Recurrent Neural Networks dan LSTM

Agung Toto Wibowo

Fakultas Informatika - Universitas Telkom

Data Sekuens

Data Sekuens

- Dalam kasus dunia nyata, terdapat beragam data
- Salah satunya adalah data sekuens (deretan atau barisan)
- Contoh:
  - Data time series seperti history curah hujan, data closing harga saham
  - Teks yang merupakan sekuens kalimat, kata, ataupun huruf
  - Suara yang merupakan barisan amplituo frekuensi, spectrum, cepstrum
  - Video yang merupakan barsan frame

What time is it?

Pemrosesan Sekuensial

- Pembelajaran pada data sekuensial
  - Perlu menyimpan informasi satu waktu untuk dapat dipergunakan pada pemrosesan waktu setelahnya.
  - Contoh: Posisi objek pada satu detik tertentu (t), dipengaruhi oleh posisi detik sebelumnya (t-1) dan juga detik sebelumnya lagi (t-2)

- Jaringan yang telah dipelajari sebelumnya (seperti ANN, CNN, ataupun DBN) tidak mampu mengingat history data

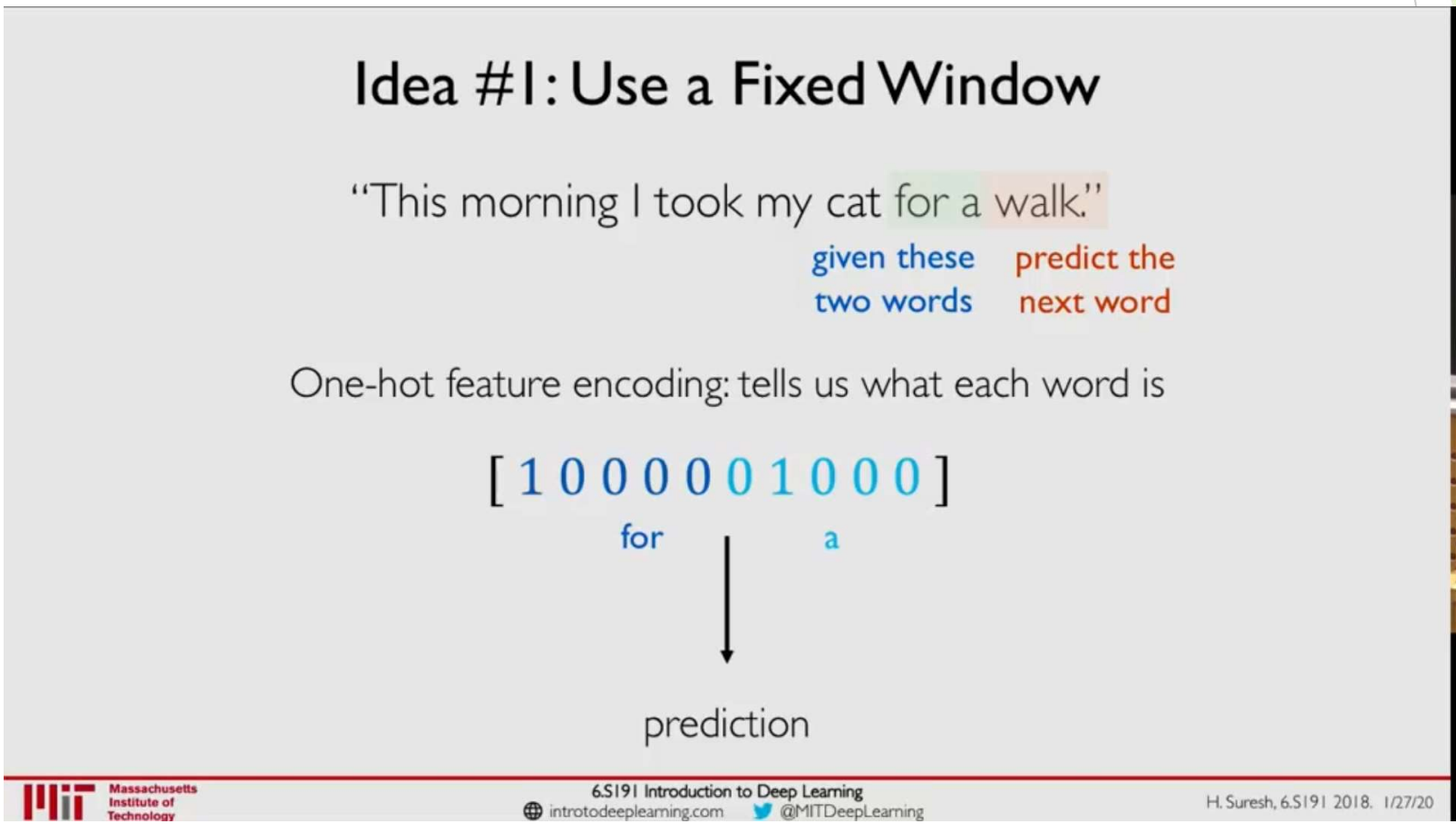
Permasalahan Pemodelan Data Sekuens

A Sequence Modeling Problem: Predict the Next Word

"This morning I took my cat for a walk."

given these words

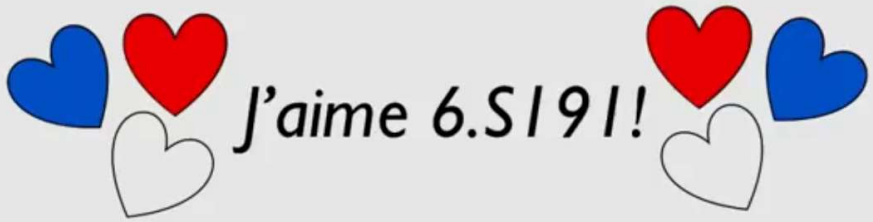
Permasalahan Pemodelan Data Sekuens



Permasalahan Pemodelan Data Sekuens

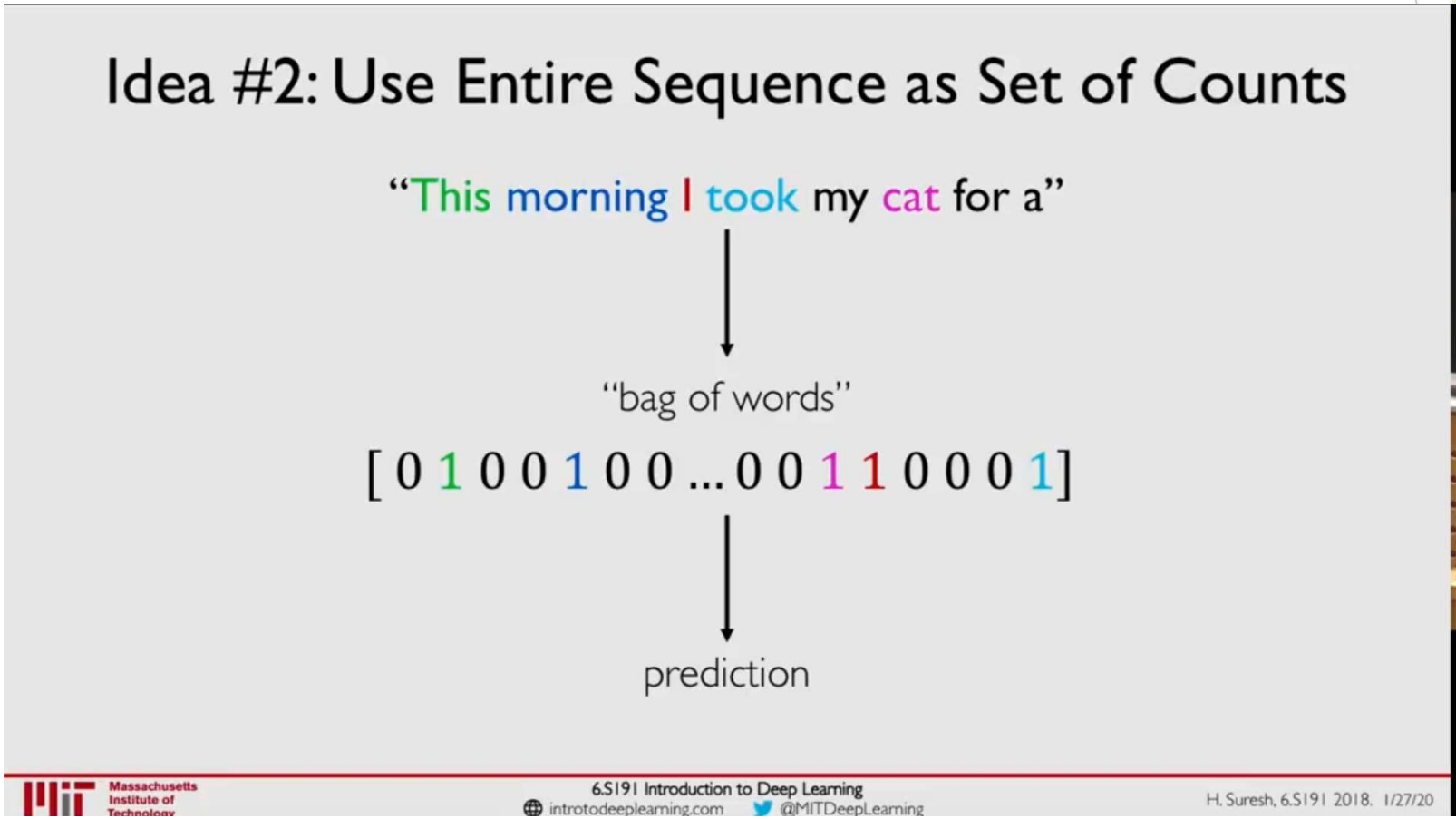
Problem #1: Can't Model Long-Term Dependencies

"**France** is where I grew up, but I now live in Boston. I speak fluent ___."



We need information from **the distant past** to accurately predict the correct word.

Permasalahan Pemodelan Data Sekuens



Permasalahan Pemodelan Data Sekuens

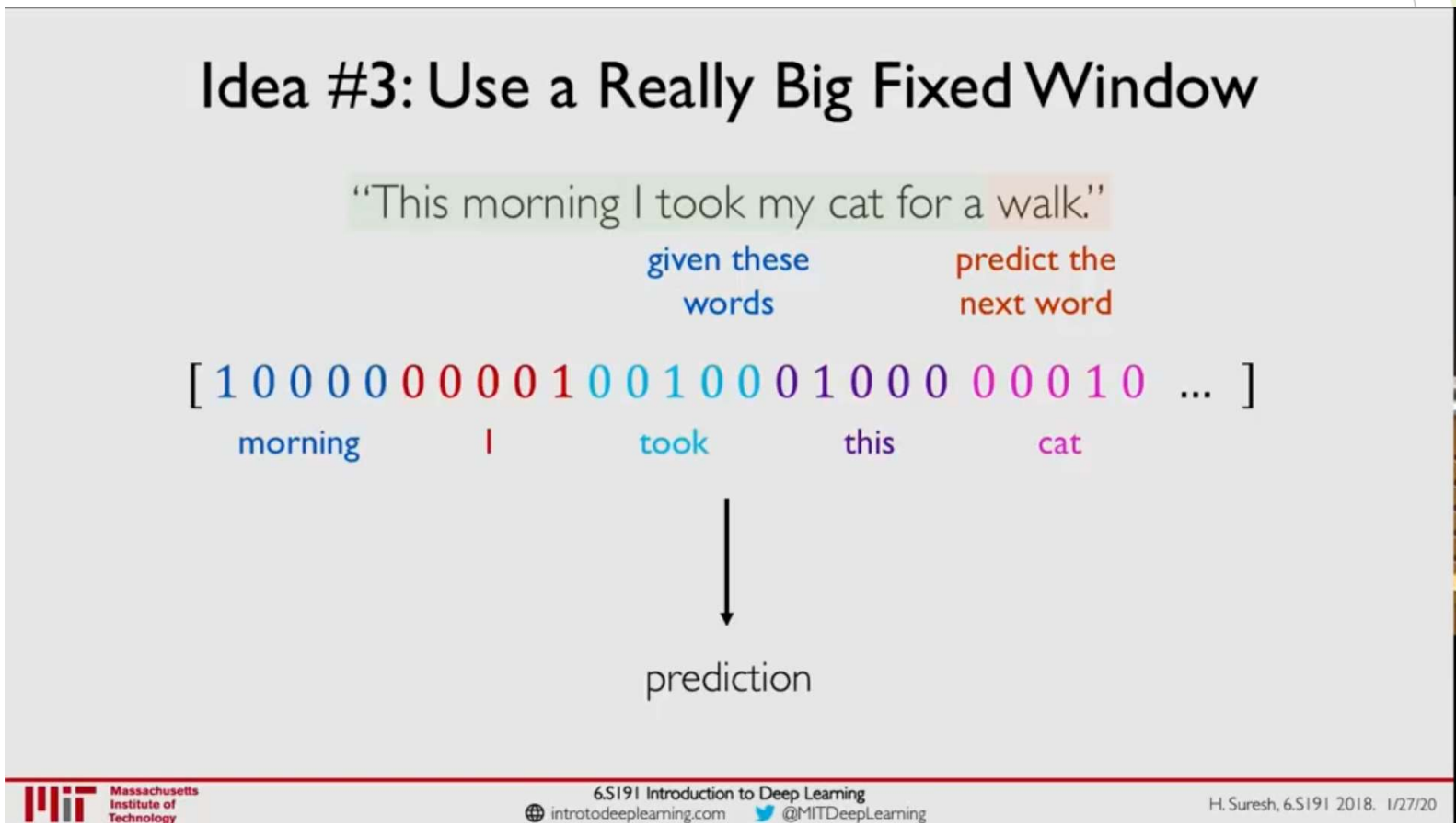
Problem #2: Counts Don't Preserve Order

The food was good, not bad at all.

vs.

The food was bad, not good at all.

Permasalahan Pemodelan Data Sekuens



Permasalahan Pemodelan Data Sekuens

Problem #3: No Parameter Sharing

[1 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 0 0 0 1 0 ...]

this morning took the cat

Each of these inputs has a **separate parameter**:

[0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 ...]

this morning

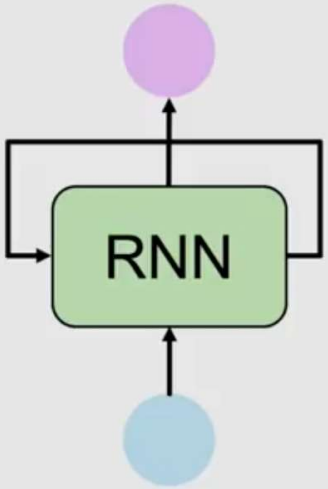
Things we learn about the sequence **won't transfer** if they appear **elsewhere** in the sequence.

Permasalahan Pemodelan Data Sekuens

Sequence Modeling: Design Criteria

To model sequences, we need to:

1. Handle **variable-length** sequences
2. Track **long-term** dependencies
3. Maintain information about **order**
4. **Share parameters** across the sequence

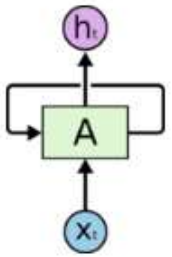


Today: **Recurrent Neural Networks (RNNs)** as an approach to sequence modeling problems

Recurent Neural Network (RNN)

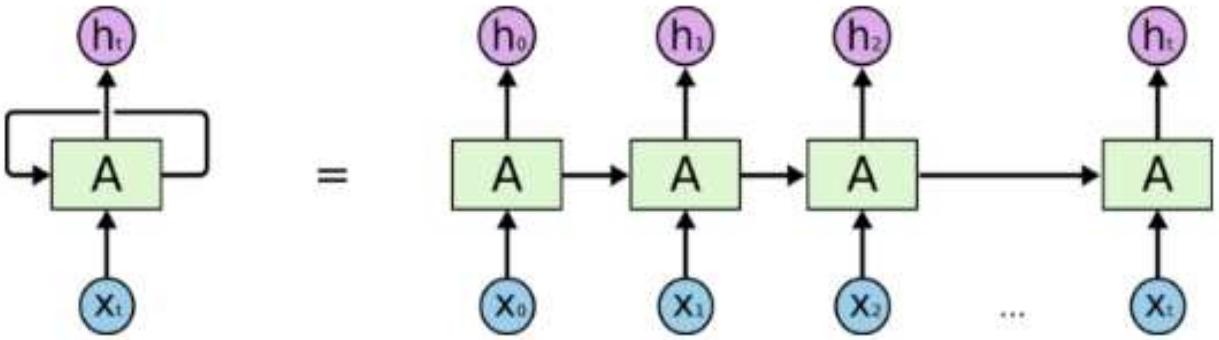
Recurrent Neural Network (RNN)

- Jaringan artificial neural networks yang memiliki loop dari satu neuron menuju ke neuron tersebut
- Loop ini berfungsi menyimpan “riwayat informasi” sebelumnya
- Riwayat informasi dan juga input pada suatu waktu akan dipergunakan untuk memproses (contoh: prediksi) data berikutnya
- Contoh:
  - Pada data percakapan, informasi diproses tidah hanya pada segmen tertentu, namun juga ada segmen sebelumnya
  - Menebak huruf yang muncul pada karakter “_” sekuens “HEL_”

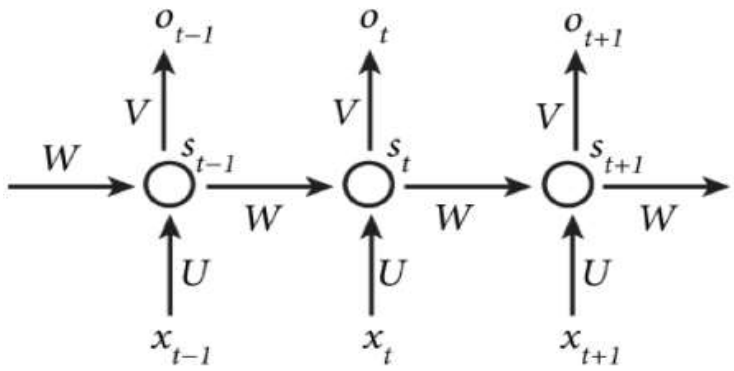


Recurrent Neural Network (RNN)

- Misal kita ingin memproses data *x* pada segemen waktu tertentu *t*
- Data x_t akan di proses oleh neuron A menjadi nilai h_t
- Hasil pemrosesan ini (h_t) akan dipergunakan untuk pemrosesan data berikutnya h_{t+1}
- Proses dilakukan beberapa kali, semakin banyak loop semakin banyak informasi yang perlu disimpan
- RNN dapat dianggap sebagai multiple copy dari sebuah network yang sama, yang meneruskan informasi ke network suksesornya.



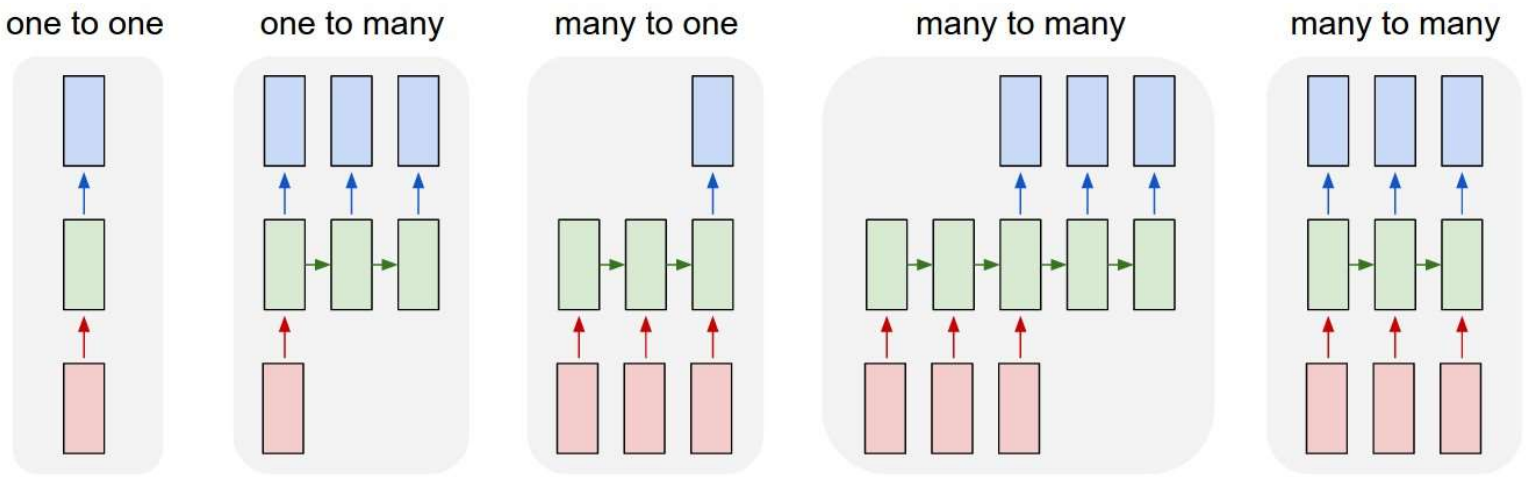
Arsitektur RNN



Dengan formula matematis, RNN dapat dijelaskan :

- *Variabel* x_{t-}, x_t, x_{t+1} merupakan data input pada time step *t-1*, *t*, *dan t+1*
- *Variabel* s_t adalah hidden state pada time step ke *t*. Hidden state ini dapat dianggap sebagai memori yang menyimpan hasil pemrosesan neuron
- Nilai s_t dihitung berdasar nilai hidden state sebelumnya (*W*) dan nilai dari sinyal input current state x_t. Atau dengan kata lain $s_t = f(Ux_t + Ws_{t-1})$
- Fungsi f adalah fungsi aktivasi dapat berupa hiperbolik (tanh) atau rectified linear unit (RelU)
- Pada kondisi awal, s_{t-1} secara default bernilai *0*.
- Variabel o_t adalah output pada state ke-*t*.
- Nilai dari o_t berupa output dari fungsi aktivasi dari hasil perhitungan pada s_t, bisa berupa softmax atau sigmoid biasa.

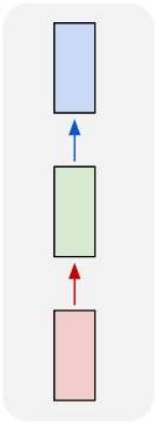
Arsitektur RNN



- Each rectangle is a vector and arrows represent functions (e.g. matrix multiply).
- Input vectors are in red, output vectors are in blue and green vectors hold the RNN's state (more on this soon).

Arsitektur RNN

one to one

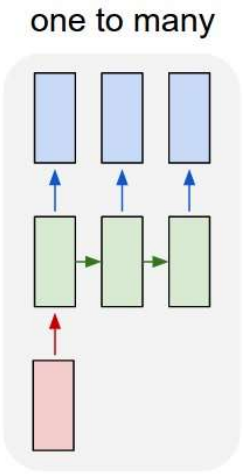


- Vanilla mode of processing without RNN,
- From fixed-sized input to fixed-sized output (e.g. image classification).

Running

https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1#:~:text=Sequence%20models%20are%20the%20machine,algorithm%20used%20in%20sequence%20models.

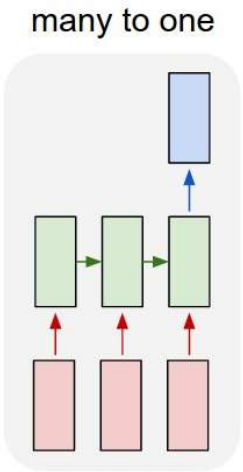
Arsitektur RNN



- Sequence output
- e.g. image captioning takes an image and outputs a sentence of words.

https://github.com/danieljl/keras-image-captioning

Arsitektur RNN

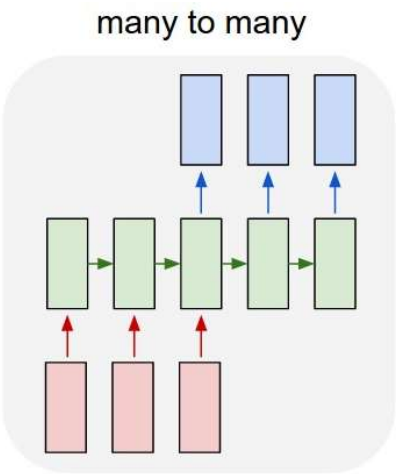


- Sequence input with single output
- e.g. sentiment analysis where a given sentence is classified as expressing positive or negative sentiment.

Review (X)	Rating (Y)
"This movie is fantastic! I really like it because it is so good!"	★★★★☆
"Not to my taste, will skip and watch another movie"	★★☆☆☆
"This movie really sucks! Can I get my money back please?"	★☆☆☆☆

https://towardsdatascience.com/sequence-models-and-recurrent-neural-networks-rnns-62cadeb4f1e1#:~:text=Sequence%20models%20are%20the%20machine,algorithm%20used%20in%20sequence%20models.

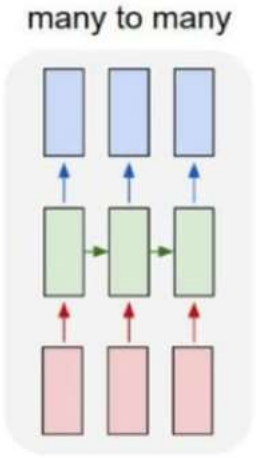
Arsitektur RNN



- Sequence input and sequence output
- e.g. Machine Translation:
  - an RNN reads a sentence in English and
  - then outputs a sentence in French

Arsitektur RNN

- Synced sequence input and output
- e.g. video classification where we wish to label each frame of the video



e.g., video classification on frame level

Formulasi RNN

Back Propagation Through Time

- RNN memiliki arsitektur yang seolah-oleh berlapis
- *Neural network* yang sederhana namun digunakan secara berulang.
- Satu *layer neural network* digunakan beberapa kali pada *time step* yang berurutan.
- Pembelajaran pada RNN juga akan menelusuri setiap lapis *neural network* pada satu rangkaian waktu.
- RNN disebut sebagai algoritma pembelajaran *Back Propagation Through Time* (BPTT)

Formulasi RNN

$$\boldsymbol{h_t = f_w(h_{t-1}, x_t)} \qquad (1)$$

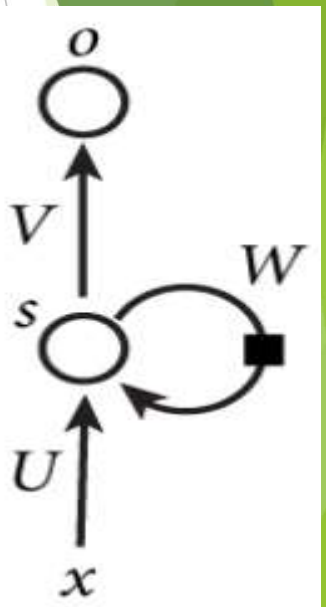
- dengan $\boldsymbol{f_w}$ adalah fungsi tanh atau fungsi RelU
- h_{t-1} didapatkan dari perhitungan *h* pada *time step* sebelumnya
- Bobot pada hidden *layer h* terdiri dari dua:
  - bobot untuk vektor ht-1 dan
  - bobot untuk vektor input xt

$$\boldsymbol{h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)} \qquad (2)$$

- Nilai dari output *layer* pada *time step* ke-*t*

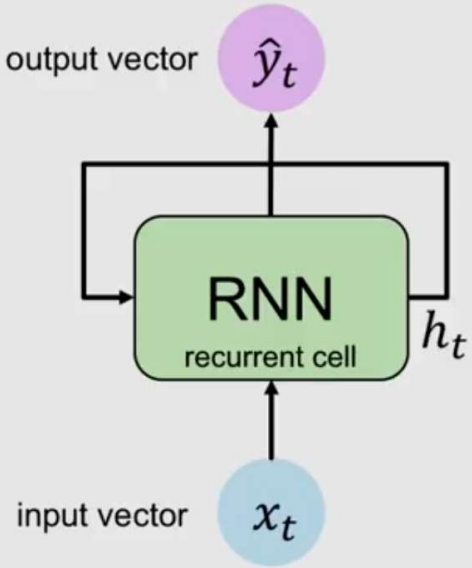
$$\boldsymbol{o_t = f_o(W_{ho}h_t + b_o)} \qquad (3)$$

- dengan $\boldsymbol{f_o}$ adalah fungsi aktivasi dari output layer biasanya sigmoid atau softmax



Formulasi RNN (perspektif lain)

Recurrent Neural Network (RNN)



Apply a **recurrence relation** at every time step to process a sequence:

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

- h_t: cell state
- f_W: function parameterized by W
- h_{t-1}: old state
- x_t: input vector at time step t

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

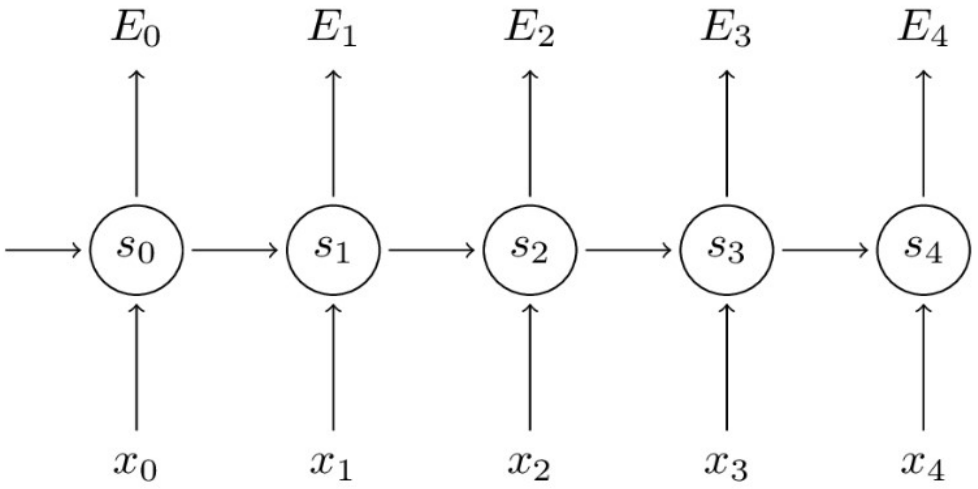
Formulasi RNN

- *Loss function* menggunakan formula *cross entropy:*

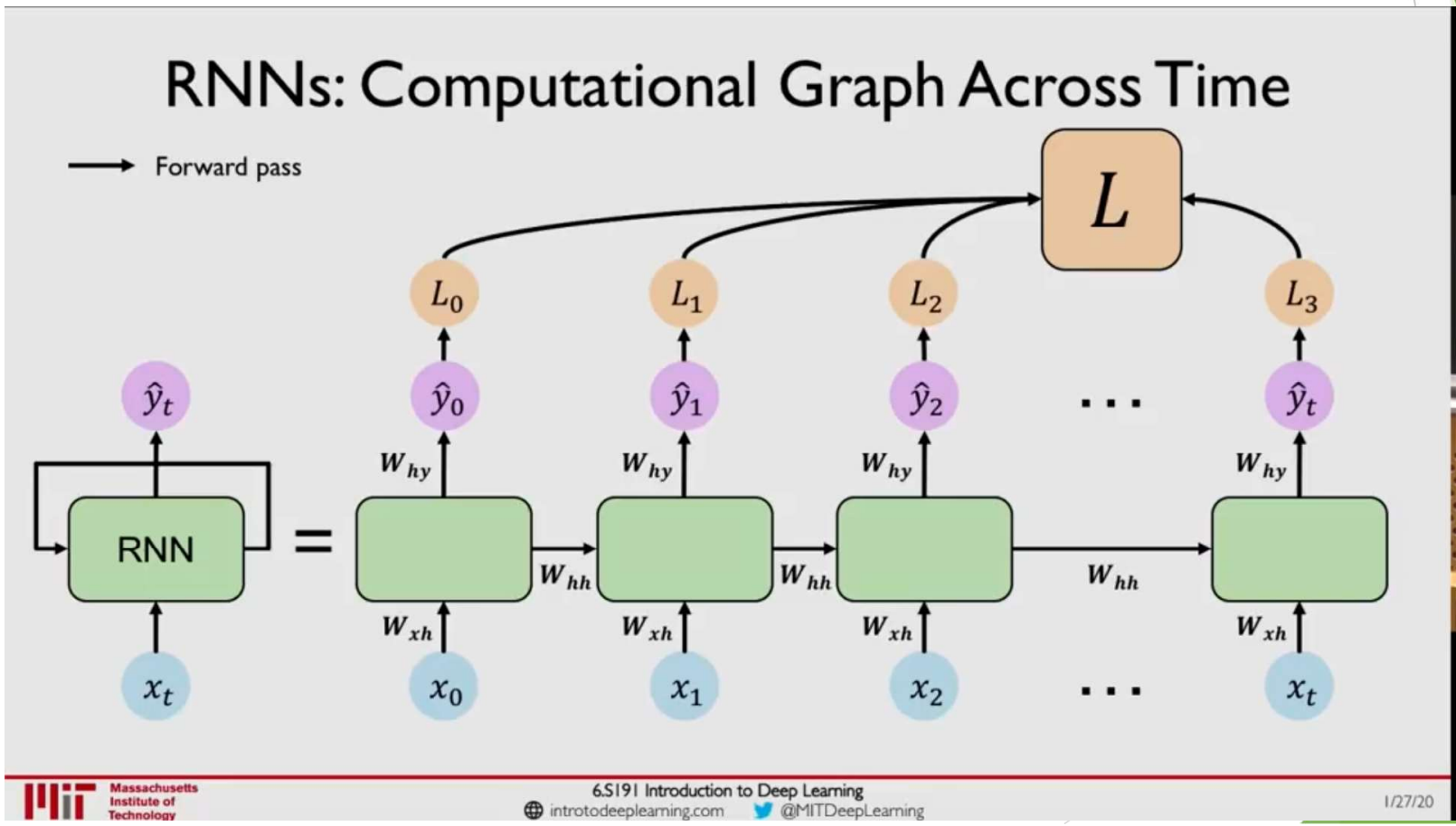
$$\boldsymbol{E_t(o_t, \hat{o}_t) = -o_t log(\hat{o}_t)} \qquad \textit{(4)}$$

- Nilai $\boldsymbol{E_t(o_t, \hat{o}_t)}$ di atas merupakan lost function pada satu *time step* ke-t. Untuk mengetahui total loss, dapat dipergunakan formula:

$$\boldsymbol{E = \sum_t E_t(o_t, \hat{o}_t) = \sum_t -o_t log(\hat{o}_t)} \qquad \textit{(5)}$$



Formulasi RNN (perspektif lain)



Formulasi RNN

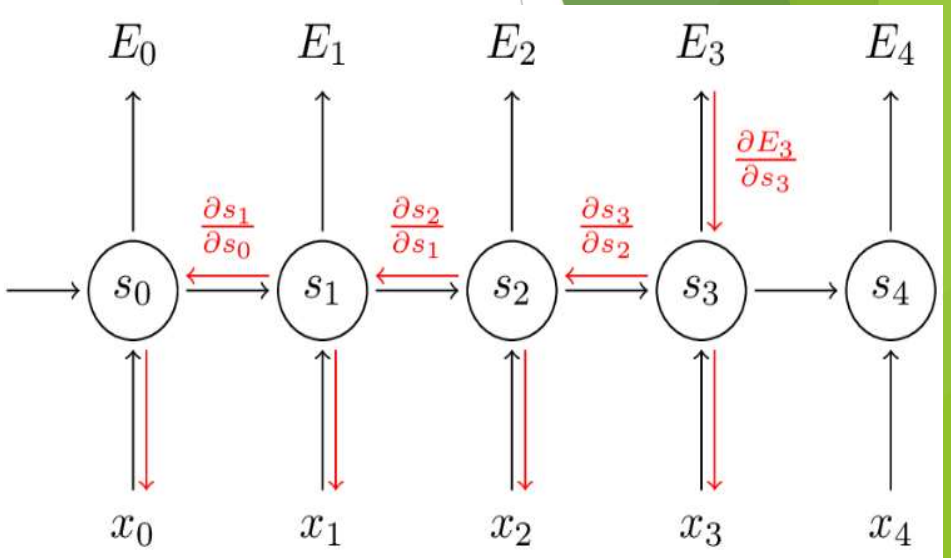
- Pencari nilai dari *U, V* dan *W* menggunakan *Gradient Descent* dengan meminimumkan *loss function*
- BPPT menghitung total dari gradient pada setiap time step

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \qquad (6)$$

- Untuk t = 3

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V} = (\hat{y}_3 - y_3) \otimes s_3 \qquad (7)$$

- Dengan $z_3 = V.s_3$ dan $\otimes$ adalah hasil perkalian dari dua vektor. Perhatikan bahwa hasil dari perhitungan tersebut hanya menggunakan informasi dari *time step t*=3



Formulasi RNN

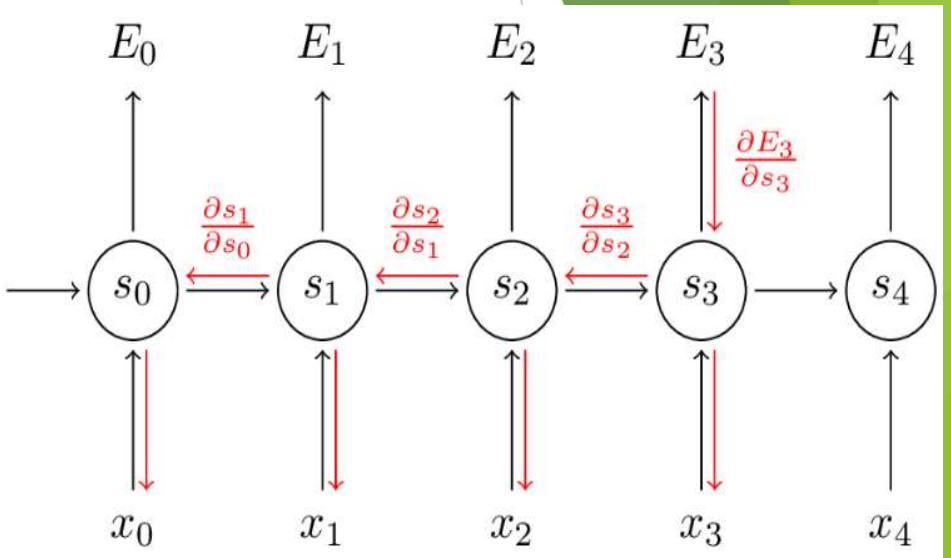
- Untuk *t* = 3, perhitungan nilai gradient terhadap W

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W} \qquad (8)$$

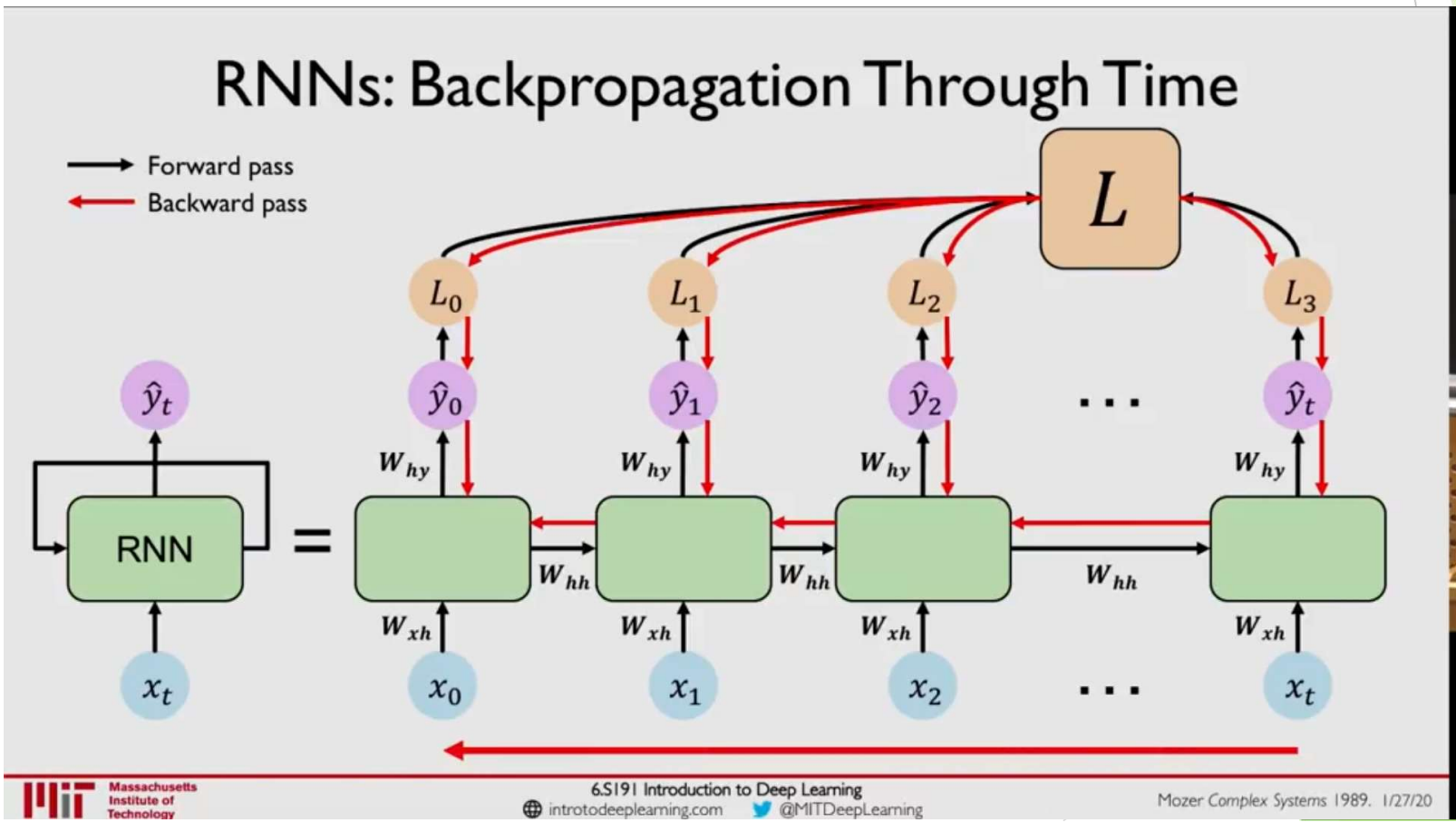
- Perhatikan bahwa $s_3 = \tanh(U_{x_t} + W_{s_2})$ bergantung dari nilai W dan s_2 yang merupakan nilai yang didapatkan pada *time step* sebelumnya, dan nilai s_2 bergantung dari nilai W dan s_1 yang didapatkan pada *time step* sebelumnya lagi
- Maka dengan aturan rantai, akan kita dapatkan perhitungan:

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_k}\frac{\partial s_k}{\partial W} \qquad (9)$$

- Semakin banyak loop dari arsitektur RNN, akan semakin kompleks proses perhitungan gradient

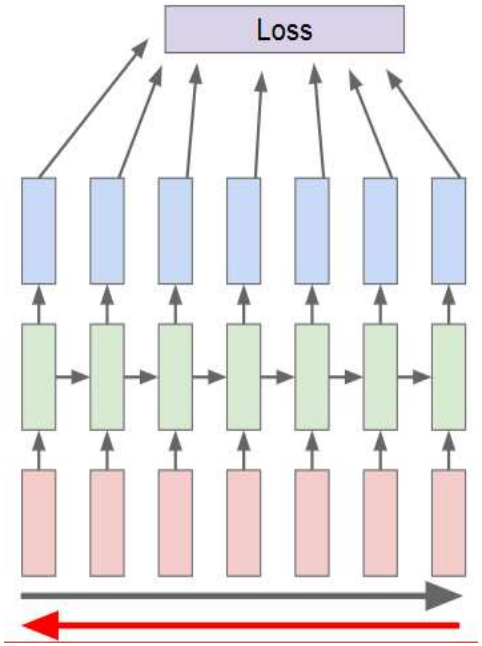


Formulasi RNN (perspektif lain)



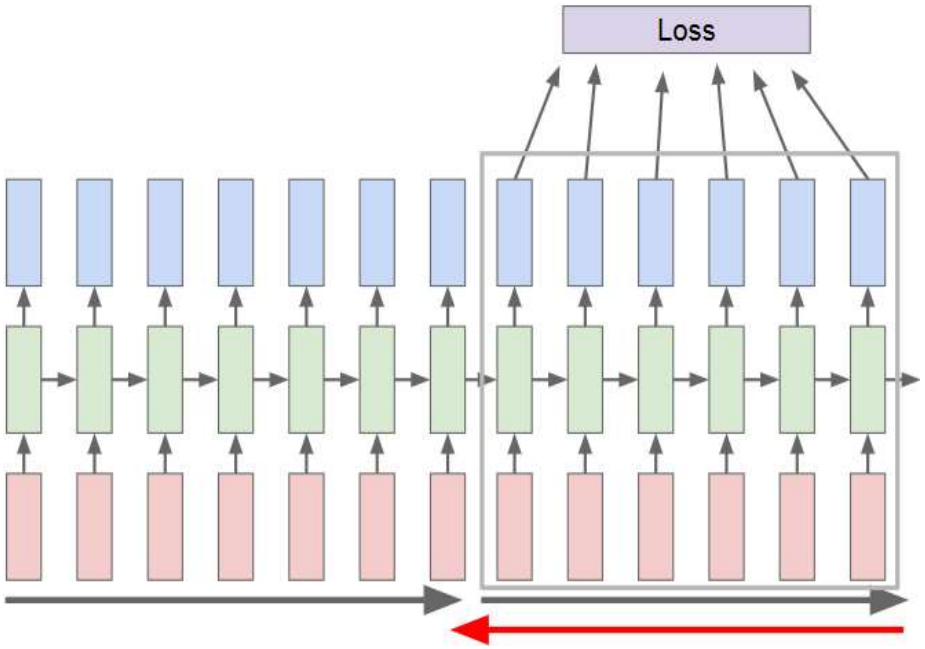
Truncated BPTT

- Perhitungan gradient sangat kompleks jika jumlah loop pada RNN sangat banyak.
- Agar lebih cepat, digunakan skema Truncated BPTT: membagi forward pass dan backward pass ke dalam subsecuences dengan jumlah *time step* yang seragam



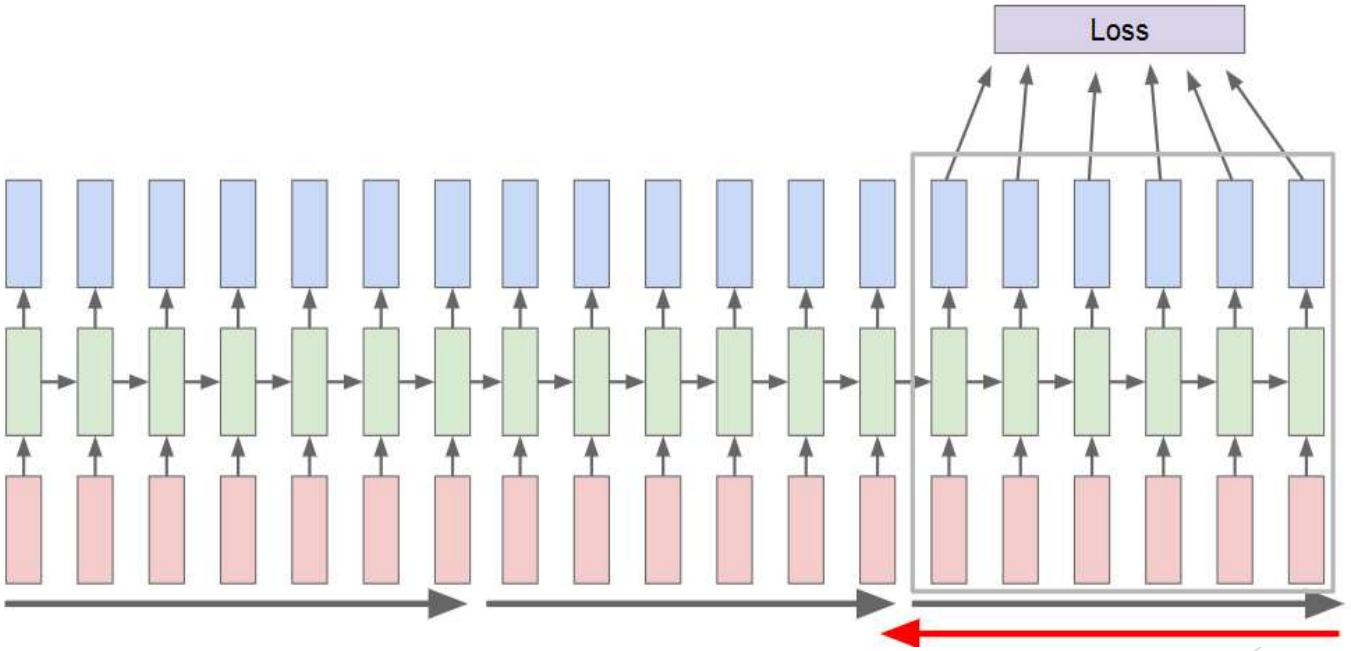
Truncated BPTT

- Perhitungan gradient sangat kompleks jika jumlah loop pada RNN sangat banyak.
- Agar lebih cepat, digunakan skema Truncated BPTT: membagi forward pass dan backward pass ke dalam subsecuences dengan jumlah *time step* yang seragam



Truncated BPTT

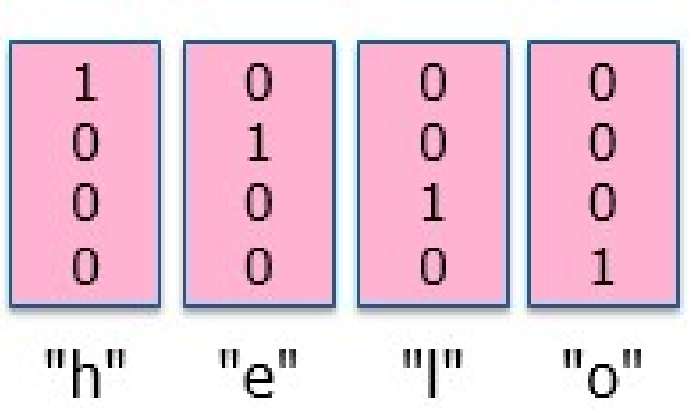
- Perhitungan gradient sangat kompleks jika jumlah loop pada RNN sangat banyak.
- Agar lebih cepat, digunakan skema Truncated BPTT: membagi forward pass dan backward pass ke dalam subsecuences dengan jumlah *time step* yang seragam



RNN untuk Data Sekuens

RNN untuk Data Sekuens - Training

- Contoh pada Character Level language model
- RNN untuk mengenali sekuens *h-e-l-l-o*
- Saat model sudah dilatih dan diberikan pola *h-e-l*, RNN bisa mengembalikan sekuens sisanya : karakter *l* dan *o*
- Representasi *one-hot-encoding* setiap karakter



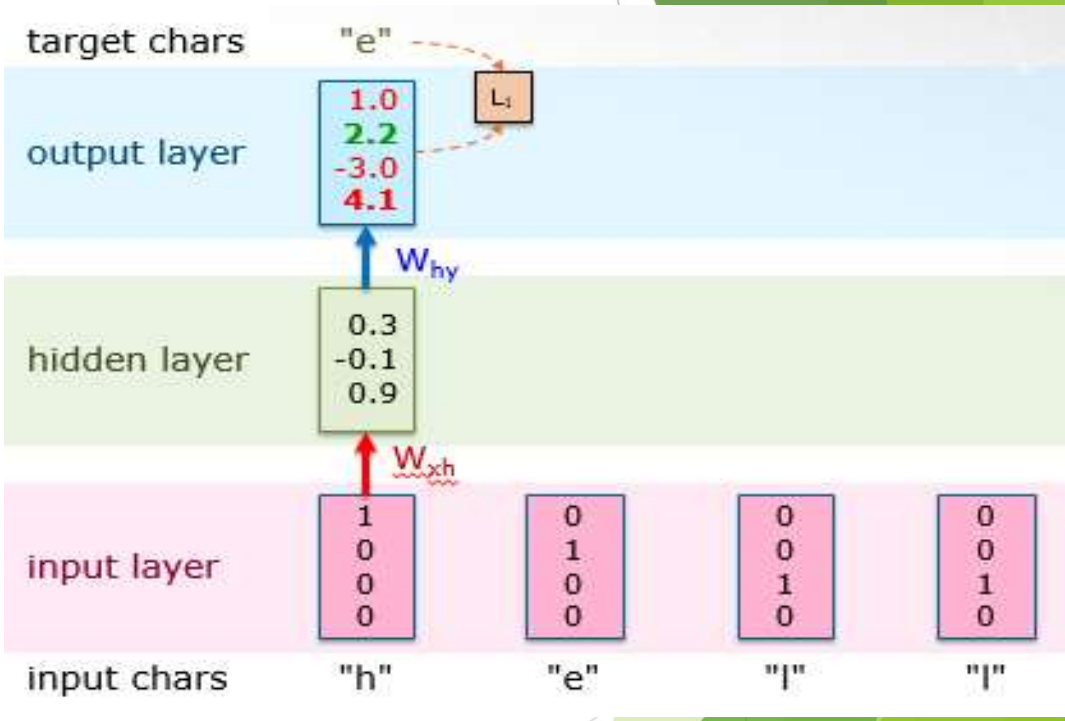
RNN untuk Data Sekuens - Training

- Parameter yang kita butuhkan pada hidden *layer* adalah W_{xh}, W_{hh}, dan b_h dan pada output *layer* adalah W_{hy} dan *by*
- Fungsi pada hidden *layer* dan output *layer* masing-masing adalah:

$$\boldsymbol{h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)} \qquad (2)$$

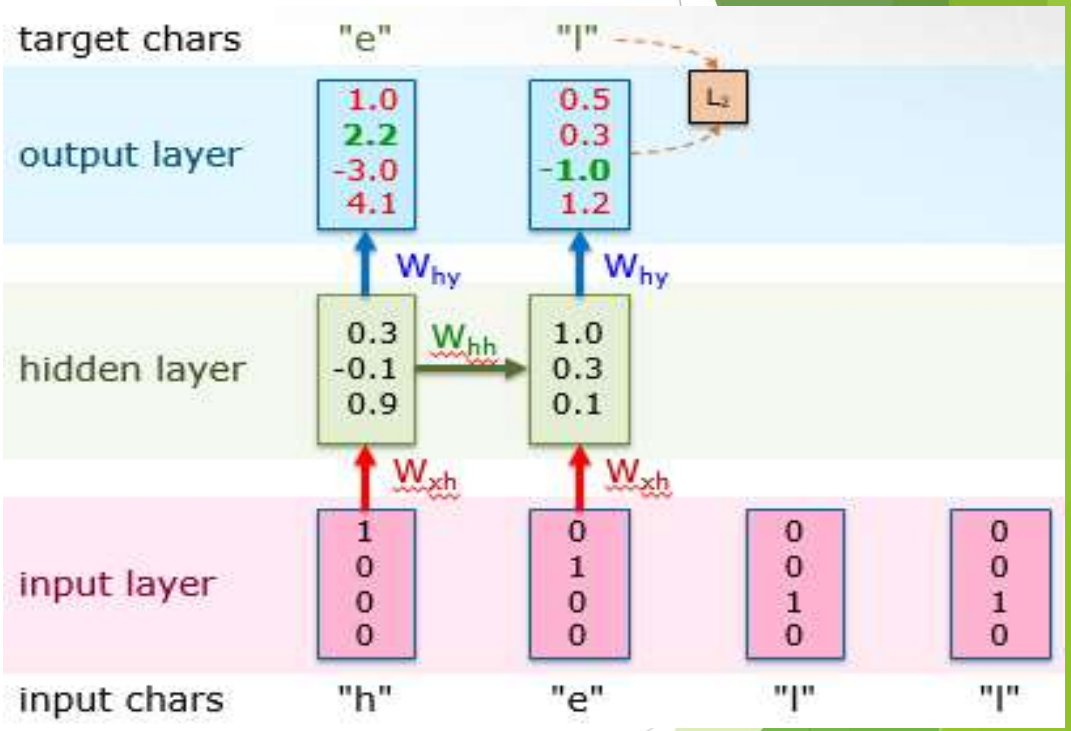
$$\boldsymbol{y_t = W_{hy}h_t} \qquad (2)$$

- Iterasi 1 (t=1), input (x_t) adalah "h" yang dikodekan dengan $[1, 0, 0, 0]^T$, Dikalikan dengan *Wxh* yang didapatkan vektor $[0.3, -0.1, 0.9]^T$
- Diforward melalui fungsi aktivasi tanh ke output *layer* dengan mengalikan dengan bobot W_{hy} dan didapatkan hasil $[1.0, 2.2, -3.0, 4.1]^T$
- Dengan softmax, diprediksi karakter berikutnya adalah "o". Kemudian loss function t=1 dihitung



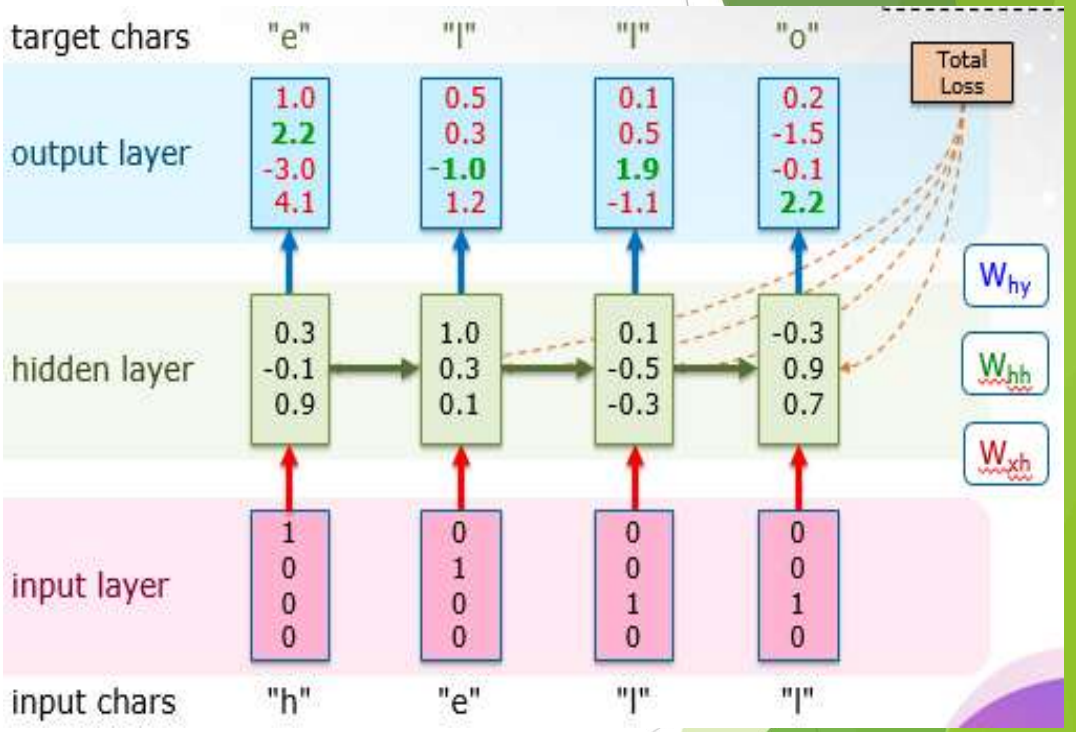
RNN untuk Data Sekuens - Training

- Perhitungan *t* = 2, pada hidden *layer* melibatkan nilai *h* dari t = 1.
- Nilai h_1 dikalikan dengan W_{hh} dan dijumlahkan dengan hasil perkalian W_{xh} dan x_2.
- Proses berikutnya sama dengan t = 1 hingga didapatkan *loss* function untuk t = 2.
- Proses tersebut dilakukan hingga timestep berakhir untuk satu sekuens



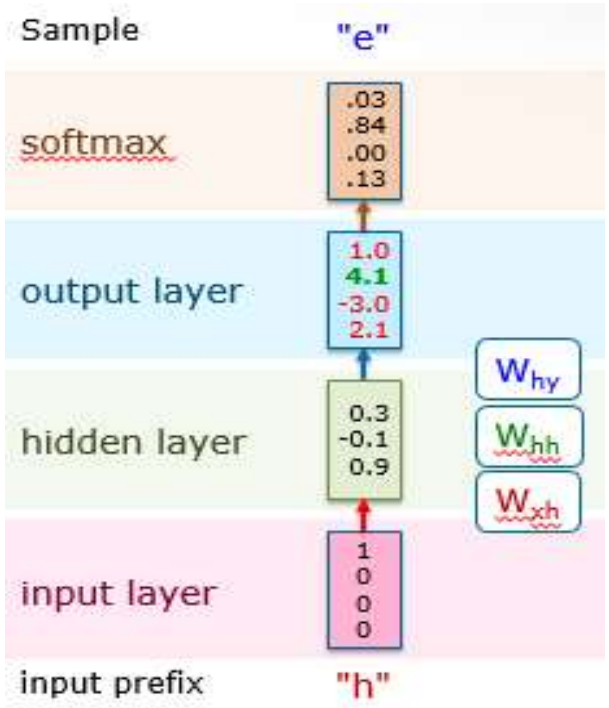
RNN untuk Data Sekuens - Training

- Total *loss* dihitung dengan menjumlahkan keseluruhan nilai *loss* function untuk seluruh timestep.
- Lalu proses berikutnya adalah menghitung gradient terhadap U, V dan W menggunakan aturan rantai.
- Lalu melakukan perubahan bobot pada U, V dan W



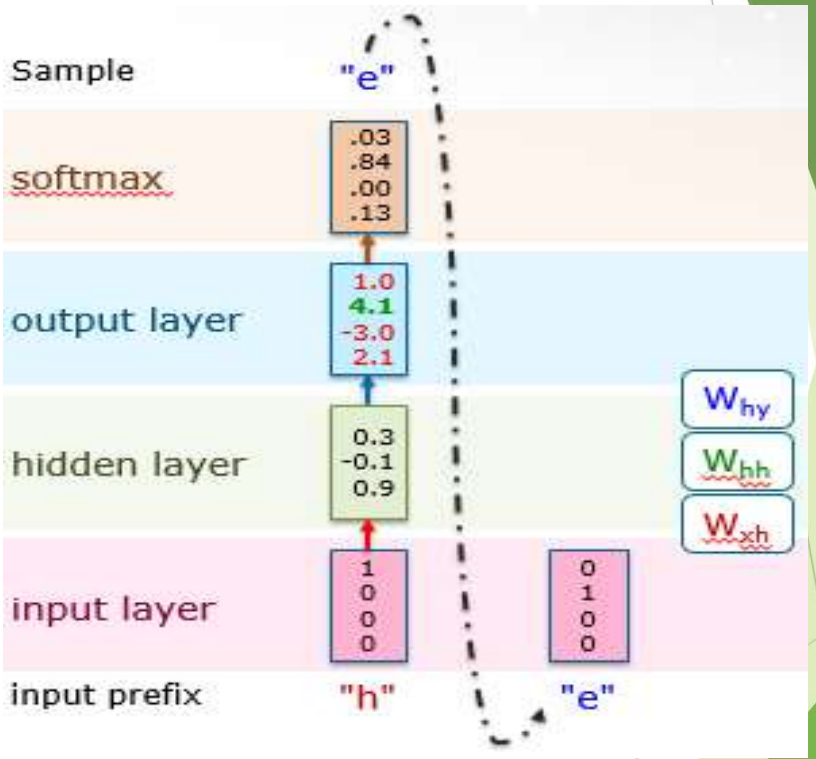
RNN untuk Data Sekuens - Testing

- Input dimulai dengan karakter pertama yaitu "*h*" dikodekan $[1,0,0,0]^T$.
- Forward pass dari input vektor, menuju hidden *layer* sampai kepada output *layer* menghasilkan nilai vektor *y*.
- Dengan fungsi softmax, didapatkan predicted karakter = *e*.



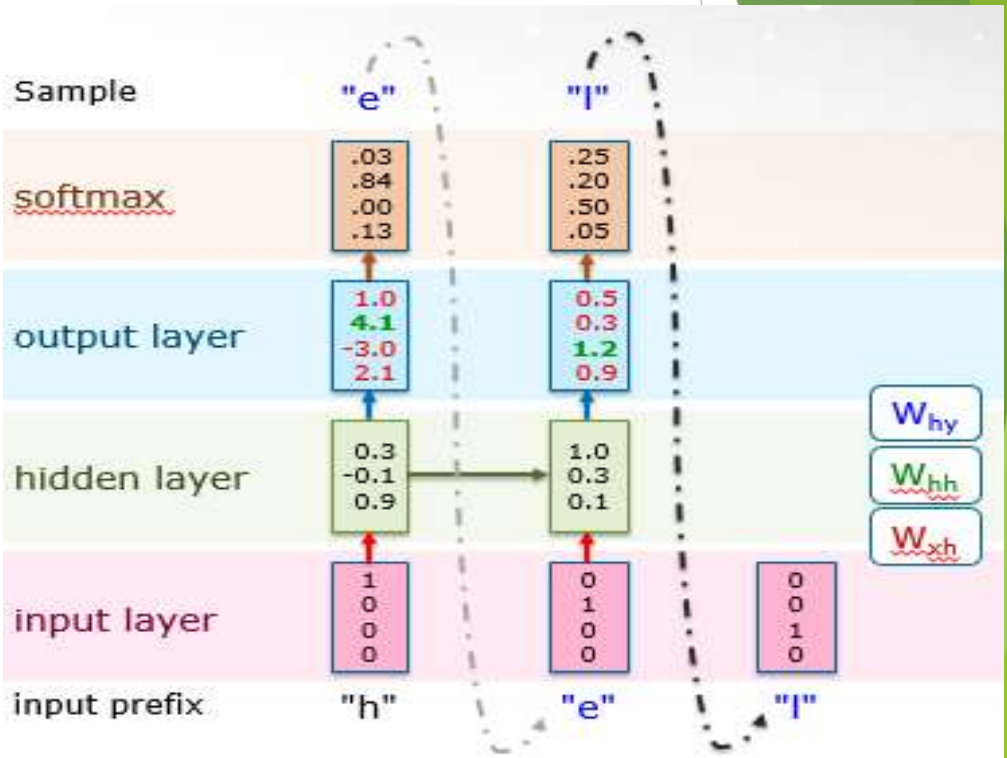
RNN untuk Data Sekuens - Testing

- Karakter *e* ini digunakan sebagai input untuk timestep berikutnya.
- Untuk *t* = 2, proses forward pass pada hidden *layer* menggunakan nilai hidden *layer t* = 1 dikalikan dengan bobot W_{hh} dan juga nilai input dikali bobot W_{xh}.
- Proses selanjutnya sama dengan timestep pertama.
- Proses tersebut dilakukan hingga seluruh input sequences diproses.



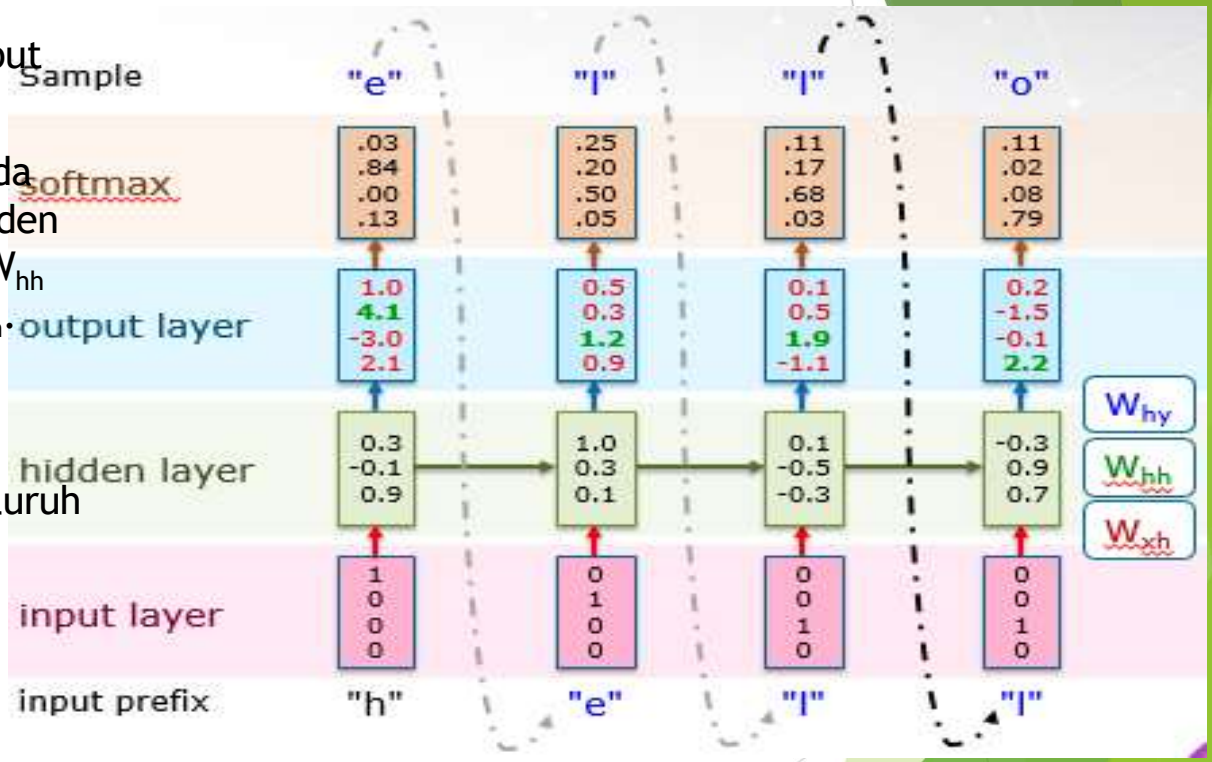
RNN untuk Data Sekuens - Testing

- Karakter *e* ini digunakan sebagai input untuk timestep berikutnya.
- Untuk *t* = 2, proses forward pass pada hidden *layer* menggunakan nilai hidden *layer t* = 1 dikalikan dengan bobot W_{hh} dan juga nilai input dikali bobot W_{xh}.
- Proses selanjutnya sama dengan timestep pertama.
- Proses tersebut dilakukan hingga seluruh input sequences diproses.

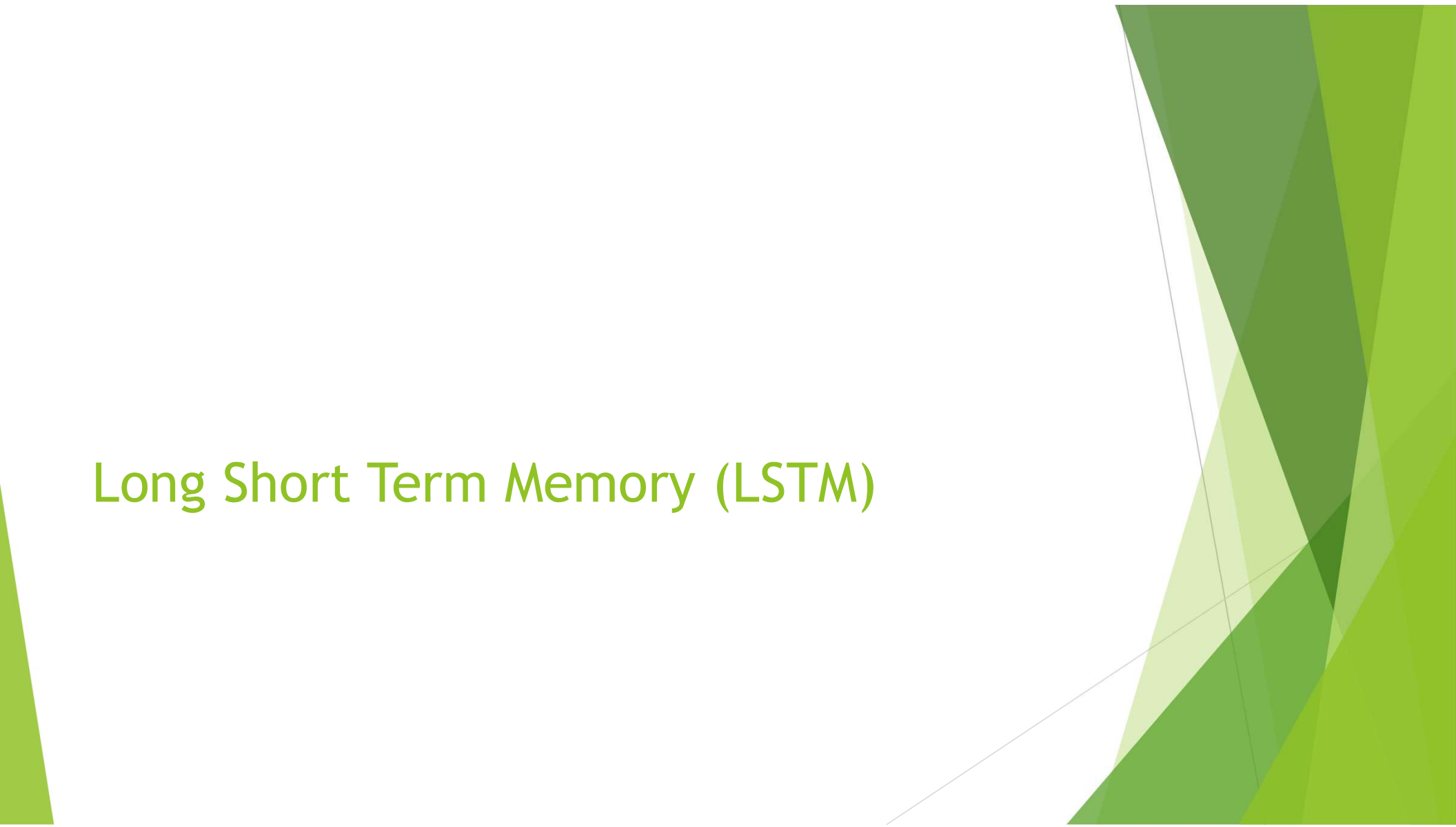


RNN untuk Data Sekuens - Testing

- Karakter *e* ini digunakan sebagai input untuk timestep berikutnya.
- Untuk *t* = 2, proses forward pass pada hidden *layer* menggunakan nilai hidden *layer t* = 1 dikalikan dengan bobot W_{hh} dan juga nilai input dikali bobot W_{xh}.
- Proses selanjutnya sama dengan timestep pertama.
- Proses tersebut dilakukan hingga seluruh input sequences diproses.

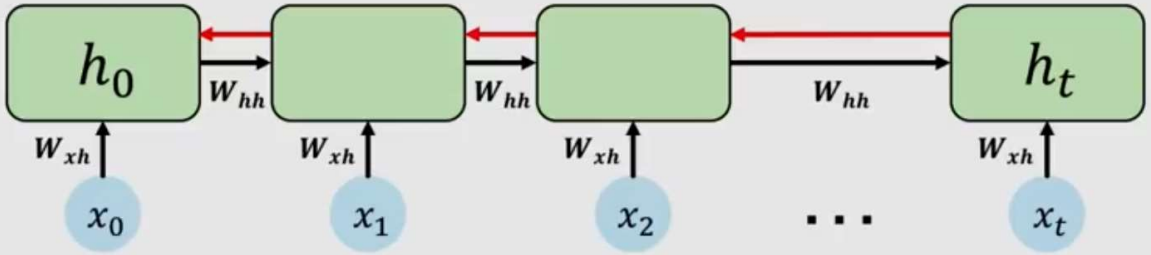


Long Short Term Memory (LSTM)



Permasalahan RNN

Standard RNN Gradient Flow: Exploding Gradients



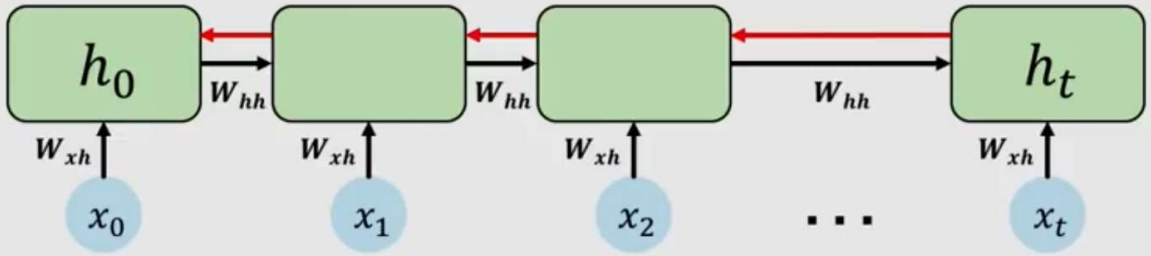
Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation!**

Many values > 1:
exploding gradients

Gradient clipping to scale big gradients

Permasalahan RNN

Standard RNN Gradient Flow: Vanishing Gradients



Computing the gradient wrt h_0 involves **many factors of W_{hh} + repeated gradient computation!**

Many values < 1:
vanishing gradients

1. Activation function
2. Weight initialization
3. Network architecture

Permasalahan RNN

The Problem of Long-Term Dependencies

Why are vanishing gradients a problem?

Multiply many **small numbers** together

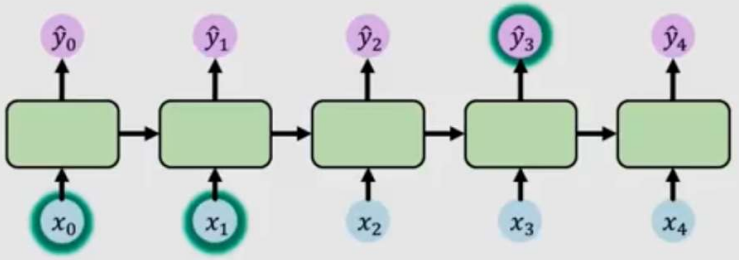
↓

Errors due to further back time steps have smaller and smaller gradients

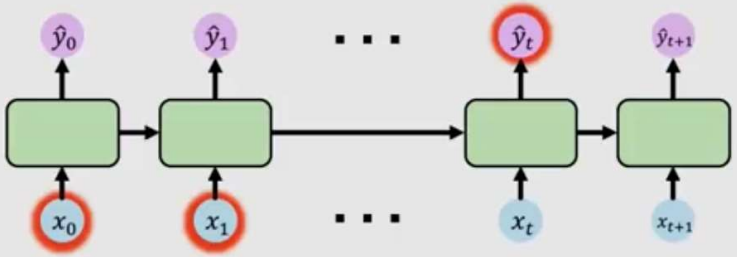
↓

Bias parameters to capture short-term dependencies

"The clouds are in the ___"

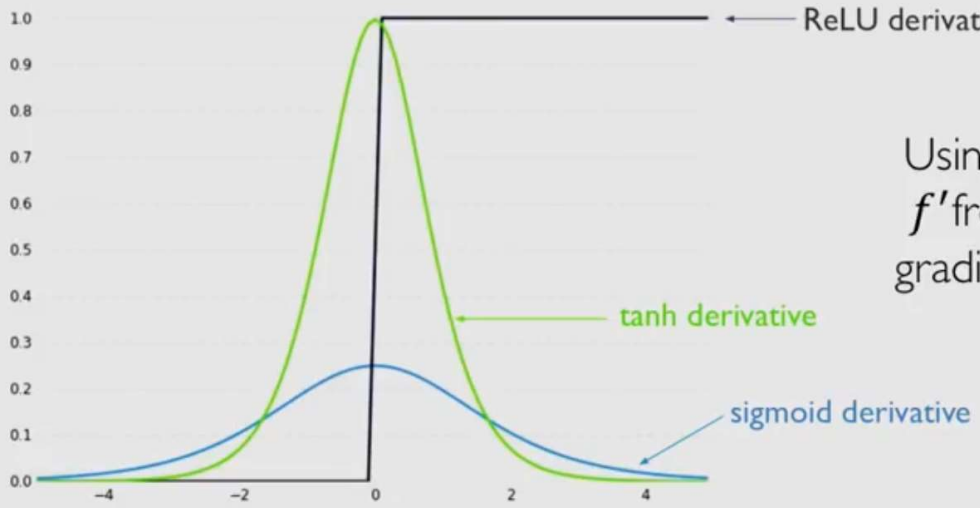


"I grew up in France, … and I speak fluent___ "



Permasalahan RNN

Trick #1: Activation Functions



Using ReLU prevents f' from shrinking the gradients when $x > 0$

Permasalahan RNN

Trick #2: Parameter Initialization

Initialize **weights** to identity matrix

Initialize **biases** to zero

$$I_n = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

This helps prevent the weights from shrinking to zero.

Permasalahan RNN

Solution #3: Gated Cells

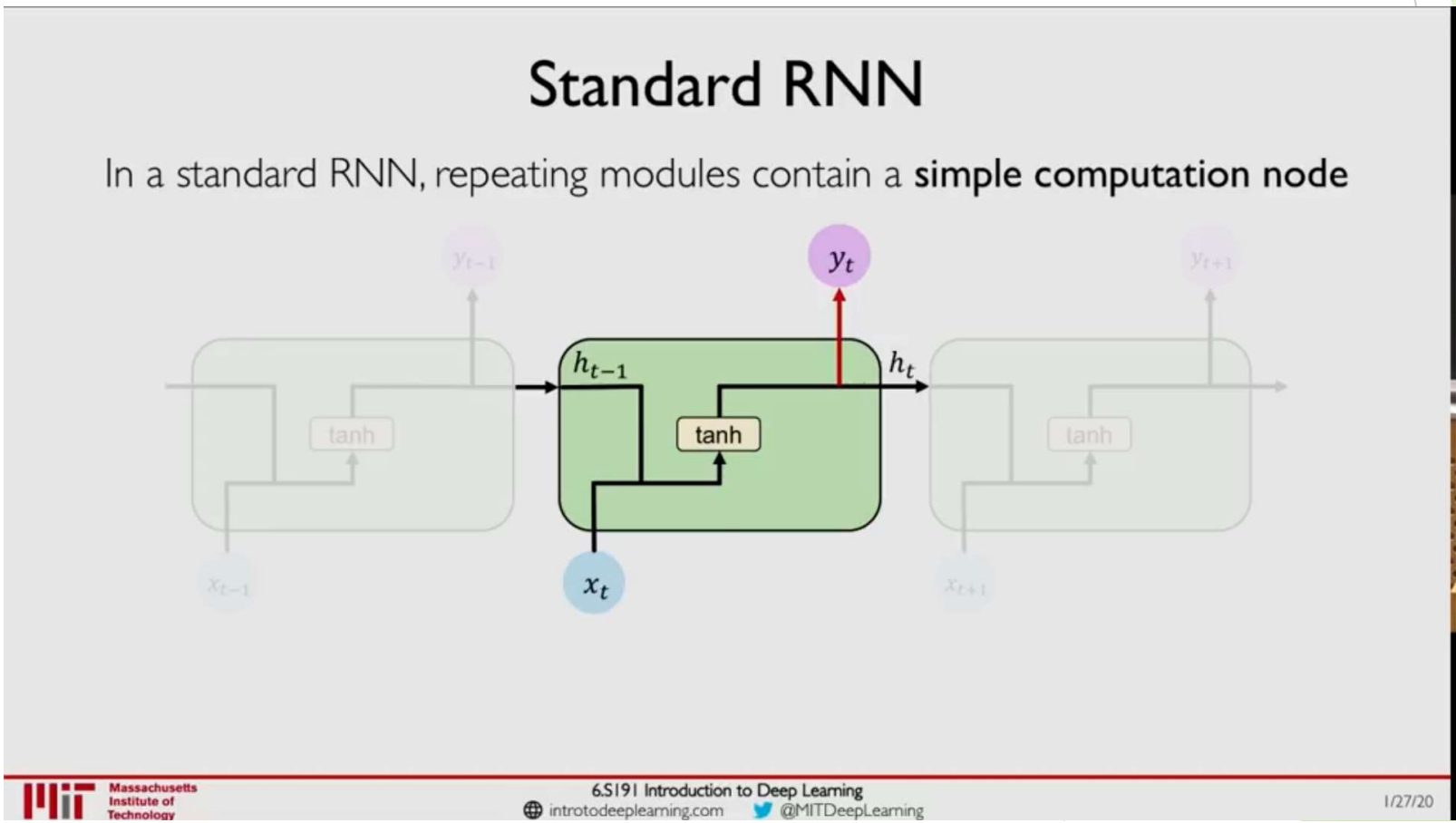
Idea: use a more **complex recurrent unit with gates** to control what information is passed through

gated cell

LSTM, GRU, etc.

Long Short Term Memory (LSTMs) networks rely on a gated cell to track information throughout many time steps.

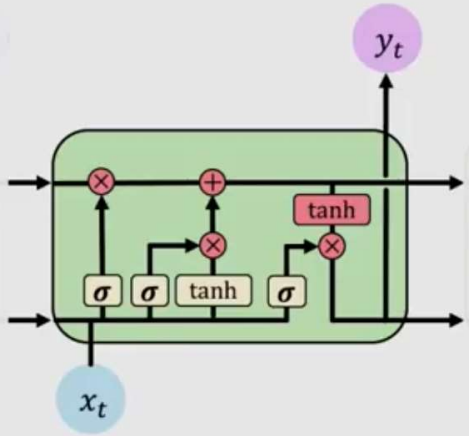
Permasalahan RNN



Long Short Term Memory (LSTM)

Long Short Term Memory (LSTMs)

LSTM modules contain **computational blocks** that **control information flow**



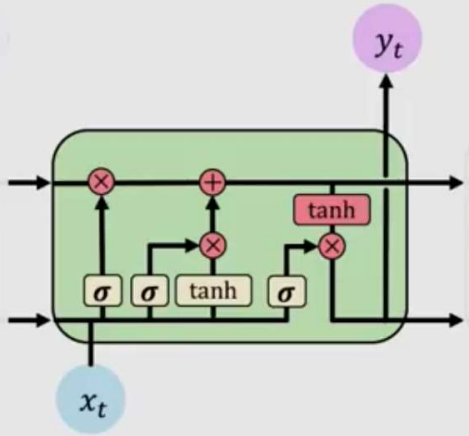
LSTM cells are able to track information throughout many timesteps

```
tf.keras.layers.LSTM(num_units)
```


Long Short Term Memory (LSTM)

Long Short Term Memory (LSTMs)

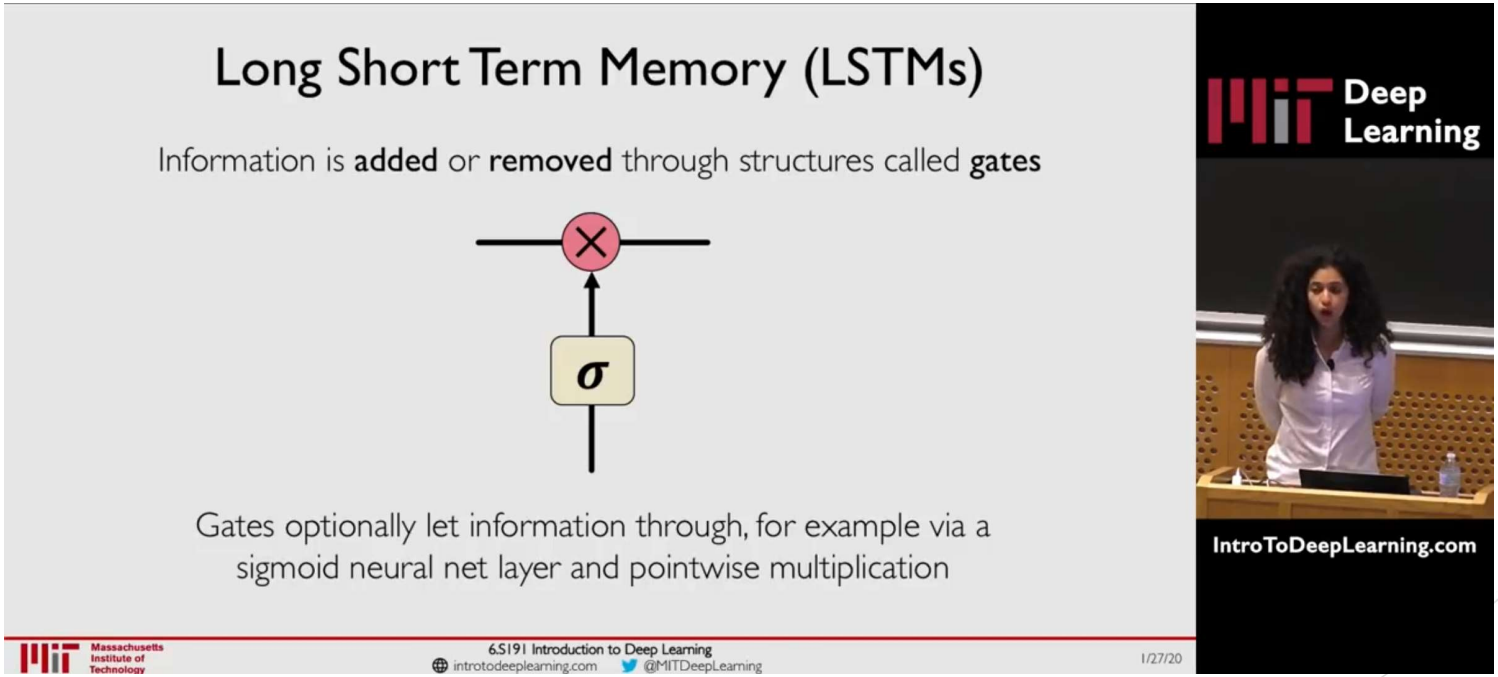
LSTM modules contain **computational blocks** that **control information flow**



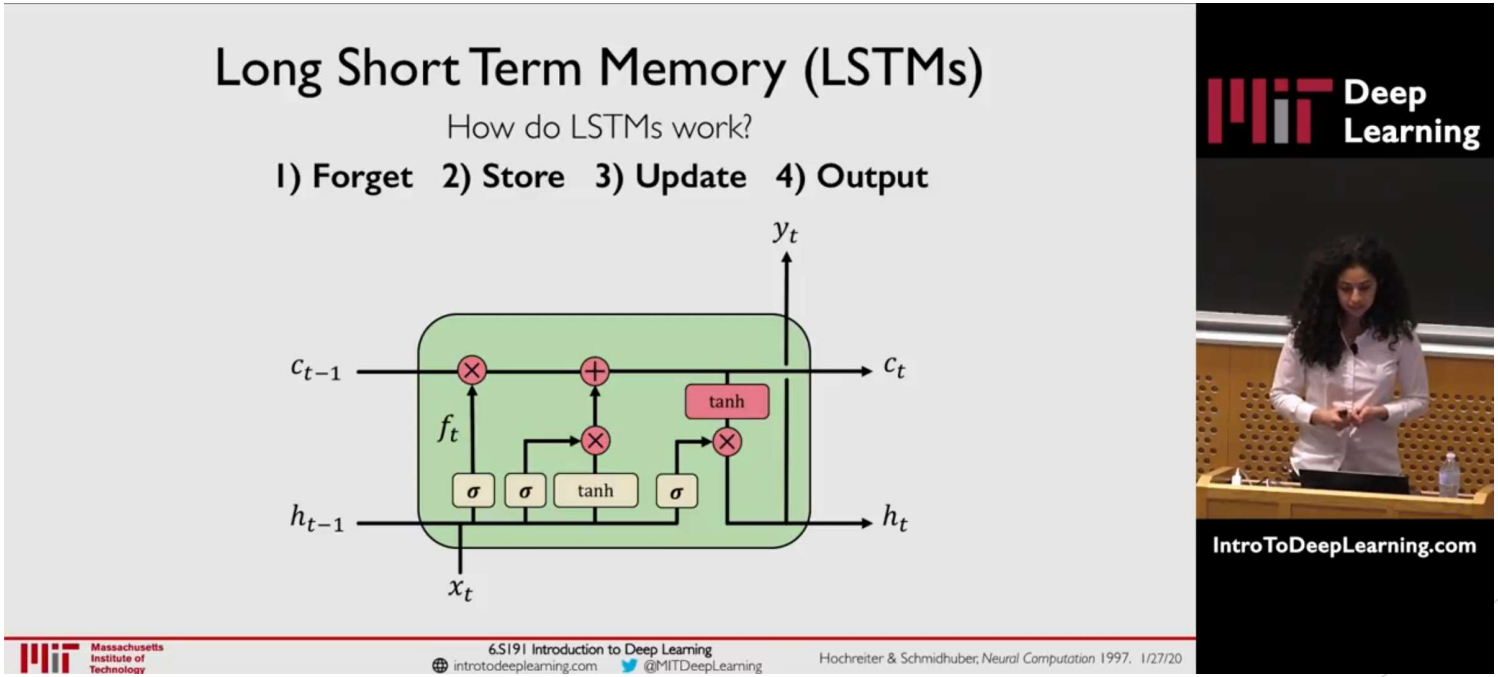
LSTM cells are able to track information throughout many timesteps

```
tf.keras.layers.LSTM(num_units)
```

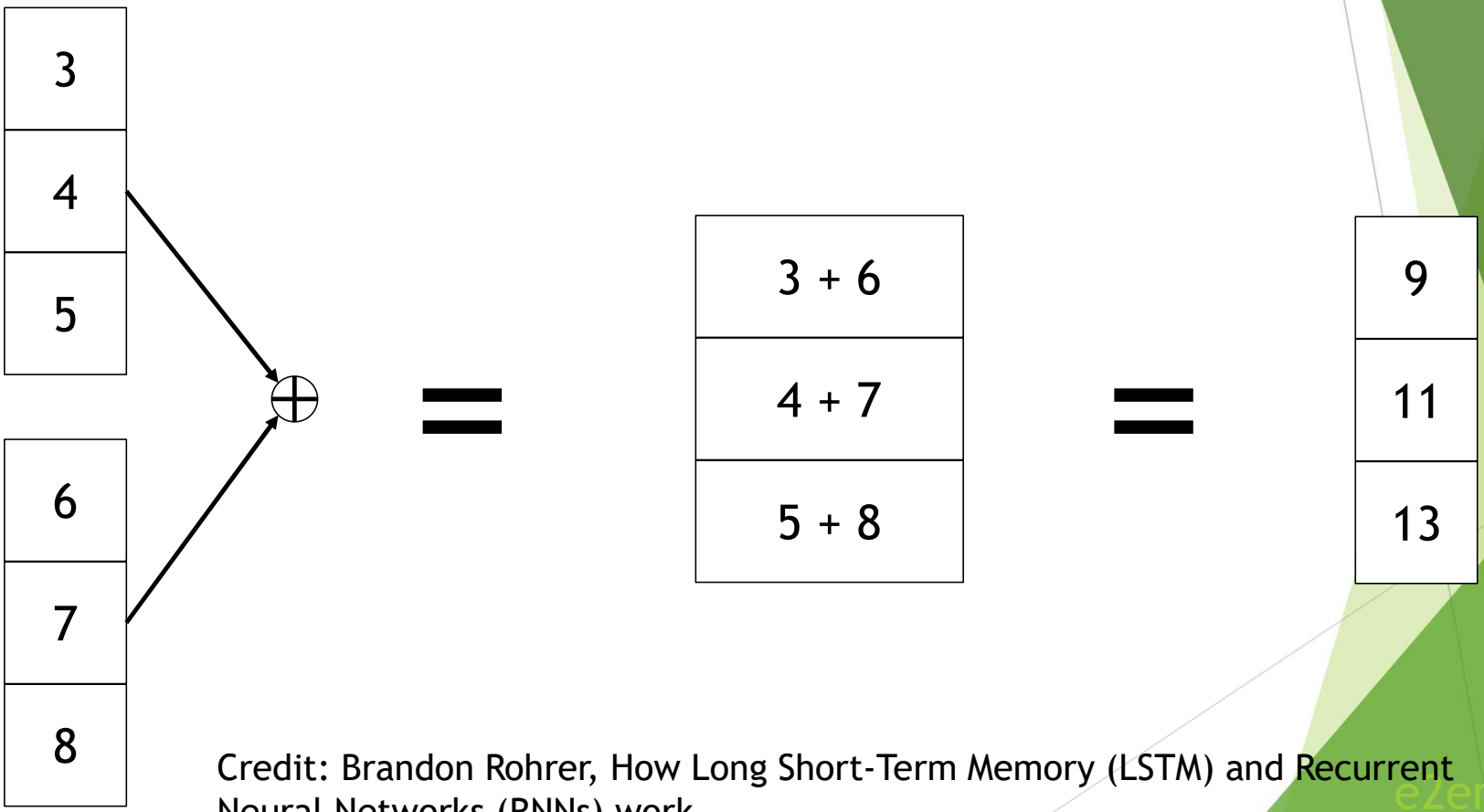

Long Short Term Memory (LSTM)



Long Short Term Memory (LSTM)

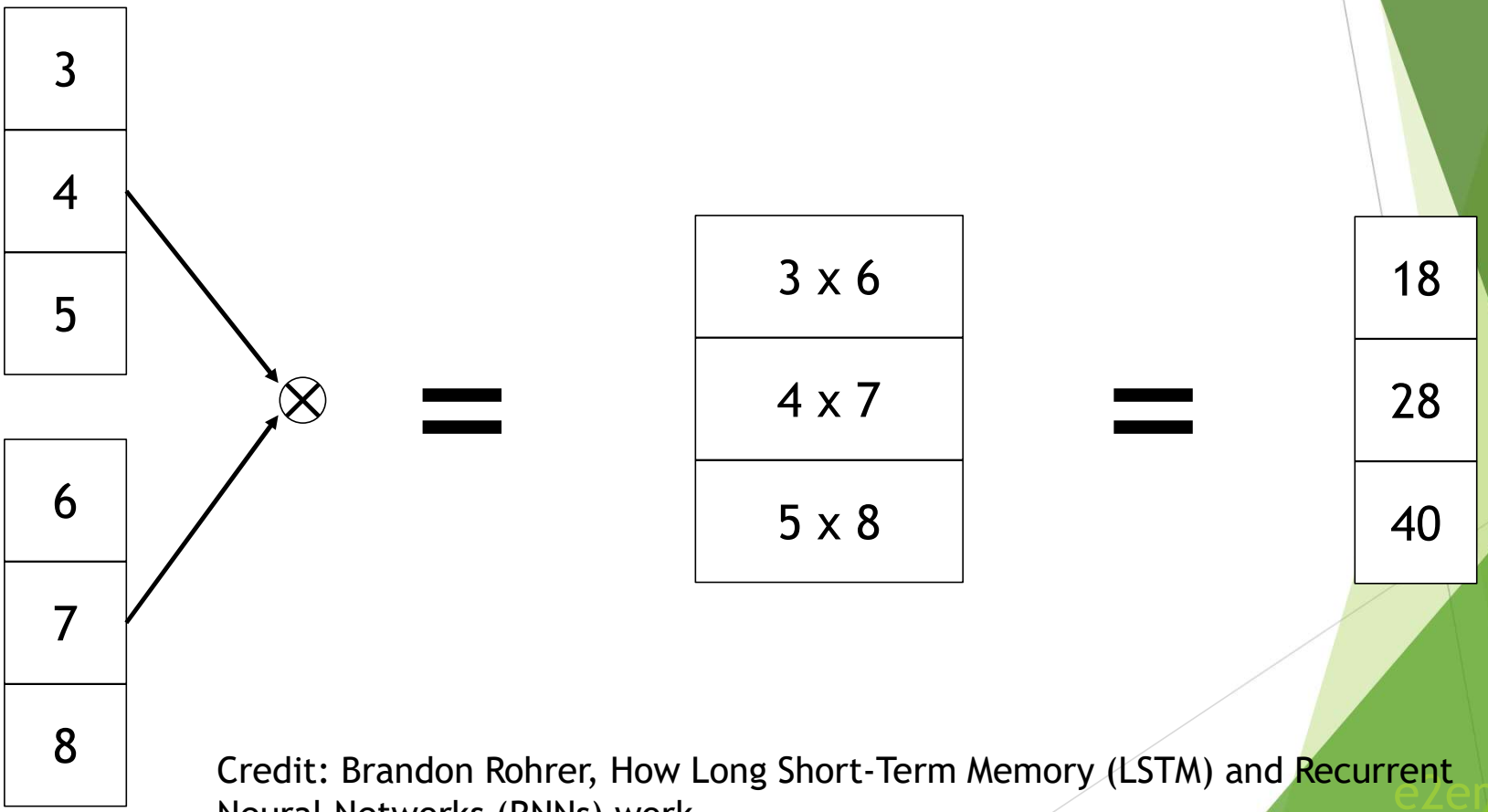


Plus junction: element-by-element addition



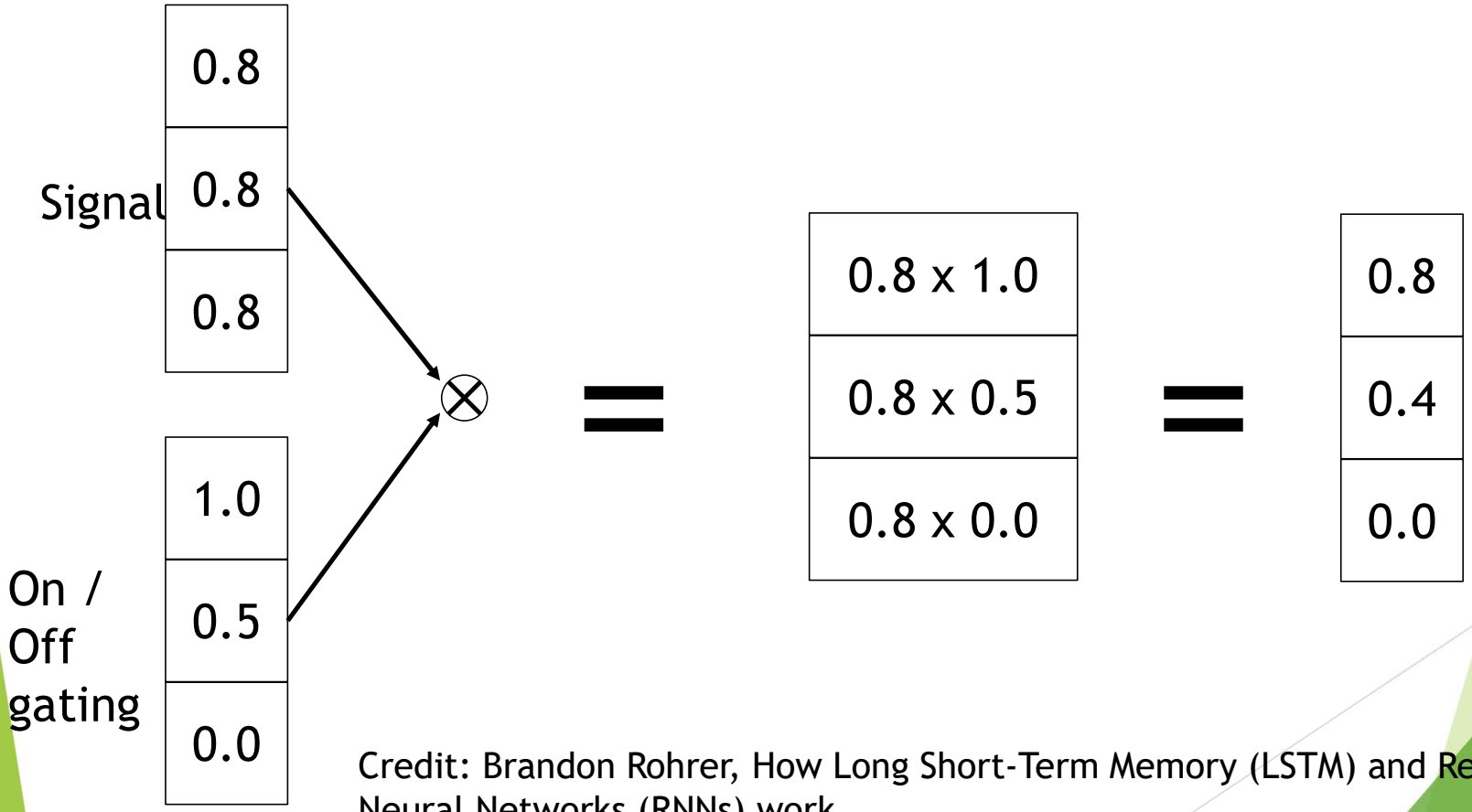
Credit: Brandon Rohrer, How Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs) work

Times junction: element-by-element multiplication



Credit: Brandon Rohrer, How Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs) work

Gating



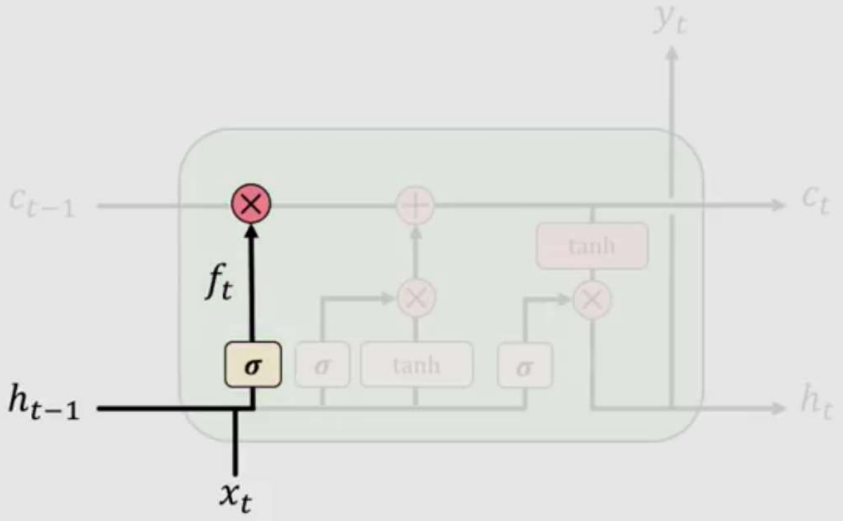
Credit: Brandon Rohrer, How Long Short-Term Memory (LSTM) and Recurrent Neural Networks (RNNs) work

Long Short Term Memory (LSTM)

Long Short Term Memory (LSTMs)

1) Forget 2) Store 3) Update 4) Output

LSTMs **forget irrelevant** parts of the previous state

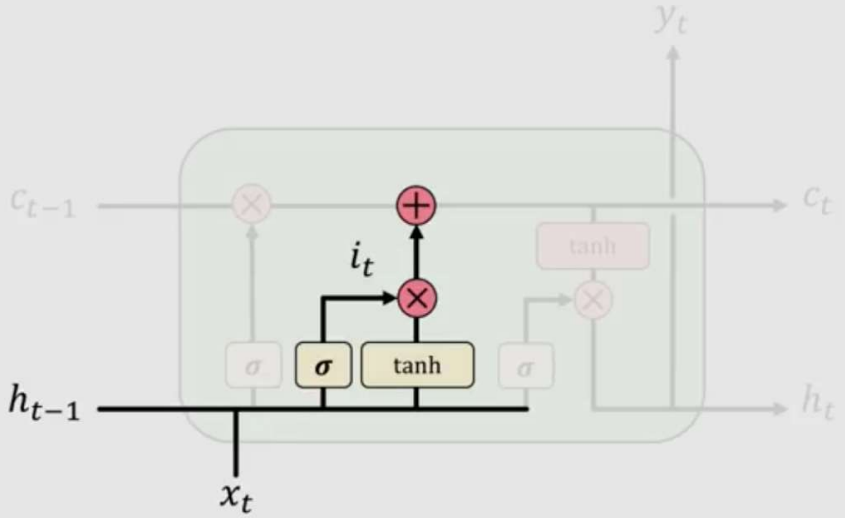


Long Short Term Memory (LSTM)

Long Short Term Memory (LSTMs)

1) Forget **2) Store** 3) Update 4) Output

LSTMs **store relevant** new information into the cell state

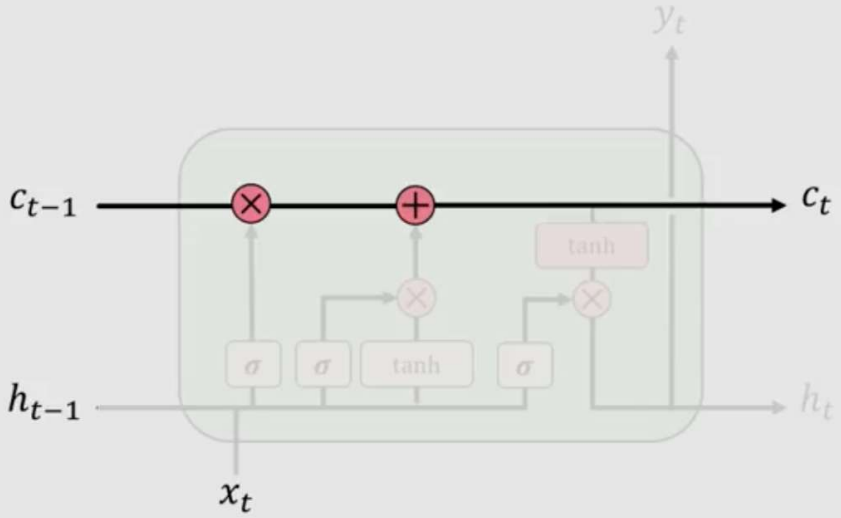


Long Short Term Memory (LSTM)

Long Short Term Memory (LSTMs)

1) Forget 2) Store **3) Update** 4) Output

LSTMs **selectively update** cell state values

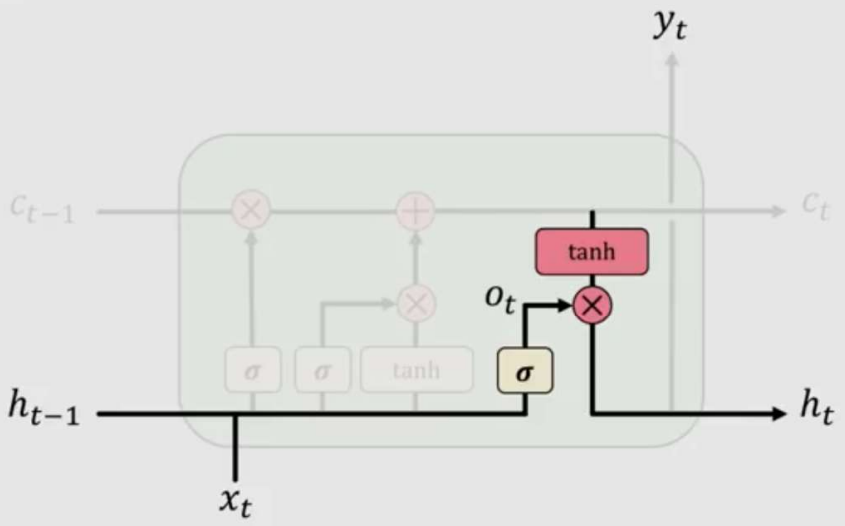


Long Short Term Memory (LSTM)

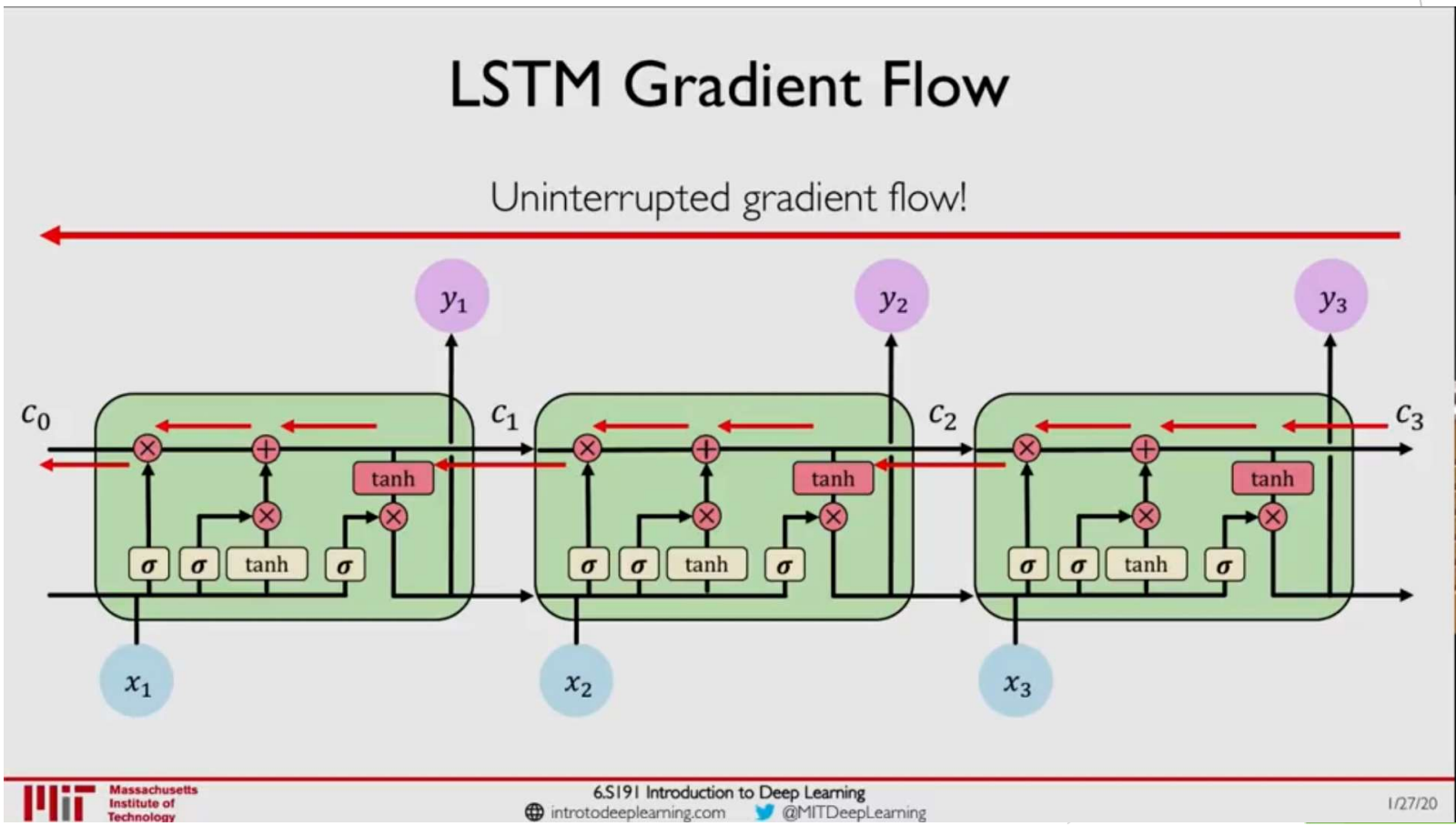
Long Short Term Memory (LSTMs)

1) Forget 2) Store 3) Update **4) Output**

The **output gate** controls what information is sent to the next time step



Long Short Term Memory (LSTM)



Long Short Term Memory (LSTM)

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information
   - **Store** relevant information from current input
   - Selectively **update** cell state
   - **Output** gate returns a filtered version of the cell state

Long Short Term Memory (LSTM)

LSTMs: Key Concepts

1. Maintain a **separate cell state** from what is outputted
2. Use **gates** to control the **flow of information**
   - **Forget** gate gets rid of irrelevant information
   - **Store** relevant information from current input
   - Selectively **update** cell state
   - **Output** gate returns a filtered version of the cell state
3. Backpropagation through time with **uninterrupted gradient flow**

Reference and Credits

- Suyanto, Kurniawan Nur Ramadhani, Satria Mandala. (2019). Deep Learning Modernisasi Machine Learning Untuk Big Data. Informatika.
- MIT Deep Learning C, Deep Sequence Modeling (Lecture 2)
  - Slide: http://introtodeeplearning.com/slides/6S191_MIT_DeepLearning_L2.pdf
  - Video: https://www.youtube.com/watch?v=qjrad0V0uJE&list=PLtBw6njQRU-rwp5__7C0oIVt26ZgjG9NI&index=2
- Brandon Rohrer, How Recurrent Neural Networks and Long Short-Term Memory Work (https://e2eml.school/how_rnns_lstm_work.html)
  - Slide: https://docs.google.com/presentation/d/1hqYB3LRwg_-ntptHxH18W1ax9kBwkaZ1Pa_s3L7R-2Y/edit#slide=id.g1dfdf4c7e7_0_204
  - Video: https://www.youtube.com/watch?v=WCUNPb-5EYI