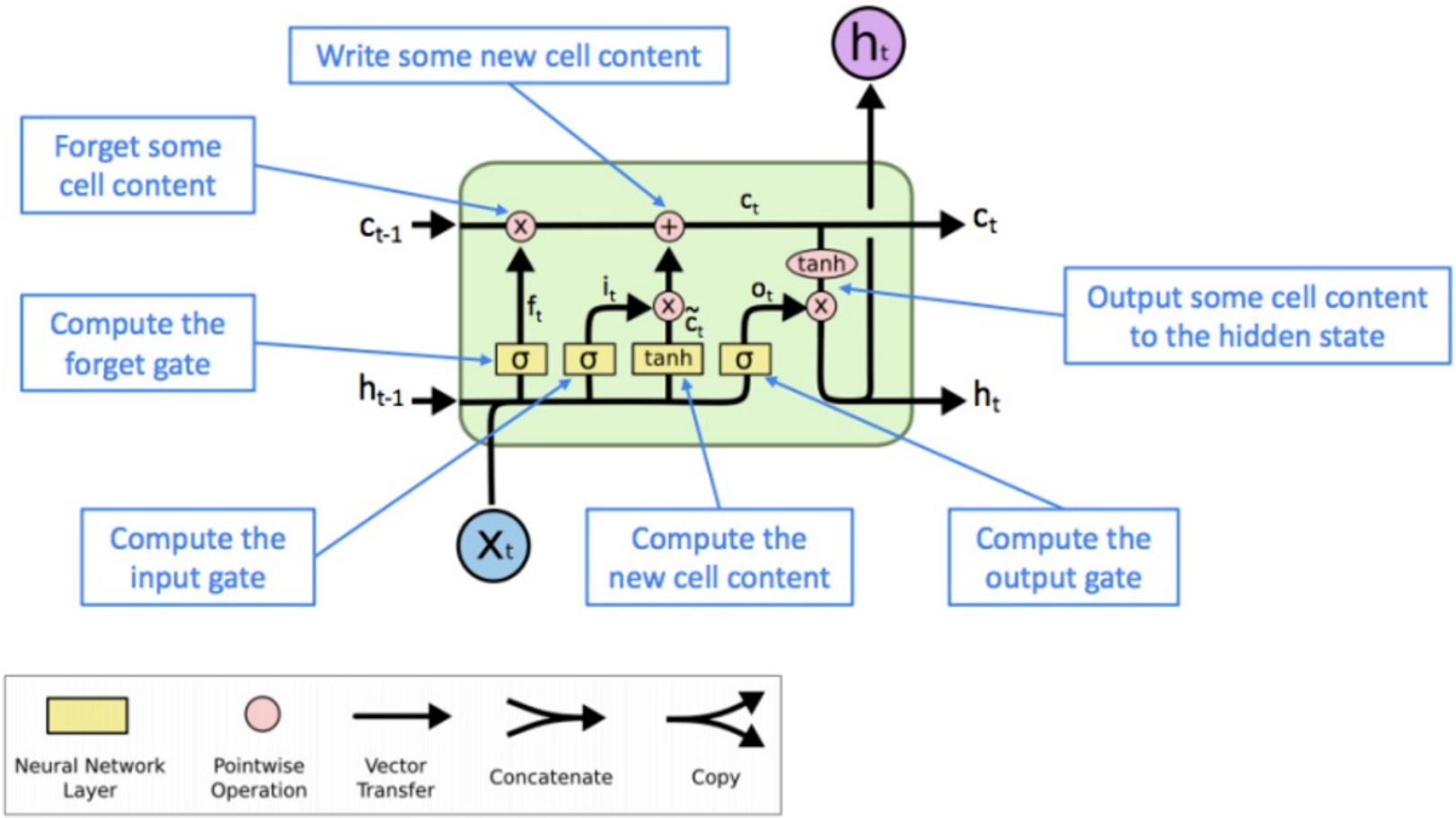


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

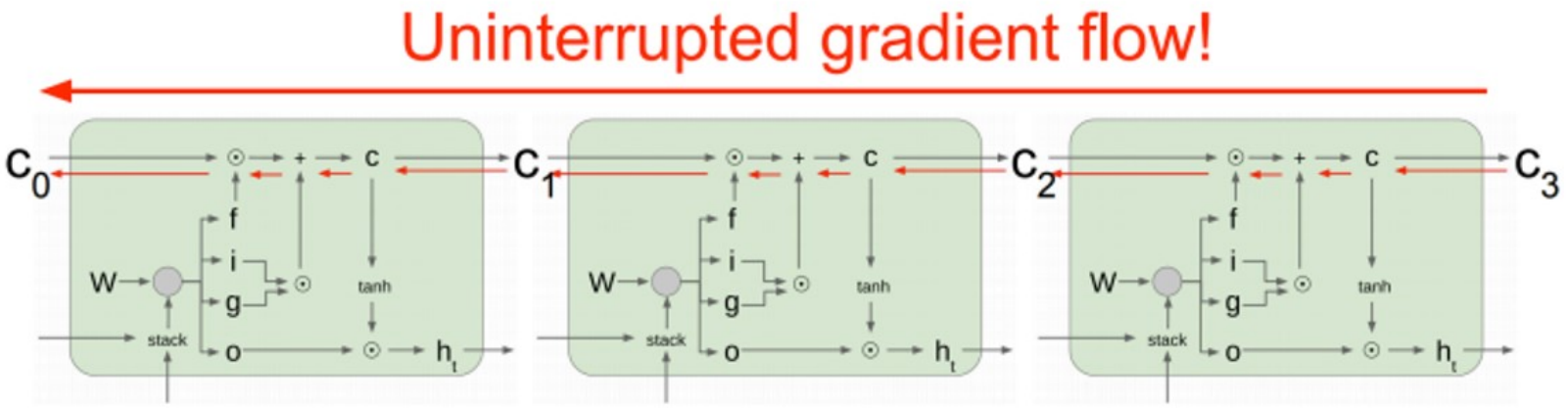
$$c_t = f \odot c_{t-1} + i \odot g$$

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

LSTM (Long Short-Term Memory) (optional)



LSTM (Long Short-Term Memory) (optional)



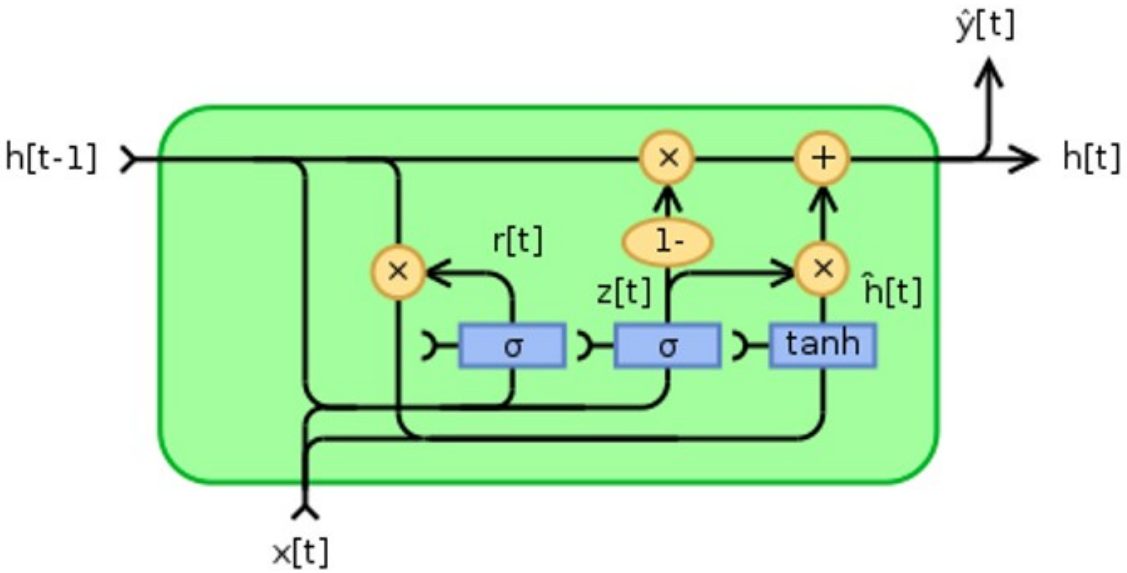
- LSTM does not guarantee that there is no vanishing (exploding) gradient.
- But it does provide an easier way for the model to learn long-distance dependencies.
- LSTMs were invented in 1997 but finally got working from 2013-2015.

1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

Other Extensions of RNN (optional)

Gated recurrent units (GRUs)

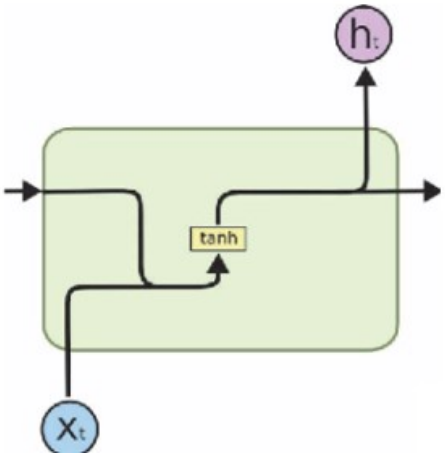
- A gating mechanism in recurrent neural networks.
- Introduced in 2014 by Kyunghyun Cho et al.
- Similar to LSTM but has fewer parameters than LSTM, as it lacks an output gate.



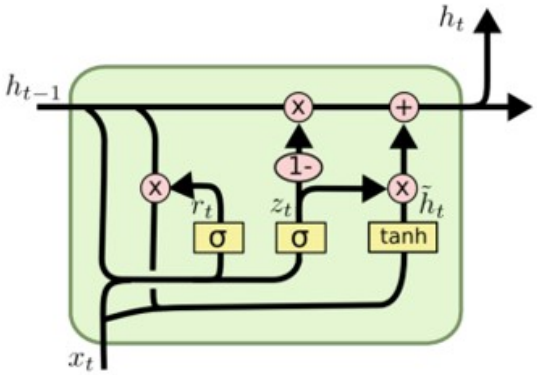
Other Extensions of RNN (optional)

Gated recurrent units (GRUs)

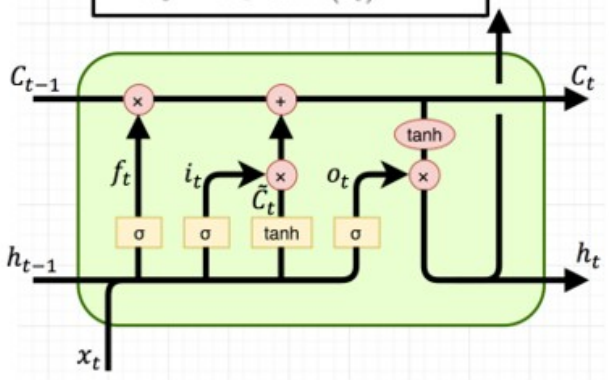
$$h_t = \tanh\left(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$



$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

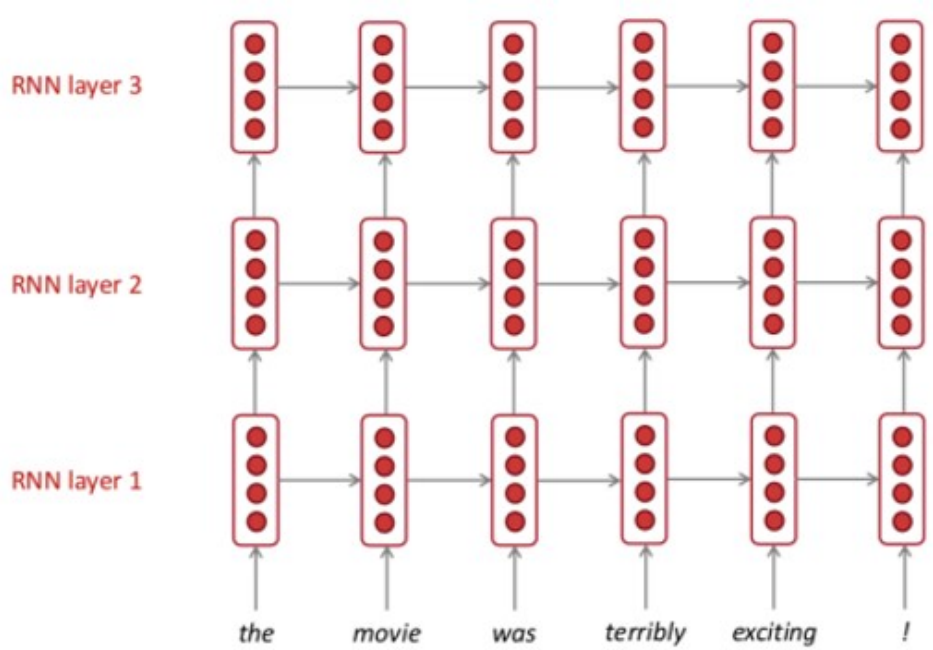


$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$
$$c_t = f \odot c_{t-1} + i \odot g$$
$$h_t = o \odot \tanh(c_t)$$



Other Extensions of RNN (optional)

- Multi-layer RNNs



The hidden states from RNN layer i are the inputs to RNN layer $i + 1$

- In practice, using 2 to 4 layers is common (usually better than 1 layer)
- Transformer-based networks can be up to 24 layers with lots of skip-connections.

1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

Limitations of Basic RNN and Its Variants

- **Difficulty with Long-Term Dependencies:** RNNs struggle to capture long-term dependencies in data due to the vanishing gradient problem. This means that the model has trouble learning from data points that are far apart in the sequence.
- **Limited Parallelization:** Unlike some other neural network architectures, RNNs process data sequentially, which makes it difficult to parallelize the computation. This can lead to slower training times compared to models that can process data in parallel.

1 Sequential Data Analysis

2 Recurrent Neural Network
- Basic RNN Architecture
- Training of Basic RNN
- LSTM
- Other Extensions of RNN (optional)
- Limitations of Basic RNN and Its Variants

3 Transformer
- Overview of Transformer
- Attention Mechanism

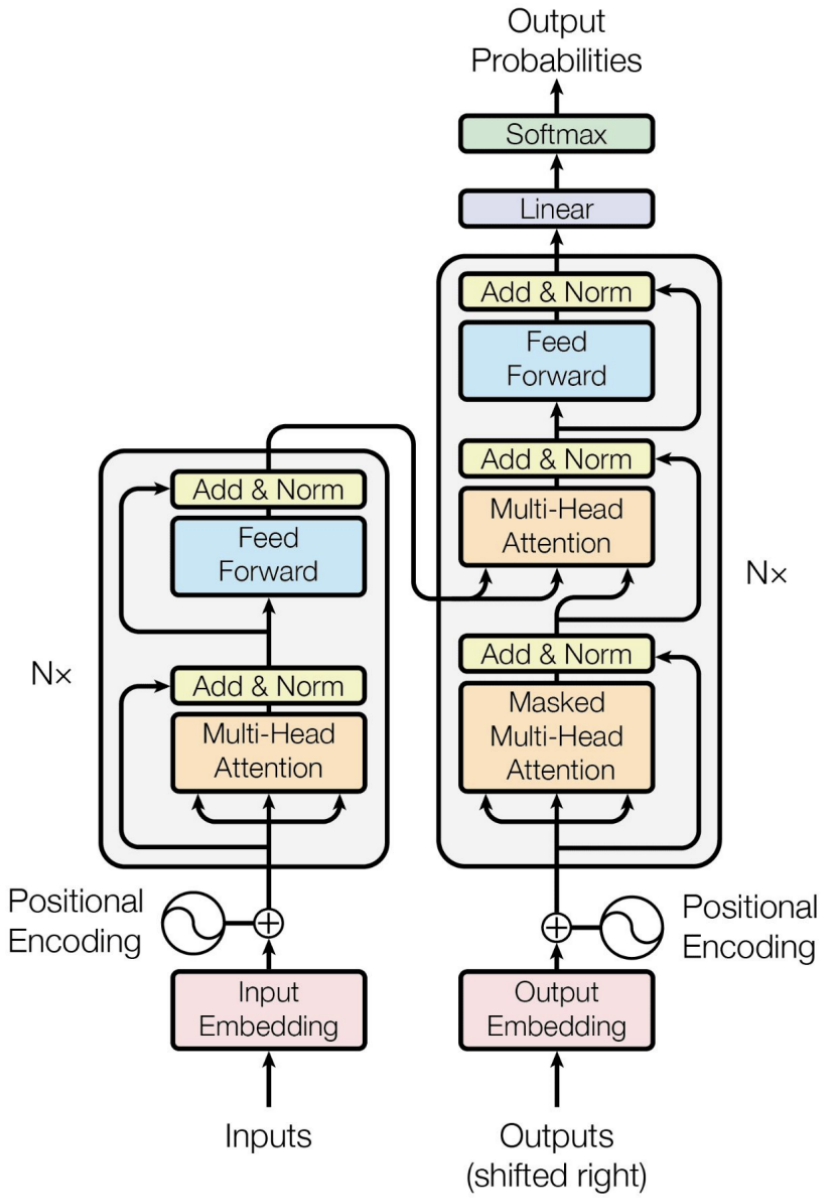
4 Summary

1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

Overview of Transformer

A transformer is a deep learning architecture developed by researchers at Google and based on the multi-head **attention mechanism**

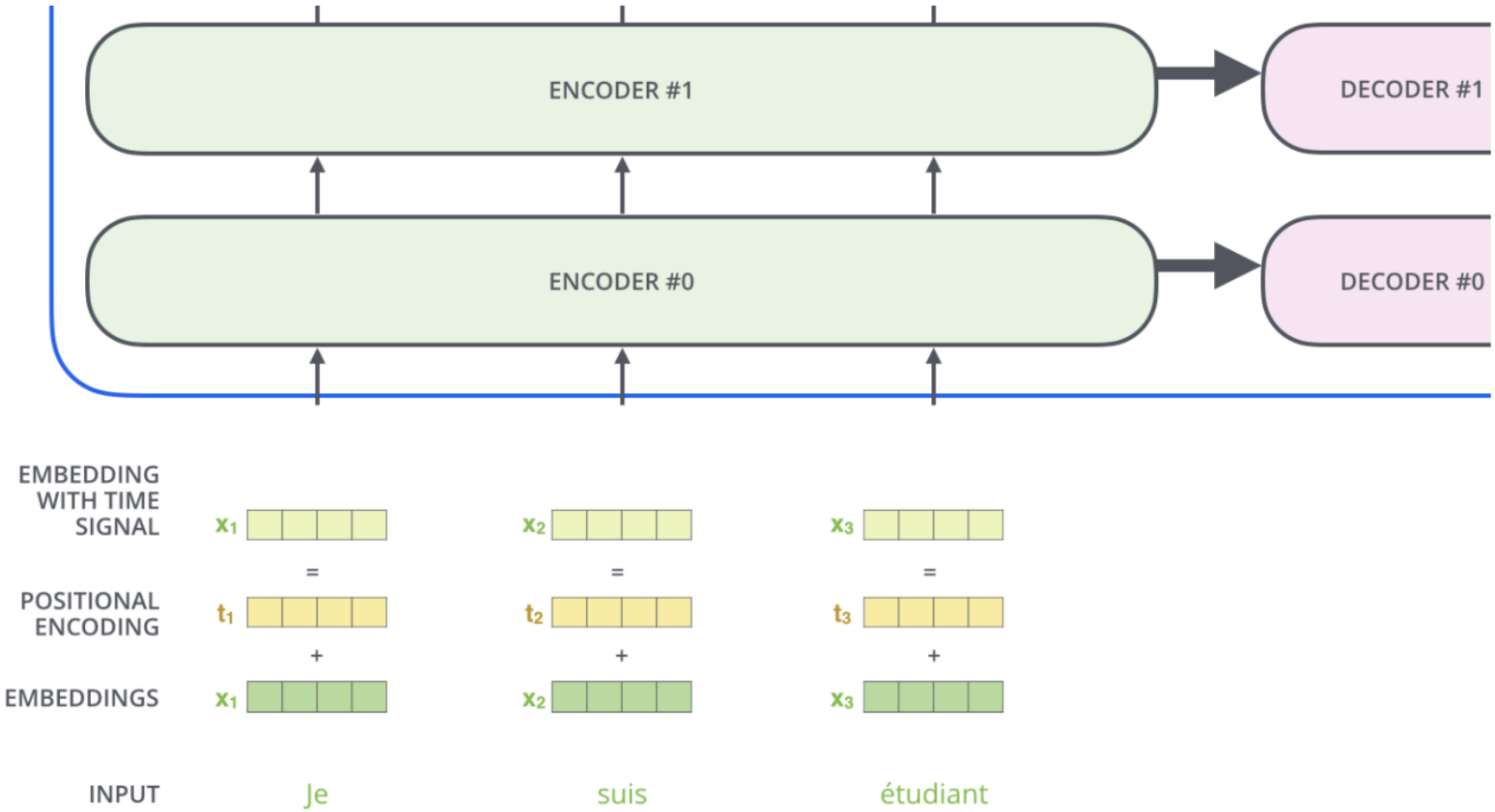
- Input embeddings
  - E.g.: represent each word as a vector of fixed size
- Position encodings
  - Represent the order of each input as a vector
  - See the blog `https://kazemnejad.com/blog/transformer_architecture_positional_encoding/`
- Attention mechanism



Vaswani et al. Attention Is All You Need. NeurIPS 2017.

Overview of Transformer

Consider a language translation problem (French to English):
Je suis étudiant $\longrightarrow$ I am a student



Each encoder: **Self-Attention** $\rightarrow$ Add & Norm $\rightarrow$ MLP $\rightarrow$ Add & Norm

1. Sequential Data Analysis
2. Recurrent Neural Network
   - Basic RNN Architecture
   - Training of Basic RNN
   - LSTM
   - Other Extensions of RNN (optional)
   - Limitations of Basic RNN and Its Variants
3. Transformer
   - Overview of Transformer
   - Attention Mechanism
4. Summary

Attention Mechanism

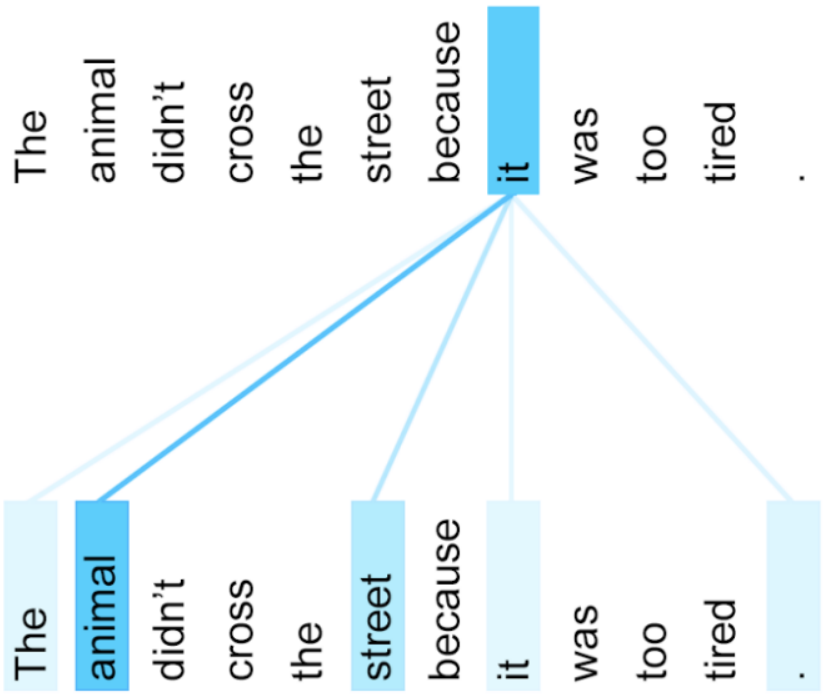
- Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ (or $\mathbf{H}$ in a hidden layer) be the embedding matrix, where n is the number of samples (e.g., words). Let $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d'}$ be three parameter matrices.
- Let $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$, $\mathbf{V} = \mathbf{X}\mathbf{W}_V$. They are often called 'query', 'key', and 'value' respectively.
- The (self-) attention mechanism is defined as

$$\mathbf{Z} = \text{attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d'}}\right)\mathbf{V}$$

- $\mathbf{A} := \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d'}}\right) \in \mathbb{R}^{n \times n}$ is called attention map.
- When $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ are computed using different features, the mechanism is called cross-attention.

Attention Mechanism

Self-attention layers learned “it” could refer to different entities in the different contexts.



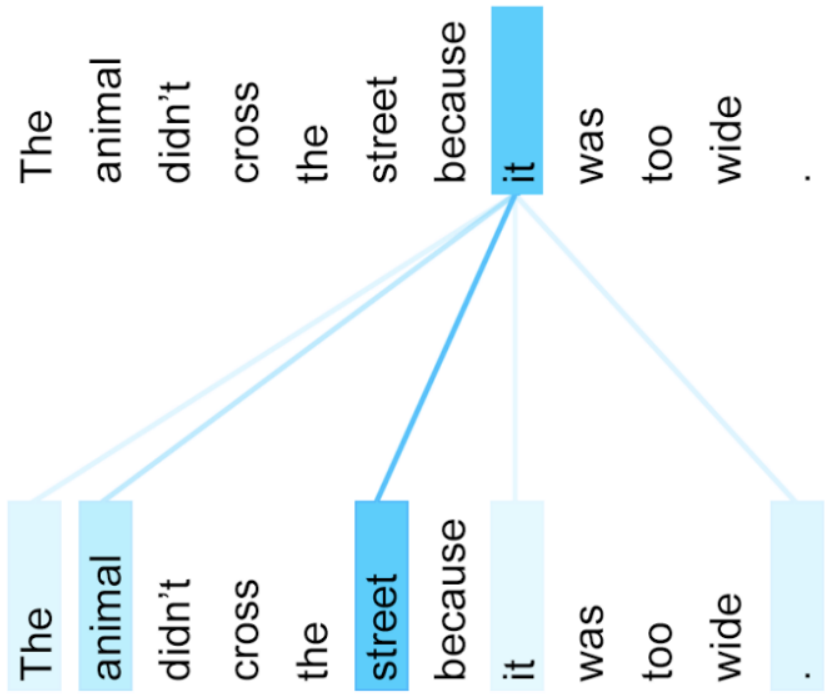
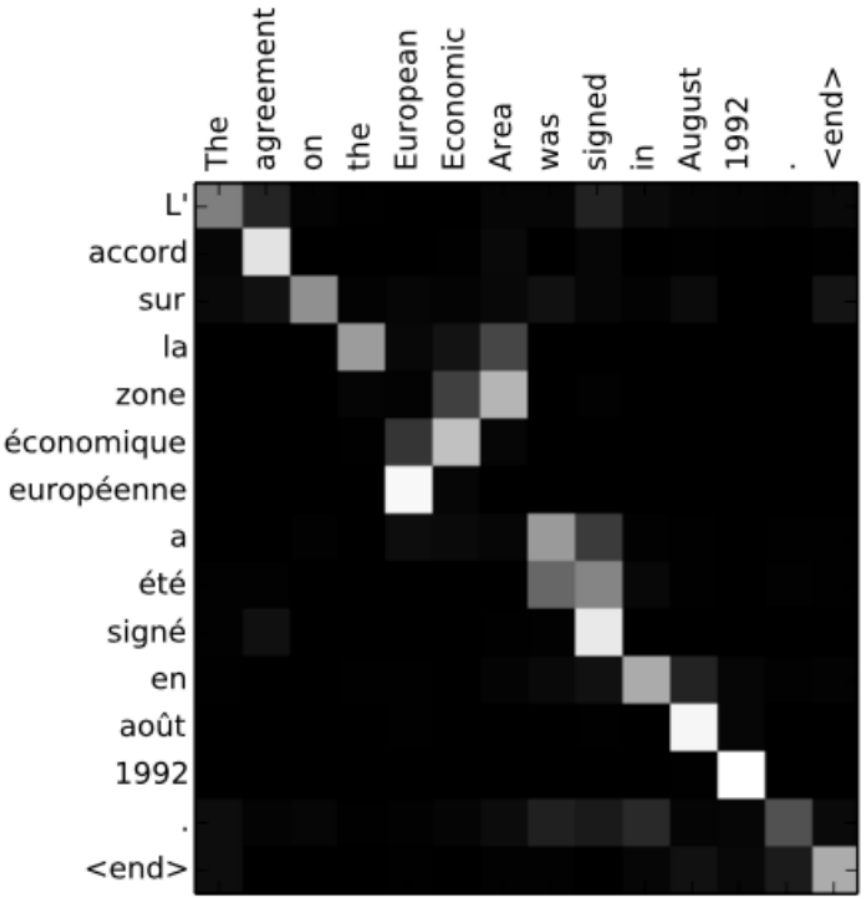


Image credit: Jimmy Ba

Attention Mechanism

Cross-attention also learns the connection between different samples.



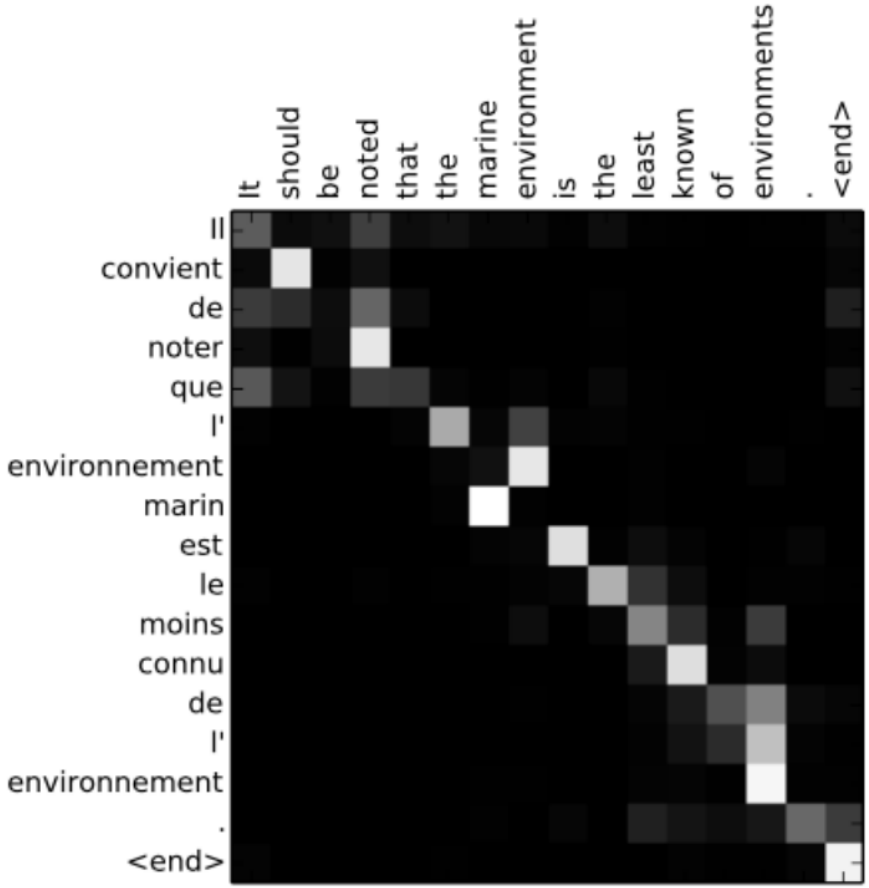
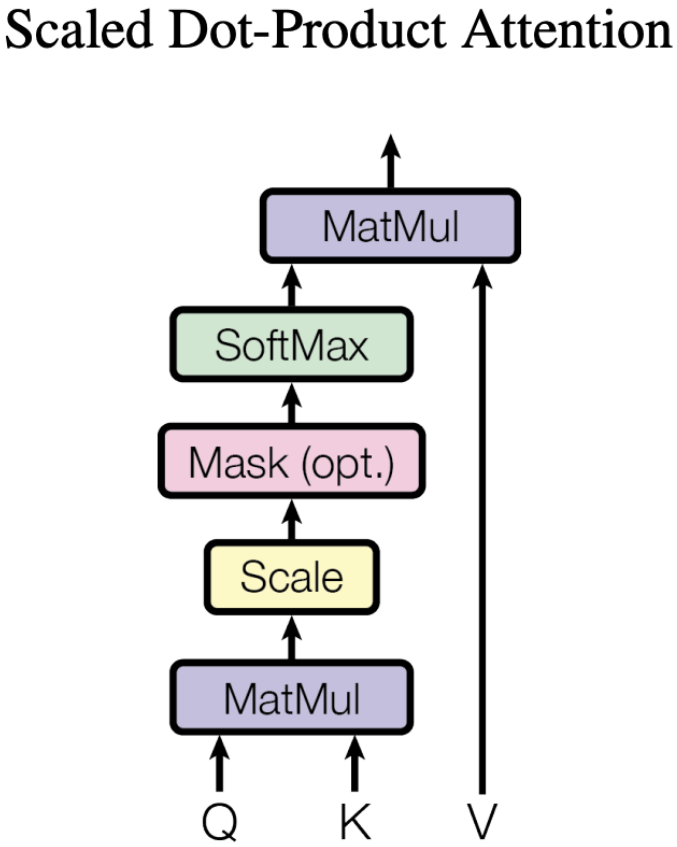
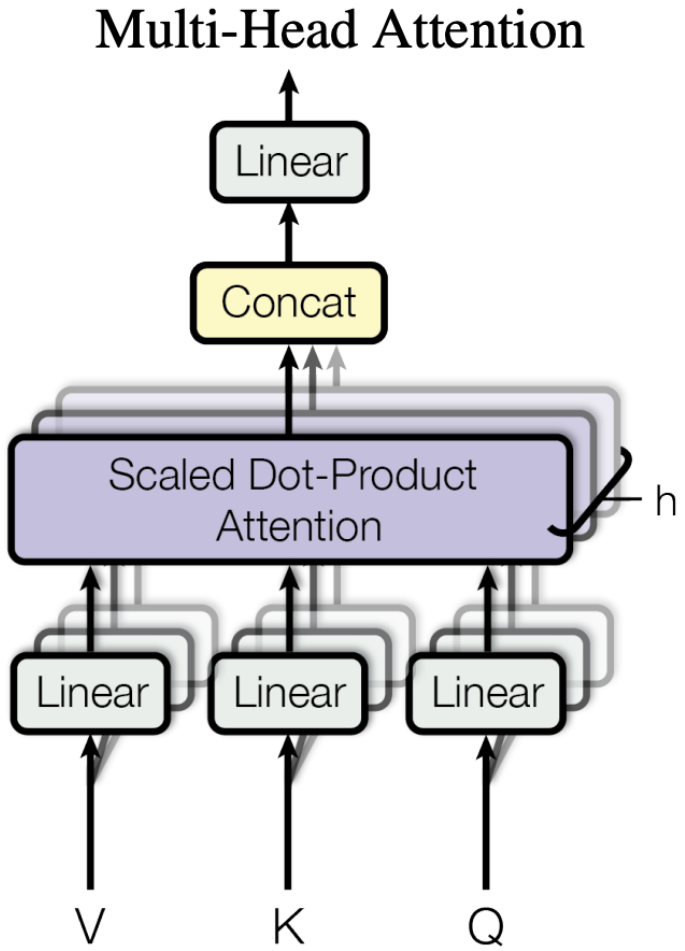


Image credit: Jimmy Ba

Attention Mechanism

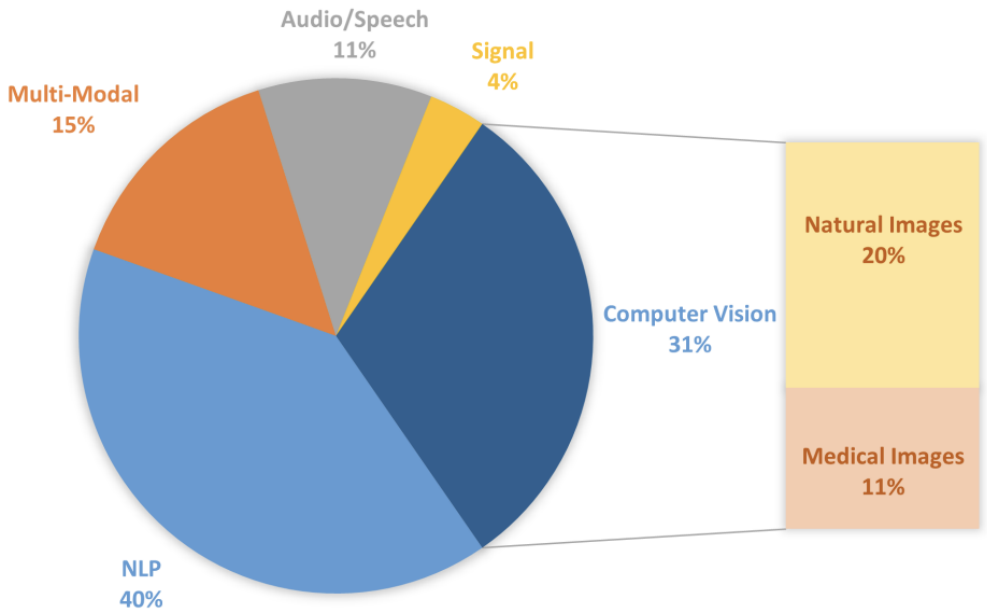




Applications of Transformer

Compared to RNNs, transformers have the following advantages:

- Easier parallelization
- More effective in handling long-range dependencies
- Higher model capacity and flexibility



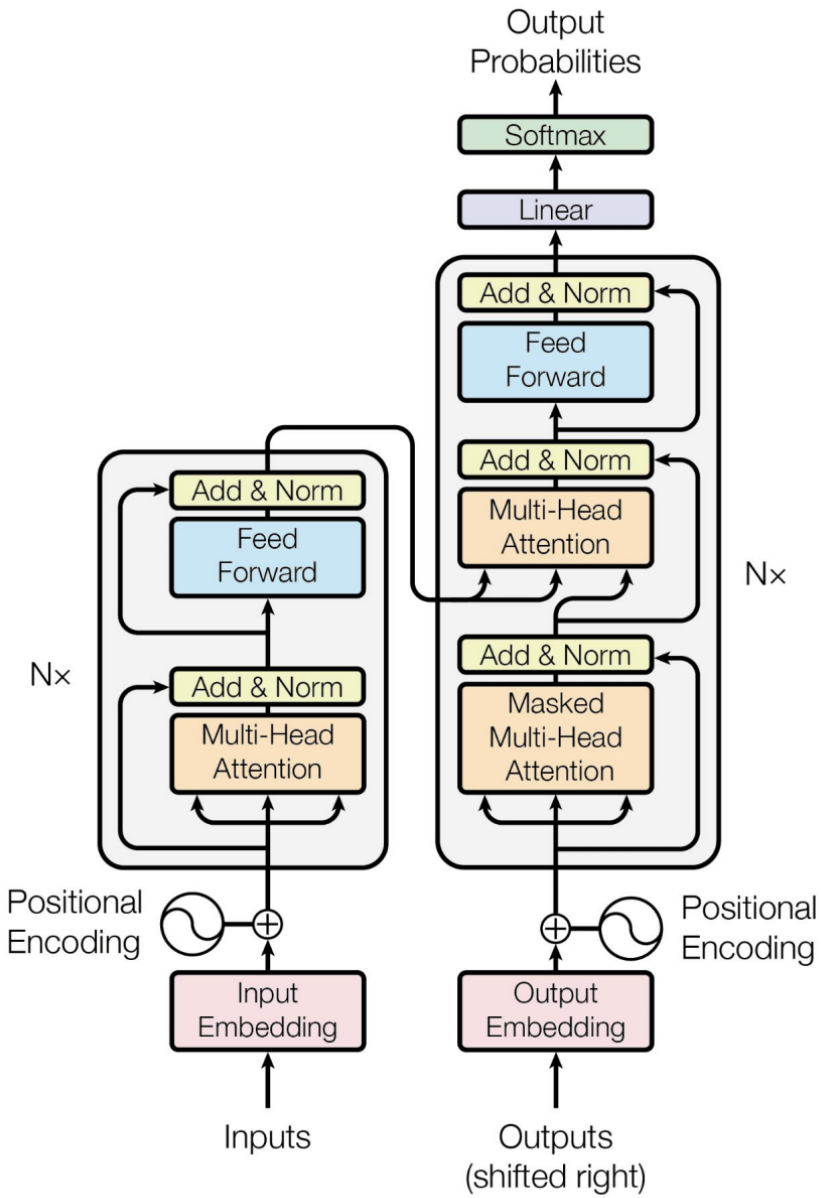


Image source: https://arxiv.org/pdf/2306.07303

Applications of Transformer

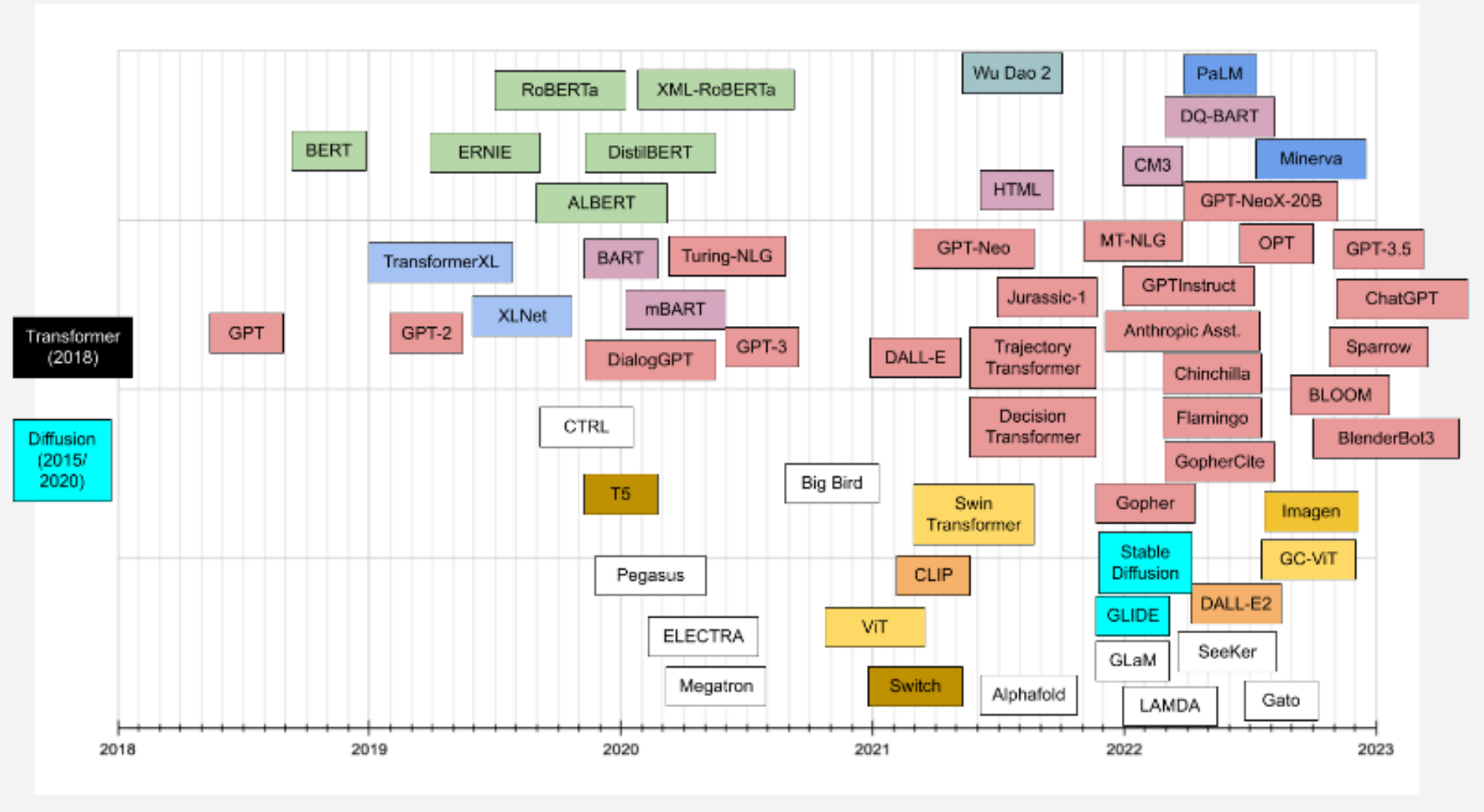


Image source: https://amatriain.net/blog/transformer-models-an-introduction-and-catalog-2d1e9039f376/

1 Sequential Data Analysis

2 Recurrent Neural Network
- Basic RNN Architecture
- Training of Basic RNN
- LSTM
- Other Extensions of RNN (optional)
- Limitations of Basic RNN and Its Variants

3 Transformer
- Overview of Transformer
- Attention Mechanism

4 Summary

Summary

- Tasks of sequential data analysis
- Basic RNN architecture
- Limitations of RNN and its variants
- Transformer
- Attention mechanism