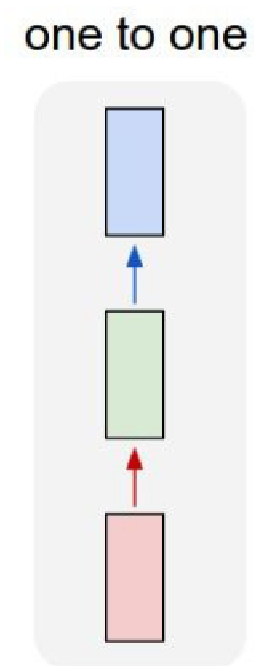
COMP3055
Machine Learning

Topic 16 – RNN,LSTM

Zheng Lu

2024 Autumn

“Vanilla” Neural Network

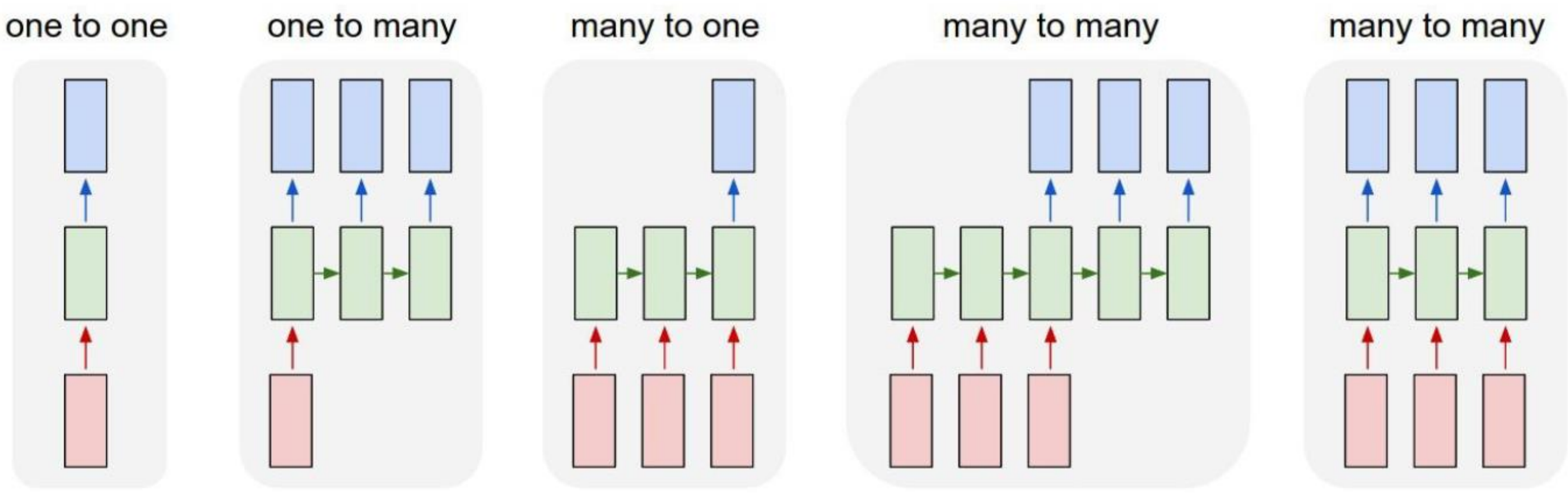


Vanilla Neural Networks

Recurrent Neural Networks

- Excellent models for problems more than one-to-one
  - Time series prediction and classification.
  - Sequence prediction and classification.
  - Simplify some problems that are difficult for multi-layer perceptron.

RNN: Process Sequences



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

Image Caption

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

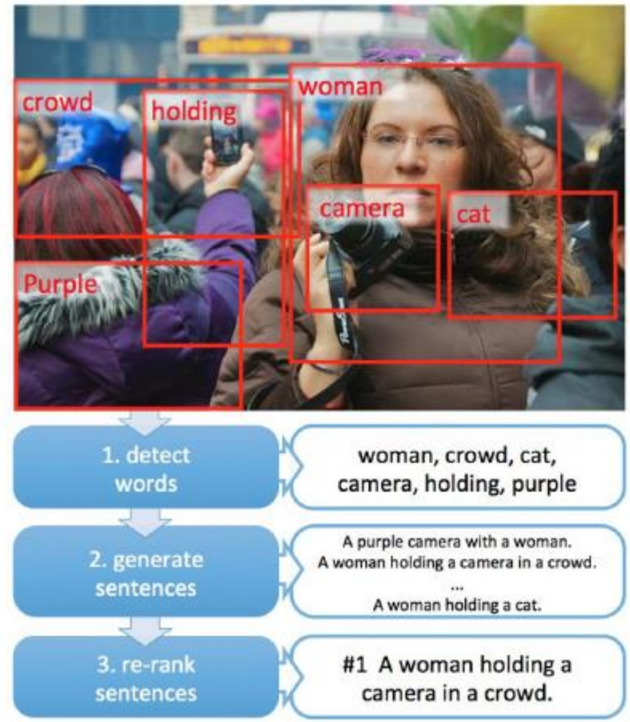
A tennis player in action on the court

Two giraffes standing in a grassy field

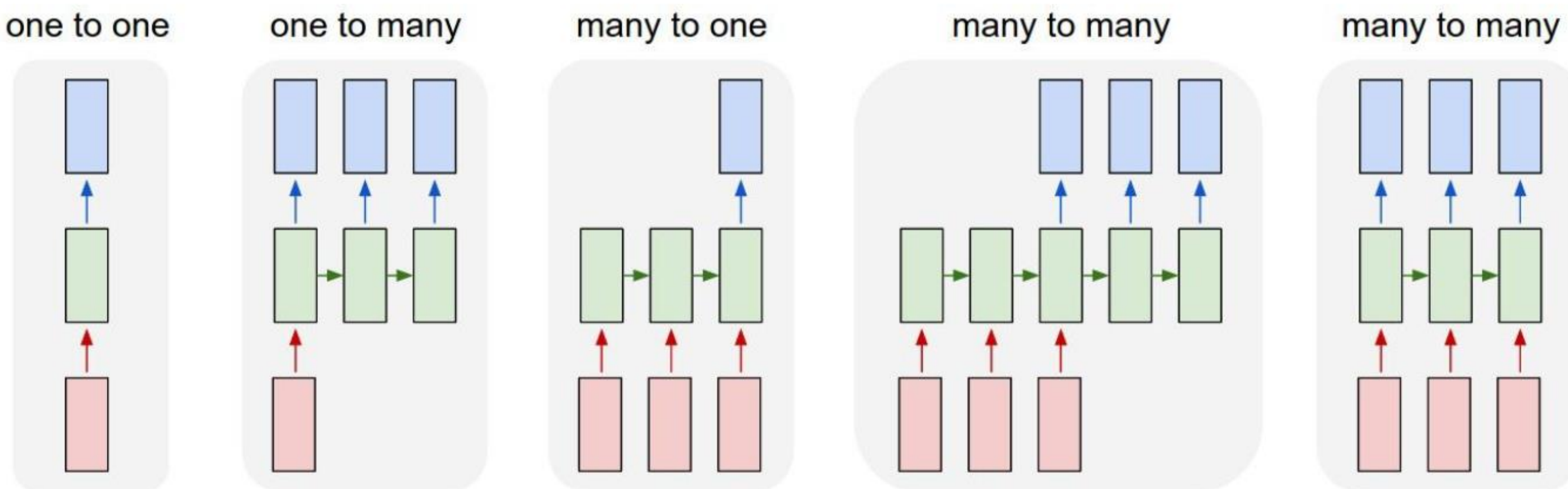
A man riding a dirt bike on a dirt track

Image Caption

a man sitting on a couch with a dog
a man sitting on a chair with a dog in his lap

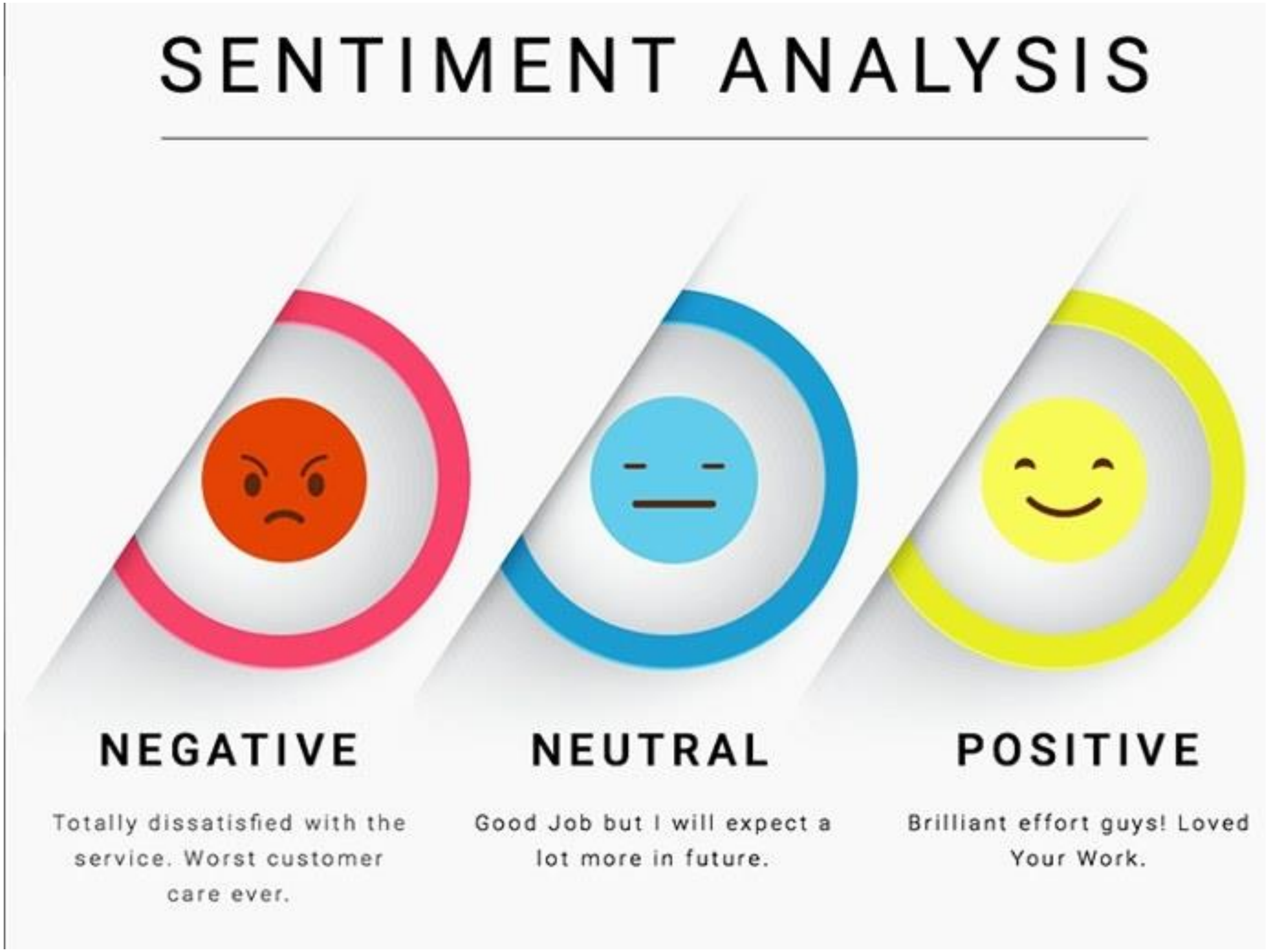


RNN: Process Sequences

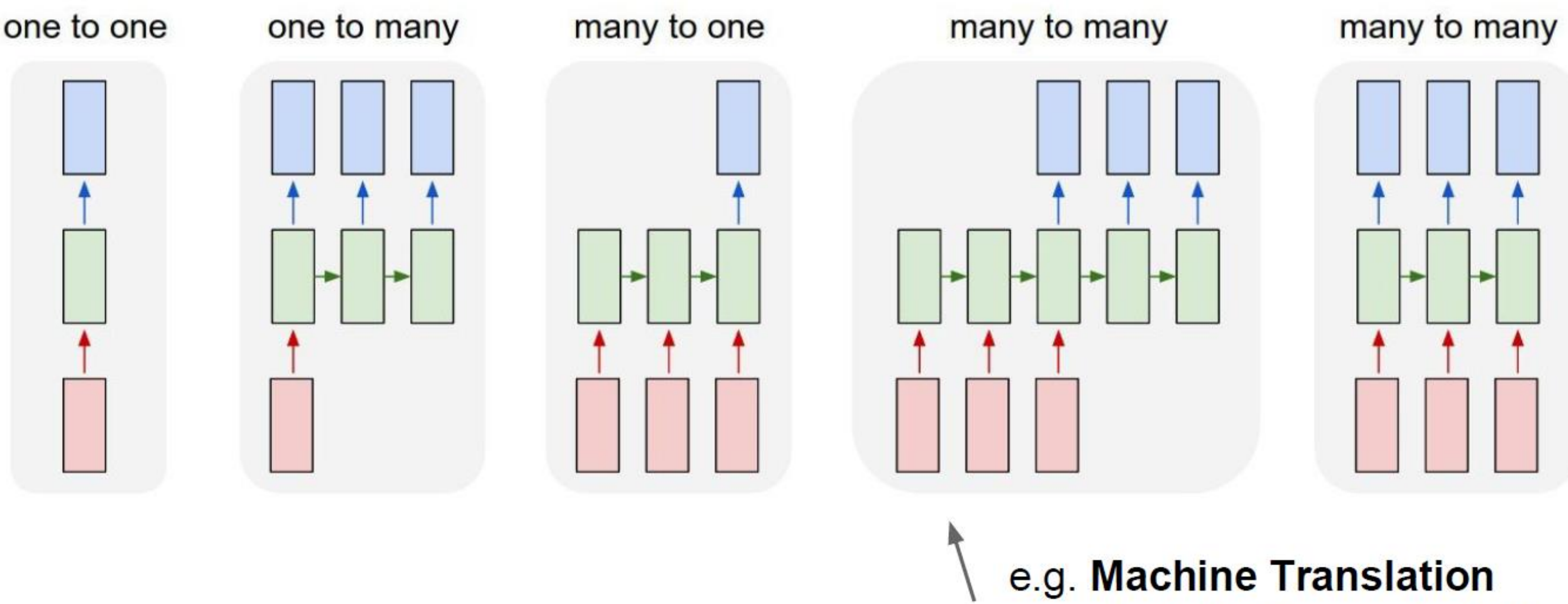


e.g. **Sentiment Classification**
sequence of words -> sentiment

Sentiment Classification

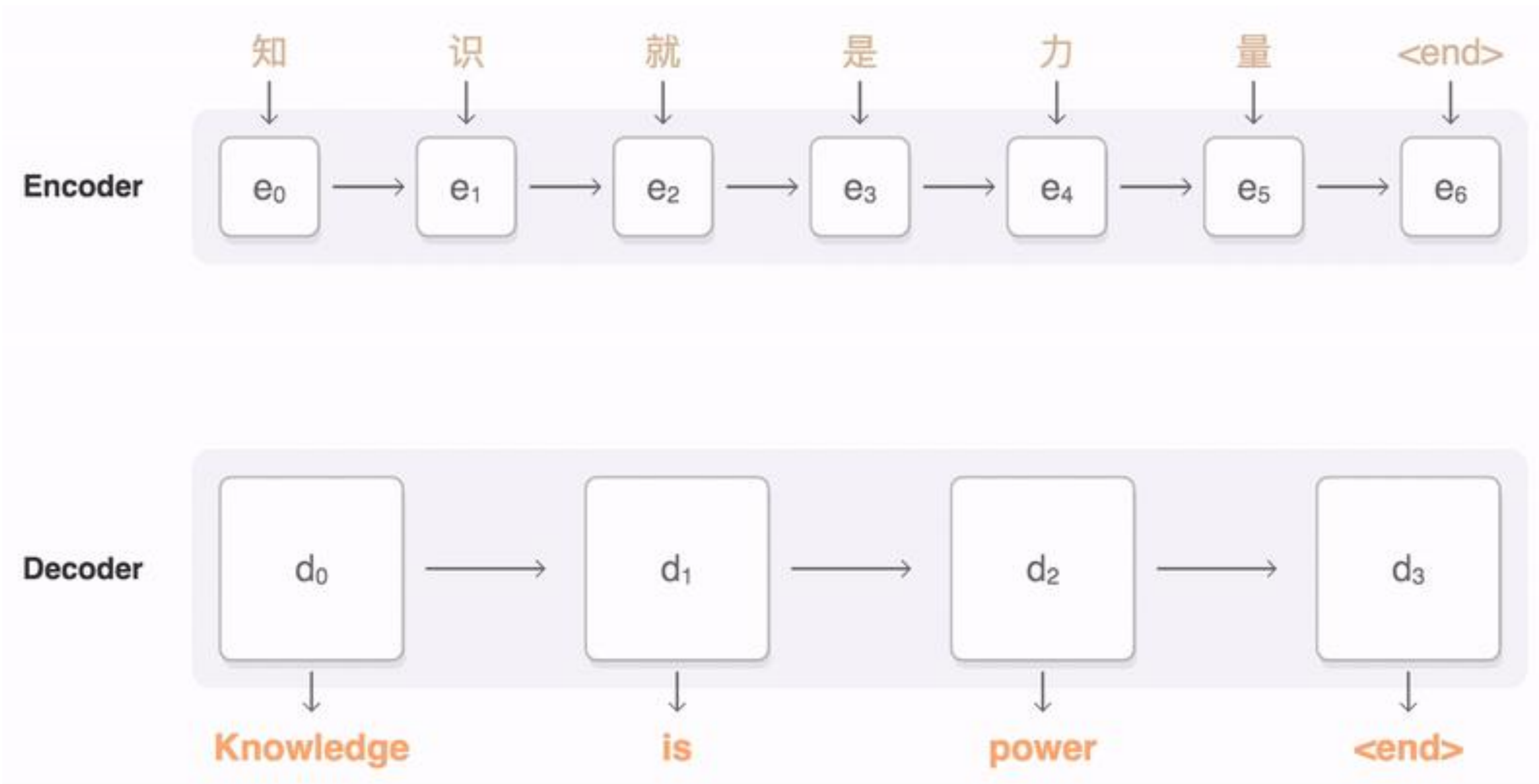


RNN: Process Sequences

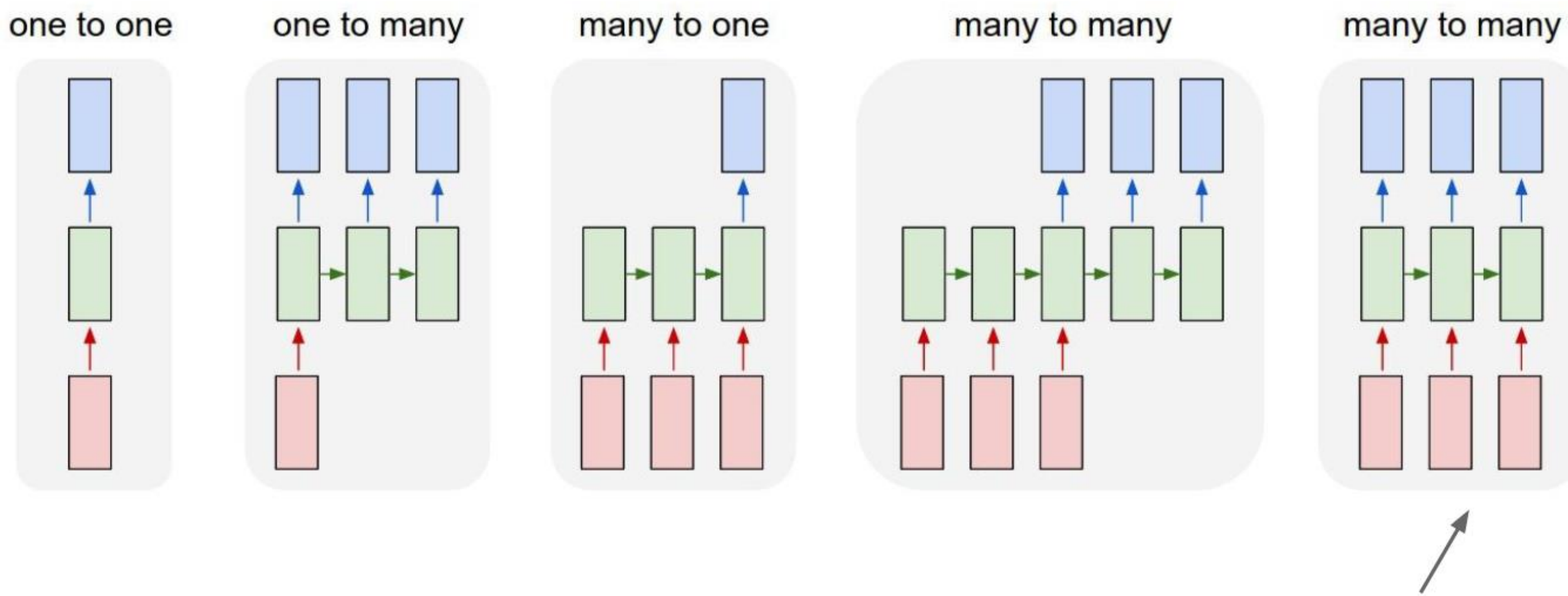


e.g. **Machine Translation**
seq of words -> seq of words

Machine Translation



RNN: Process Sequences



e.g. **Video classification on frame level**

Video Classification (frame level)

RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

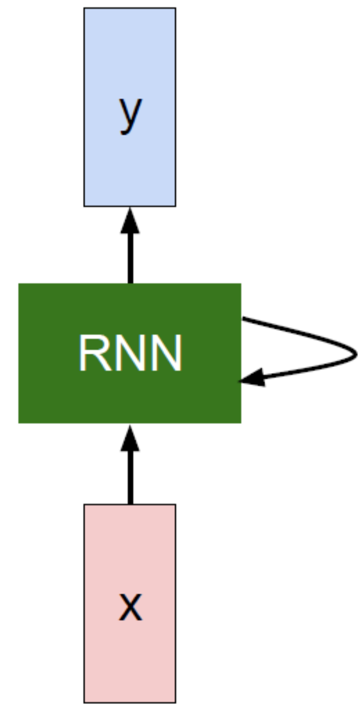
$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

some function with parameters W

old state

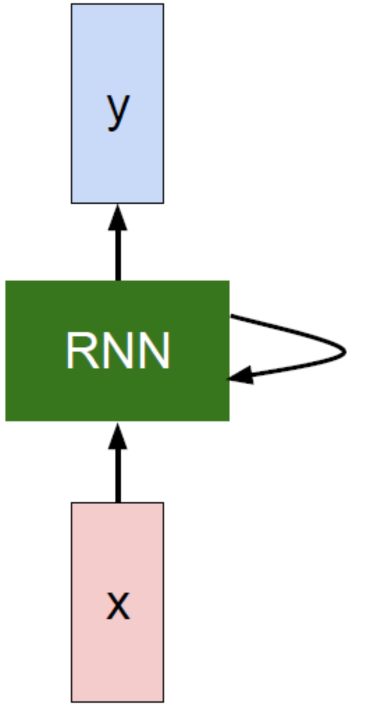
input vector at some time step



RNN

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

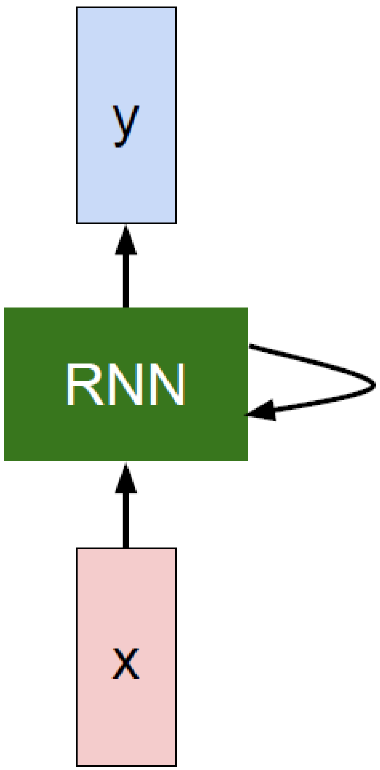
$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

RNN

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

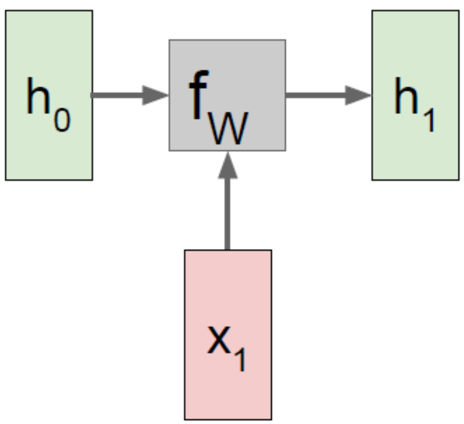


$$h_t = f_W(h_{t-1}, x_t)$$

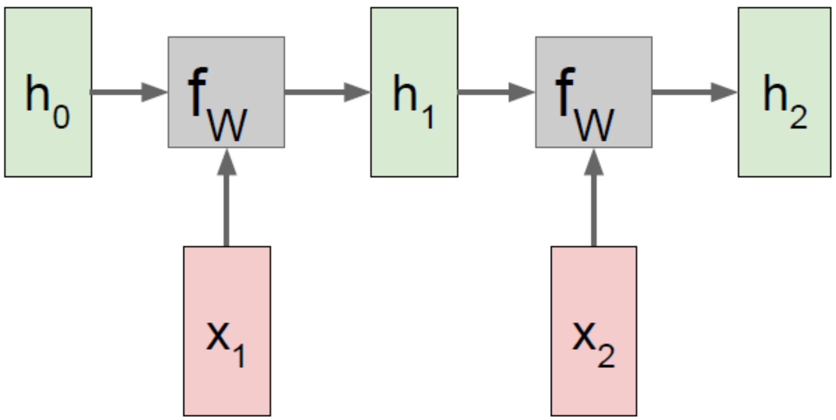
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

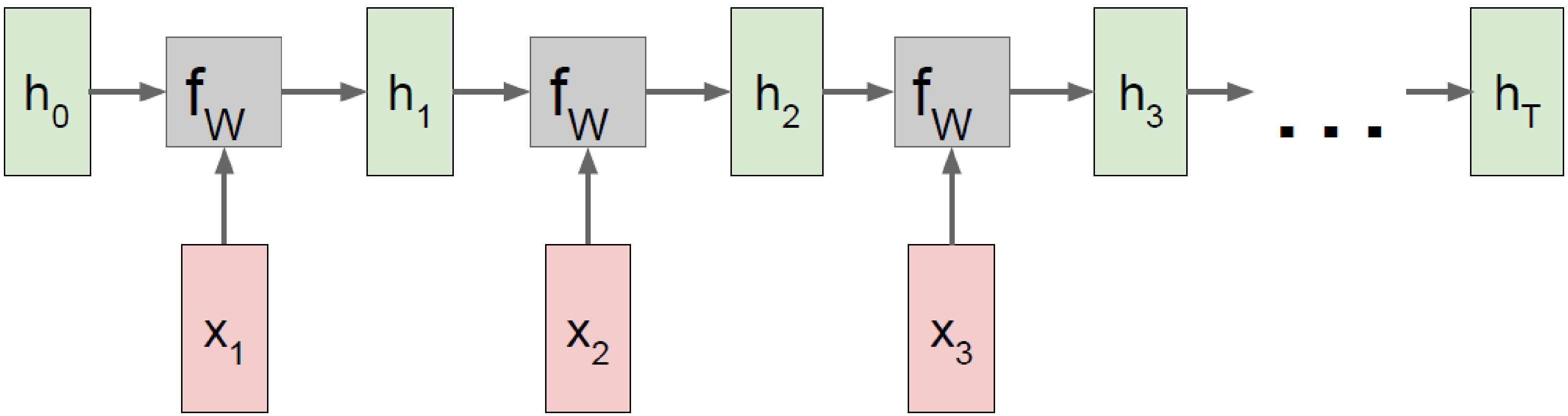
RNN: Computational Graph



RNN: Computational Graph

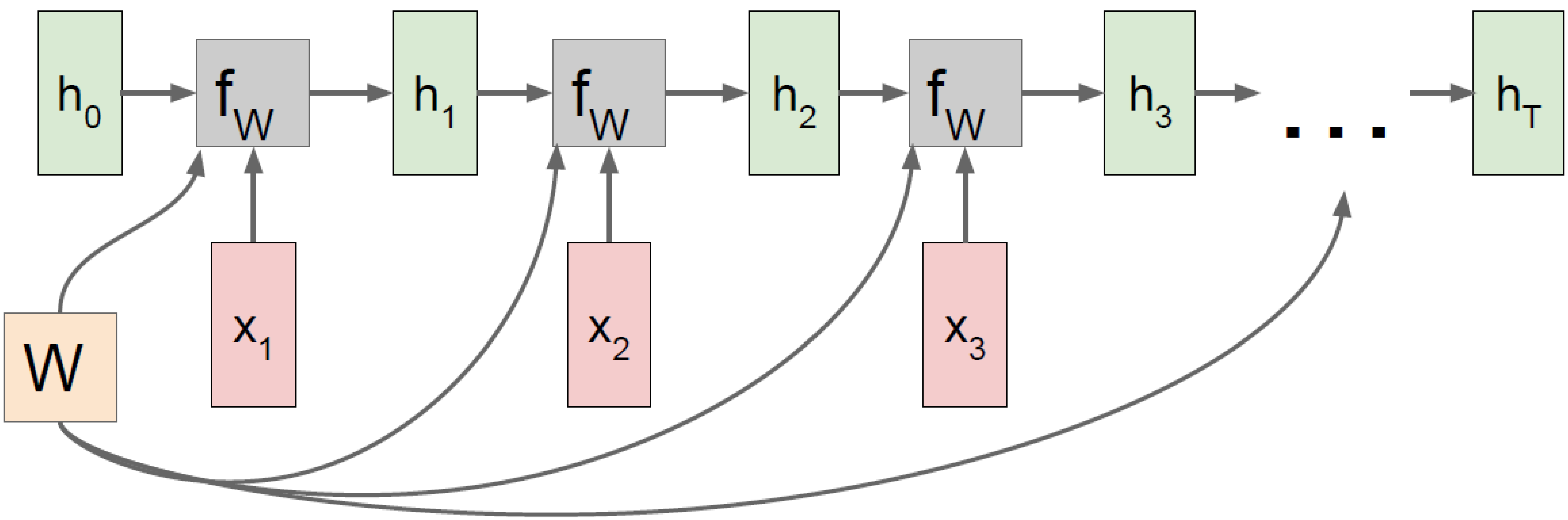


RNN: Computational Graph

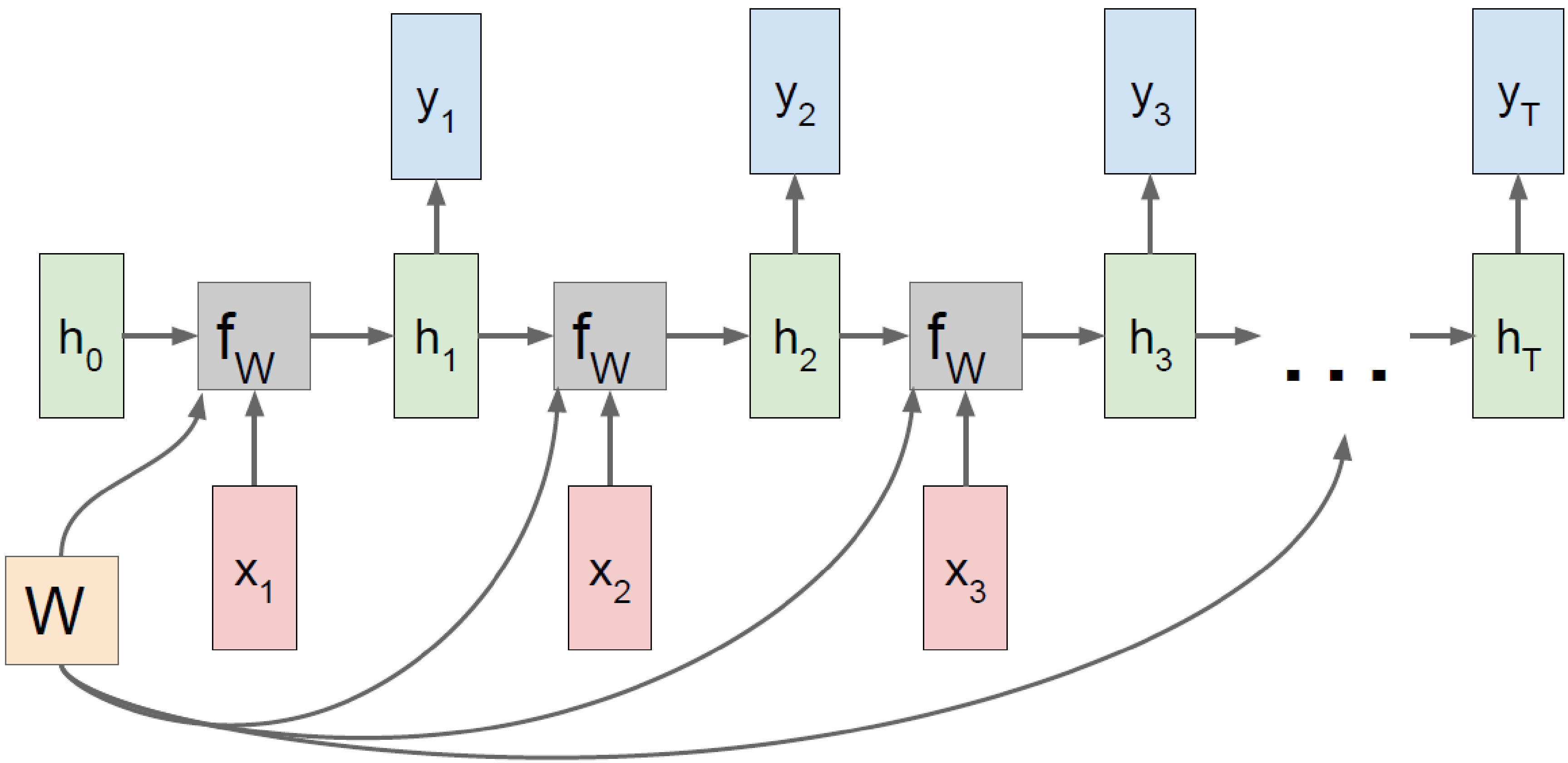


RNN: Computational Graph

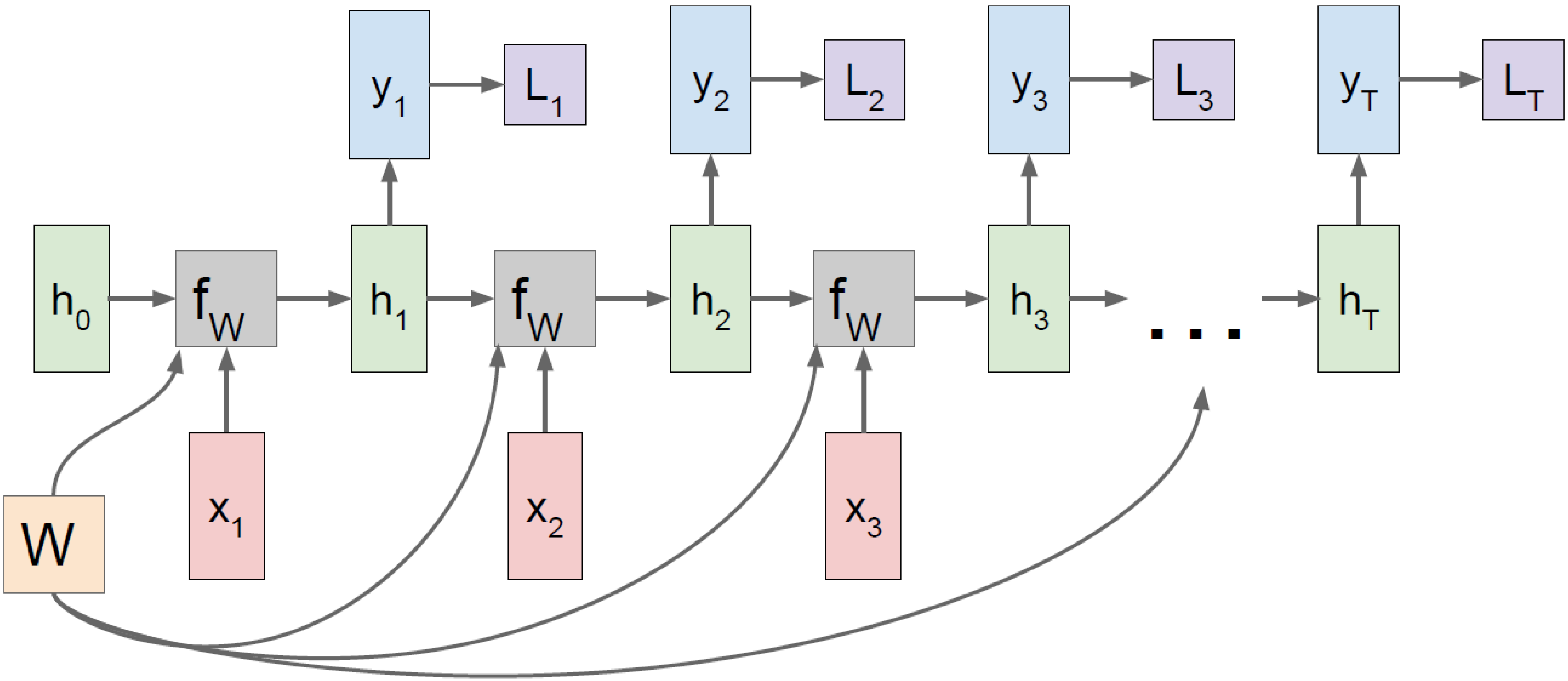
Re-use the same weight matrix at every time-step



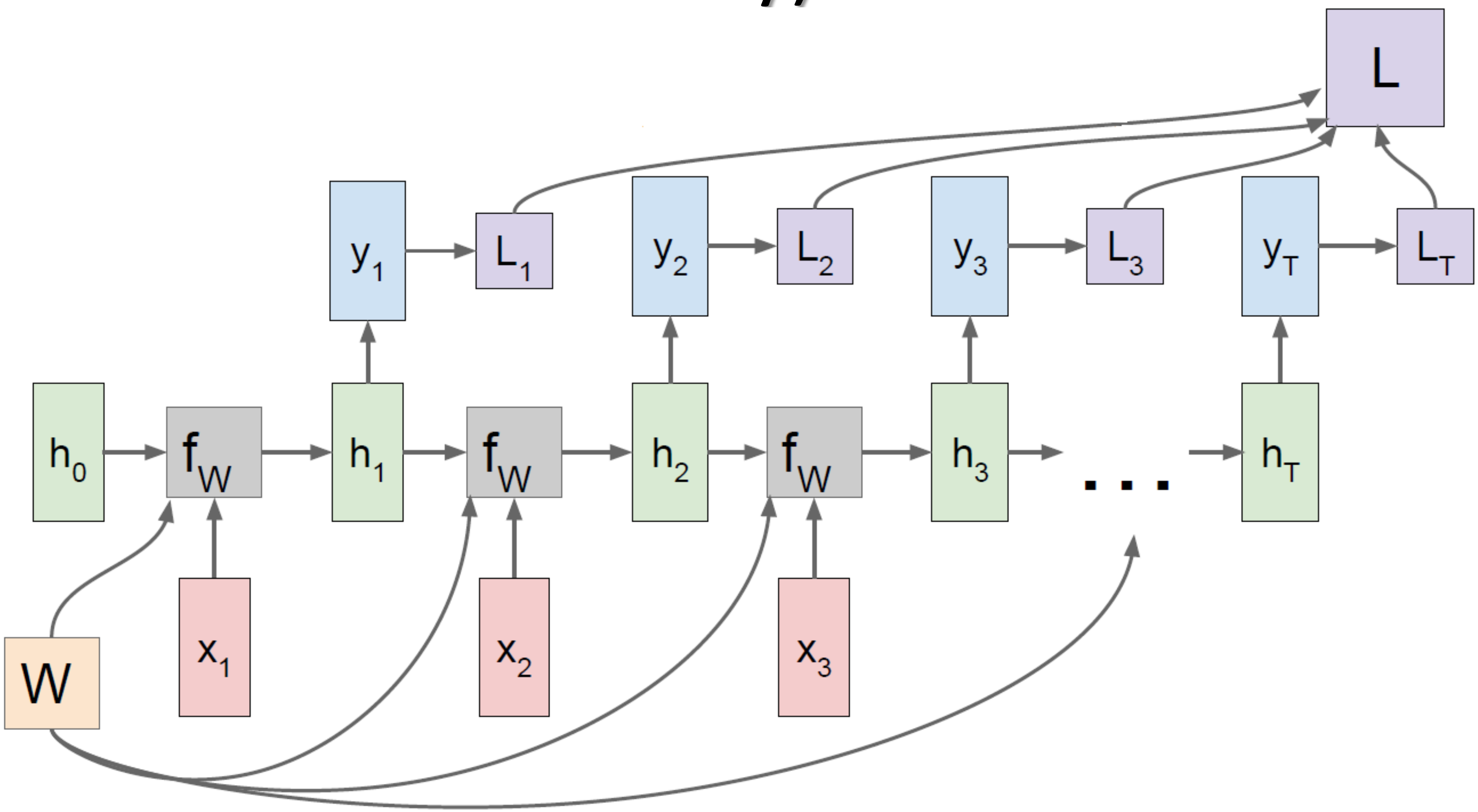
RNN: Computational Graph (Many to Many)



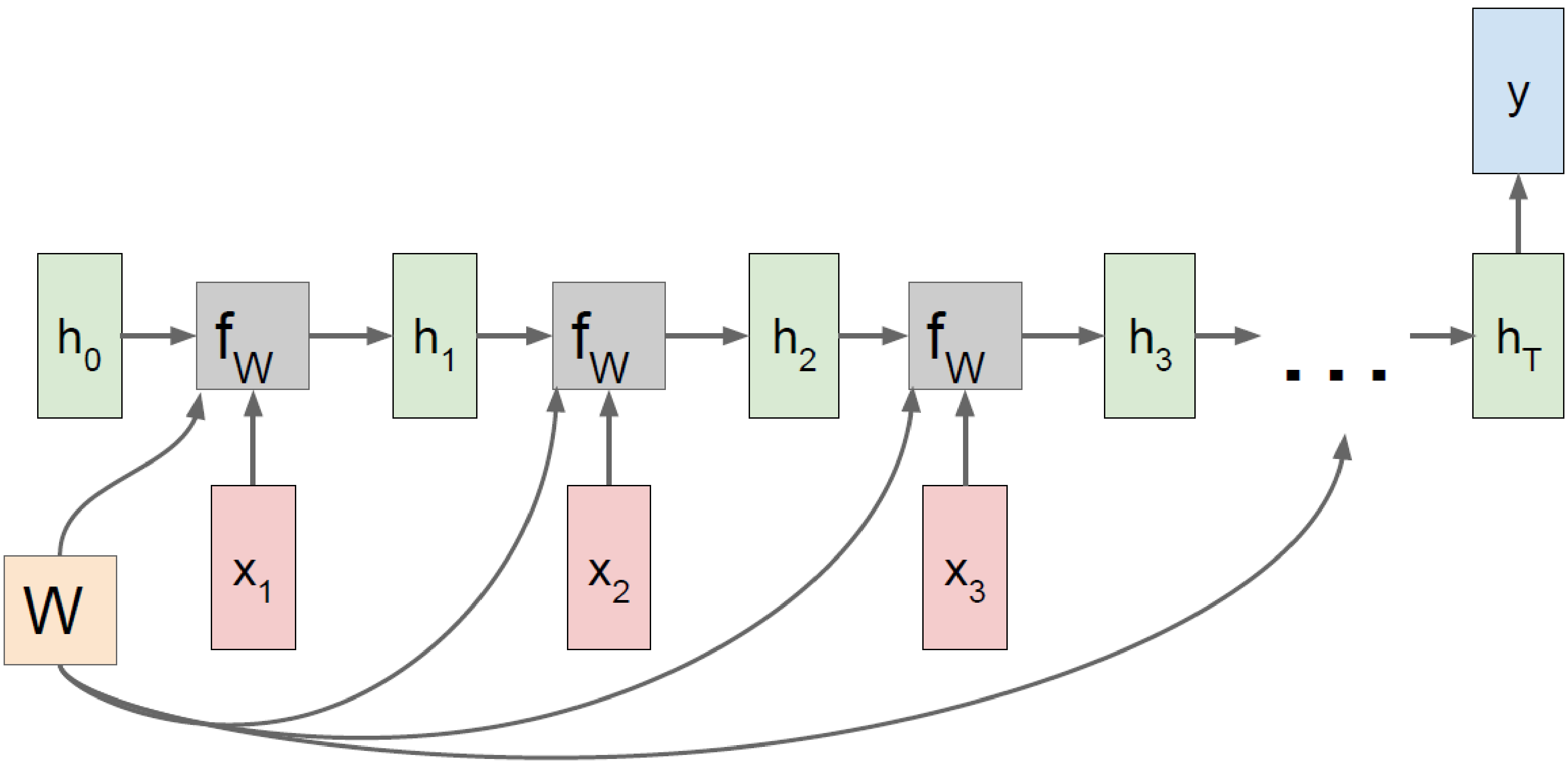
RNN: Computational Graph (Many to Many)



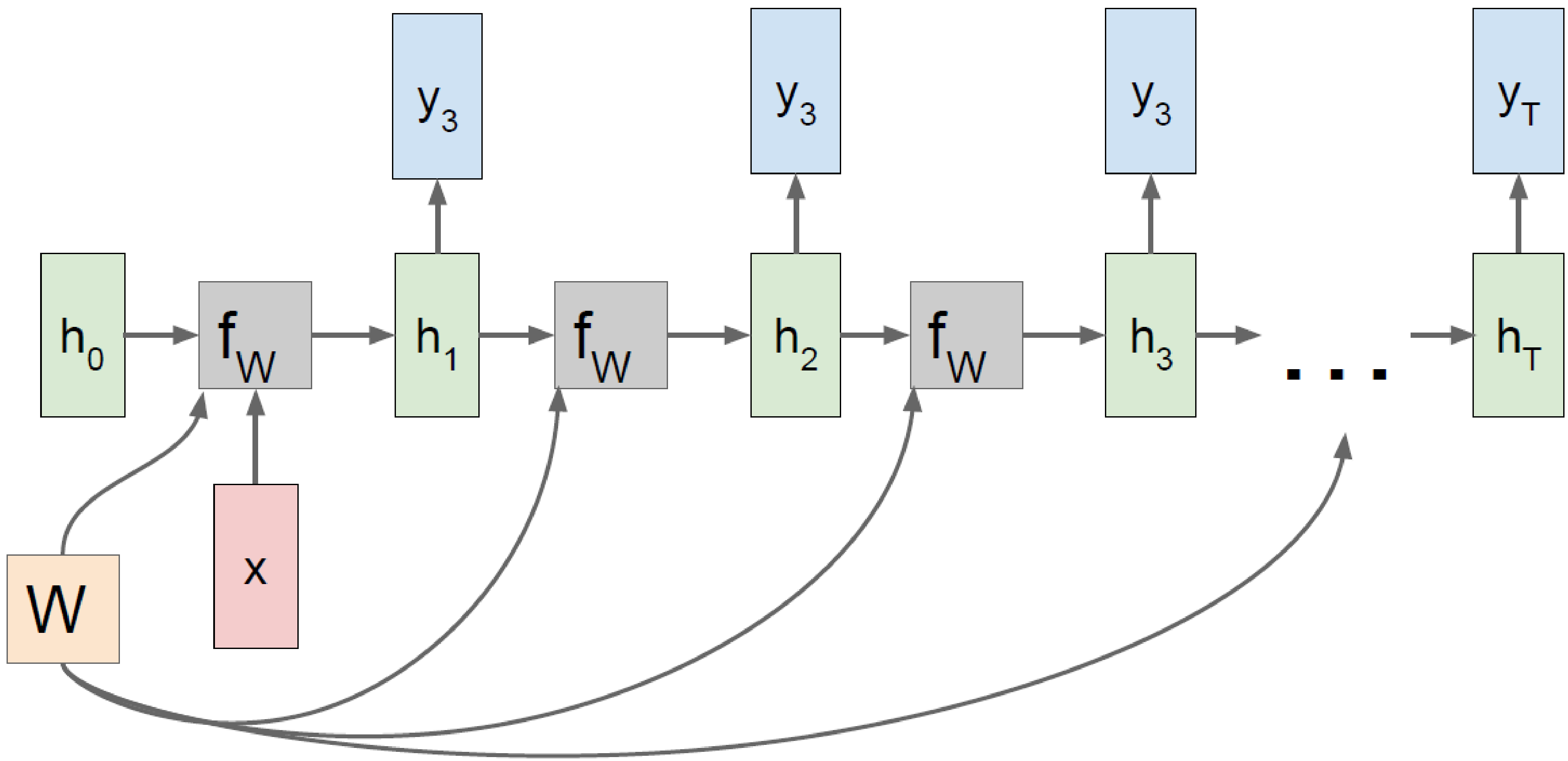
RNN: Computational Graph (Many to Many)



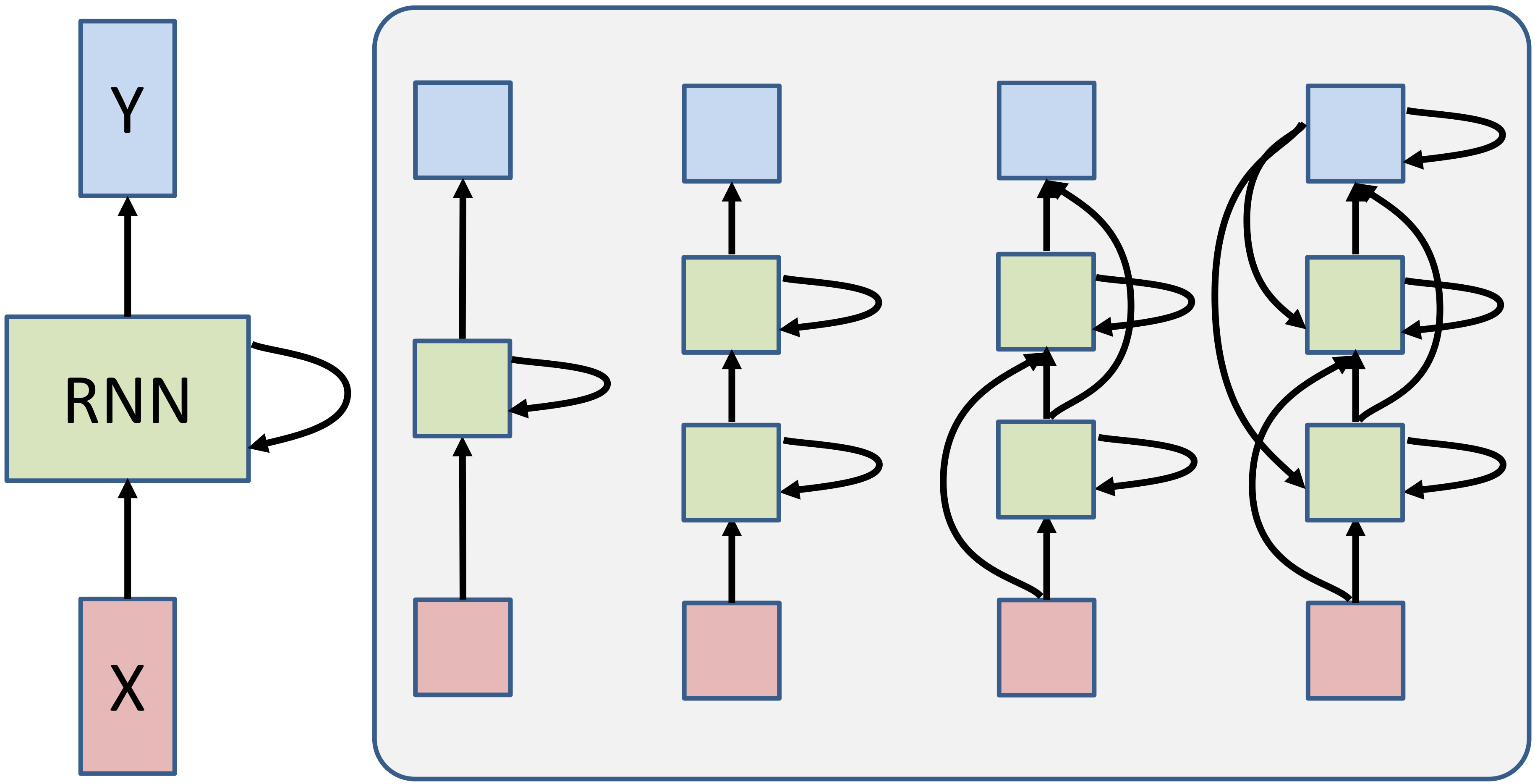
RNN: Computational Graph (Many to One)



RNN: Computational Graph (One to Many)



RNN: other design



RNN can be designed very sophisticatedly with different layers different ways of recurrency

RNN

- In theory RNN retains information from the infinite past.
  - All past hidden state has influence on the future state.

- In practice RNN has little response to the early states.
  - Little memory over what seen before.
  - The hidden outputs blowup or shrink to zeros.
  - The “memory” also depends on activation functions.
  - ReLU and Sigmoid do not work well. Tanh is OK but still not “memorize” for too long.

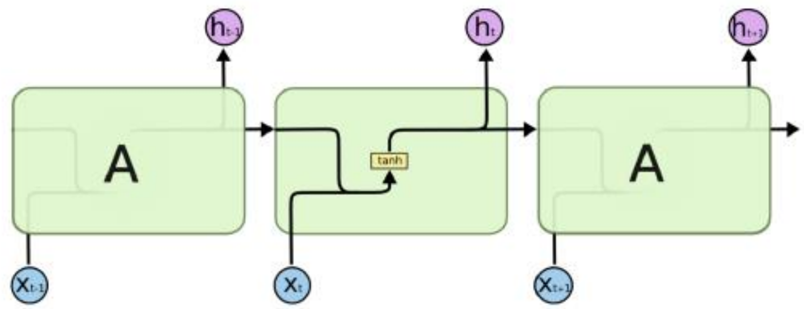
- Vanishing gradient problem
  - Deeper layers do not have meaningful weights.

Long-Term Dependency

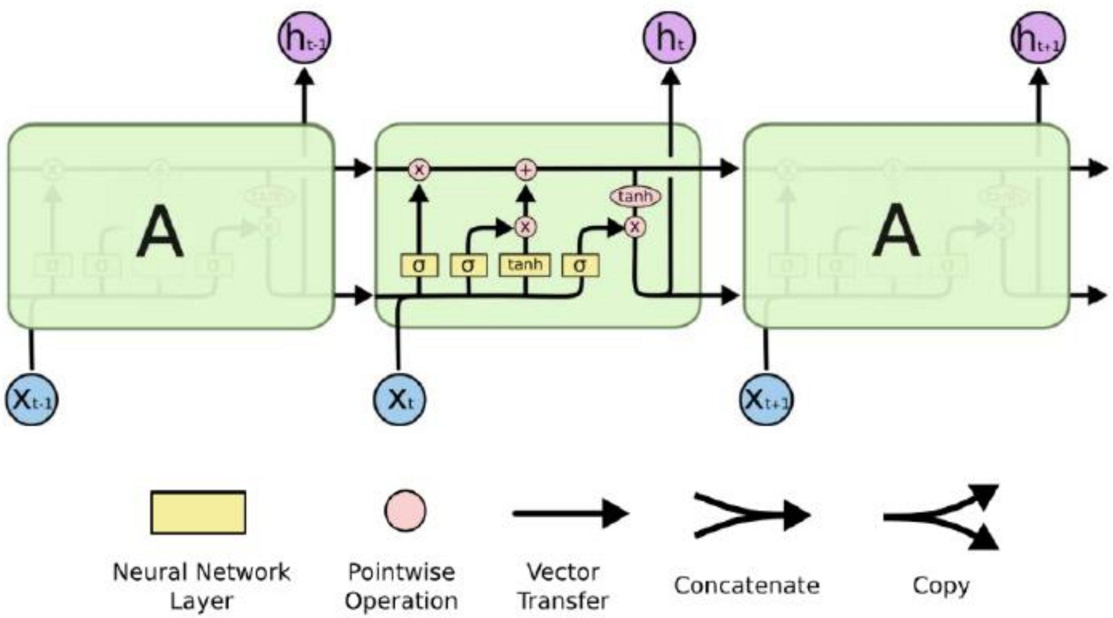
- In theory, vanilla RNNs can handle arbitrarily long term dependence
- In practice, it's difficult
- Long-term dependency:
  - **Bob** likes **apples**. He is hungry and decided to have a snack. So now he is eating an **apple**.

LSTM: Gates Regulate

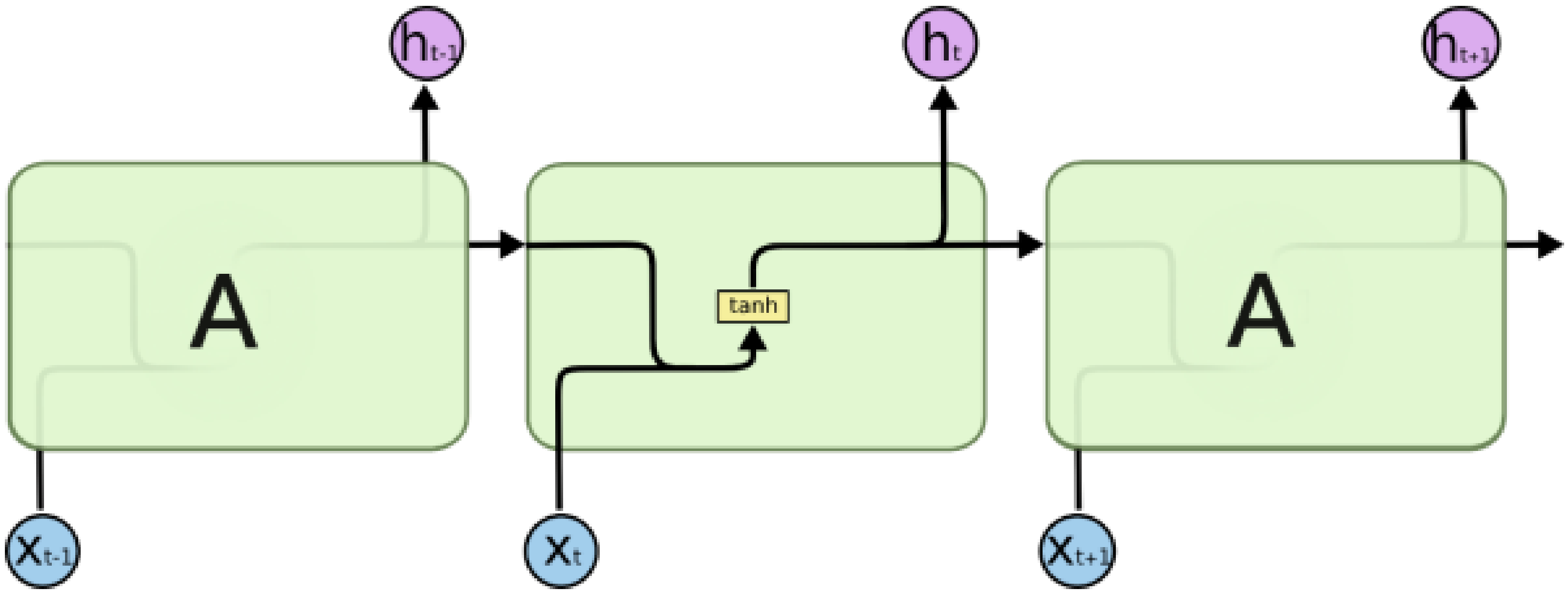
Vanilla RNN:



LSTM:

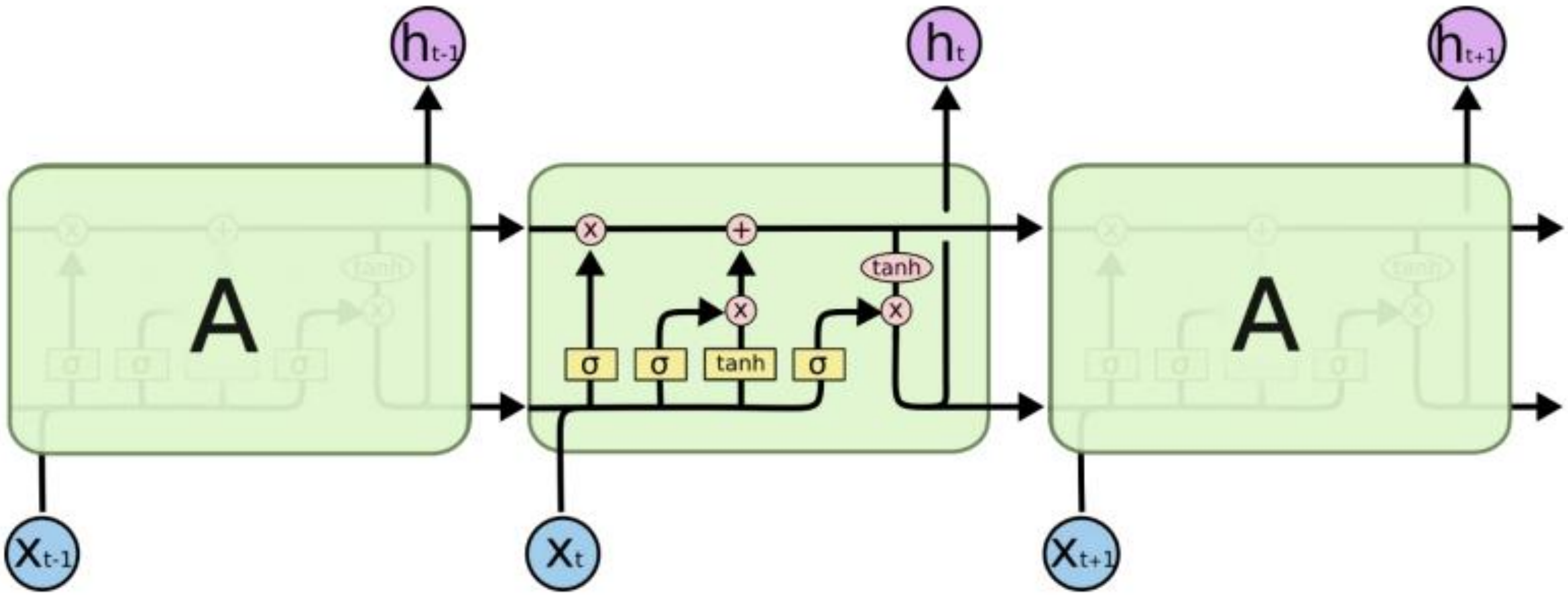


RNN VS LSTM

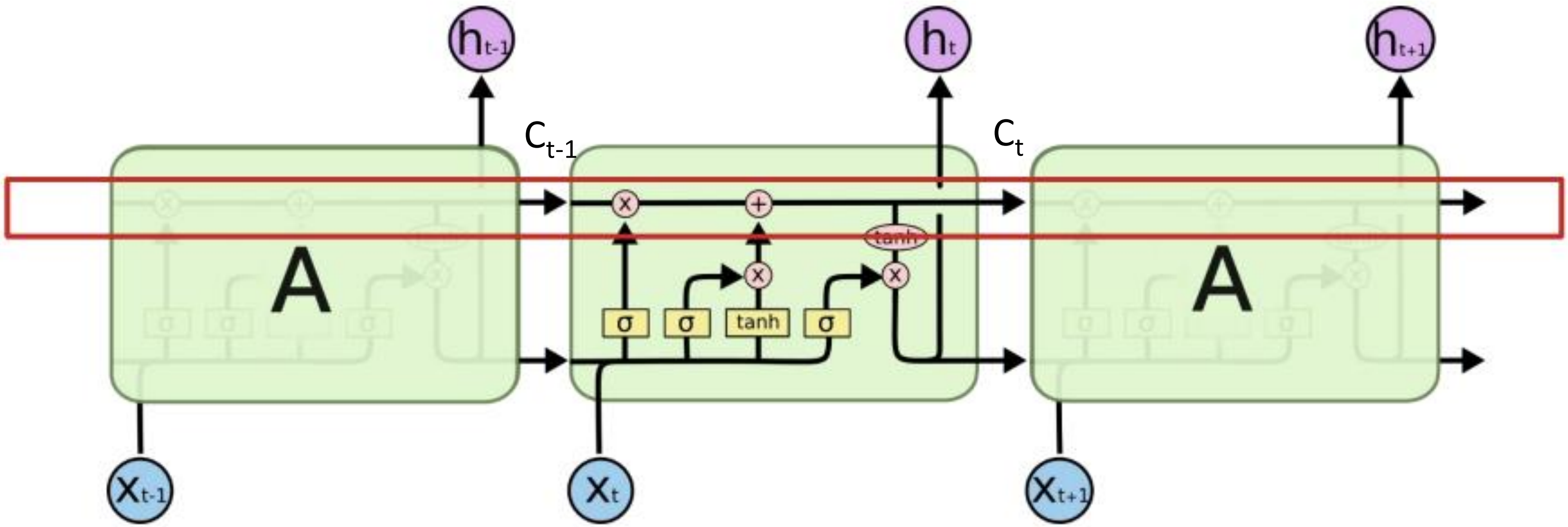


- Recurrent neurons receive past recurrent outputs and current input as inputs.
- Processed through a tanh() activation function
- Current recurrent output passed to next higher layer and next time step.

LSTM



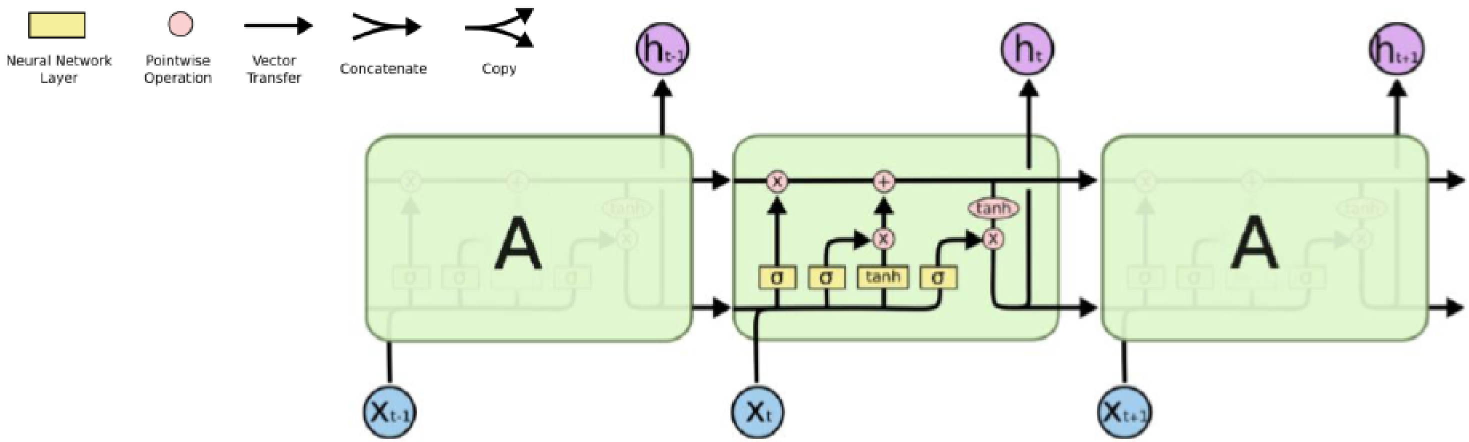
LSTM



Constant Error Carousel

- Key of LSTM: a remembered cell state
- C_t is the linear history carried by the constant error carousel.
- Carries information through and only effected by a gate
    - Addition of history (gated).

LSTM

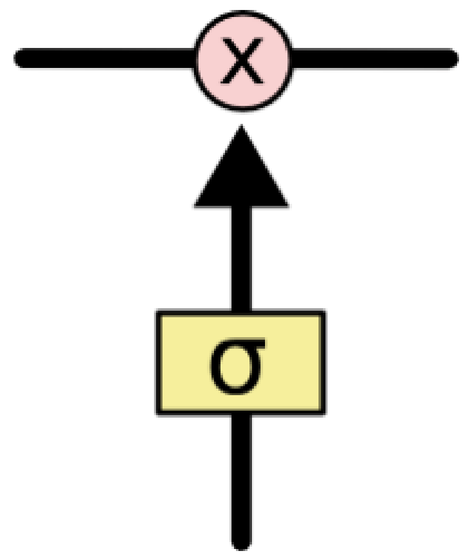


Bob and Alice are having lunch. Bob likes apples. Alice likes oranges. **She is eating an orange.**

Conveyer belt for **previous state** and **new data**:

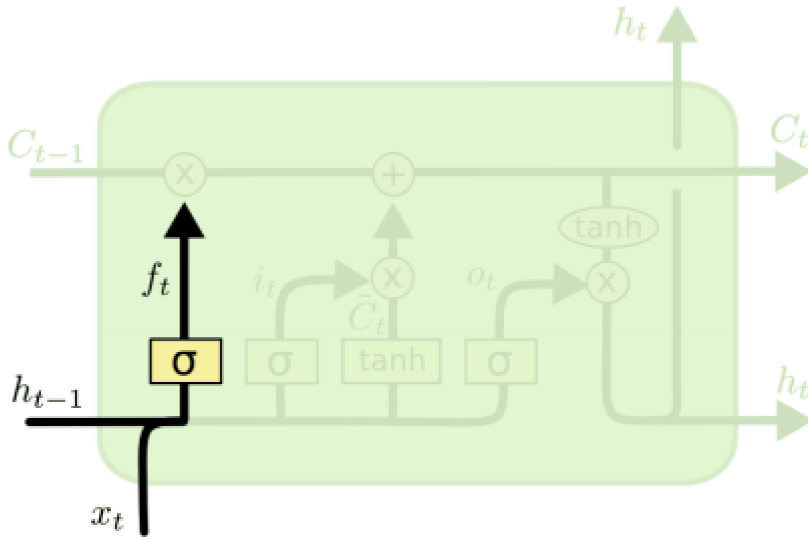
1. Decide what to forget (state)
2. Decide what to remember (state)
3. Decide what to output (if anything)

LSTM - Gate



- A simple sigmoid function to project output in range (0, 1).
  - Information is let through (~1)
  - Information is not let through (~0)
- ⨂: element-wise multiplication.

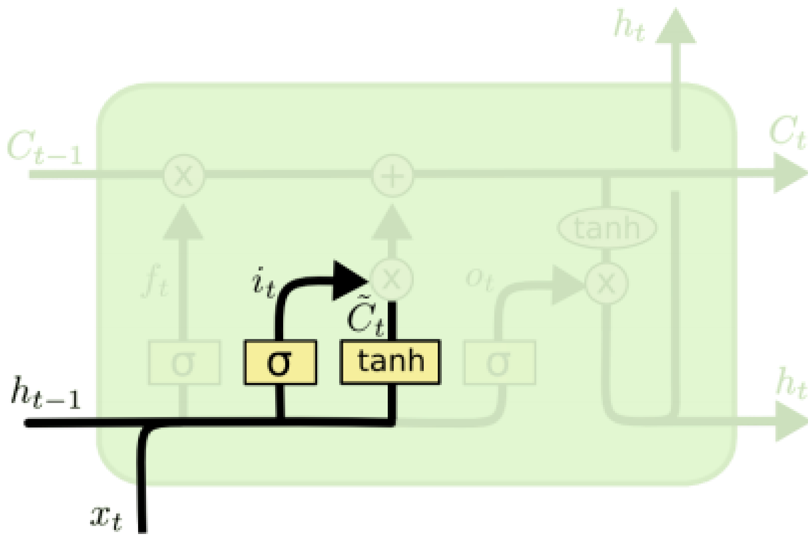
LSTM – Forget Gate



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

- The first gate determines whether to carry over the history or forget it
  - Called "forget" gate.
  - Actually, determine how much history to carry over.
  - The memory C and hidden state h are distinguished.

LSTM – Input Gate

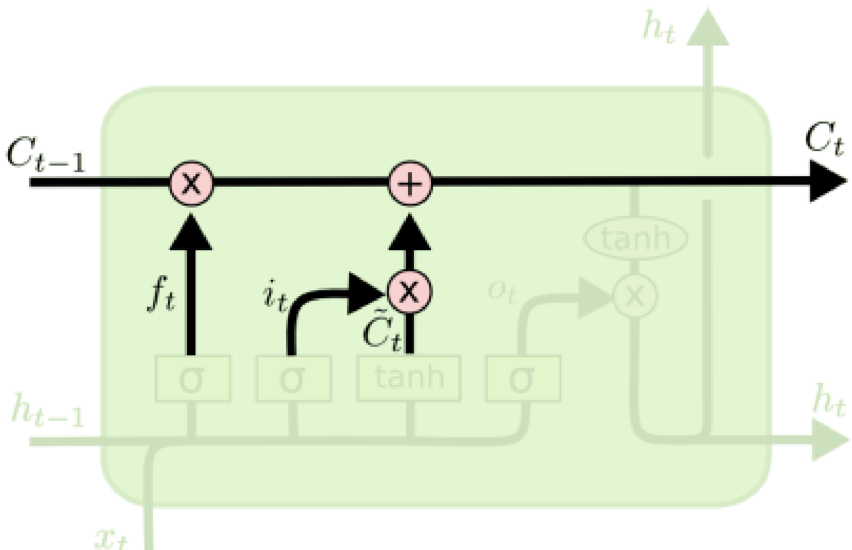


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

The second gate has two parts

- A *tanh* unit determines if there is something new or interesting in the input.
- A gate decides if it is worth remembering.

LSTM – Memory Cell Update

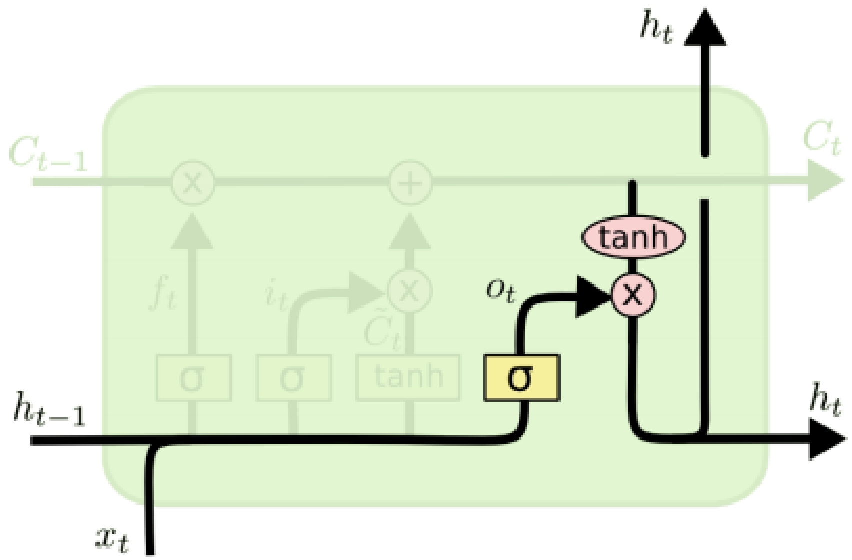


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Add the output of input gate to the current memory cell

- After the forget gate.
- ⊕: Element-wise addition.
- Perform the forgetting and the state update

LSTM – Output and Output Gate



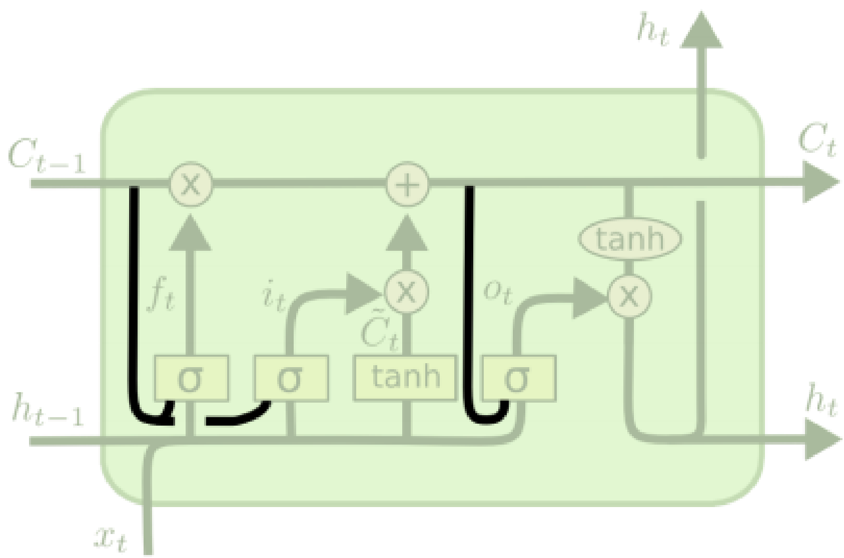
$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

The output of the memory cell

- Similar to input gate.
- A *tanh* unit over the memory to output in range [-1, 1].
- A *sigmoid* unit [0,1] decide the filtering.
- Note the memory is carried through without *tanh*.

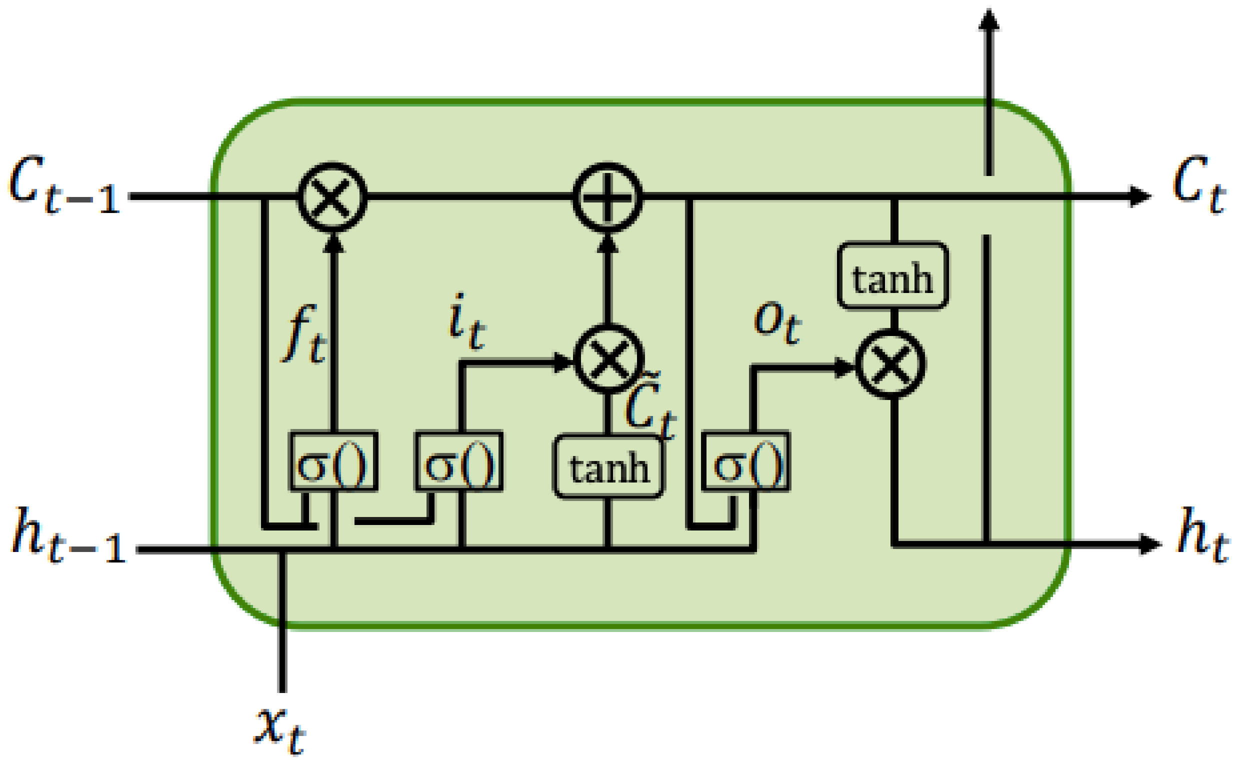
LSTM – the "Peephole" Connection



$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Let the memory cell directly influence the gates!

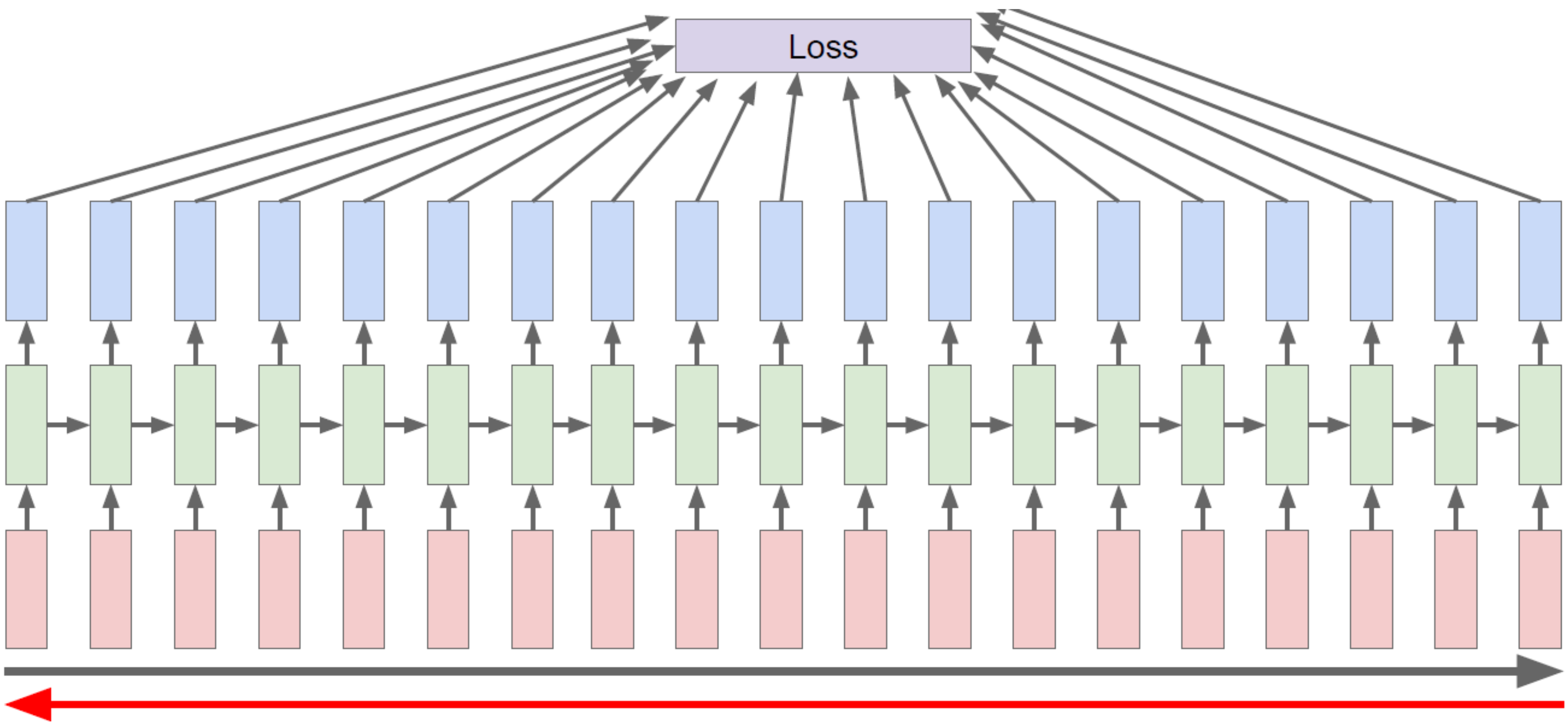
The Complete LSTM Unit



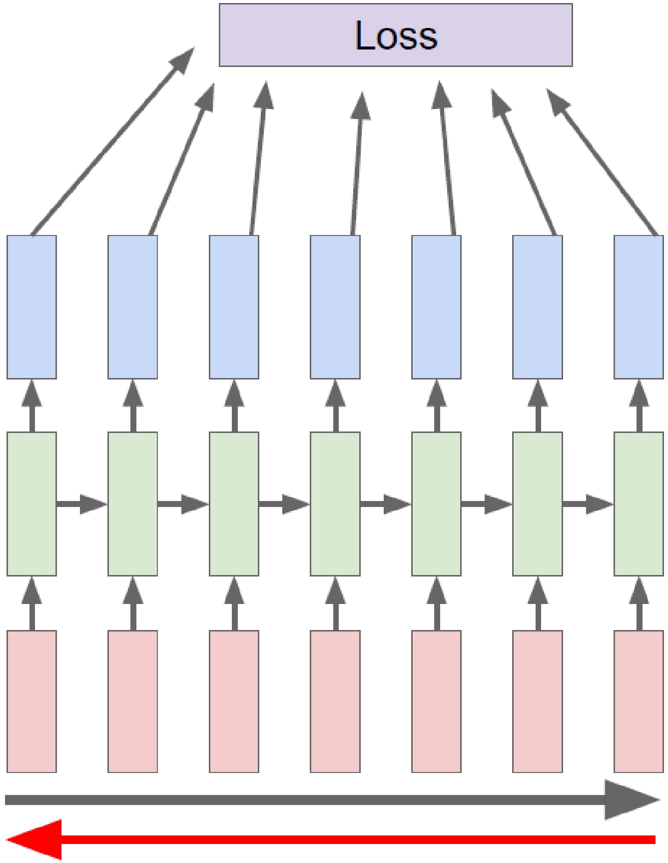
Input, output, forget gates with peephole connection

Back Propagation Through Time (BPTT)

Forward through entire sequence to compute loss, then backward through entire sequence to compute gradient

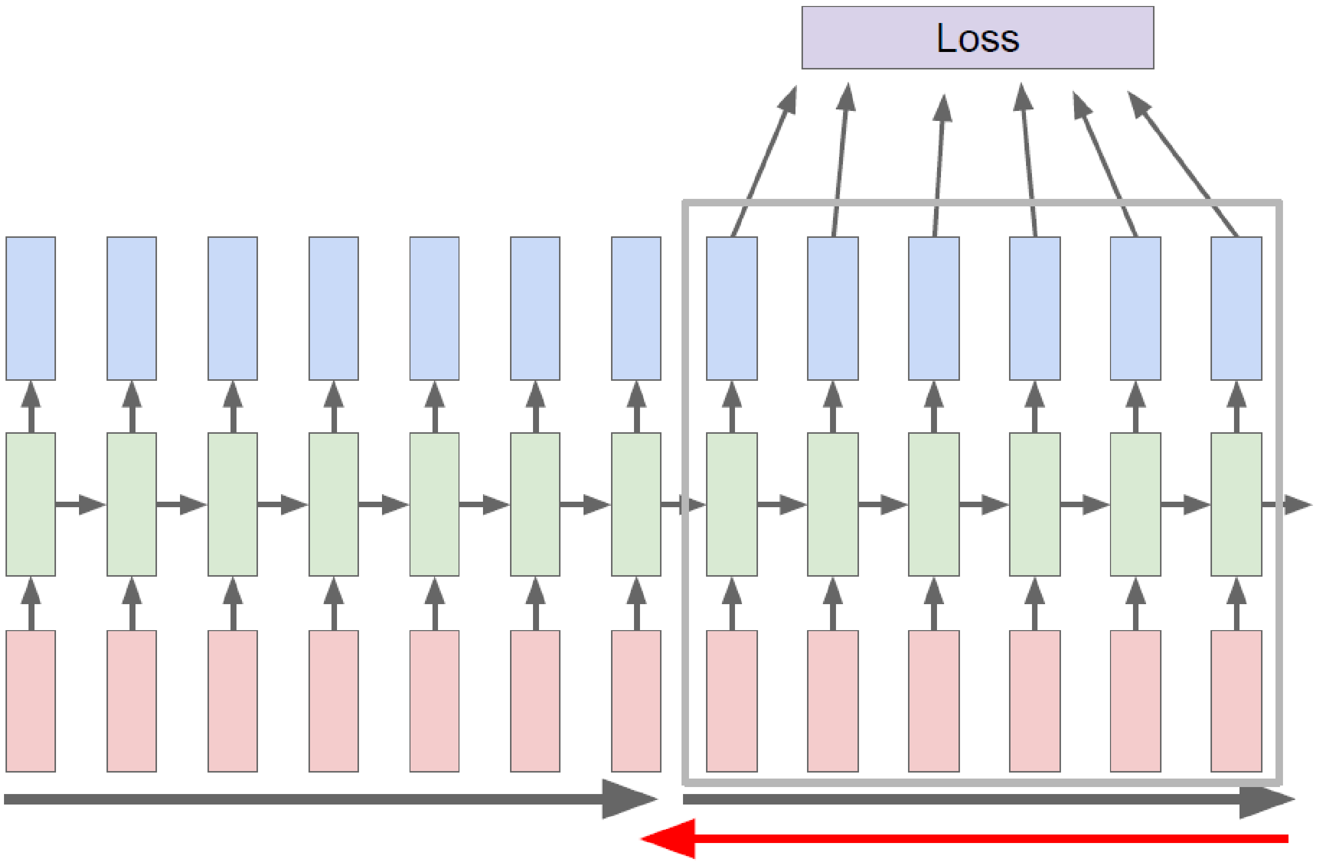


Truncated BPTT



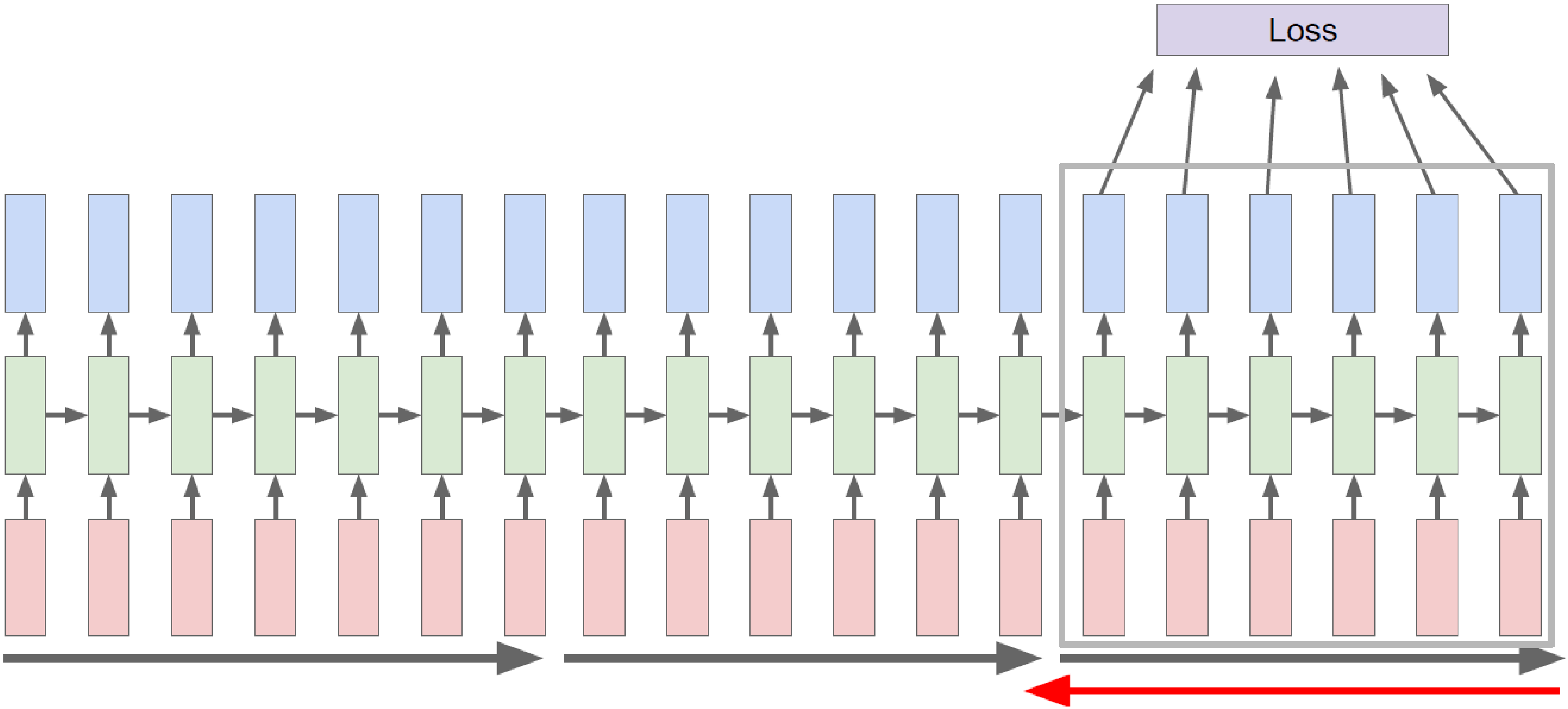
Run forward and backward through chunks of the sequence instead of whole sequence

Truncated BPTT



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

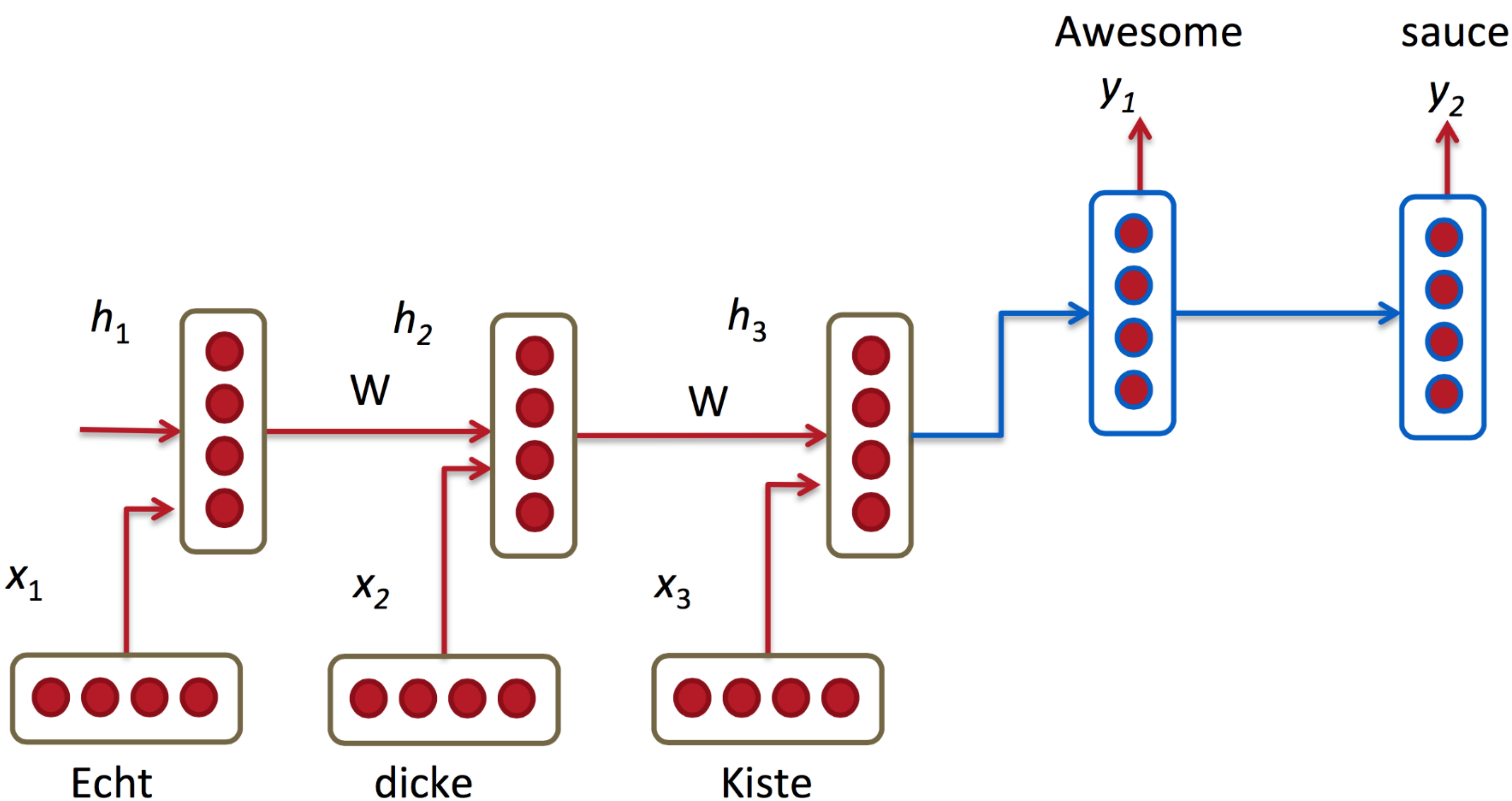
Truncated BPTT



Applications of LSTM

- Nowadays, considered as the default models for sequence labeling tasks.
- Does not suffer from Vanishing Gradient problem.
- Very powerful, especially in deeper networks.
- Very useful when you have a lot of data.

Machine Translation



Machine Translation

Method	test BLEU score (ntst14)
Bahdanau et al. [2]	28.45
Baseline System [29]	33.30
Single forward LSTM, beam size 12	26.17
Single reversed LSTM, beam size 12	30.59
Ensemble of 5 reversed LSTMs, beam size 1	33.00
Ensemble of 2 reversed LSTMs, beam size 12	33.27
Ensemble of 5 reversed LSTMs, beam size 2	34.50
Ensemble of 5 reversed LSTMs, beam size 12	**34.81**

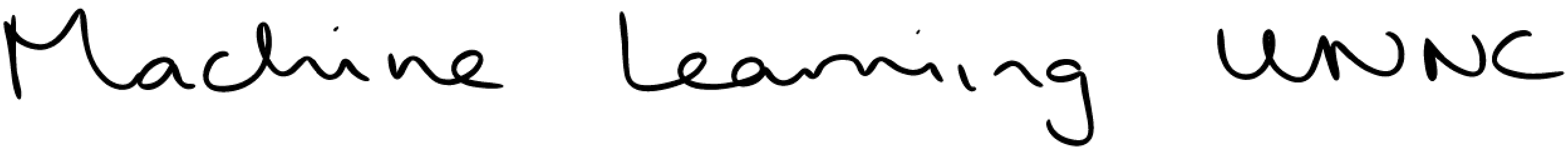
Table 1: The performance of the LSTM on WMT'14 English to French test set (ntst14). Note that an ensemble of 5 LSTMs with a beam of size 2 is cheaper than of a single LSTM with a beam of size 12.

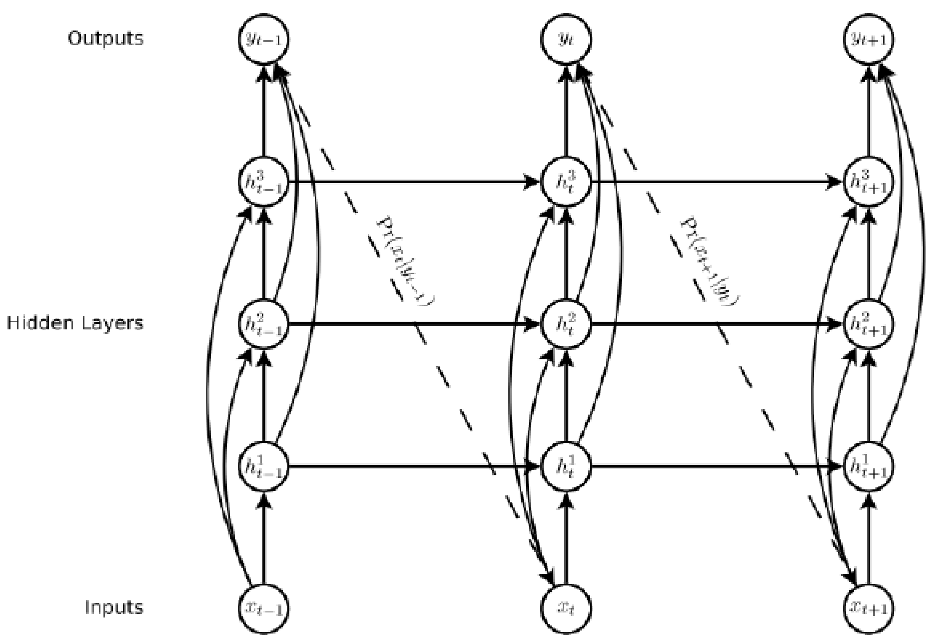
Method	test BLEU score (ntst14)
Baseline System [29]	33.30
Cho et al. [5]	34.54
Best WMT'14 result [9]	**37.0**
Rescoring the baseline 1000-best with a single forward LSTM	35.61
Rescoring the baseline 1000-best with a single reversed LSTM	35.85
Rescoring the baseline 1000-best with an ensemble of 5 reversed LSTMs	**36.5**
Oracle Rescoring of the Baseline 1000-best lists	$\sim$45

Sequence to Sequence Learning by Sutskever et al. 2014

Handwriting Generation from Text

- Input: Machine Learning UNNC





Alex Graves. "**Generating sequences with recurrent neural networks.**" (2013).

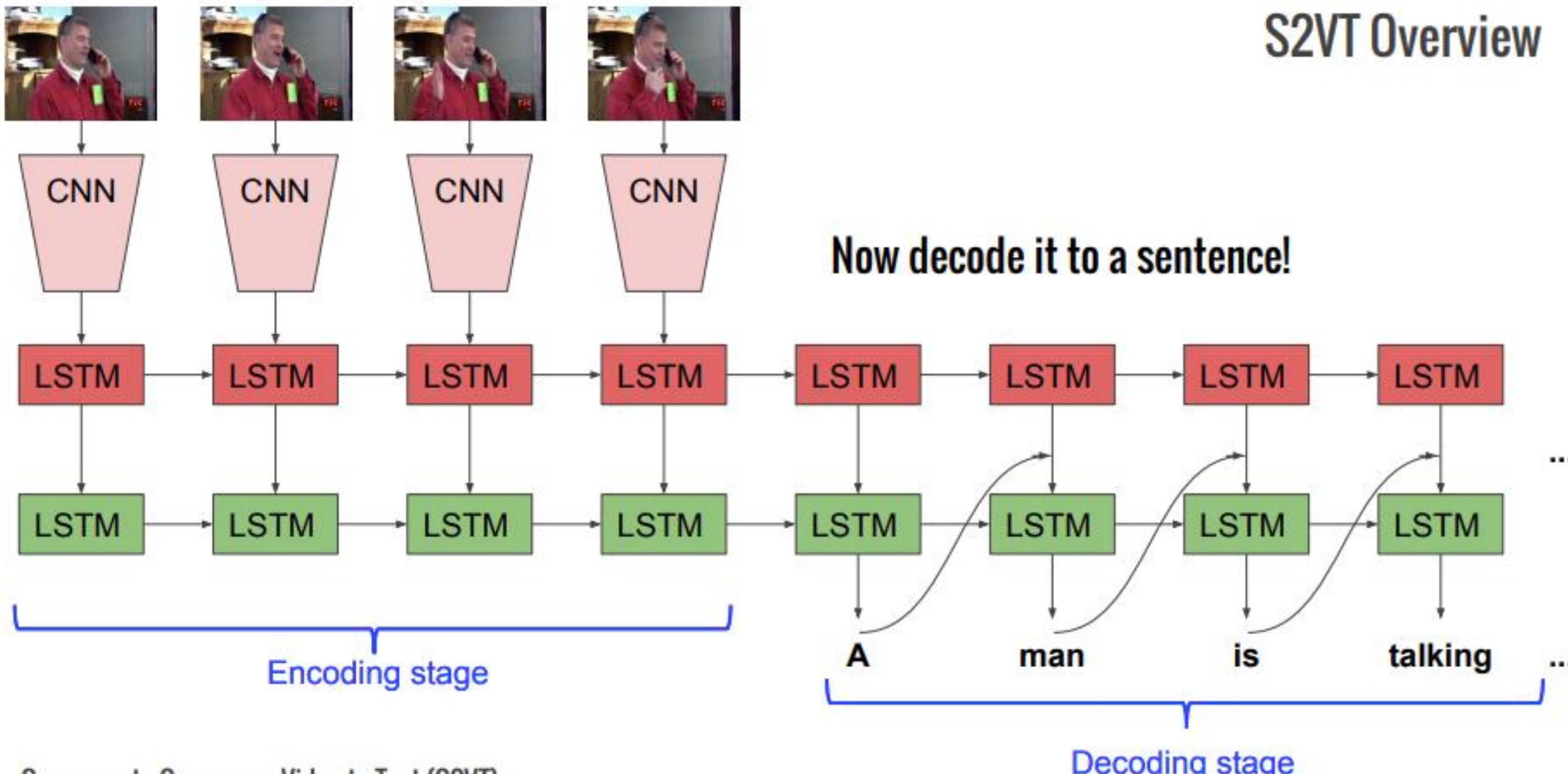
Applications of LSTM

- Sequence to sequence: video to text

Objective

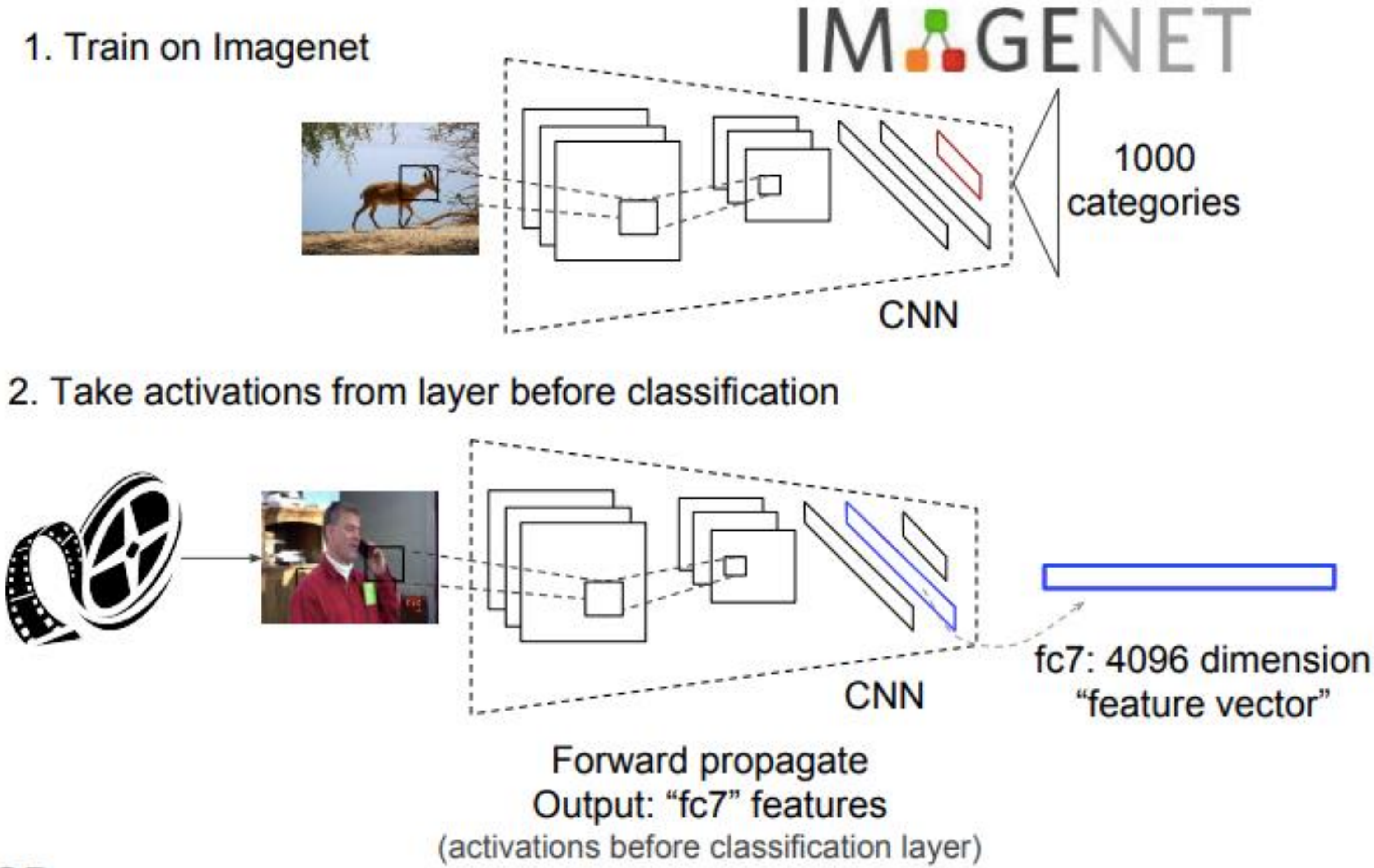
A monkey is pulling a dog's tail and is chased by the dog.

Video to Text



Sequence to Sequence - Video to Text (S2VT)
S. Venugopalan, M. Rohrbach, J. Donahue, R. Mooney, T. Darrell, K. Saenko

Video to Text



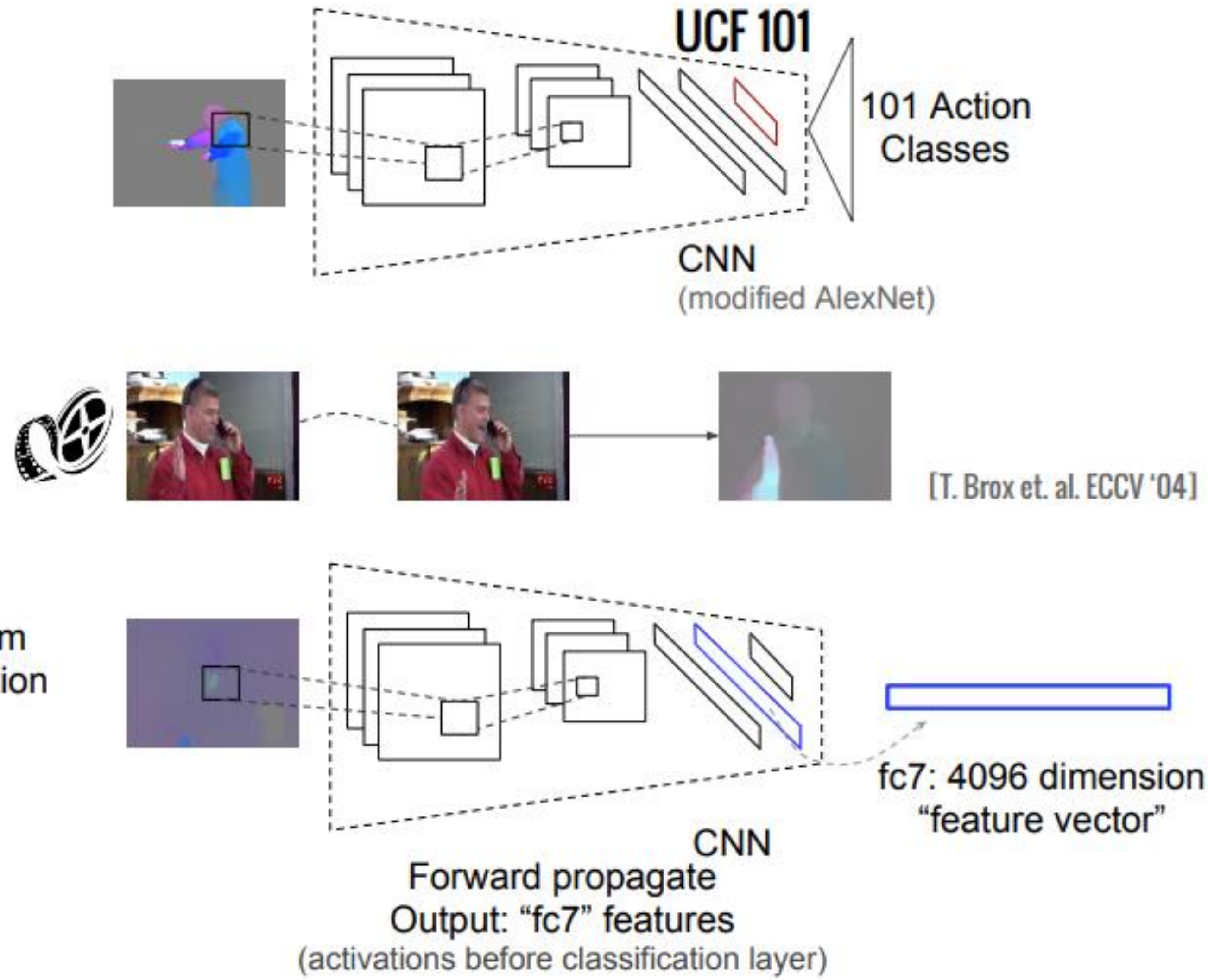
Frames: RGB

Video to Text

1. Train CNN on Activity classes

2. Use optical flow to extract flow images.

3. Take activations from layer before classification



Frames: Flow

Video to Text

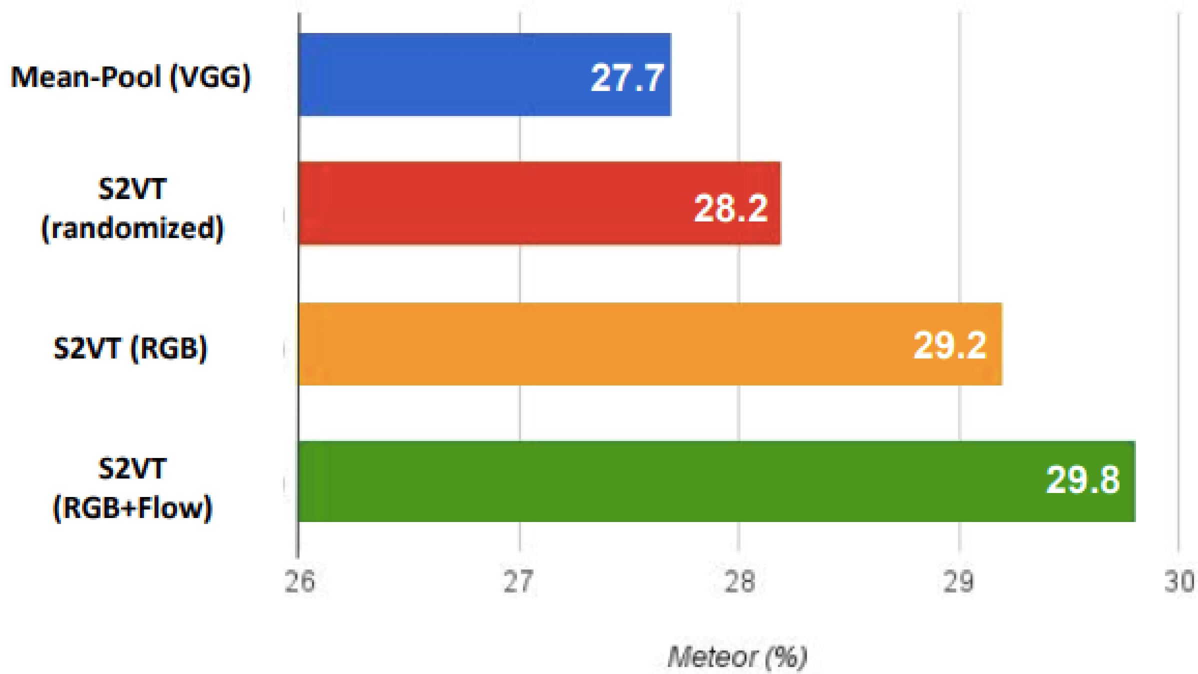
Dataset: Youtube

- ~2000 clips
- Avg. length: 11s per clip
- **~40 sentence per clip**
- ~81,000 sentences

- A man is **walking** on a **rope**.
- A man is **walking** across a **rope**.
- A man is **balancing** on a **rope**.
- A man is **balancing** on a **rope** at the beach.
- A man **walks** on a **tightrope** at the beach.
- A man is **balancing** on a **volleyball net**.
- A man is **walking** on a **rope** held by poles
- A man **balanced** on a **wire**.
- The man is **balancing** on the **wire**.
- A man is **walking** on a **rope**.
- A man is **standing** in the sea shore.

Video to Text

Results (Youtube)



METEOR: MT metric. Considers alignment, para-phrases and similarity.

Video to Text

Video to Text

Evaluation on movie corpus

M-VAD

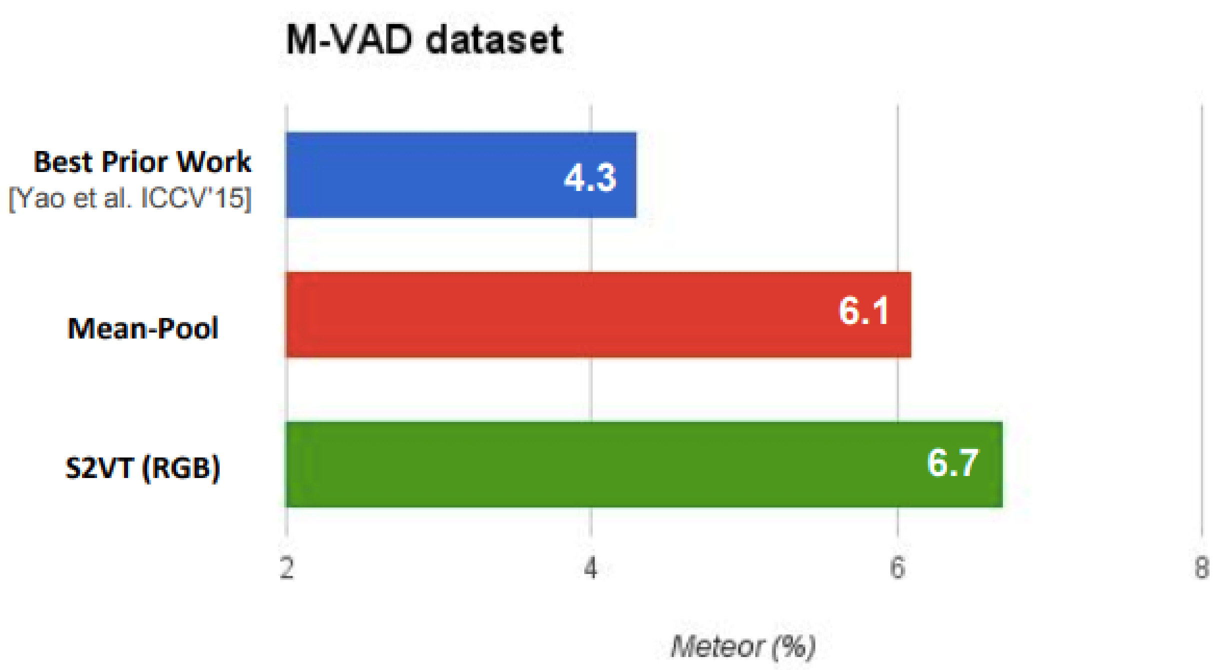
- Univ. of Montreal
- DVS alignment: automated speech extraction
- 92 movies
- 46,009 clips
- Avg. length: 6.2s per clip
- **1-2 sentences per clip**
- 56,634 sentences

The Land Rover pulls away.

Three bodyguards quickly jump into a nearby car and follow her.

Video to Text

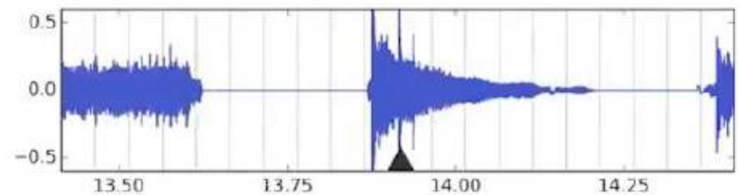
Results (M-VAD Movie Corpus)



Adding Audio to Silent Film

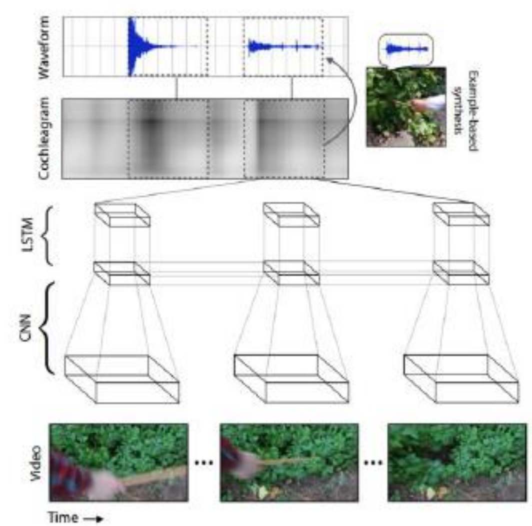
https://www.youtube.com/watch?v=0FW99AQmMc8

Silent video

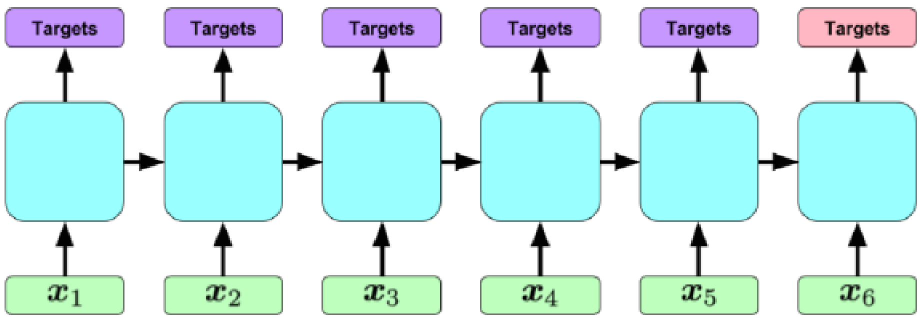


Predicted soundtrack

Owens, Andrew, Phillip Isola, Josh McDermott, Antonio Torralba, Edward H. Adelson, and William T. Freeman. **"Visually Indicated Sounds."** (2015).



Medical Diagnosis



- **Input:** patients electronic health record (EHR) data over multiple visits (meaning, variable length sequences)
- **Output:** 128 diagnoses

Top 6 diagnoses measured by F1 score

Label	*F1*	AUC	Precision	Recall
Diabetes mellitus with ketoacidosis	0.8571	0.9966	1.0000	0.7500
Scoliosis, idiopathic	0.6809	0.8543	0.6957	0.6667
Asthma, unspecified with status asthmaticus	0.5641	0.9232	0.7857	0.4400
Neoplasm, brain, unspecified	0.5430	0.8522	0.4317	0.7315
Delayed milestones	0.4751	0.8178	0.4057	0.5733
Acute Respiratory Distress Syndrome (ARDS)	0.4688	0.9595	0.3409	0.7500

Lipton et al. "Learning to diagnose with LSTM recurrent neural networks." (2015).

Stock Market Prediction

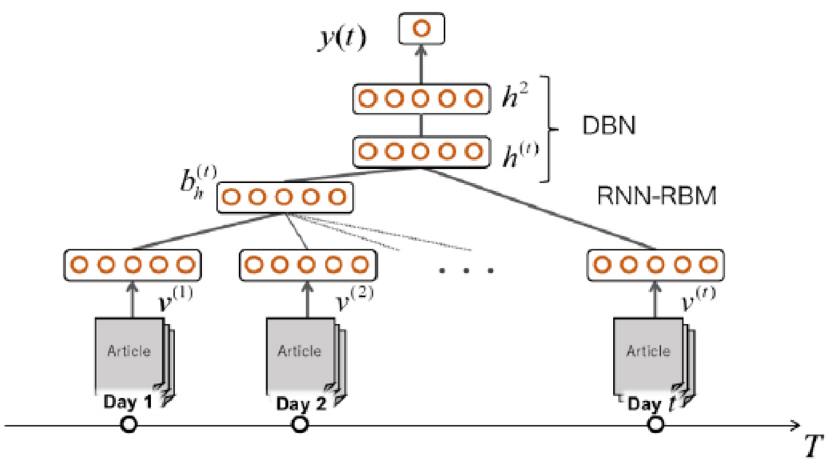


Table 3. Test error rates for stock price prediction

Brands	Baseline	SVM	DBN	RNN-RBM + DBN
Nikkei Average	49.57	48.73	45.50	**43.62**
Hitachi	35.71	37.29	32.00	**32.00**
Toshiba	39.52	41.95	**38.50**	**38.50**
Fujitsu	40.00	40.25	**32.00**	34.00
Sharp	42.00	47.88	**40.00**	**40.00**
Sony	43.00	47.46	41.43	**40.95**
Nissan Motor	40.00	45.34	39.50	**37.00**
Toyota Motor	44.29	53.39	43.81	**42.38**
Canon	43.81	53.39	43.00	**39.11**
Mitsui	46.96	47.88	**41.43**	**41.43**
Mitsubishi	43.81	49.15	43.33	**40.43**
Average	42.61	46.61	40.05	**39.04**

Table 5. Comparison of test error rates after a significant financial crisis

Brands	SVM	RNN-RBM + DBN
Nikkei Average	51.61	**38.70**
Hitachi	61.29	**32.25**
Toshiba	54.83	**38.70**
Fujitsu	45.16	**32.25**
Sharp	58.06	**45.16**
Sony	**41.93**	**41.93**
Nissan Motor	**29.03**	35.48
Toyota Motor	48.38	**45.16**
Canon	**54.83**	**54.83**
Mitsui	41.93	**38.70**
Mitsubishi	29.03	**25.80**
Average	46.92	**39.00**

Yoshihara et al. "Leveraging temporal properties of news events for stock market prediction." 2015.

Audio Classification