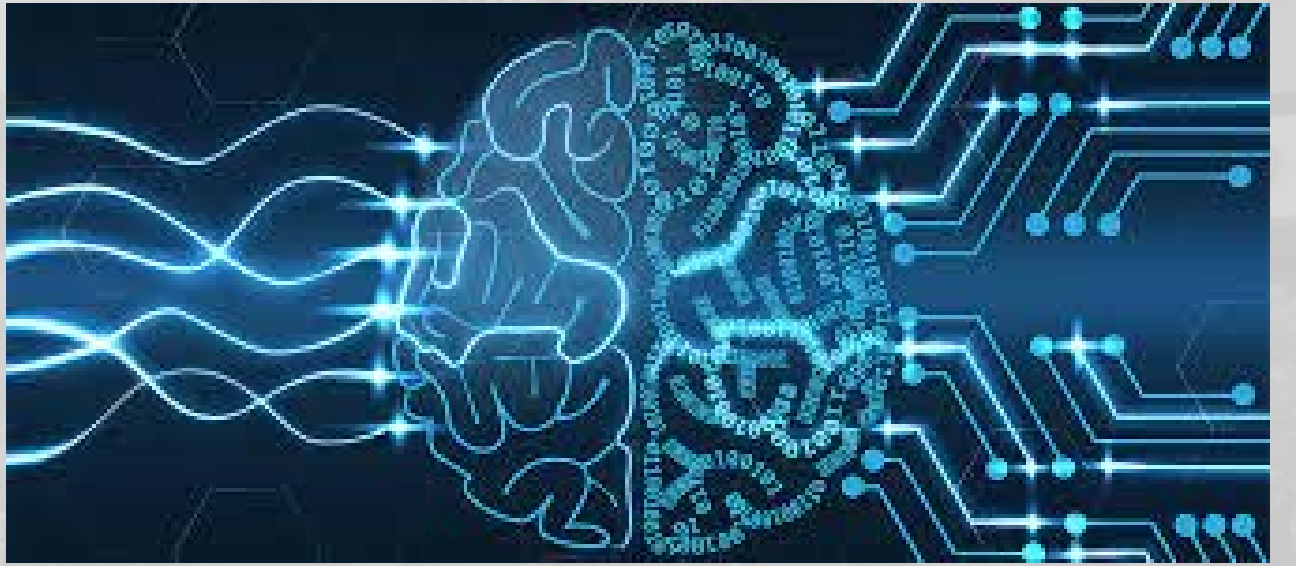
14620323

DEEP LEARNING



Deep Sequence Modelling

PENGAMPU

Dr. Fajar Astuti Hermawati, S.Kom.,M.Kom.

Bagus Hardiansyah, S.Kom.,M.Si

Andrey Kartika Widhy H., S.Kom., M.Kom.

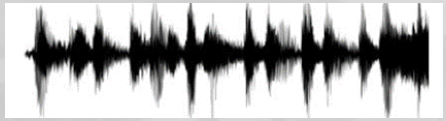
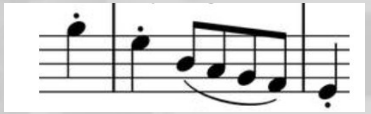
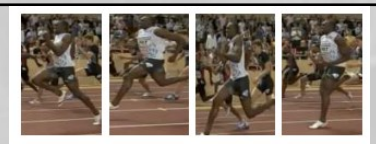
Capaian Pembelajaran

- Sub-CPMK-4: Mampu **menyelesaikan masalah komputasi kompleks** dengan menerapkan prinsip-prinsip Pemodelan Sekuensial (Sequence Modeling) dalam pemelajaran mendalam [C3, A3]

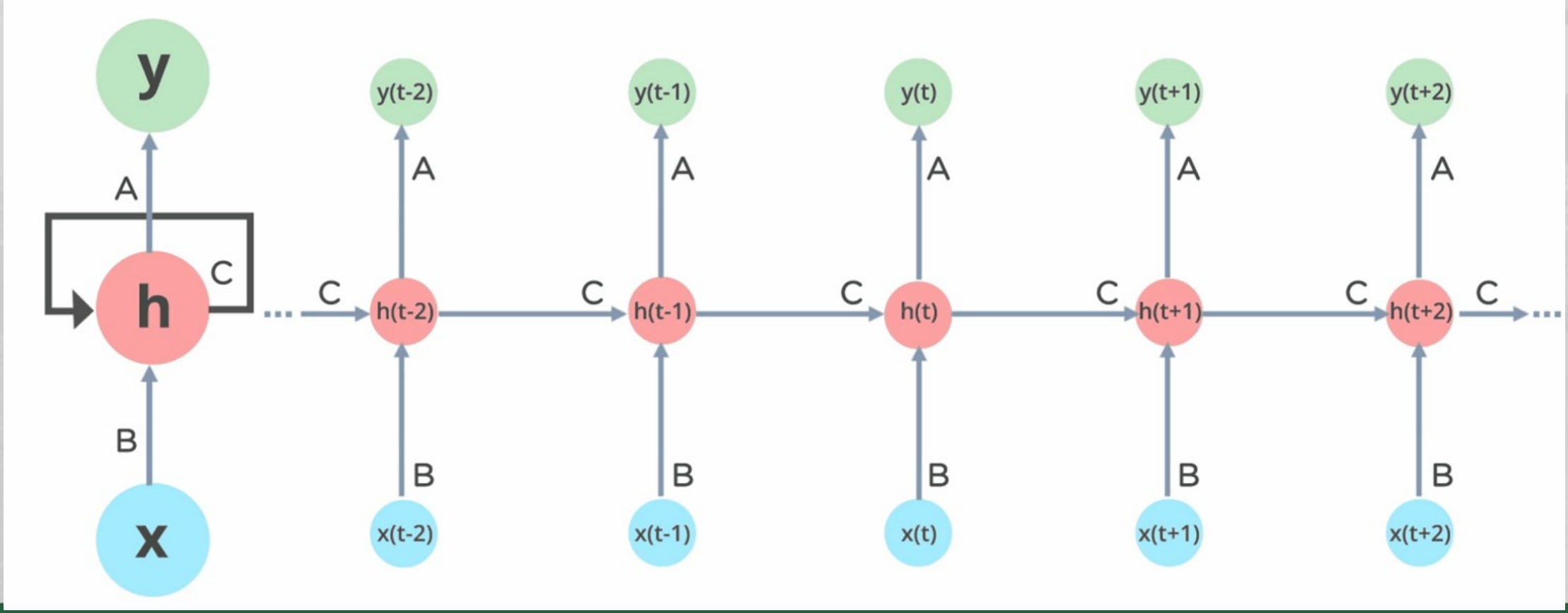
Bahan Kajian

- Recurrent Neural Network Model
- Why sequence models?
- Backpropagation through time
- Different types of RNNs
- Vanishing gradients with RNNs
- LSTM (long short term memory) unit

Examples of sequence data

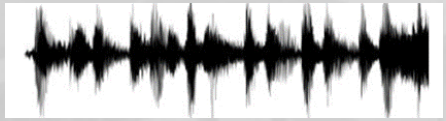
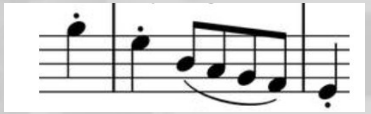
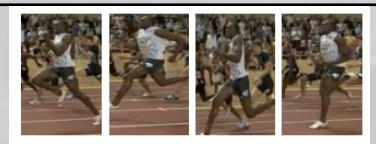
Task	Input	Output
Speech recognition		“The quick brown fox jumped over the lazy dog.”
Music generation	∅	
Sentiment classification	“There is nothing to like in this movie.”	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	Do you want to sing with me?
Video activity recognition		Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger	Yesterday, Harry Potter met Hermione Granger

Gambaran RNN



Recurrent Neural Network Model

Examples of sequence data

Task	Input	Output
Speech recognition		“The quick brown fox jumped over the lazy dog.”
Music generation	∅	
Sentiment classification	“There is nothing to like in this movie.”	
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	Do you want to sing with me?
Video activity recognition		Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger	Yesterday, Harry Potter met Hermione Granger

Recurrent Neural Network Model

- Recurrent neural networks, or RNNs (Rumelhart et al., 1986a), are a family of neural networks for processing sequential data.
- A recurrent neural network is a neural network that is specialized for processing a sequence of values $x^{(1)}, \ldots, x^{(\tau)}$.
- For the simplicity of exposition, we refer to RNNs as operating on a sequence that contains vectors $x^{(t)}$ with the time step index t ranging from 1 to τ.
- In practice, recurrent networks **usually operate on minibatches** of such sequences, with a different sequence length τ for each member of the minibatch.
- We have omitted the minibatch indices to simplify notation.

Motivating example: Name entity recognition

- Jika diberikan input kalimat:
  - x: Harry Potter and Hermione Granger invented a new spell.
- Dengan NLP, input kalimat x diatas akan diproses untuk mendapatkan token-token (dengan beberapa pengolahan awal yang diperlukan):
  - <harry> <potter> <and> <hermione> <granger> <invented> <a> <new> <spell>
- Rangkaian token tersebut dapat dinotasikan sebagai:
  - <harry> <potter> <and> <hermione> <granger> <invented> <a> <new> <spell>
  - $x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ $x^{(5)}$ $x^{(6)}$ $x^{(7)}$ $x^{(8)}$ $x^{(9)}$

Motivating example: Name entity recognition

- Output yang diharapkan pada problem NER tersebut adalah mengenali kata atau token yang merupakan nama orang, sehingga output yang diharapkan dapat dinotasikan sebagai berikut:
  - <harry> <potter> <and> <hermione> <granger> <invented> <a> <new> <spell>
  - $y^{(1)}$ $y^{(2)}$ $y^{(3)}$ $y^{(4)}$ $y^{(5)}$ $y^{(6)}$ $y^{(7)}$ $y^{(8)}$ $y^{(9)}$
  - 1 1 0 1 1 0 0 0 0

Representing words

- Dalam permasalahan Name entity recognition (NER), dataset berupa kumpulan kata (vocabulary) yang setiap katanya telah dilabeli dengan jenis kata (POS tagging)

$$\begin{bmatrix} a \\ aaron \\ \vdots \\ and \\ \vdots \\ harry \\ \vdots \\ potter \\ \vdots \\ zulu \end{bmatrix}$$

Representing words

- Misalkan dalam vocabulary kita, diketahui index (posisi) dari token sbb:
  - and = 367 Invented = 4700 A = 1 New = 5976 Spell = 8376 Harry = 4075 Potter = 6830 Hermione = 4200 Gran… = 4000

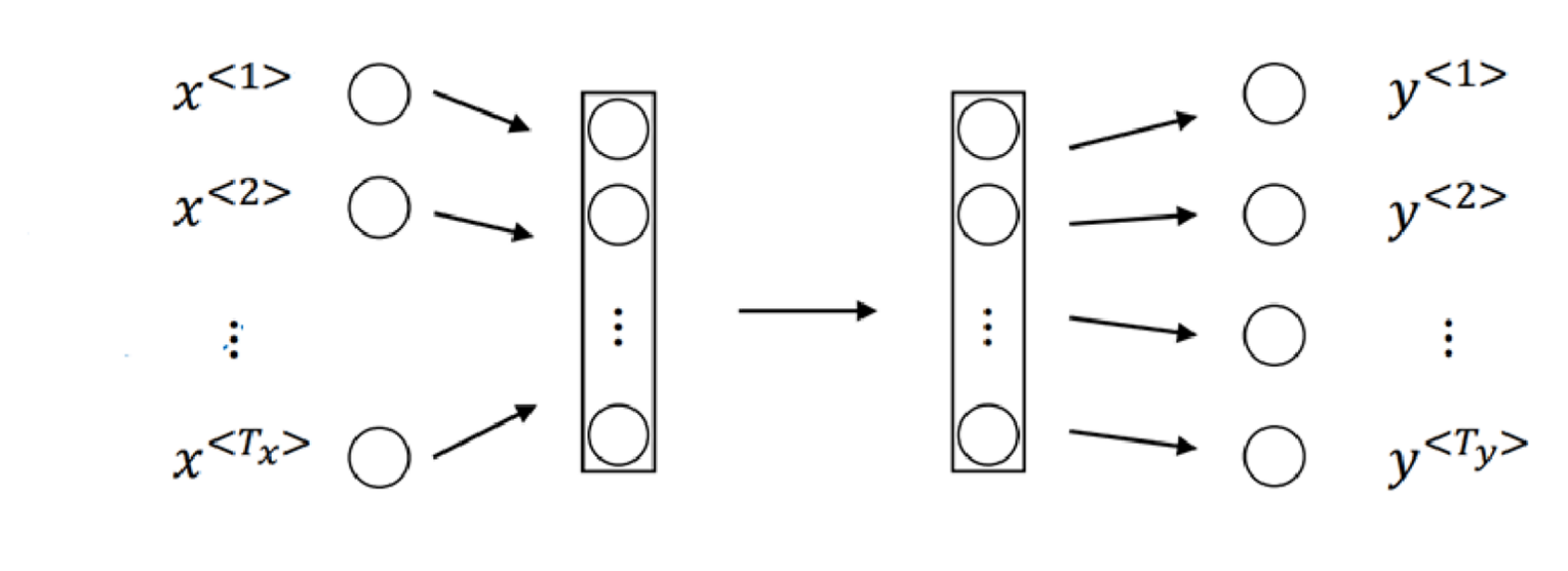
$$\begin{bmatrix} a \\ aaron \\ \vdots \\ and \\ \vdots \\ harry \\ \vdots \\ potter \\ \vdots \\ zulu \end{bmatrix}$$

Representing words

- Maka setiap token input dapat dinyatakan dalam vector sebagai berikut:
  - <harry> <potter> <and> <hermione> <granger> <invented> <a> <new> <spell>
  - $x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$ $x^{(5)}$ $x^{(6)}$ $x^{(7)}$ $x^{(8)}$ $x^{(9)}$

Why sequence models?

Why not a standard network?



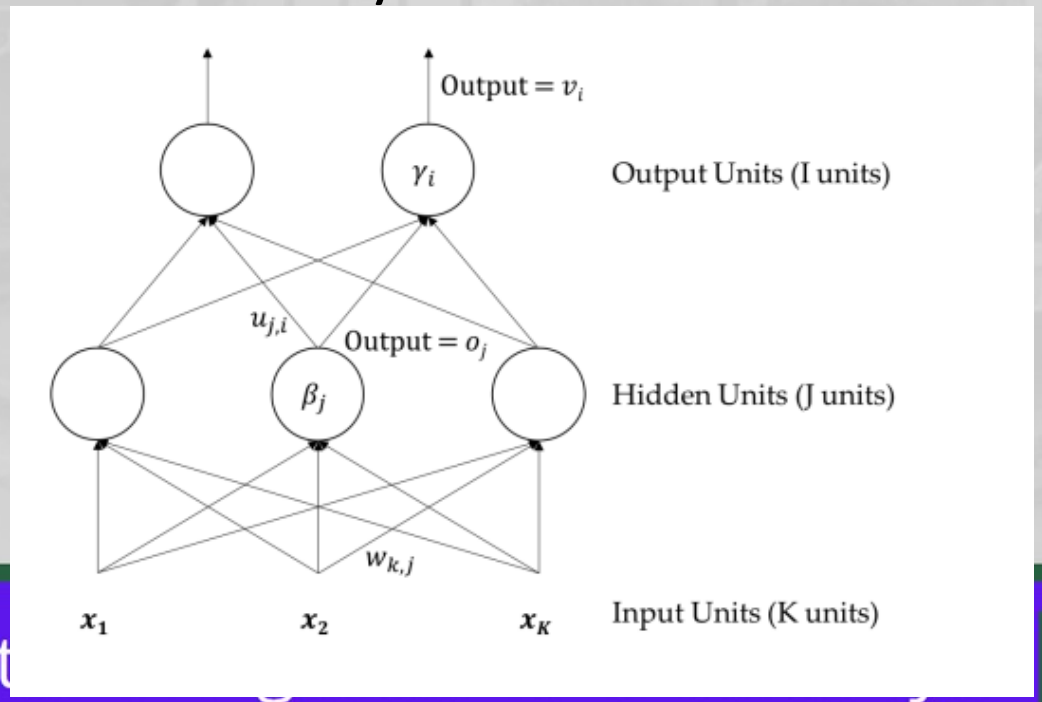
- Problems:
  - Input, output dapat memiliki panjang yang berbeda dalam contoh yang berbeda.
  - Tidak berbagi fitur yang dipelajari di berbagai posisi teks.

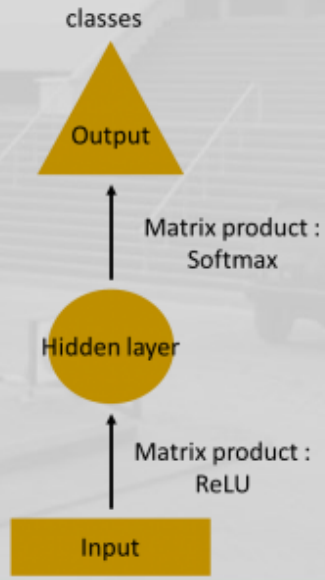
Why not a standard network?

- Katakanlah kita akan memprediksi kata berikutnya dalam sebuah kalimat.
- Mari kita coba menyelesaikannya menggunakan multilayer perceptron (MLP).
- Jadi apa yang terjadi di MLP.

Why not a standard network?

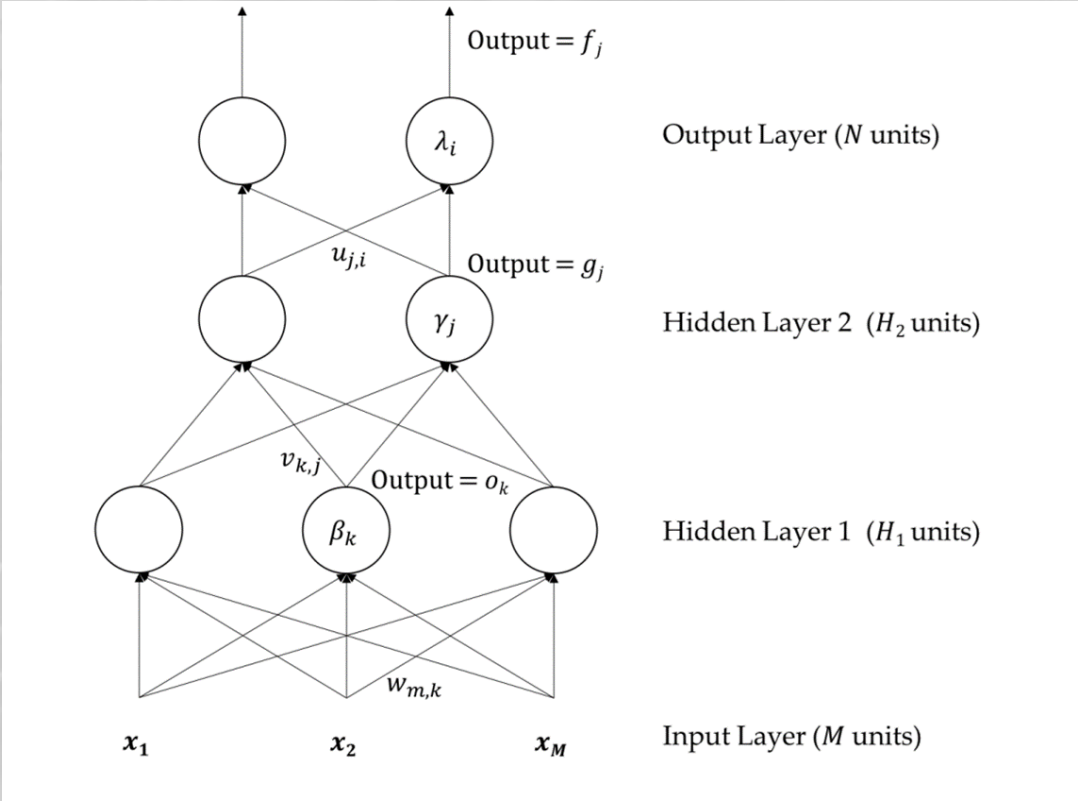
- Dalam bentuk paling sederhana, kita memiliki lapisan masukan, lapisan tersembunyi, dan lapisan keluaran.
- Lapisan masukan menerima masukan, aktivasi lapisan tersembunyi diterapkan dan akhirnya memberikan keluaran.





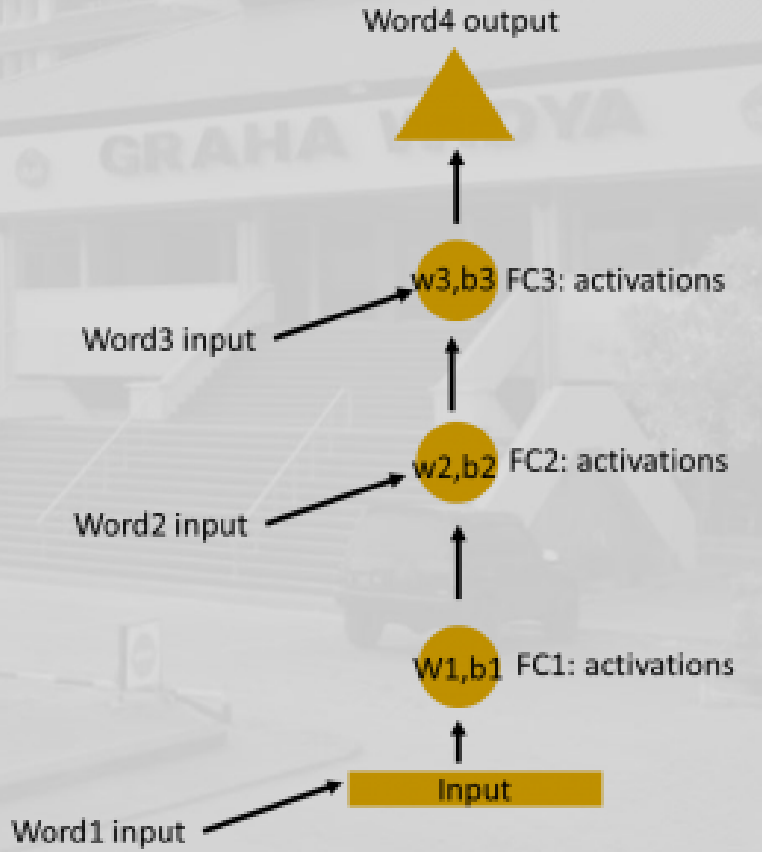
Why not a standard network?

- Mari kita memiliki jaringan yang lebih dalam, di mana ada banyak lapisan tersembunyi.
- Jadi di sini, lapisan input menerima input, aktivasi lapisan tersembunyi pertama diterapkan dan kemudian aktivasi ini dikirim ke lapisan tersembunyi berikutnya, dan aktivasi berturut-turut melalui lapisan untuk menghasilkan keluaran.
- Setiap lapisan tersembunyi dicirikan oleh bobot dan biasnya sendiri.



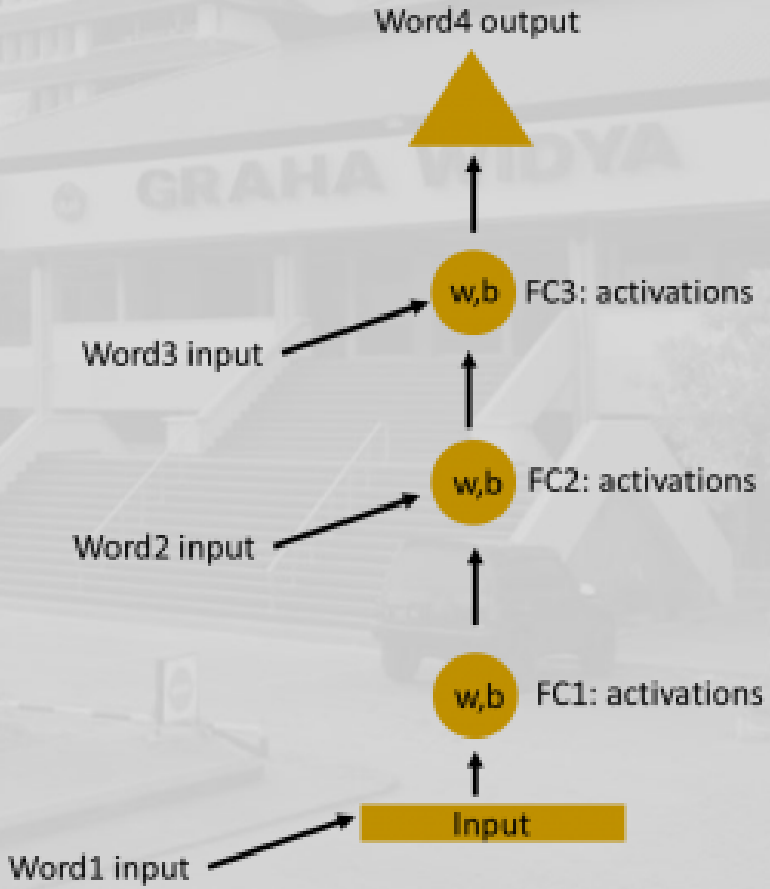
Why not a standard network?

- Karena setiap lapisan tersembunyi memiliki bobot dan aktivasinya sendiri, mereka berperilaku secara independen.
- Sekarang tujuannya adalah untuk mengidentifikasi hubungan antara input yang berurutan.
- Bisakah kita menyediakan input ke lapisan tersembunyi?



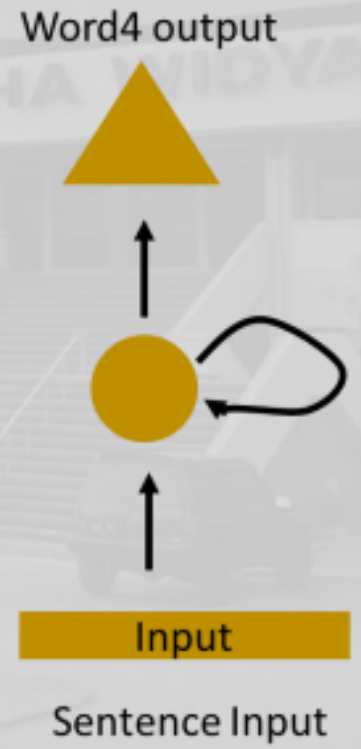
Why not a standard network?

- Di sini, bobot dan bias dari lapisan tersembunyi ini berbeda.
- Dan karenanya masing-masing lapisan ini berperilaku secara independen dan tidak dapat digabungkan bersama.
- Untuk menggabungkan lapisan-lapisan tersembunyi ini bersama-sama, kita akan memiliki bobot dan bias yang sama untuk lapisan-lapisan tersembunyi ini.



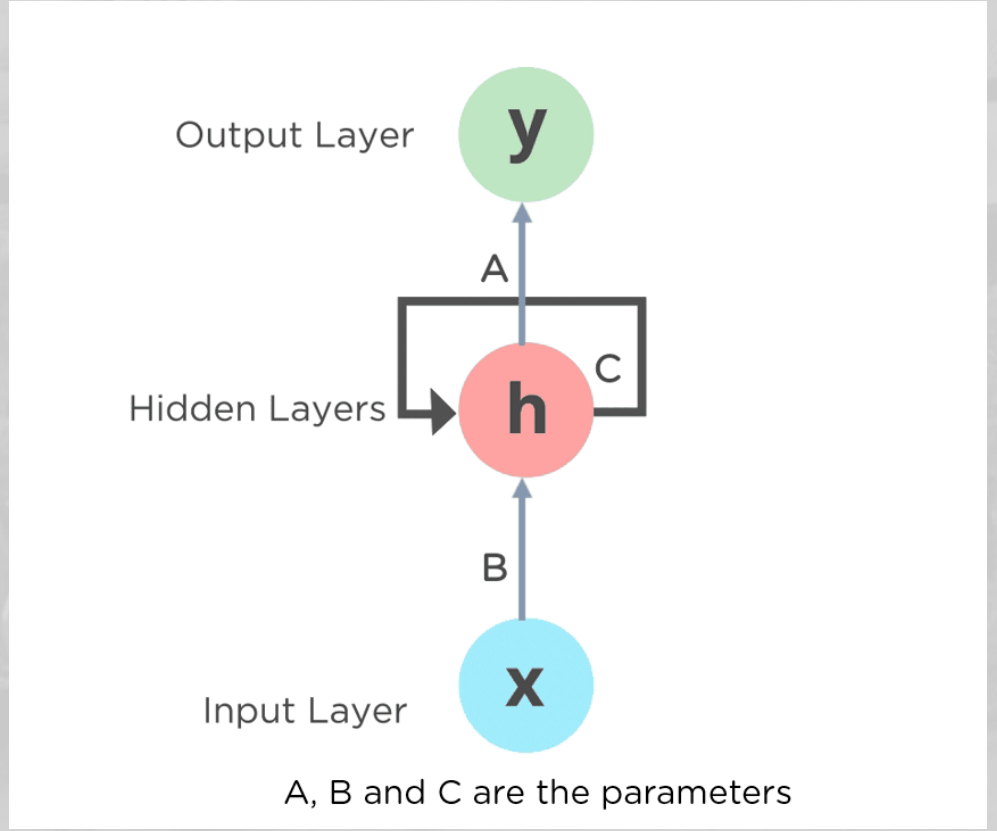
Why not a standard network?

- Kita sekarang dapat menggabungkan layer-layer ini bersama-sama, bahwa bobot dan bias dari semua layer tersembunyi adalah sama.
- Semua lapisan tersembunyi ini dapat di-looping.



Why not a standard network?

- Jadi seperti memberikan input ke lapisan tersembunyi.
- Sepanjang waktu, bobot neuron berulang akan sama karena sekarang menjadi neuron tunggal.
- Jadi neuron berulang menyimpan keadaan input sebelumnya dan menggabungkannya dengan input saat ini sehingga mempertahankan beberapa hubungan input saat ini dengan input sebelumnya.



Unfolding Computational Graphs

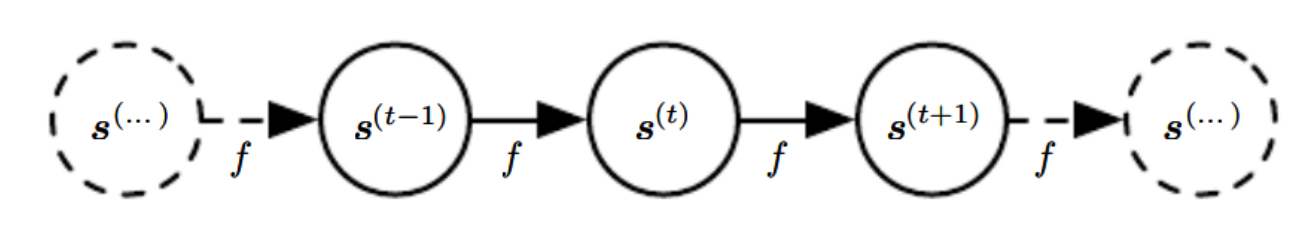
- **Computational Graphs** adalah cara untuk memformalkan struktur sekumpulan komputasi, seperti yang terlibat dalam pemetaan input dan parameter ke output dan kerugian.
- Gagasan untuk membuka perhitungan rekursif atau berulang menjadi grafik komputasi yang memiliki struktur berulang, biasanya sesuai dengan rangkaian peristiwa.

Unfolding Computational Graphs

- Consider the classical form of a dynamical system

$$s^{(t)} = f(s^{(t-1)}; \theta),$$

  - where $s^{(t)}$ is called the state of the system
- Equation is **recurrent** because the definition of *s* at time *t* refers back to the same definition at time *t* − 1.
- The classical dynamical system described by equation, illustrated as an **unfolded computational graph**



Unfolding Computational Graphs

- Each node in graph, represents the state at some time t, and the function f maps the state at t to the state at $t+1$.
- The same parameters (the same value of θ used to parametrize f) are used for all time steps.
- For a finite number of time steps τ, the graph can be **unfolded** by applying the definition $\tau - 1$ times.
  - For example, if we **unfold** equation for $\tau = 3$ time steps, we obtain

$$\begin{aligned} \boldsymbol{s}^{(3)} &= f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}) \\ &= f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta}). \end{aligned}$$

Unfolding Computational Graphs

- Such an expression can now be represented by a traditional directed acyclic computational graph
- As another example, let us consider a dynamical system driven by an external signal $x(t)$, Eq. (4)

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}),$$

- where we see that the state now contains information about the **whole past sequence**

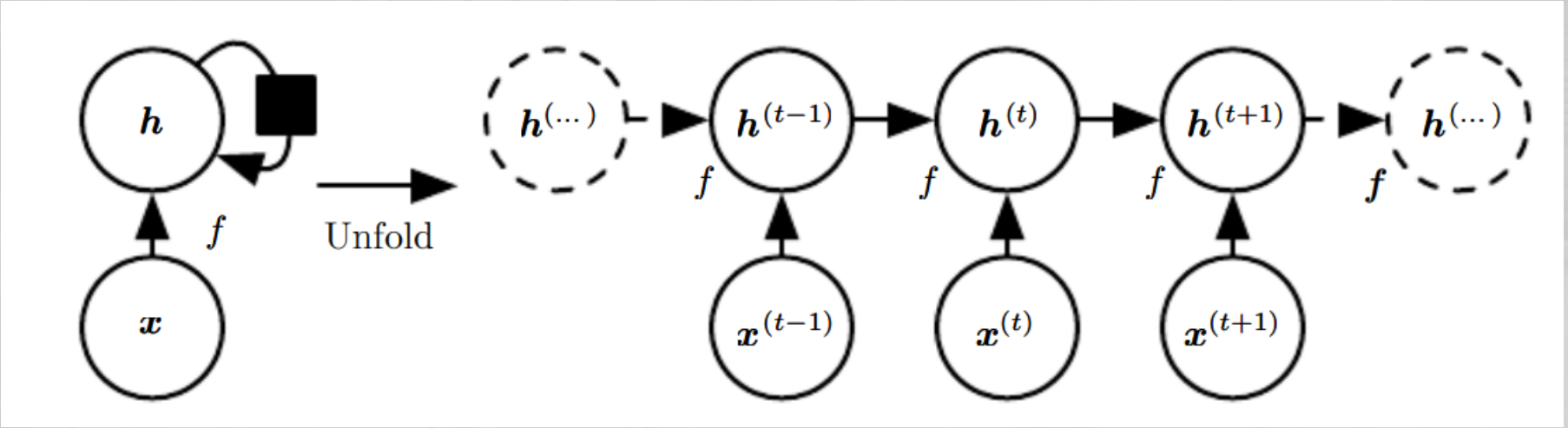
Unfolding Computational Graphs

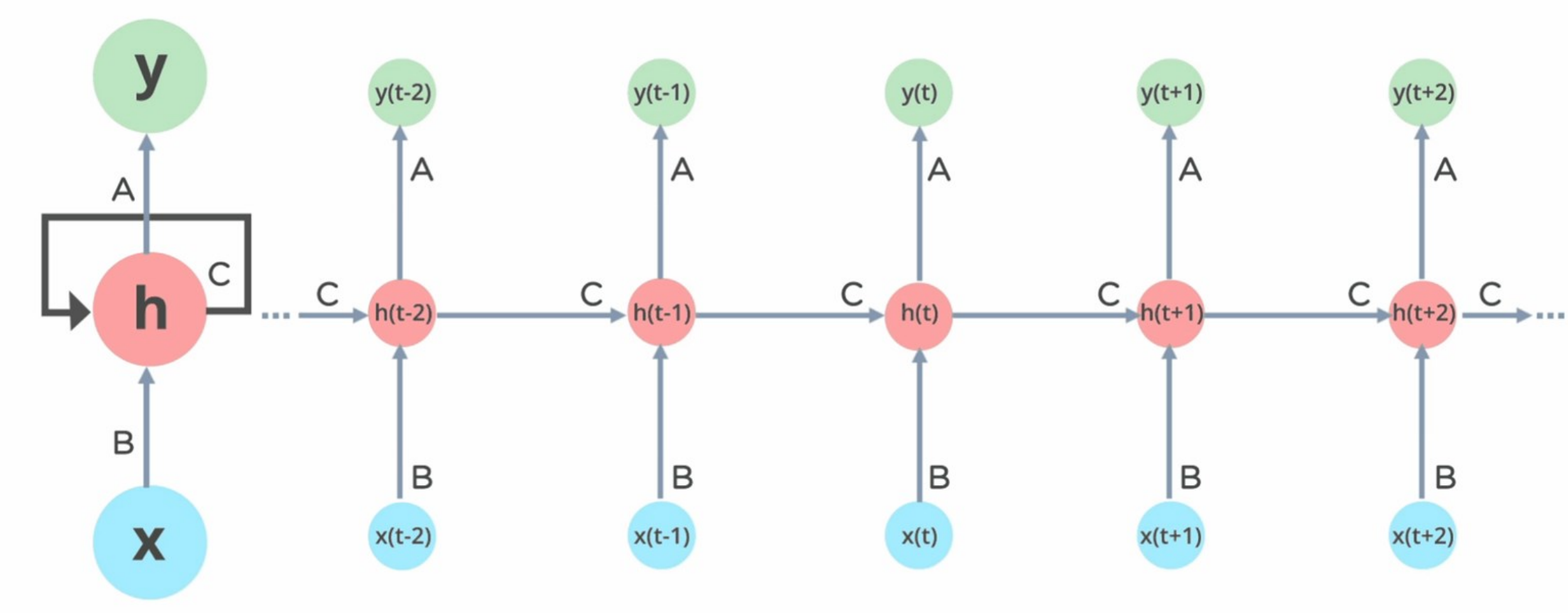
- Many recurrent neural networks use the following equation (5) or a similar equation to define the values of their hidden units.
- To indicate that the state is the hidden units of the network, we now rewrite equation (4) using the variable h to represent the state,

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}),$$

Unfolding Computational Graphs

- illustrated in the following figure, typical RNNs will add extra architectural features such as output layers that read information out of the state $\boldsymbol{h}$ to make predictions




y
A
h
C
B
x
y(t-2)
y(t-1)
y(t)
y(t+1)
y(t+2)
h(t-2)
h(t-1)
h(t)
h(t+1)
h(t+2)
x(t-2)
x(t-1)
x(t)
x(t+1)
x(t+2)

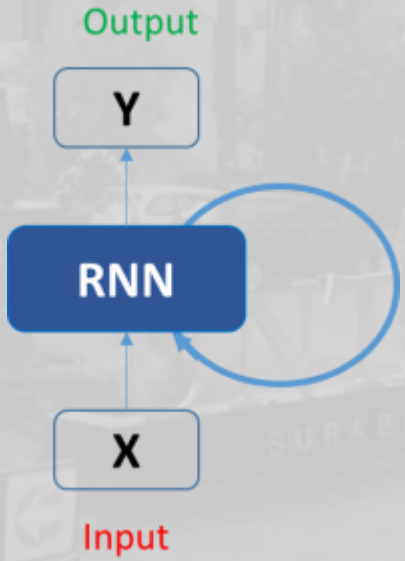
Backpropagation through time

Recurrent Neuron

- Misalkan kita ambil RNN level karakter di mana kita memiliki kata "Hello".
- Jadi diberikan 4 huruf pertama yaitu h, e, l, l dan meminta jaringan untuk memprediksi huruf terakhir yaitu 'o’.
- Jadi disini kosakata tugasnya hanya 4 huruf {h,e,l,o}.
- Dalam skenario kasus nyata yang melibatkan pemrosesan bahasa alami, kosakata mencakup kata-kata di seluruh basis data wikipedia, atau semua kata dalam suatu bahasa.
- Di sini untuk kesederhanaan kami telah mengambil satu set kosakata yang sangat kecil.

Recurrent Neuron

- Mari kita lihat bagaimana struktur recurrent digunakan untuk memprediksi huruf kelima pada kata “hello”.



Recurrent Neuron

- Dalam struktur di gambar , blok RNN biru, menerapkan sesuatu yang disebut rumus perulangan ke vektor masukan dan juga keadaan sebelumnya.
- Dalam hal ini, huruf "***h***" tidak ada yang mendahuluinya, mari kita ambil huruf "***e***".
- Jadi pada saat huruf "**e**" diberikan ke jaringan, rumus perulangan diterapkan pada huruf "**e**" dan keadaan sebelumnya yaitu huruf "**h**".

Recurrent Neuron

- Ini dikenal sebagai berbagai **langkah waktu input**.
- Jadi jika pada waktu *t* inputnya adalah "**e**", pada waktu *t*-1 inputnya adalah "**h**".
- Rumus perulangan diterapkan pada **e** dan **h** keduanya. dan kita mendapatkan keadaan baru.
- Rumus untuk keadaan saat ini dapat ditulis sebagai –

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}),$$

- Dimana, $\boldsymbol{h}^{(t)}$ adalah state baru, $\boldsymbol{h}^{(t-1)}$ adalah state sebelumnya sedangkan $\boldsymbol{x}^{t}$ adalah input saat ini.

Recurrent Neuron

- Kita sekarang memiliki status input sebelumnya, bukan input itu sendiri, karena neuron input akan menerapkan transformasi pada input kita sebelumnya.
- Jadi setiap input yang berurutan disebut sebagai **langkah waktu (time step)**.

Recurrent Neuron

- Dalam hal ini kita memiliki empat input untuk diberikan ke jaringan, selama rumus perulangan, fungsi yang sama dan bobot yang sama diterapkan ke jaringan pada setiap langkah waktu.
- Mengambil bentuk paling sederhana dari jaringan saraf berulang (***recurrent neural network***), misalkan fungsi aktivasi adalah **tanh**, bobot pada neuron berulang adalah **Whh** dan bobot pada input neuron adalah **Wxh**, kita dapat menulis persamaan keadaan pada waktu ***t*** sebagai :

$$h^{(t)} = \tanh(w_{hh}h^{(t)} + w_{xh}x^{(t)})$$

Recurrent Neuron

- Neuron berulang (***Recurrent neuron***) dalam kasus ini hanya mempertimbangkan keadaan sebelumnya.
- Untuk urutan yang lebih panjang, persamaan dapat melibatkan banyak keadaan seperti itu.
- Setelah keadaan akhir dihitung, kita dapat melanjutkan untuk menghasilkan output.
- Sekarang, setelah status saat ini dihitung, kita dapat menghitung status keluaran sebagai-

$$y^{(t)} = w_{hy} h^{(t)}$$

Recurrent Neuron

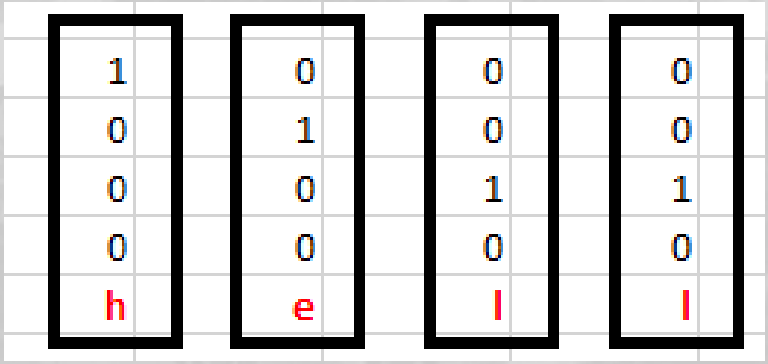
- Ringkasan langkah-langkah dalam neuron berulang (Recurrent Neuron):
  1. Satu langkah waktu input dipasok ke jaringan yaitu $\boldsymbol{x}^{(t)}$ dipasok ke jaringan
  2. Kemudian menghitung keadaan saat ini menggunakan kombinasi input saat ini dan keadaan sebelumnya yaitu menghitung $\boldsymbol{h}^{(t)}$
  3. $\boldsymbol{h}^{(t)}$ saat ini menjadi $\boldsymbol{h}^{(t-1)}$ untuk langkah waktu berikutnya
  4. Kita dapat melakukan langkah waktu sebanyak yang diminta oleh masalah dan menggabungkan informasi dari semua keadaan sebelumnya

Recurrent Neuron

- Ringkasan langkah-langkah dalam neuron berulang (Recurrent Neuron):
  5. Setelah semua langkah waktu selesai, keadaan akhir saat ini digunakan untuk menghitung output $\boldsymbol{y}^{(t)}$
  6. Keluaran kemudian dibandingkan dengan keluaran aktual dan kesalahan dihasilkan
  7. Kesalahan kemudian disebarkan kembali ke jaringan untuk memperbarui bobot (***back propagation***) dan jaringan dilatih

Forward Propagation in a Recurrent Neuron

- Misalkan input dinyatakan sebagai berikut:



1	0	0	0
0	1	0	0
0	0	1	1
0	0	0	0
h	e	l	l

- Inputnya adalah *one hot encoded*.
- Seluruh kosakata kita adalah {**h**,**e**,**l**,**o**} dan oleh karena itu kita dapat dengan mudah menyandikan masukan.

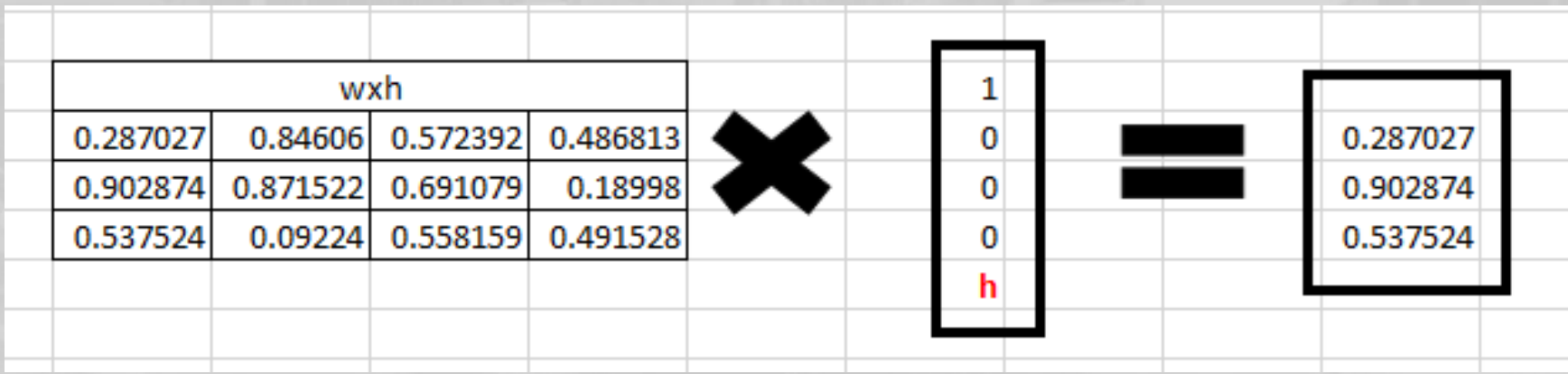
Forward Propagation in a Recurrent Neuron

- Sekarang neuron input akan mengubah input ke keadaan tersembunyi (hidden state) menggunakan bobot w_{xh}.
- Misalkan inisialisasi bobot secara acak sebagai matriks 3*4 –

wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

Forward Propagation in a Recurrent Neuron

- **Step 1**: Sekarang untuk huruf "***h***", untuk hidden state, kita membutuhkan $W_{xh}*X^{(t)}$. Dengan perkalian matriks, kita dapatkan sebagai



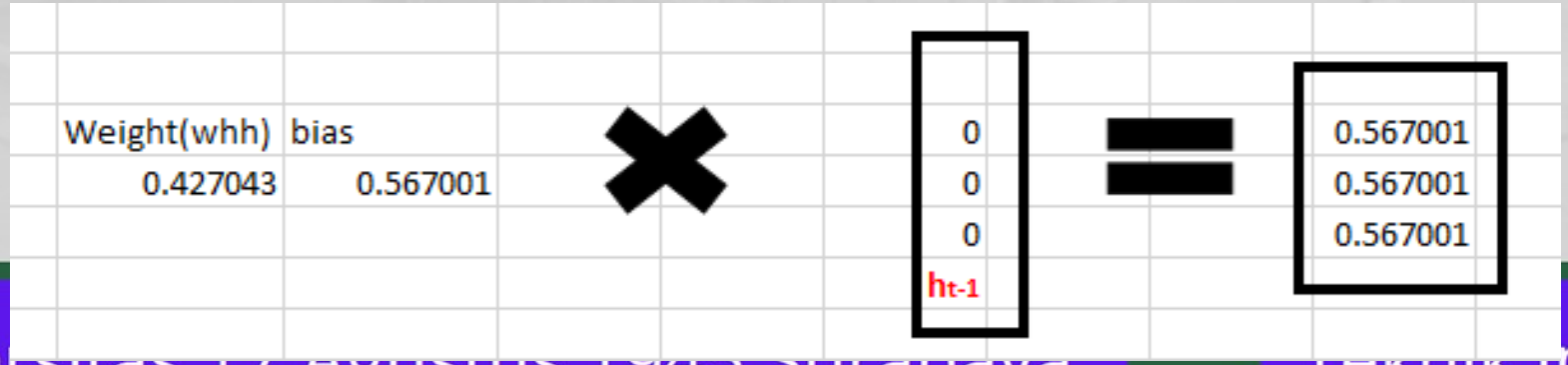
wxh			
0.287027	0.84606	0.572392	0.486813
0.902874	0.871522	0.691079	0.18998
0.537524	0.09224	0.558159	0.491528

1
0
0
0
h

0.287027
0.902874
0.537524

Forward Propagation in a Recurrent Neuron

- **Step 2**: Sekarang pindah ke neuron berulang (recurrent neuron), kita memiliki W_{hh} sebagai bobot yang merupakan matriks 1*1 dan bias yang juga merupakan matriks 1* 0.427043 0.56700
- Untuk huruf "**h**", keadaan sebelumnya adalah [0,0,0] karena tidak ada huruf sebelumnya.
- Jadi untuk menghitung -> (w_{hh}*$\boldsymbol{h}^{t-1}$+bias)

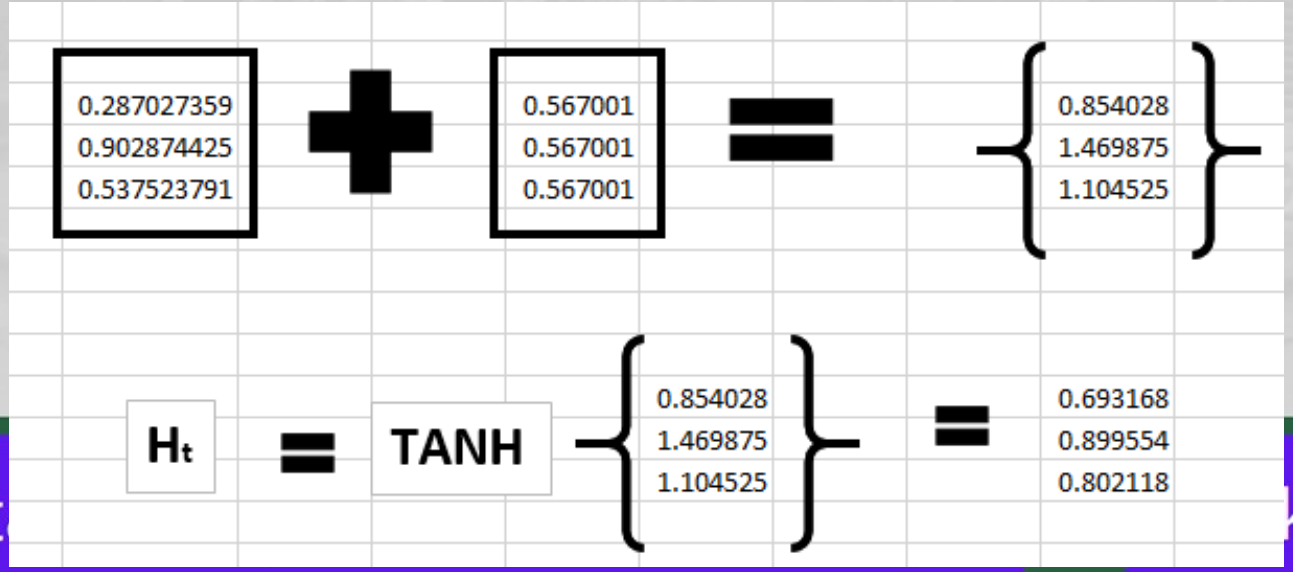


Forward Propagation in a Recurrent Neuron

- **Step 3**: Sekarang kita bisa mendapatkan status saat ini sebagai

$$h^{(t)} = \tanh(w_{hh}h^{(t)} + w_{xh}x^{(t)})$$

- Karena untuk ***h***, tidak ada status tersembunyi sebelumnya, kita menerapkan fungsi *tanh* ke keluaran ini dan mendapatkan status saat ini



Forward Propagation in a Recurrent Neuron

- **Step 4**: Sekarang kita pergi ke keadaan berikutnya. "***e***" sekarang dipasok ke jaringan. Output yang diproses dari $\boldsymbol{h}^{t}$, sekarang menjadi $\boldsymbol{h}^{t-1}$, sedangkan yang *one hot encoded* ***e***, adalah x^{t}. Sekarang mari kita hitung status $\boldsymbol{h}^{t}$ saat ini.
- $(W_h * h^{t-1} + \text{bias})$ akan menjadi

Whh*Ht-1+Bias	=	0.427043	×	0.69316804 0.89955366 0.8021184	+	0.567001	=	0.863013 0.951149 0.90954

Forward Propagation in a Recurrent Neuron

- Wxh*xt akan menjadi

wxh			
0.287027359	0.84606	0.572392	0.486813
0.902874425	0.871522	0.691079	0.18998
0.537523791	0.09224	0.558159	0.491528

×

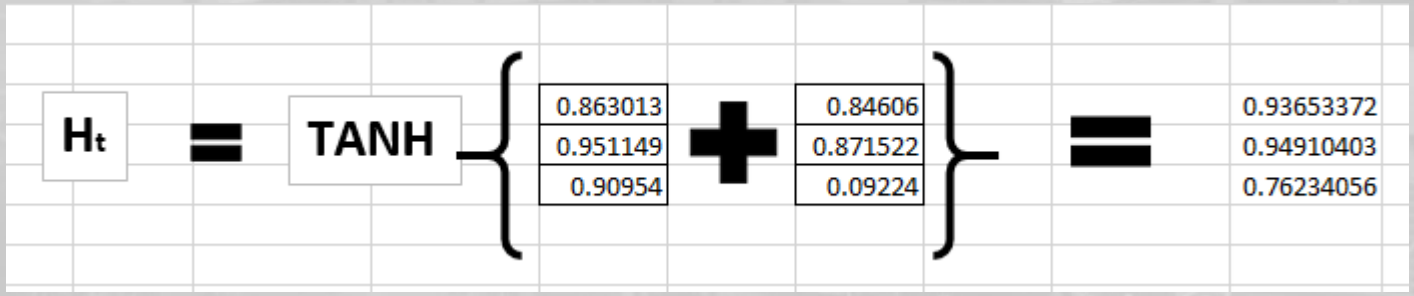
0
1
0
0
e

=

0.84606
0.871522
0.09224

Forward Propagation in a Recurrent Neuron

- **Step 5**: Sekarang menghitung h^t untuk huruf "e",



- Sekarang ini akan menjadi h^{t-1} untuk keadaan selanjutnya dan neuron berulang akan menggunakan ini bersama dengan karakter baru untuk memprediksi yang berikutnya.

Forward Propagation in a Recurrent Neuron

- **Step 6:** Pada setiap keadaan, jaringan saraf berulang akan menghasilkan keluaran juga. Mari kita hitung y^t untuk huruf e.

$$y^{(t)} = w_{hy} h^{(t)}$$

why		
0.37168	0.974829459	0.830034886
0.39141	0.282585823	0.659835709
0.64985	0.09821557	0.334287084
0.91266	0.32581642	0.144630018

×

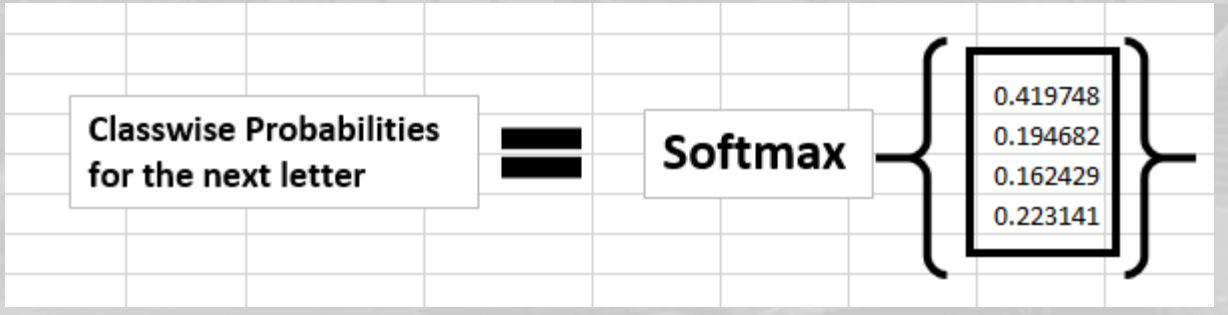
Ht
0.936534
0.949104
0.762341

=

yt
1.90607732
1.13779113
0.95666016
1.27422602

Forward Propagation in a Recurrent Neuron

- **Step 7**: Probabilitas untuk huruf tertentu dari kosakata dapat dihitung dengan menerapkan fungsi softmax. jadi kita akan memiliki softmax(y^t)



Softmax Function:

- Fungsi softmax sering digunakan pada lapisan keluaran RNN untuk tugas klasifikasi multi-kelas.
- Fungsi ini mengubah output jaringan menjadi distribusi probabilitas atas kelas yang mungkin. Rumus untuk fungsi softmax adalah:
  - softmax(x) = e^x / ∑(e^x)

Forward Propagation in a Recurrent Neuron

- Jika kita mengonversi probabilitas ini untuk memahami prediksi, kita melihat bahwa model mengatakan bahwa huruf setelah "**e**" harus menjadi **h**, karena probabilitas tertinggi adalah huruf "**h**".
- Apakah ini berarti kita telah melakukan kesalahan? Tidak, jadi di sini hampir tidak melatih jaringan. Kita baru saja menunjukkan dua huruf. Jadi itu belum belajar apa-apa.

Back propagation in a Recurrent Neural Network(BPTT)

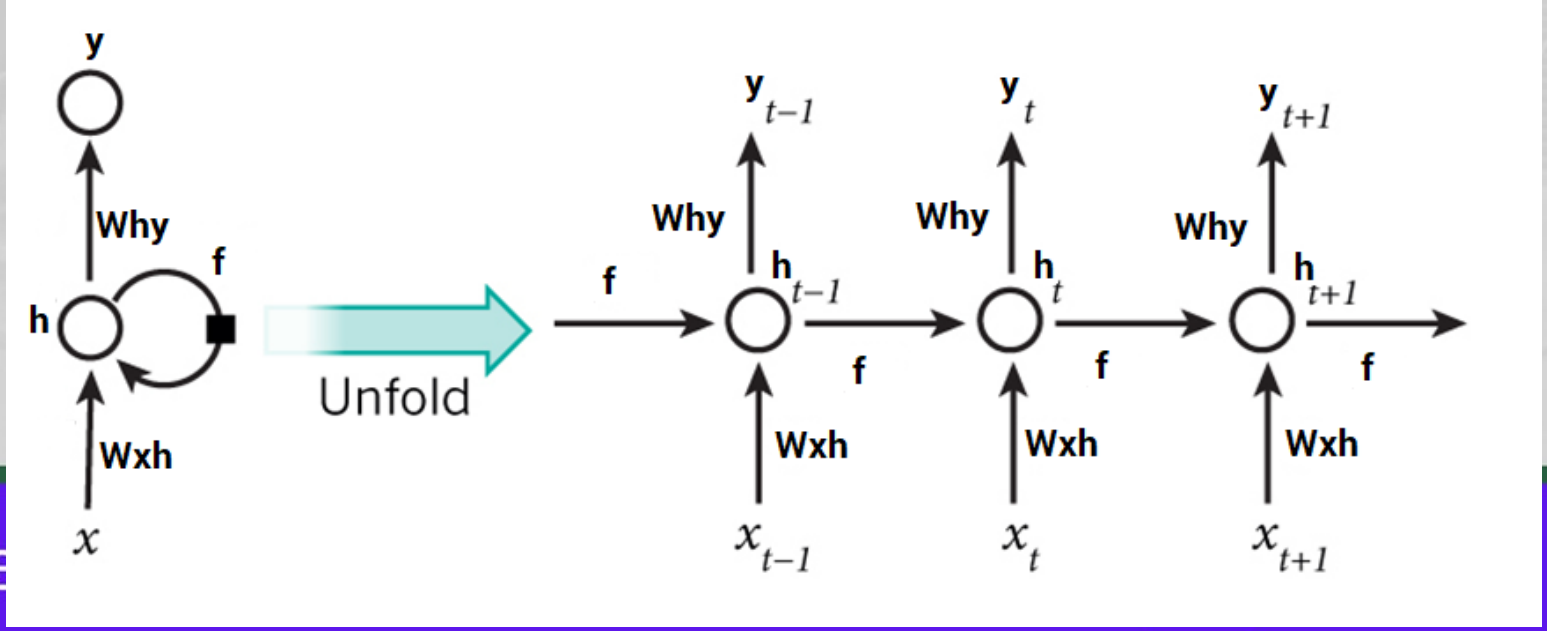
- Sekarang pertanyaan BESAR berikutnya yang kita hadapi adalah bagaimana cara kerja Back propagation dalam kasus Recurrent Neural Network. Bagaimana bobot diperbarui saat ada umpan balik?

Back propagation in a Recurrent Neural Network(BPTT)

- Membayangkan bagaimana bobot akan diperbarui jika terjadi jaringan saraf berulang, mungkin sedikit menantang.
- Jadi untuk memahami dan memvisualisasikan propagasi balik, mari buka loop jaringan pada setiap langkah waktu.
- Dalam RNN kita mungkin atau mungkin tidak memiliki keluaran pada setiap langkah waktu.
- Dalam kasus perambatan maju (feed forward), input masuk dan bergerak maju pada setiap langkah waktu.

Back propagation in a Recurrent Neural Network(BPTT)

- Dalam kasus perambatan mundur (back propagation) dalam kasus ini, kita secara kiasan kembali ke masa lalu untuk mengubah bobot, oleh karena itu kita menyebutnya perambatan balik melalui waktu, **Back propagation through time (BPTT).**



Back propagation in a Recurrent Neural Network(BPTT)

- Dalam kasus RNN, jika y^t adalah nilai prediksi dan $\bar{y}^t$ adalah nilai sebenarnya, error dihitung sebagai cross entropy loss :
- $Et(\bar{y}t, yt) = -\bar{y}t \log(yt)$
- $E(\bar{y}, y) = -\sum \bar{y}t \log(yt)$

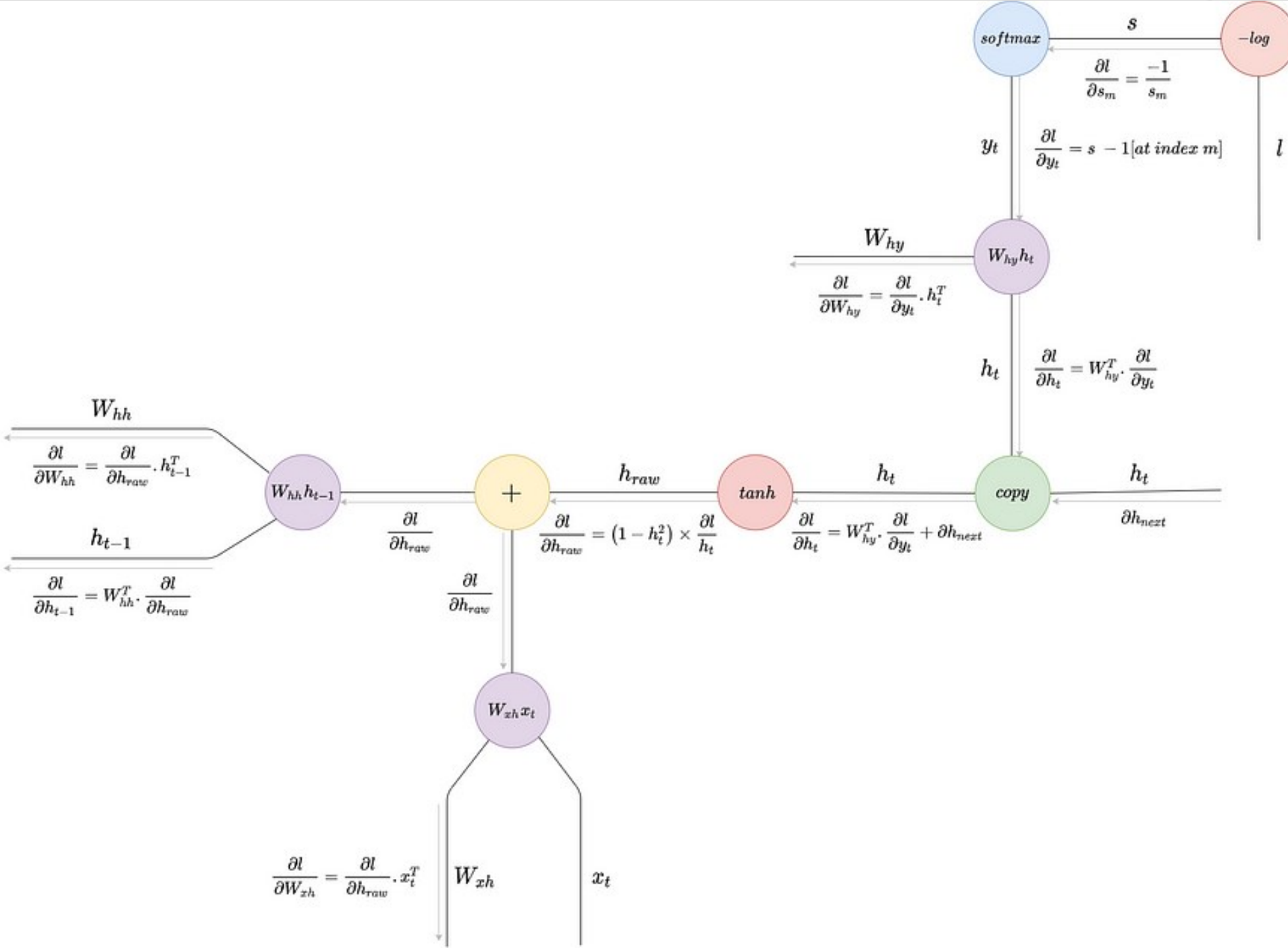
Back propagation in a Recurrent Neural Network(BPTT)

- biasanya memperlakukan urutan penuh (kata) sebagai satu contoh pelatihan, sehingga kesalahan total, jumlah kesalahan pada setiap langkah waktu (karakter).
- Bobot seperti yang bisa kita lihat sama di setiap langkah waktu.

Back propagation in a Recurrent Neural Network(BPTT)

1. Kesalahan lintas entropi pertama kali dihitung menggunakan keluaran saat ini dan keluaran aktual
2. Ingatlah bahwa jaringan tidak dibuka untuk semua langkah waktu
3. Untuk jaringan yang tidak diloop, gradien dihitung untuk setiap langkah waktu sehubungan dengan parameter bobot
4. Sekarang bobotnya sama untuk semua langkah waktu, gradien dapat digabungkan bersama untuk semua langkah waktu
5. Bobot kemudian diperbarui untuk neuron berulang dan lapisan padat

Computational Graphs



Back propagation in a Recurrent Neural Network(BPTT)

- Jaringan yang tidak digulung terlihat seperti jaringan saraf biasa.
- Dan algoritma propagasi balik mirip dengan jaringan saraf biasa, hanya saja kita menggabungkan gradien kesalahan untuk semua langkah waktu.
- Sekarang menurut Anda apa yang mungkin terjadi, jika ada 100 langkah waktu.
- Ini pada dasarnya akan memakan waktu sangat lama bagi jaringan untuk berkumpul karena setelah membuka gulungan jaringan menjadi sangat besar.

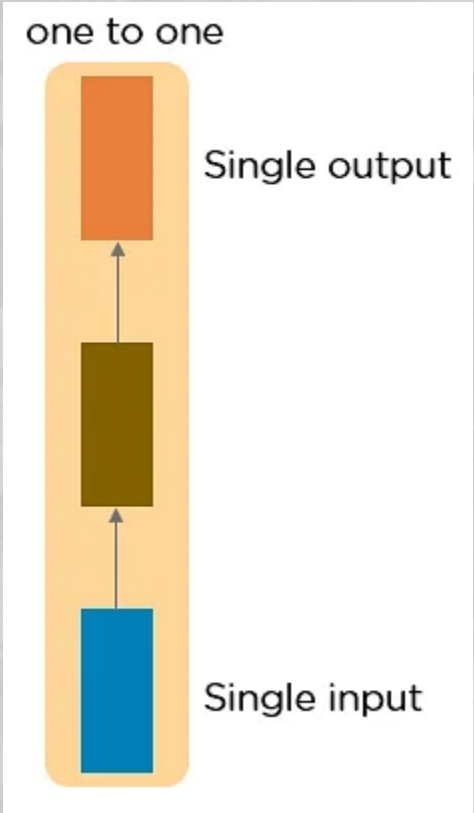
Different types of RNNs

Jenis Recurrent Neural Networks

- Ada empat jenis Recurrent Neural Networks:
  - One to One
  - One to Many
  - Many to One
  - Many to Many

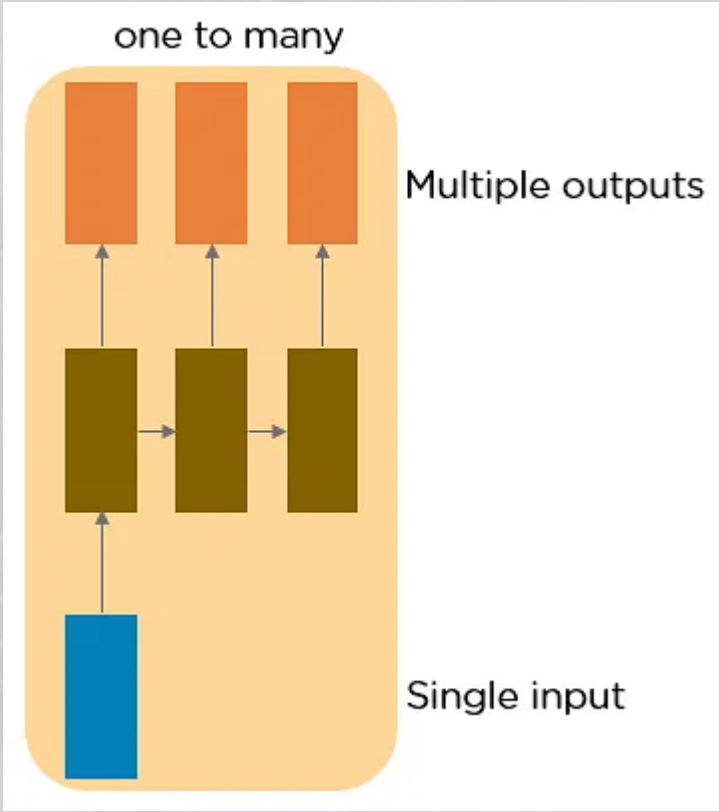
One to One RNN

- Jenis jaringan syaraf ini dikenal sebagai Vanilla Neural Network.
- Ini digunakan untuk masalah pembelajaran mesin umum, yang memiliki input tunggal dan output tunggal.



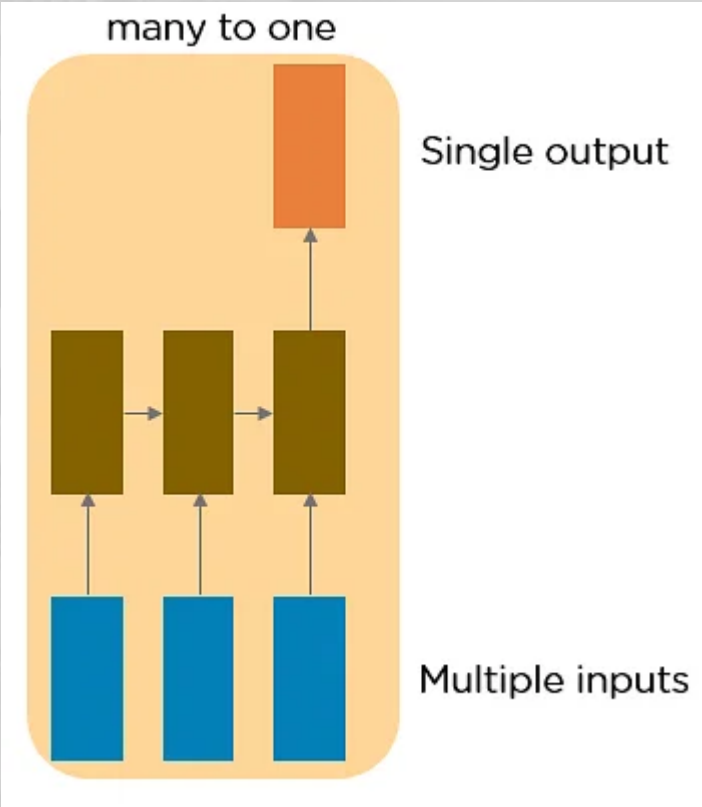
One to Many RNN

- Jenis jaringan saraf ini memiliki input tunggal dan banyak output. Contohnya adalah keterangan gambar



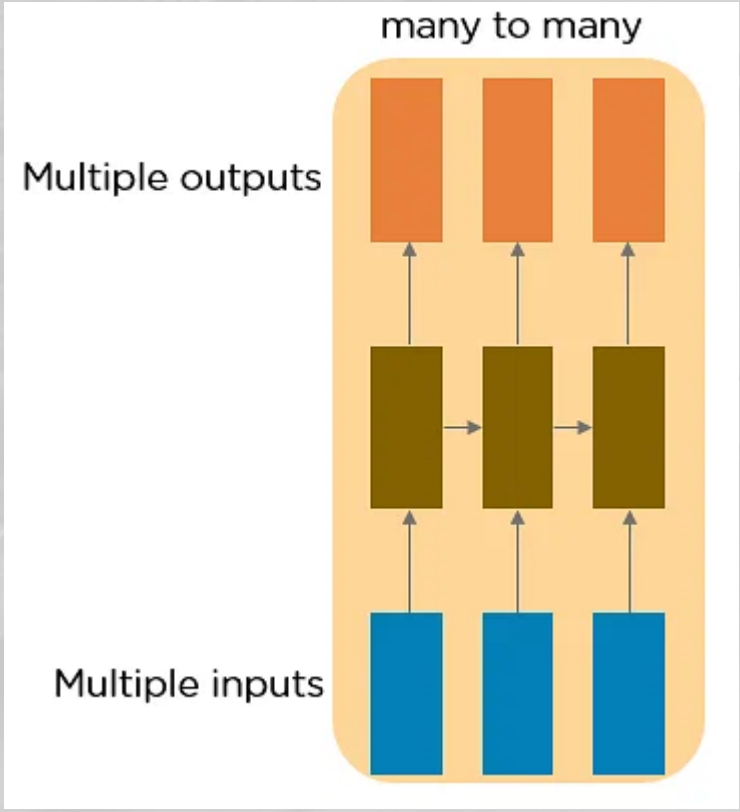
Many to One RNN

- RNN ini mengambil urutan input dan menghasilkan satu output. Analisis sentimen adalah contoh yang baik dari jaringan semacam ini di mana kalimat tertentu dapat diklasifikasikan sebagai ekspresi sentimen positif atau negatif.



Many to Many RNN

- RNN ini mengambil urutan input dan menghasilkan urutan output. Terjemahan mesin adalah salah satu contohnya.



Vanishing gradients with RNNs

Vanishing gradients with RNNs

- RNN bekerja berdasarkan fakta bahwa hasil dari suatu informasi bergantung pada keadaan sebelumnya atau n langkah waktu sebelumnya.
- RNN reguler mungkin mengalami kesulitan dalam mempelajari dependensi jarak jauh.
- Misalnya jika kita memiliki kalimat seperti “Pria yang memakan pizza saya berambut ungu”.
- Dalam hal ini, gambaran rambut ungu adalah untuk laki-laki dan bukan untuk pizza.
- Jadi ini adalah ketergantungan yang panjang (***long dependency***).

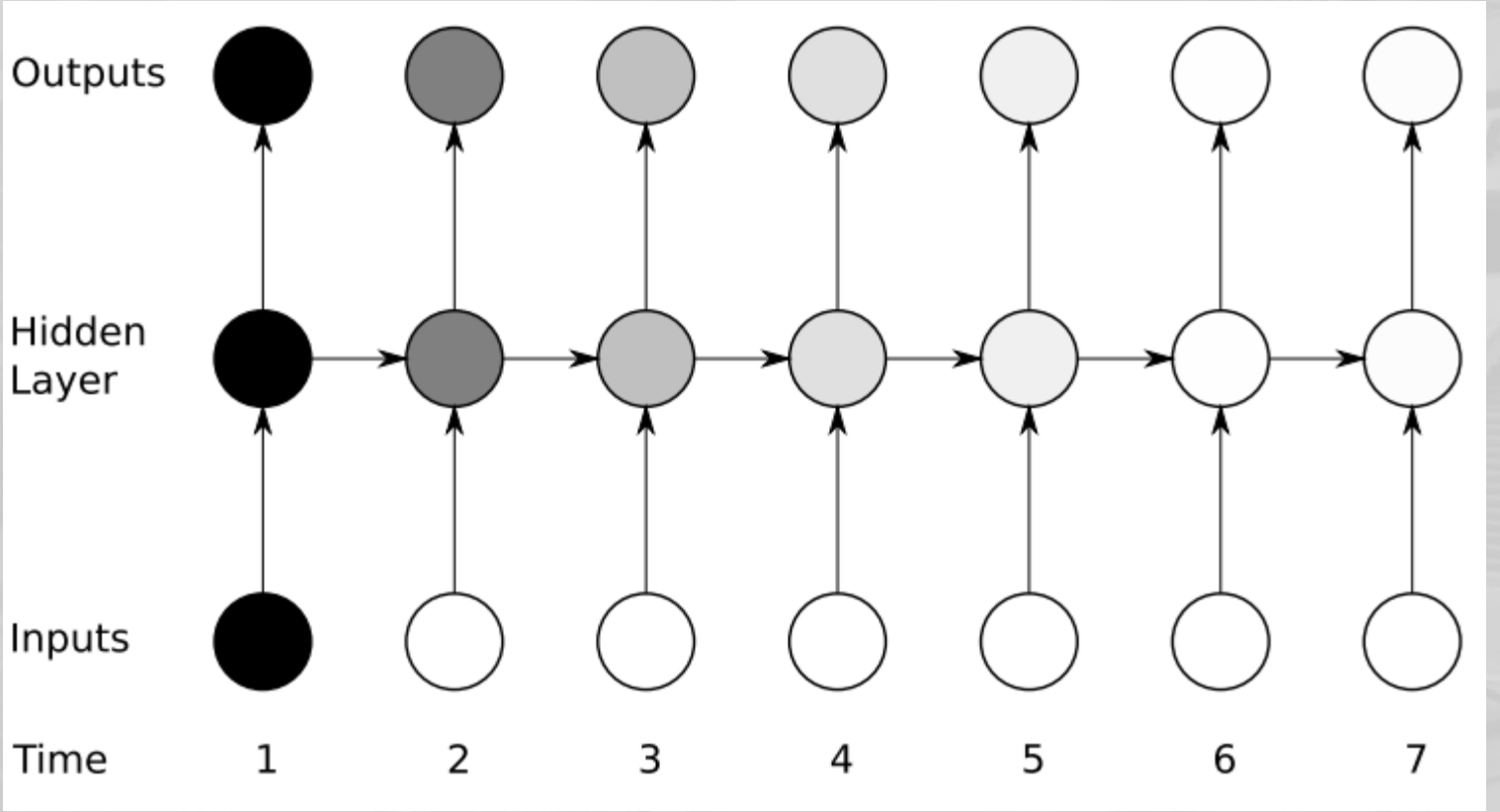
Vanishing gradients with RNNs

- Jika kita mem-backpropagate kesalahan dalam kasus ini, kita perlu menerapkan aturan rantai.
- Untuk menghitung kesalahan setelah langkah ketiga kali sehubungan dengan yang pertama
  - ∂E/∂W = ∂E/∂y3 *∂y3/∂h3 *∂h3/∂y2 *∂y2/∂h1 .. dan ada ketergantungan yang panjang

Vanishing gradients with RNNs

- Di sini kita menerapkan aturan rantai dan jika salah satu dari gradien mendekati 0, semua gradien akan bergegas ke nol secara eksponensial dengan cepat karena perkalian.
- Keadaan seperti itu tidak akan lagi membantu jaringan untuk mempelajari apa pun.
- Ini dikenal sebagai masalah gradien menghilang (*vanishing gradient problem*)

Vanishing gradients with RNNs



Vanishing gradients with RNNs

- Masalah gradien menghilang jauh lebih mengancam dibandingkan dengan masalah gradien meledak (*exploding gradient problem*), di mana gradien menjadi sangat besar karena satu atau beberapa nilai gradien menjadi sangat tinggi.
- Alasan mengapa masalah Vanishing gradient lebih memprihatinkan adalah bahwa masalah gradien yang meledak dapat dengan mudah diselesaikan dengan memotong gradien pada nilai ambang batas yang telah ditentukan.

Vanishing gradients with RNNs

- Untungnya ada cara untuk menangani masalah gradien menghilang juga. Ada arsitektur seperti LSTM (Long Short term memory) dan GRU (Gated Recurrent Units) yang dapat digunakan untuk menangani masalah vanishing gradient.

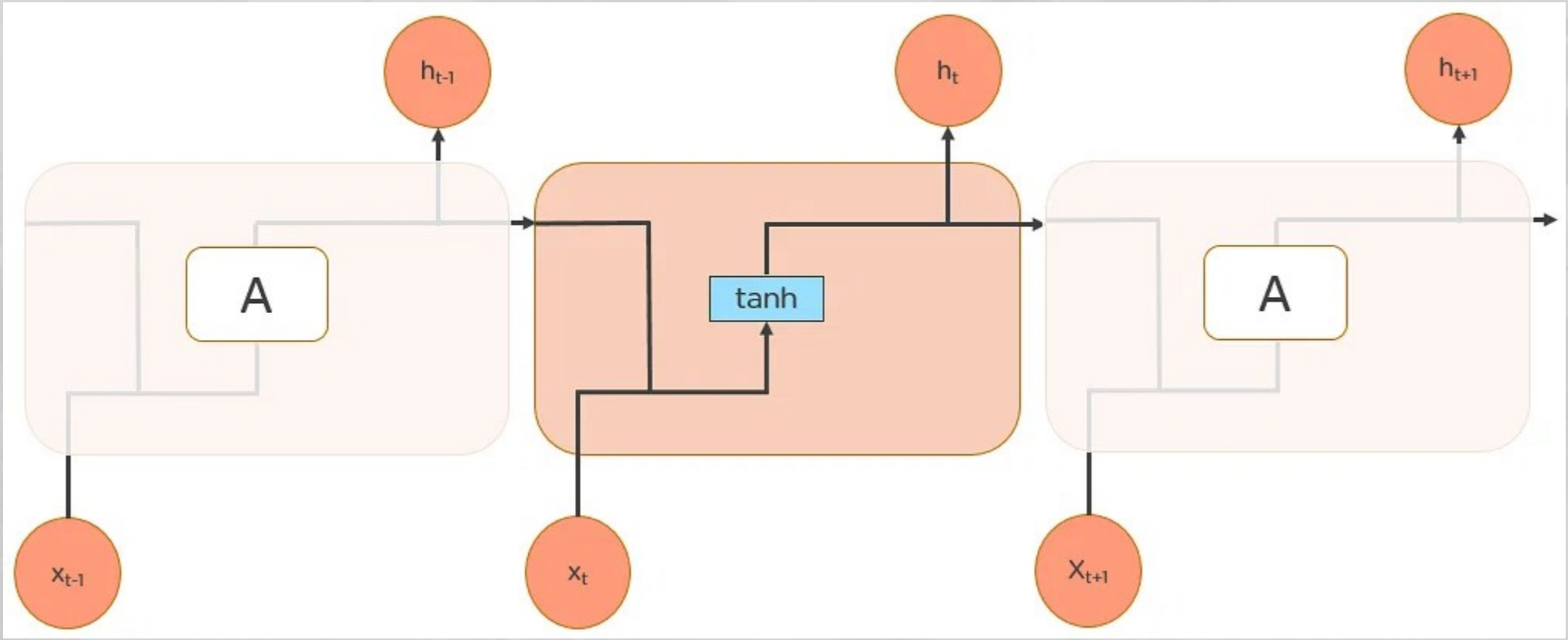
Long Short-Term Memory Networks

Long Short-Term Memory Networks

- LSTM adalah jenis RNN khusus — yang mampu mempelajari ketergantungan jangka panjang dengan mengingat informasi untuk waktu yang lama adalah perilaku default.

- Semua RNN berbentuk rantai modul berulang dari jaringan saraf.
- Dalam RNN standar, modul berulang ini akan memiliki struktur yang sangat sederhana, seperti satu lapisan tanh.

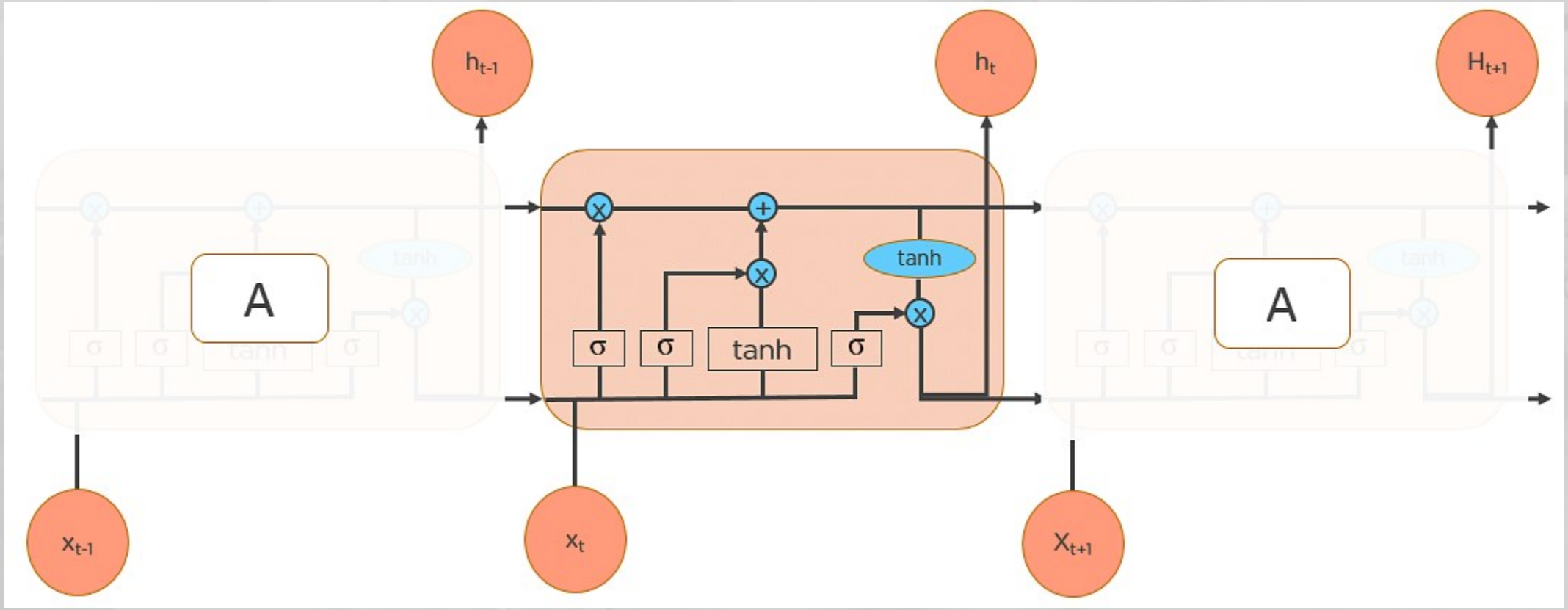
Long Short-Term Memory Networks



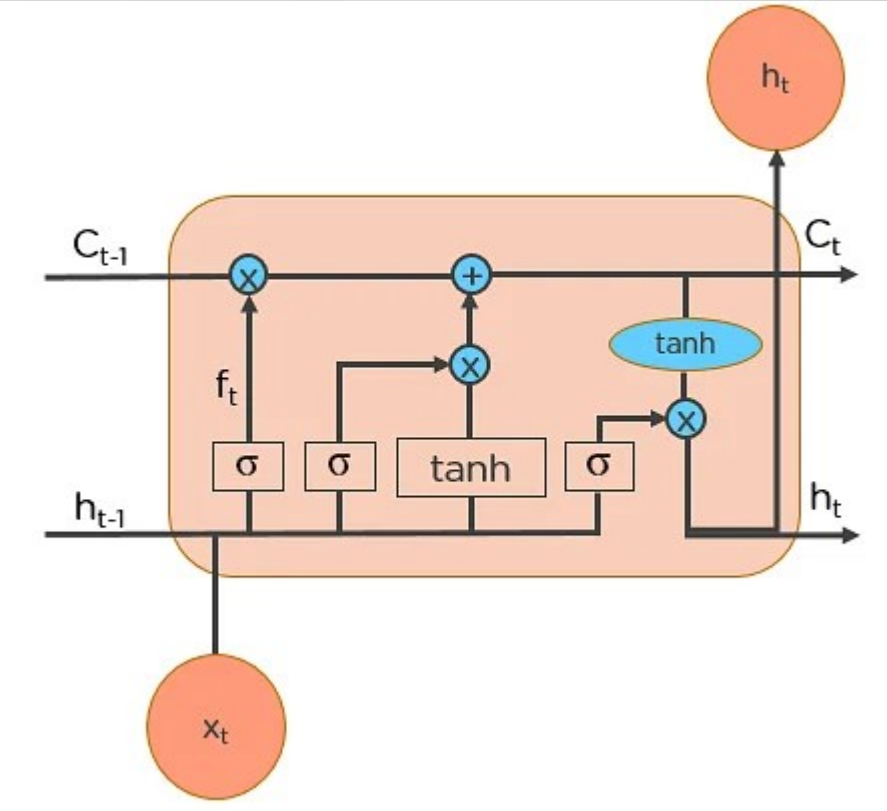
Long Short-Term Memory Networks

- LSTM juga memiliki struktur seperti rantai, tetapi modul berulang memiliki struktur yang sedikit berbeda. Alih-alih memiliki satu lapisan jaringan saraf, empat lapisan yang berinteraksi berkomunikasi dengan luar biasa.

Long Short-Term Memory Networks



Workings of LSTMs in RNN



Workings of LSTMs in RNN

- **Langkah 1**: Tentukan Berapa Banyak Data Masa Lalu yang Harus Diingat
  - Langkah pertama dalam LSTM adalah memutuskan informasi mana yang harus dihilangkan dari sel pada langkah waktu tertentu.
  - Fungsi sigmoid menentukan ini.
  - Itu melihat keadaan sebelumnya (ht-1) bersama dengan input xt saat ini dan menghitung fungsinya.

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

f_t = forget gate
Decides which information to delete that is not important from previous time step

Workings of LSTMs in RNN

- Perhatikan dua kalimat berikut:
  - Output dari h(t-1) menjadi “Alice is good in Physics. John, on the other hand, is good at Chemistry.”
  - Input saat ini di x(t) menjadi “John plays football well. He told me yesterday over the phone that he had served as the captain of his college football team.”
  - Gerbang lupa menyadari mungkin ada perubahan konteks setelah menemui perhentian penuh pertama. Itu dibandingkan dengan kalimat input saat ini di x (t). Kalimat berikutnya berbicara tentang John, sehingga informasi tentang Alice dihapus. Posisi subjek dikosongkan dan ditugaskan kepada John.

Workings of LSTMs in RNN

- **Langkah 2**: Putuskan Berapa Banyak Unit Ini Menambah Keadaan Saat Ini
  - Di lapisan kedua, ada dua bagian.
  - Salah satunya adalah fungsi sigmoid, dan yang lainnya adalah fungsi tanh.
  - Dalam fungsi sigmoid, ia memutuskan nilai mana yang akan dilewati (0 atau 1). fungsi tanh memberi bobot pada nilai-nilai yang diteruskan, menentukan tingkat kepentingannya (-1 hingga 1).

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

i_t = input gate
Determines which information to let through based on its significance in the current time step

Workings of LSTMs in RNN

- Dengan input saat ini di x(t), gerbang input menganalisis informasi penting — John bermain sepak bola, dan fakta bahwa dia adalah kapten tim kampusnya adalah penting.
- "*He told me yesterday over the phone*" kurang penting; karenanya dilupakan. Proses penambahan beberapa informasi baru ini dapat dilakukan melalui input gate.

Workings of LSTMs in RNN

- **Langkah 3**: Putuskan Bagian Apa dari Status Sel Saat Ini yang Menghasilkan Keluaran
  - Langkah ketiga adalah memutuskan apa yang akan dihasilkan.
  - Pertama, kita menjalankan layer sigmoid, yang menentukan bagian mana dari status sel yang membuatnya menjadi output.
  - Kemudian, kami menempatkan status sel melalui tanh untuk mendorong nilai antara -1 dan 1 dan mengalikannya dengan keluaran gerbang sigmoid.

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

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

o_t = output gate
Allows the passed in information to impact the output in the current time step

Workings of LSTMs in RNN

- Mari pertimbangkan contoh ini untuk memprediksi kata berikutnya dalam kalimat: *"John played tremendously well against the opponent and won for his team. For his contributions, brave ____ was awarded player of the match."*
- Mungkin ada banyak pilihan untuk ruang kosong. Input "Brave" saat ini adalah kata sifat, dan kata sifat menggambarkan kata benda. Jadi, "John" bisa menjadi keluaran terbaik setelah pemberani.

References

- Goodfellow, I; Bengio,Y.; Courville, A (2016). Deep Learning. MIT Press pp: 224 – 270
- https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn
- https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/
- https://towardsdatascience.com/backpropagation-in-rnn-explained-bdf853b4e1c2