Deep Learning

LSTM & GRU

Smartr

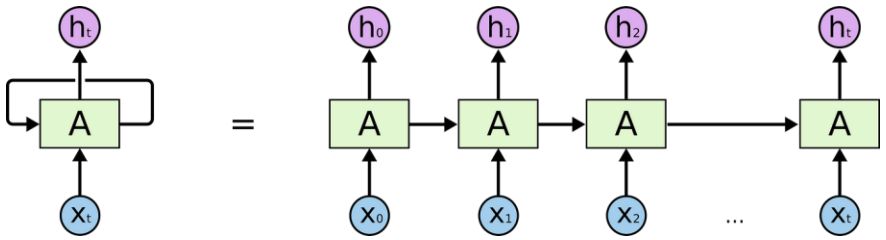
LSTM

Long Short-Term Memory (LSTM) är en typ av RNN som är designad för att komma ihåg saker över en längre tid. Vad som skrivs tidigt i en mening kan ah stor påverkan på vad något betyder senare.

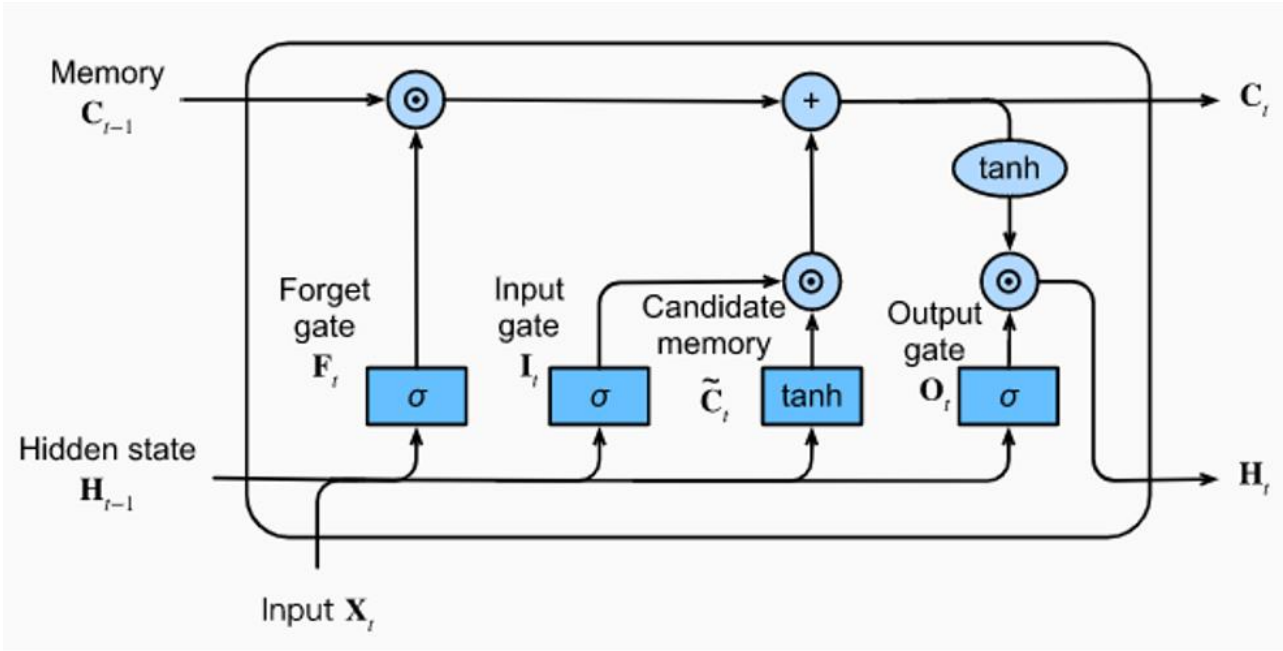
Genom detta minne får LSTMs bättre förståelse

av texter.

LSTM



RNN



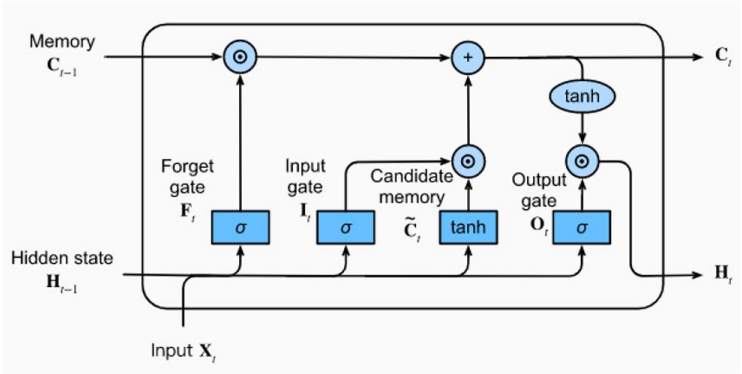
LSTM

Smartr

LSTM

En LSTM cell innehåller

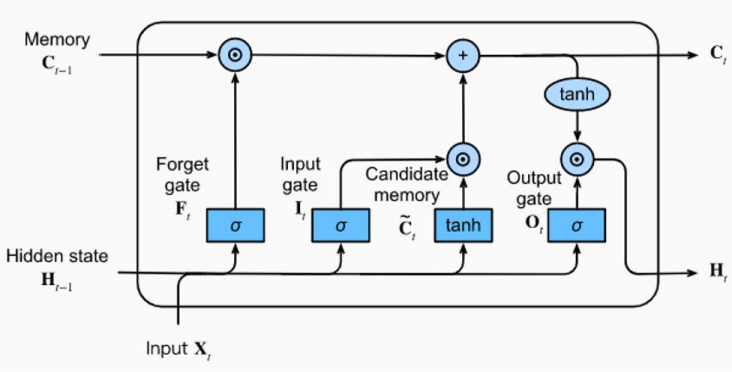
- Memory (Cell state)- Cellens minne
- Cendidate memory - innehåller information från input som kan vara intressant, men allt hamnar inte i cell minnet.
- Forget gate – vilken information ska kastas bort.
- Input gate – vilken ny information ska läggas till i cell statet.
- Output gate – vilken information ska användas för att beräkna output.



LSTM

Informationsflöde

- Tre inputs, **xt** (ord i en mening) och **ht-1** (föregeående hidden state) och föregående cell state (**ct-1**)
- Forget gate bestämmer vilken information som ska sparas i cellen. Den föregeående hidden state och input för att generera en vektor som innehåller värden mellan 0 och 1. 0 innebär att glömma information helt, 1 innebär att spara all ifnormation.
- Input gate bestämmer vilka värden som ska bli uppdaterade i cellen.
- Genom att kombinera information från forget gate och input gate vet cellen vilken information som ska bort och vilken information som ska adderas/uppdateras.
- Slutligen skapas en output och ett nytt hidden state som kan användas av nästa cell.

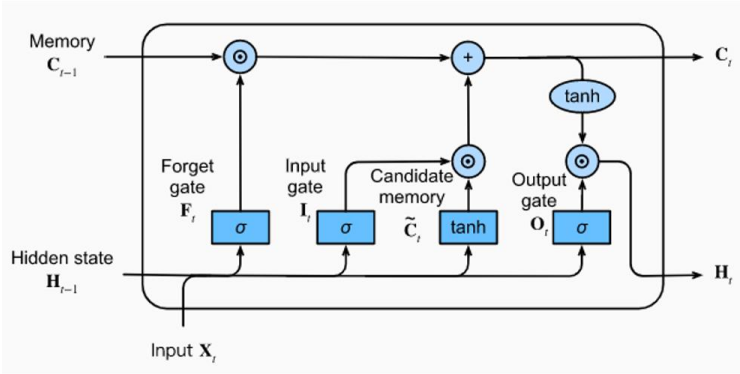


Smartr

LSTM

Informationsflöde

I LSTM kan vi se Memory (cell state) som det långa minnet och hidden state som det korta minnet.



LSTM

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

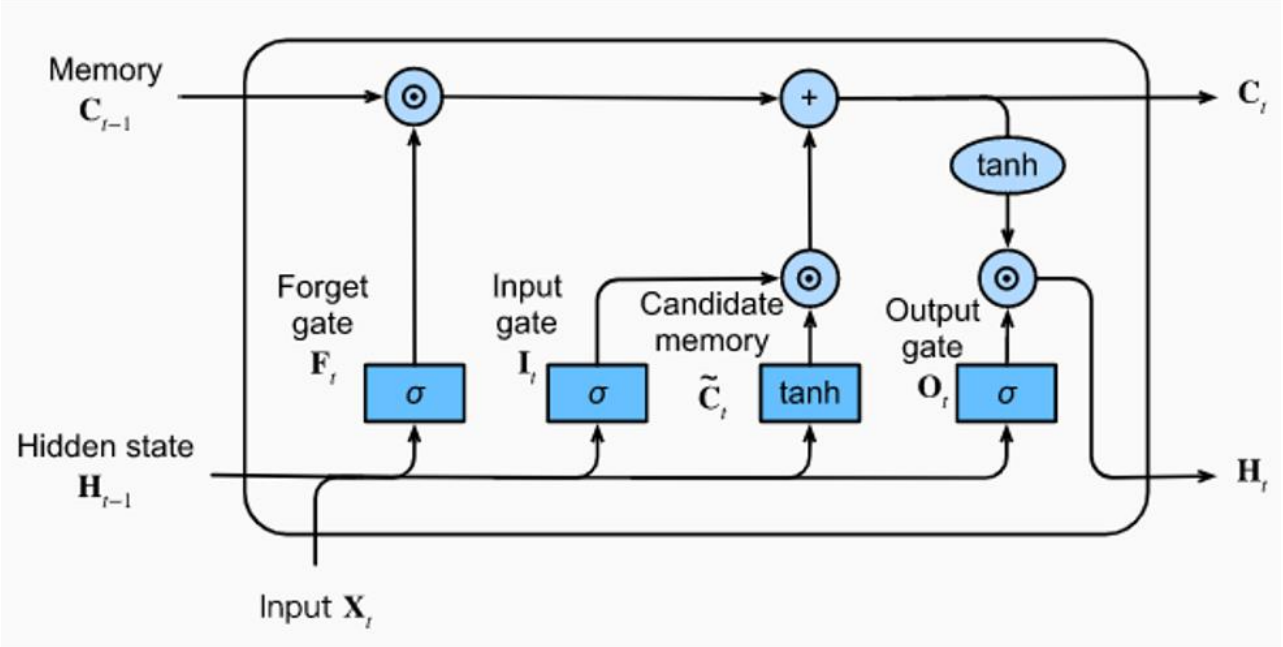
Input Gate: $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$

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

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

Output Gate: $o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$

Hidden State: $h_t = o_t * \tanh(C_t)$

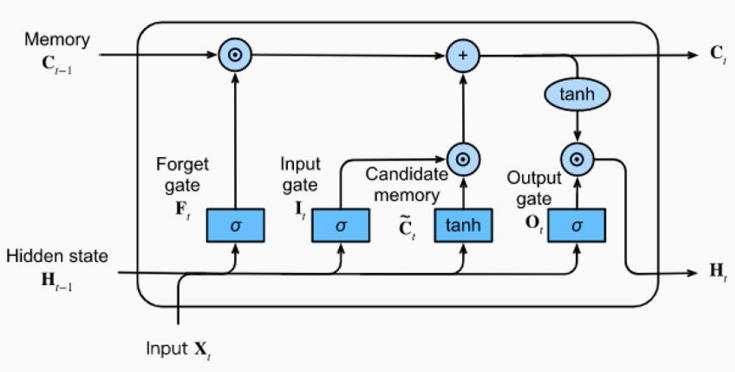


Smartr

LSTM

Fördelar och nackdelar

- Hanterar långa beroenden, i tid, ord, meningar, osv.
  - Detta är den stora fördelen med LSTM (och det är verkligen en stor fördel).
- Ökad komplexitet
  - Med större kraft kommer ofta ökad komplexitet och därmed längre träningstider (och prediktionstider)
- Känslig för overfitting
  - Mer komplexa nätverk kan hålla mer information och därmed finns en risk för overfitting
  - Behöver ofta använda regulariseringsmetoder (typ dropout).

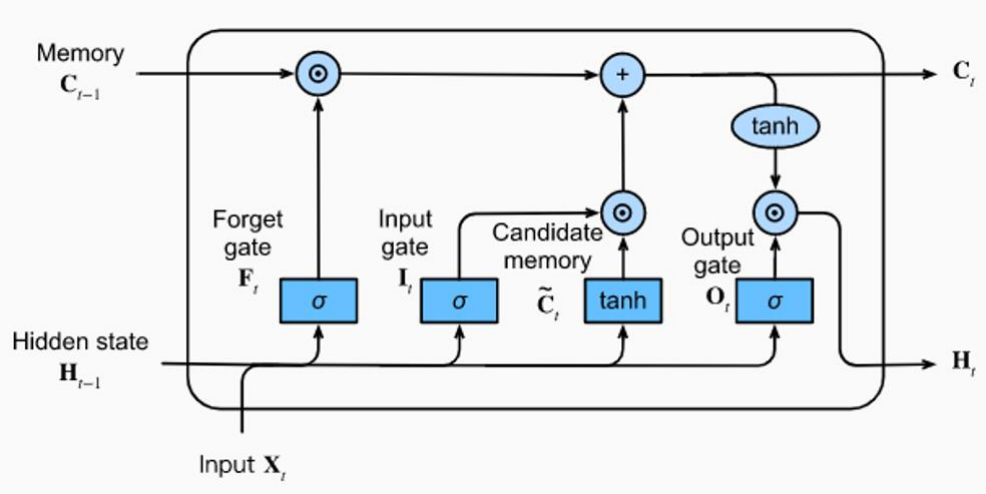


GRU

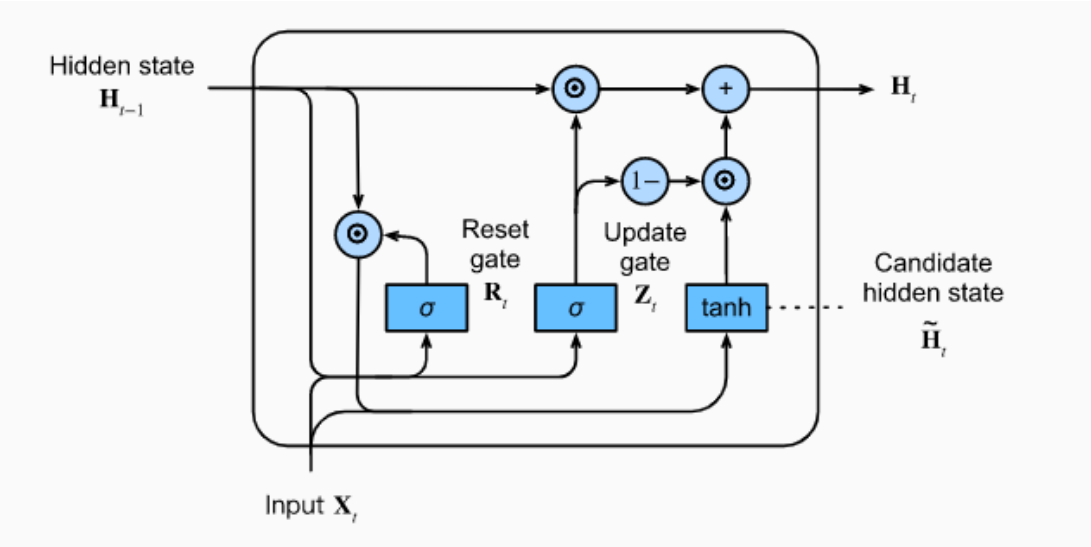
Gated Recurrent Units (GRUs) kom som en alternativ lösning på LSTMs problem med att vara väldigt komplexa. GRUs designades för att förenkla och optimera LSTM, men fortfarande behålla dess fördelar.

Varje GRU cell har en enklare struktur och kan därför tränas snabbare, men klarar inte lika komplexa problem som en LSTM modell.

GRU



LSTM



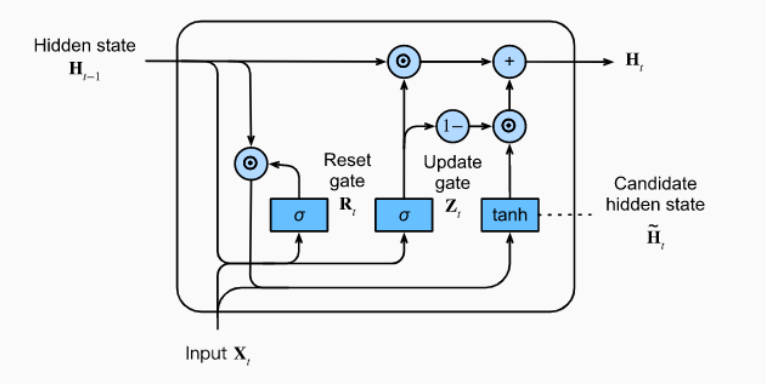
GRU

Smartr

GRU

En GRU cell innehåller

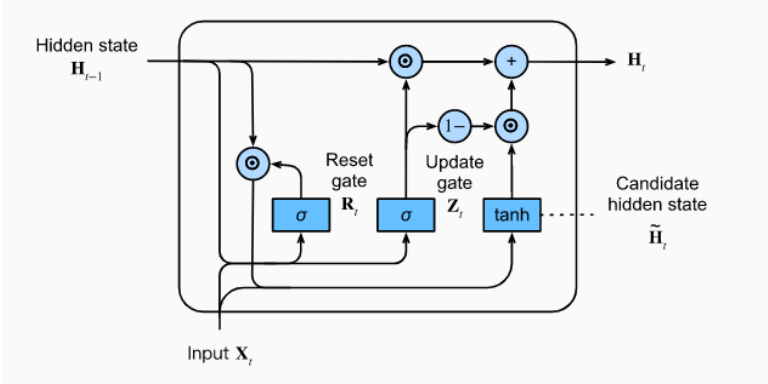
- Update gate - Bestämmer hur mycket av tidigare information som ska skickas vidare framåt (lite som kombinationen av forget och input gate i LSTM)
- Reset gate - Bestämmer hur mycket av tidigare information som ska glömmas bort.



GRU

Informationsflöde

- Två inputs, **xt** (ord i en mening) och **ht-1** (föregeående hidden state). Precis som RNN.
- Reset gate bestämmer hur mycket av föregående hidden state som ska kombineras med nuvarande input.
- Update gate bestämmer hur mycket av cellens output som ska bestå av ny information (nuvarande input) och hur mycket som ska bestå av gammal information (föregående hidden state).
- Kombinationen av input och vad som kommer ut från reset gate och update gate beräknar cellens hidden state (output).



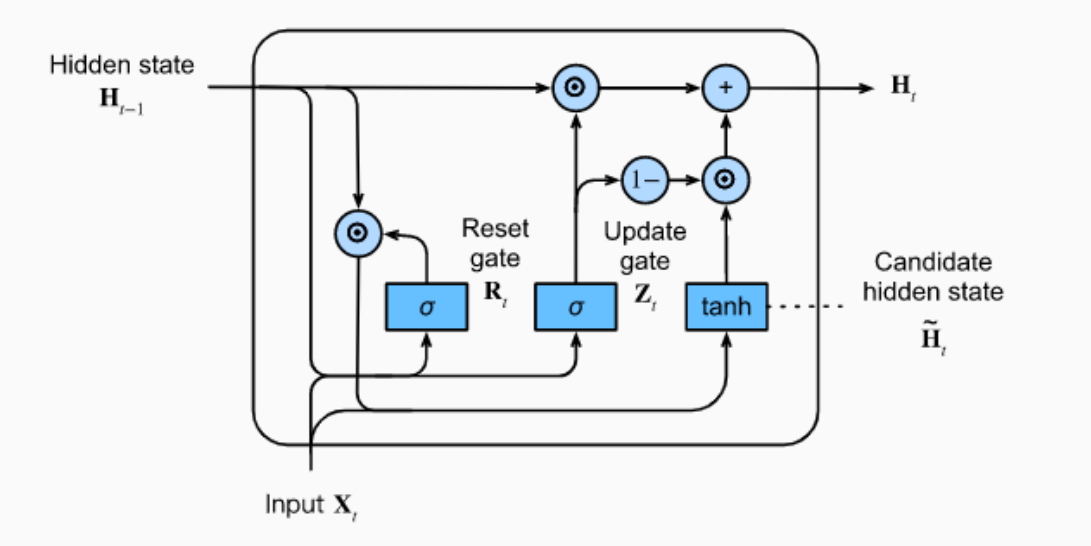
GRU

Update Gate: $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$

Reset Gate: $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$

Candidate Hidden State: $\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$

Final Hidden State: $h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$

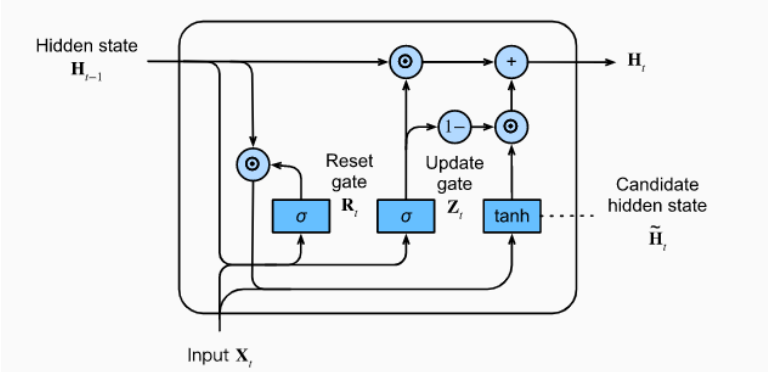


GRU

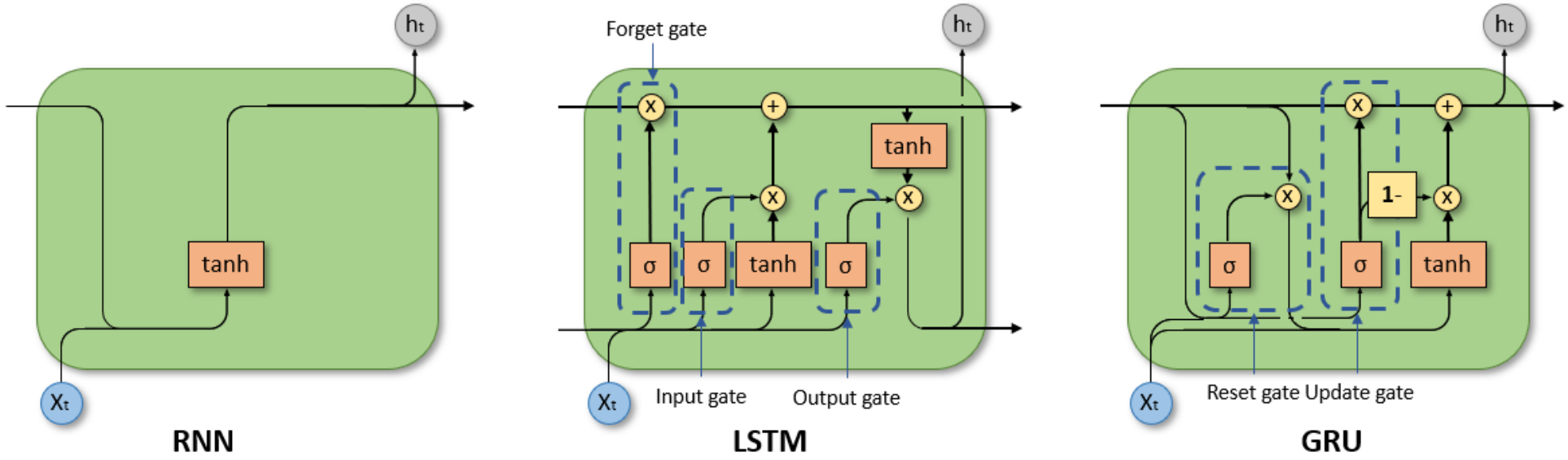
GRU

Fördelar och nackdelar

- Simplare arkitektur än LSTM, men mer komplex än vanligt RNN.
  - Lagom!
- Presterar likvärdigt med LSTM på många problem, men är snabbare att träna.
- Kan inte hantera långa beroenden lika bra som LSTM.
- Kan inte lösa lika svåra problem som LSTM.



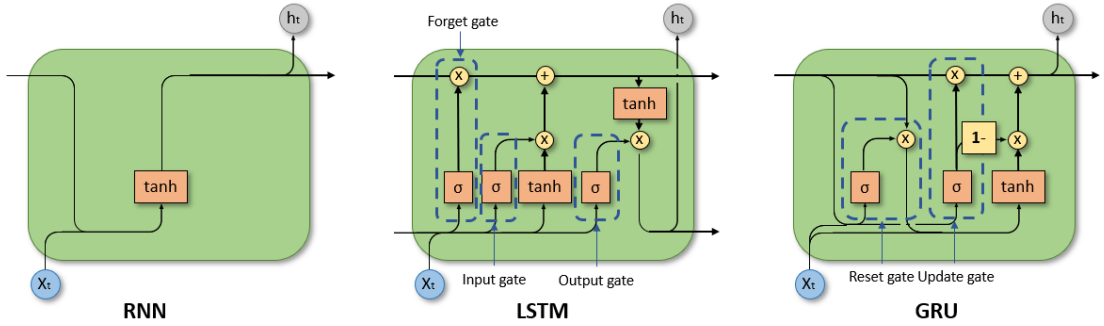
RNN


ht
tanh
Xt
RNN
Forget gate
ht
tanh
σ
σ
tanh
σ
Xt
Input gate
Output gate
LSTM
ht
1-
σ
σ
tanh
Xt
Reset gate Update gate
GRU

RNN

Vilken struktur vi ska använda beror på flera saker.

- Hur komplext är problemet
- Hur mycket beräknigskraft har jag?
  - Både vid träning och i produktion
- Hur hög modellprestanda behöver jag?
  - Och hur hög kan jag få med de olika modellerna?
- Hur se min data ut?
  - I tidseriedata kan karaktären av tidsserien göra att olika strukturer fungerar bättre.



Smartr