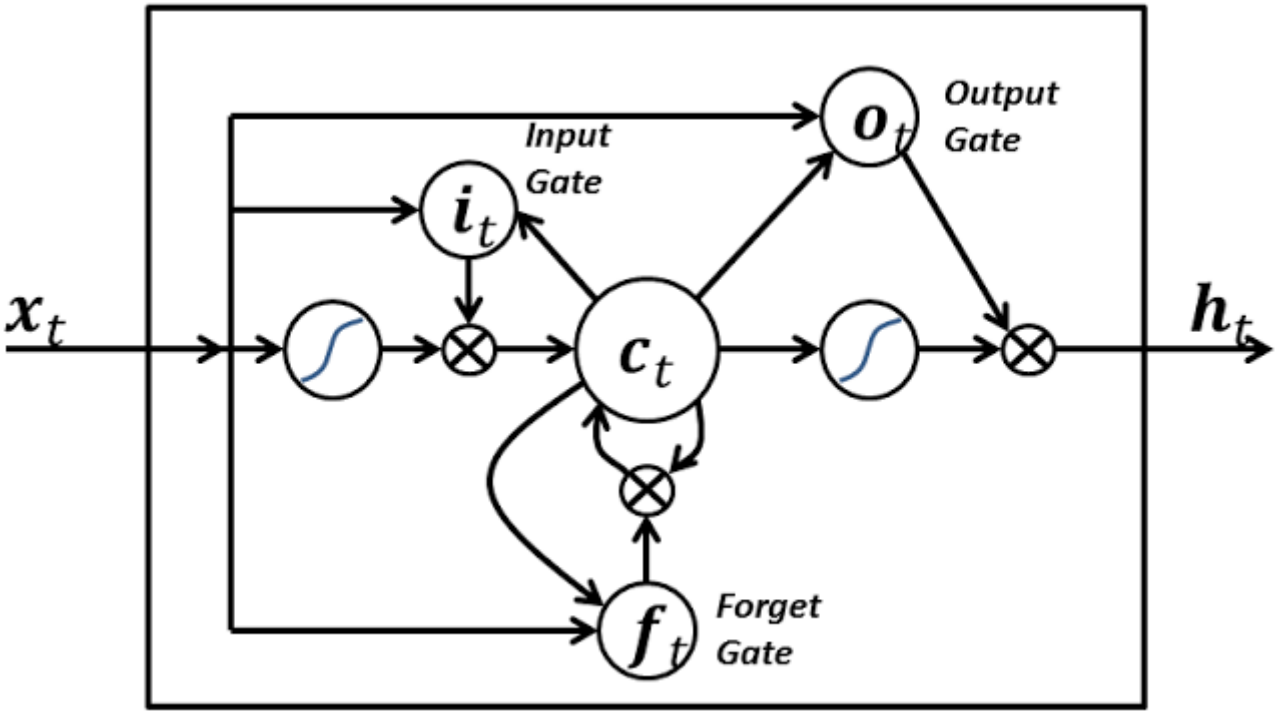
PENGENALAN LSTM (LONG SHORT TERM MEMORY)



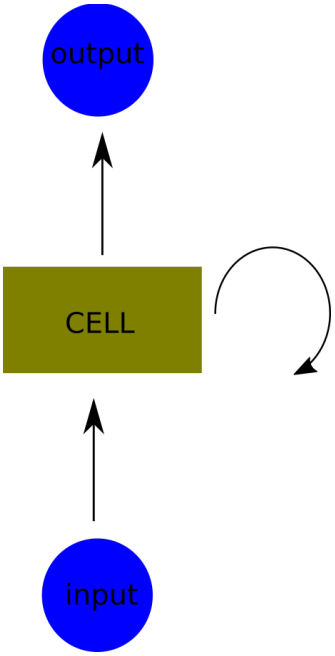
By

Muhammad Ryan

RNN

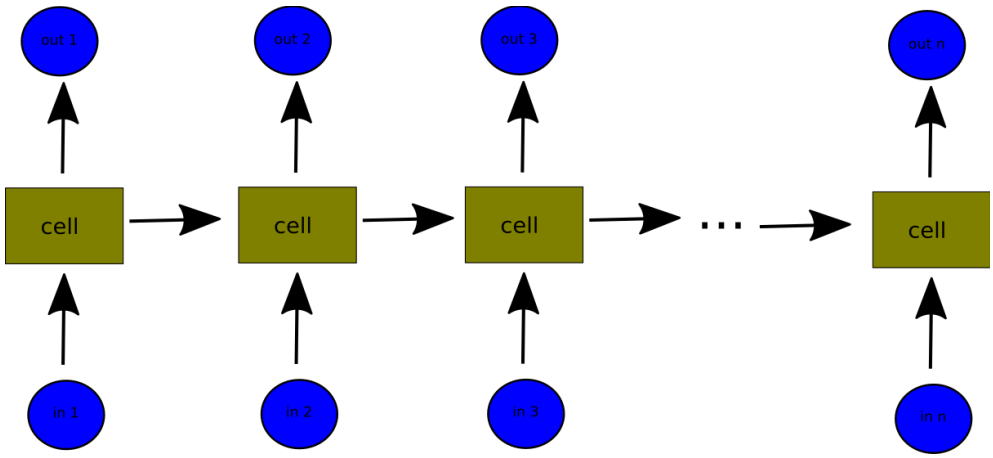
Sebelum lebih jauh membahas LSTM, kita harus membahas terlebih dahulu apa itu RNN. Mengapa harus membahas ini terlebih dahulu? Karena LSTM adalah varian dari RNN.

RNN (Recurrent Neural Network) adalah salah satu jenis ANN (Artificial Neural Network) yang mana pada bagian “inti”nya (yang disebut sebagai cell) terjadi loop. Artinya output dari cell ini akan menjadi inputnya lagi.



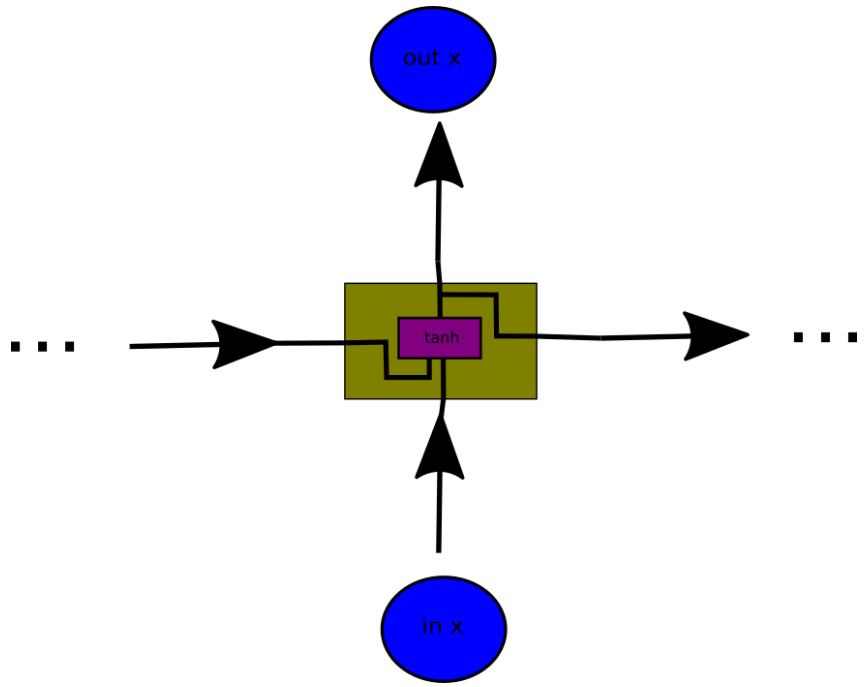
Gambar 1: Struktur dari RNN secara umum.

Jika struktur pada Gambar 1 ini di uraikan, maka struktur dari RNN ini akan lebar tengahnya selebar panjang pola data yang ingin “dipelajari” oleh RNN seperti pada Gambar 2. Ya, RNN sepertinya di desain khusus untuk menghandle data berurutan (sequence data).



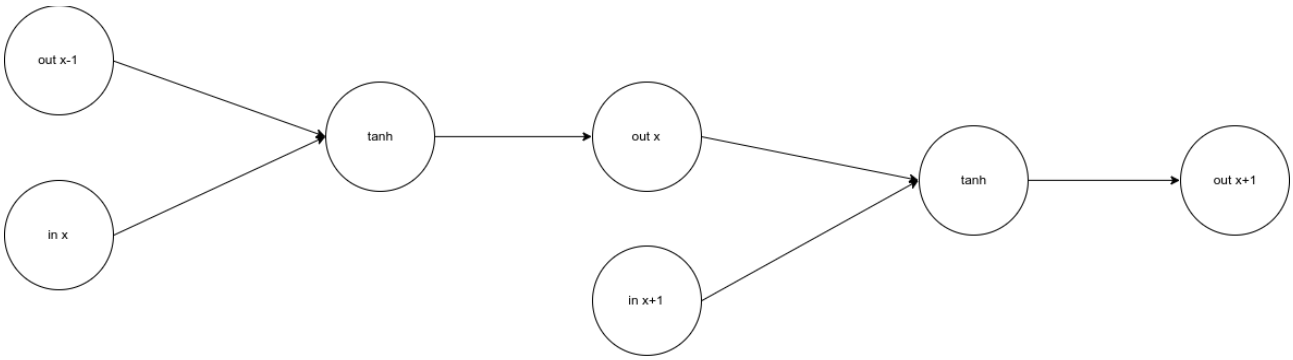
Gambar 2: Struktur RNN setelah di uraikan. Terlihat output dari RNN adalah sepanjang cell atau sepanjang data berurut yang menjadi inputnya.

Semua komputasi pada RNN di lakukan pada cell. Pada RNN yang paling sederhana, cell hanya berisi 1 neuron dengan fungsi aktivasi yang biasanya dipakai adalah tanh atau sigmoid.



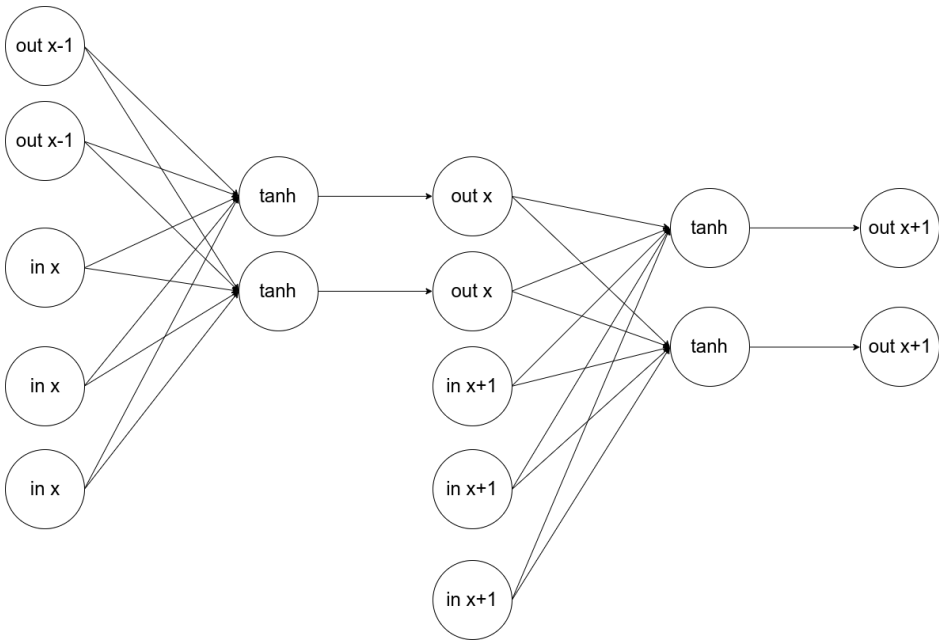
Gambar 3: Isi dari cell RNN sederhana yaitu 1 neuron dengan fungsi aktivasi tanh. Neuron ini mengambil input keluaran cell urutan x-1 dan data berurut urutan x yang kemudian outputnya sebagai out x dan juga sebagai input untuk cell selanjutnya (x+1).

Jika di ilustrasikan sebagai struktur ANN biasa, komputasi dalam cell pada Gambar 3 akan terlihat seperti Gambar 4 dibawah ini.



Gambar 4: Struktur RNN dengan cell berisi 1 neuron yang memakai fungsi aktivasi tanh jika digambarkan dengan bagan yang biasa digunakan untuk menggambarkan struktur ANN.

Ini adalah skema paling sederhana dimana RNN ini hanya mempunyai 1 hidden neuron atau hidden state dan data berurutnya juga hanya untuk 1 variabel. Bagaimana jika misalnya RNN yang di inginkan punya 2 hidden neuron dan ada 3 variabel untuk data berurutnya? Maka akan seperti Gambar 5 di bawah ini.

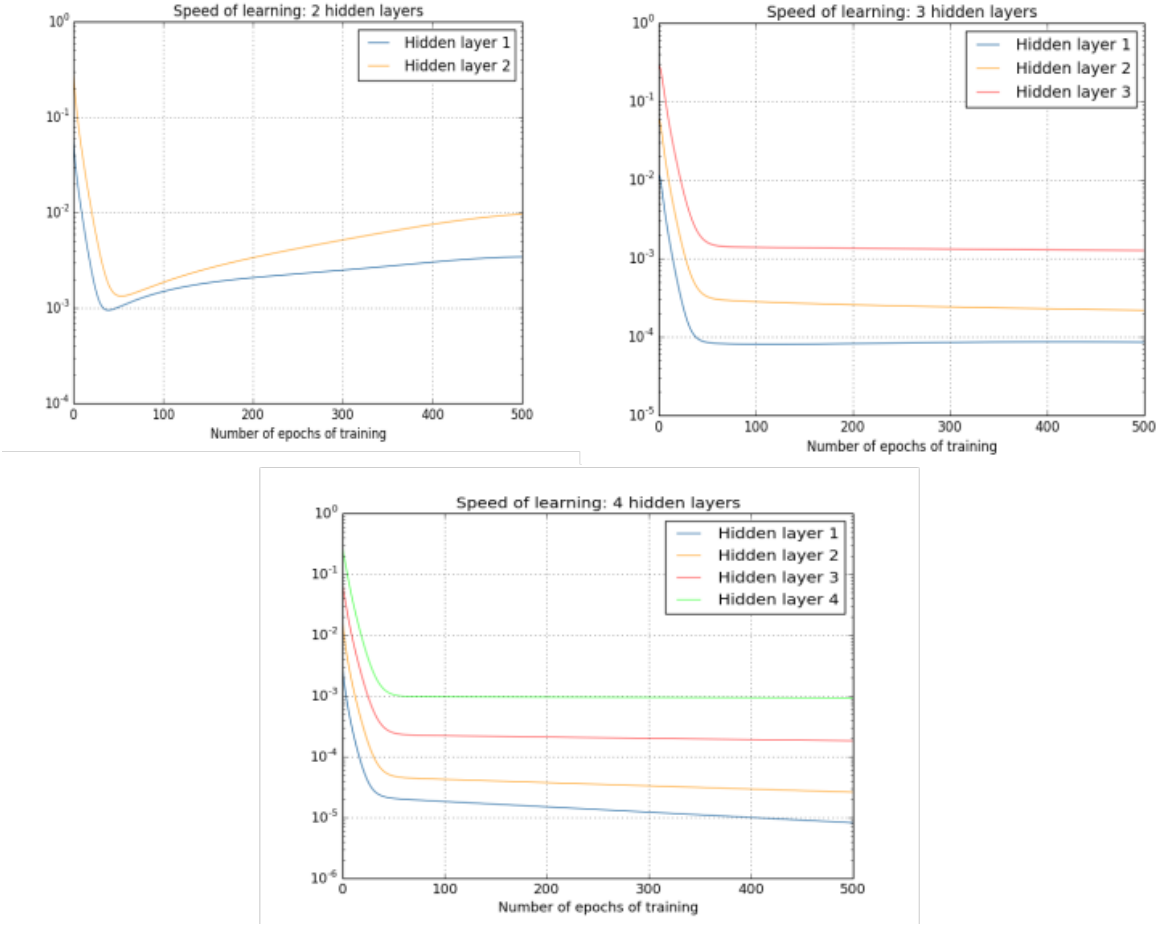


Gambar 5: Struktur RNN dengan 2 hidden neuron dan 3 variabel input data berurut. Total semua input pada 1 urutan (ada 3 variabel) dan semua output (ada 2 sesuai jumlah hidden neuron) menjadi input dari 2 hidden neuron tersebut. Terlihat banyaknya output dari RNN sama dengan hidden neuron yang dideklarasikan.

Sehingga size output dari RNN dengan panjang data beurutnya **n**, hidden neuronnya **t** adalah **n x t**. Banyaknya variabel pada data berurut tidak mempengaruhi size output RNN.

Akan tetapi seperti DNN (Deep Neural Network) pada umumnya, RNN sederhana atau varian lainnya mengalami masalah *vanishing gradient* seiring bertambahnya panjang data berurut yang ingin ditraining kepada RNN tersebut. Vanishing gradient adalah situasi dimana nilai gradient yang digunakan untuk mengupdate weight pada neuron pada sesi training ANN bernilai 0 atau mendekati

0. Ini dikarenkan penggunaan fungsi aktivasi sigmoid atau tanh yang memetakan input menjadi output yang jarak antar marginnya kecil. Ini memengaruhi sensitivitas layer neuron dalam mengupdate weightnya, jadi saat dimana errornya besar dan harus mengupdate weight secara signifikan juga, yang terjadi malah nilai weight yang diupdate tidak signifikan. Semakin banyak layer yang ditumpuk, semakin lambat pembelajaran (mengupdate weight) yang dilakukan oleh neuron pada ANN.

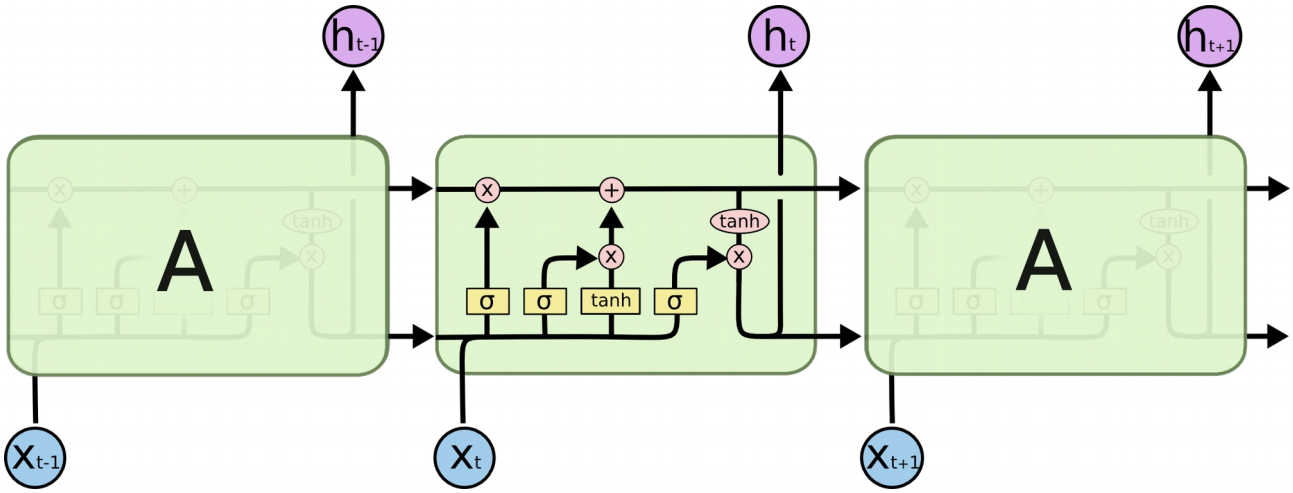


Gambar 6: Contoh ilustrasi kecepatan belajar seiring bertambahnya layer untuk hidden layer. Terlihat semakin banyak layer yang ditumpuk semakin lambat pembelajaran yang terjadi pada neuron saat training. Kecepatan belajar di sini mengacu pada besar kecilnya update weigth pada neuron di hidden layer. Pada RNN banyaknya tumpukan hidden layer mengacu kepada panjangnya cell berantai atau panjangnya data berurut yang ingin di training ke RNN.

Ini menyimpulkan bahwa RNN biasa tidak cocok untuk mempelajari pola data berurut yang panjang. Lalu apa solusinya? **Solusinya adalah menggunakan LSTM** yang menjadi topik utama pembahasan di sini.

LSTM

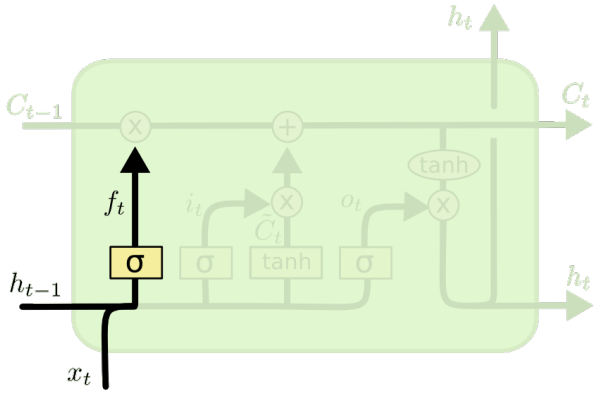
Di penjelasan RNN sebelumnya telah di perlihatkan RNN sederhana dengan cell yang hanya berisi 1 layer neuron dengan fungsi aktivasi tanh. Pada LSTM, isi cell menjadi lebih kompleks dari yang sekedar 1 layer neuron tadi dan inilah yang menjadikan LSTM bisa mempelajari pola panjang dari data berurut karena situasi vanishing gradient dicegah (saya masih belum tahu sebatas mana LSTM bisa mencegah situasi itu). Yang membedakan LSTM dengan RNN vanila tadi hanya isi cellnya, selebihnya secara prinsip sama.



Gambar 7: Struktur LSTM. persegi panjang kuning melambangkan layer neuron (jadi bisa ada 2 atau lebih neuron pada masing-masing kotak kuning tersebut jika hidden neuron yang dideklarasikan adalah 2 atau lebih) sedangkan bulatan berwarna merah muda melambangkan operasi element-wise. Panah hitam berkelok-kelok ini melambangkan aliran informasi di dalam cell dan antar cell maupun keluar cell (output h).

Bisa dilihat, jadi cell LSTM punya 2 hasil keluaran, yang 1 adalah keluaran sebenarnya yang diteruskan lagi ke cell selanjutnya dan menjadi output dari cell itu dan satunya lagi adalah *cell state* (C_t). Bisa dibilang ide basic dari LSTM adalah mengupdate cell state dengan operasi-operasi element-wise di setiap cell LSTM.

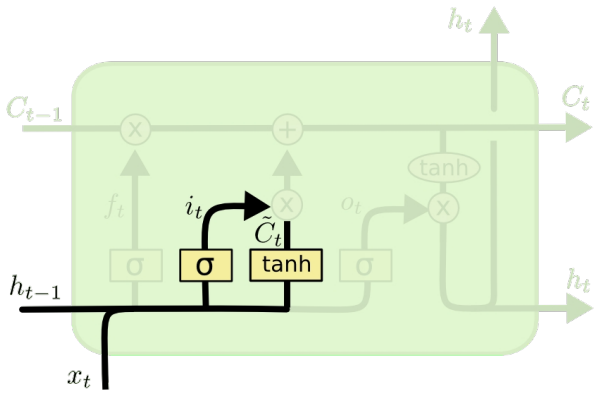
Ke-4 layer neuron tersebut biasa disebut gate. Yang paling kiri (layer neuron dengan fungsi aktivasi sigmoid diperlihatkan pada Gambar 8) adalah *forget gate*, yaitu gate yang menentukan apakah informasi dari input X_t dan output h_{t-1} pantas lewat atau tidak. Pada outputnya, mendekati 1 artinya "biarkan lewat" sedangkan mendekati 0 artinya "lupakan/abaikan informasi ini". Bisa dilihat output dari gate ini akan "di adu" dengan cell state melalu operasi perkalian element-wise. Jadi, "biarkan lewat" atau "lupakan/abaikan informasi ini" mengacu pada informasi/aliran data pada cell state. Yang perlu dicatat adalah, saat melewati gate ini maupun gate lainnya, input yang awalnya adalah vektor sepanjang output dari cell sebelumnya + input baru dari data berurut menjadi sepanjang hidden neuron yang dideklarasikan (coba lihat lagi bagian penjelasan RNN tentang size dari output RNN) sehingga operasi element-wise (bulatan warna merah muda) menjadi memungkinkan. Menurut beberapa referensi, forget gate adalah kunci dari kesuksesan LSTM mencegah vanishing gradient atau kasus sebaliknya exploding gradient (pembelajaran atau update weight menjadi terlalu cepat atau besar).



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

Gambar 8: Alur informasi pada forget gate.

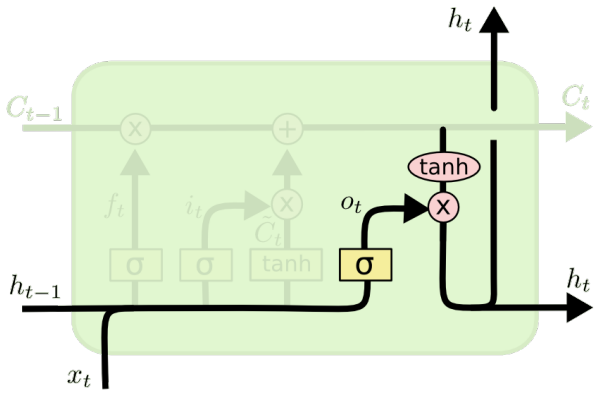
Kemudian disebelahnya ada *input gate* (layer neuron dengan fungsi aktivasi sigmoid dan sebelahnya yang mempunyai fungsi aktivasi tanh). Gate ini menentukan bagian mana yang akan di update. Bisa dilihat pada Gambar 9, setelah gate operasi penjumlahan secara element-wise. Yup, secara harfiah cell state di update dengan output dari gate ini.



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

Gambar 9: Alur informasi yang melewati input gate

Lalu terakhir adalah *output gate* (layer neuron dengan fungsi aktivasi sigmoid di paling kanan pada jajaran layer neuron). Ouput gate ini tidak berkontribusi untuk cell state, tapi gate inilah yang membedakan cell state dan output yang sebenarnya.



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

Gambar 10: Alur informasi yang melewati output gate. Output gate ini tidak berkontribusi untuk perkembangan cell state.

Sisanya yang tidak dijelaskan bisa dilihat alurnya pada Gambar 7, sepertinya cukup sederhana untuk dipahami. Seperti yang sudah dibahas sebelumnya, selain isi cell LSTM yang lebih kompleks, LSTM tetaplah salah satu varian dari RNN sehingga yang lainnya dari LSTM secara prinsip sama dengan RNN, size output dari LSTM adalah [panjang data berurut x banyaknya hidden neuron yang dideklarasikan].

Catatan Akhir

Dokumen ini saya buat karena saya berasumsi dengan menjelaskan segala sesuatunya dalam bentuk tulisan, di situ sebenarnya kita sedang menguji pemahaman kita sendiri yang pada akhirnya kita menjadi tahu kalau pemahaman saya sendiri mungkin masih ada yang kurang dan ini terbukti. Sesudah menyelesaikan dokumen ini pemahaman saya tentang LSTM dan RNN menjadi lebih baik dibandingkan sebelumnya saya menulis ini. Mengapa tidak ditulis di blog saja seperti biasanya? Saya sedang mencoba cara baru dan sepertinya untuk sementara cara ini lebih nyaman dan lebih banyak manfaatnya (pemahaman setelah menulis ini menjadi lebih mantap mungkin karena effort yang saya lakukan untuk menjelaskan lebih besar daripada saat menulis di blog). Selain itu, pastinya diharapkan yang membaca ini menjadi lebih paham tentang isi materi yang dibawakan.

Untuk memahami materi pada dokumen ini tentu saja harus paham dulu tentang ANN secara mendasar (mungkin informasi/peringatan ini lebih cocok jika ditaruh di bagian depan?). Penjelasan pada dokumen ini tidak menyertakan koding walaupun sebenarnya saya sudah membuatnya karena dirasa kurang perlu. Koding bagus untuk menjelaskan tentang implementasinya sedangkan untuk pengenalan tidak perlu karena tidak menjelaskan apa-apa mengenai subjek yang ingin dikenalkan, begitulah cara saya belajar.

Setelah menulis ini, mungkin juga saya akan menulis tentang pengenalan ANN mendasar ataupun yang lainnya di kesempatan mendatang nanti.

Referensi

http://colah.github.io/posts/2015-08-Understanding-LSTMs/, di akses pada 29 Agustus 2017

https://stats.stackexchange.com/questions/140537/why-do-rnns-have-a-tendency-to-suffer-from-vanishing-exploding-gradient, di akses pada 30 Agustus 2017

http://neuralnetworksanddeeplearning.com/chap5.html, di akses pada 30 Agustus 2017

https://en.wikipedia.org/wiki/Long_short-term_memory, di akses pad 30 Agustus 2017 (gambar sampul depan)