CISC6000 Deep Learning
LSTM/GRU

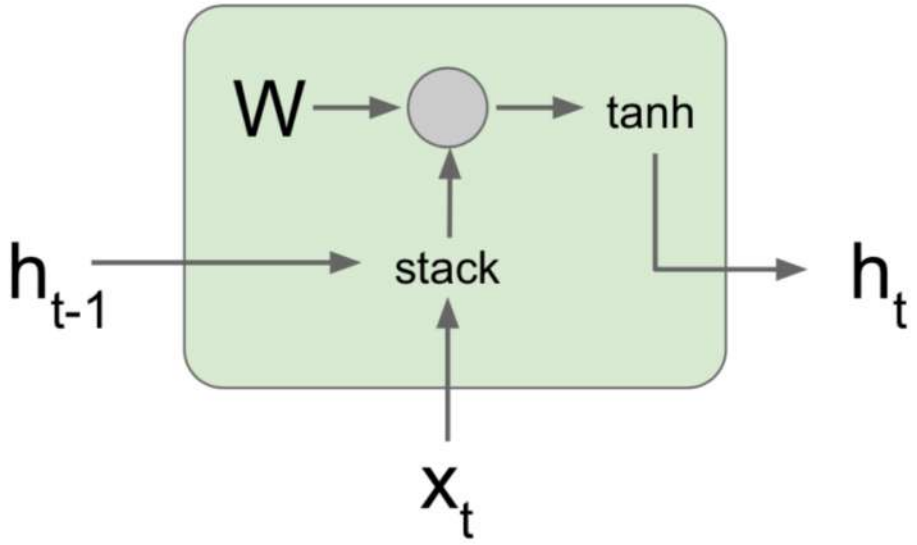
Dr. Yijun Zhao

Fordham University

Limitations of RNN

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



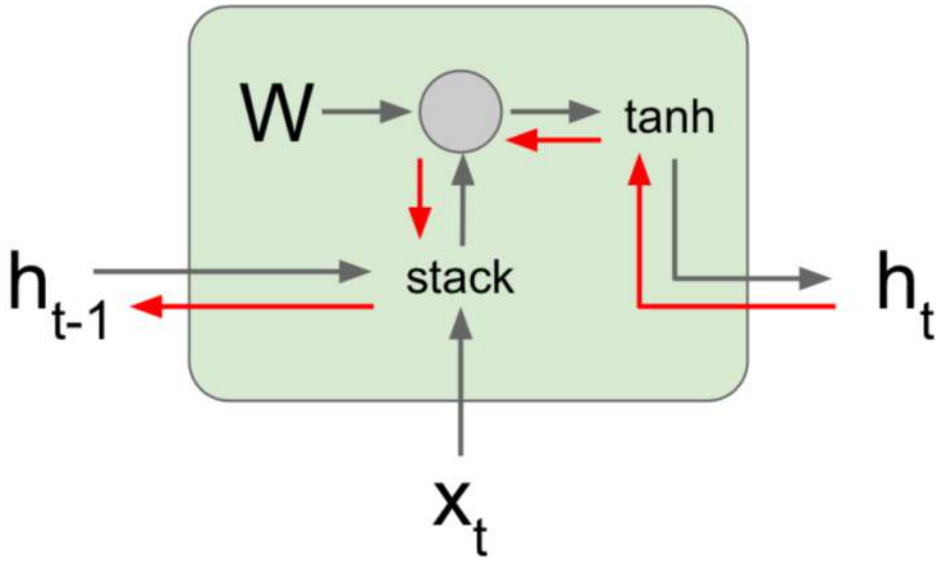
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Limitations of RNN

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

Backpropagation from h_t to h_{t-1} multiplies by W

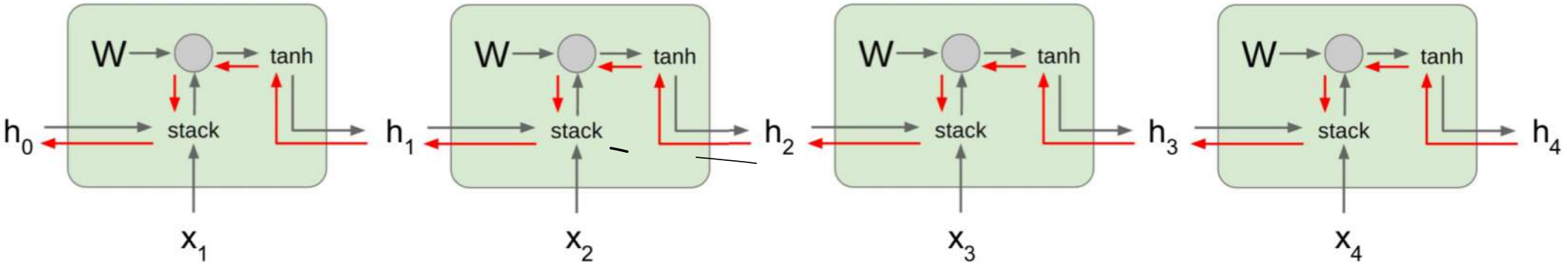


$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= \tanh\left((W_{hh} \quad W_{hx})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= \tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Limitations of RNN

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013

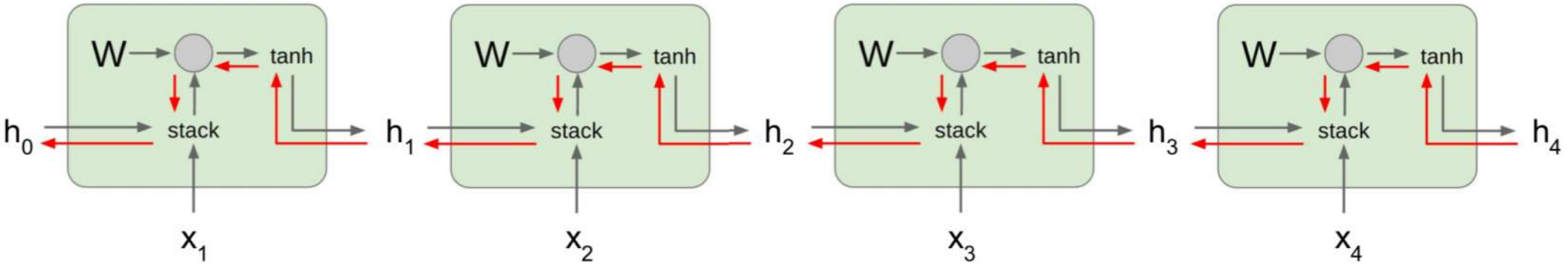


Computing gradient of h_0 involves many factors of W

Limitations of RNN

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

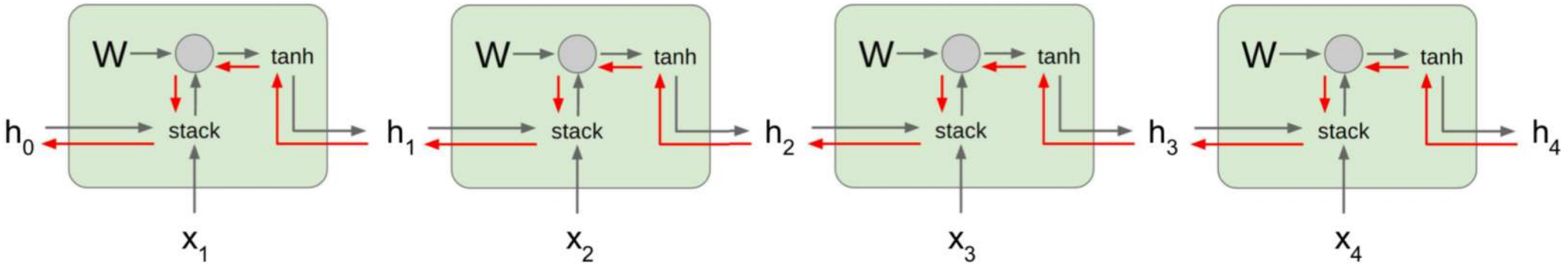
Largest singular value > 1:
Exploding gradients

Largest singular value < 1:
Vanishing gradients

Limitations of RNN

Vanilla RNN Gradient Flow

Bengio et al, "Learning long-term dependencies with gradient descent is difficult", IEEE Transactions on Neural Networks, 1994
Pascanu et al, "On the difficulty of training recurrent neural networks", ICML 2013



Computing gradient of h_0 involves many factors of W (and repeated tanh)

Largest singular value > 1:
Exploding gradients

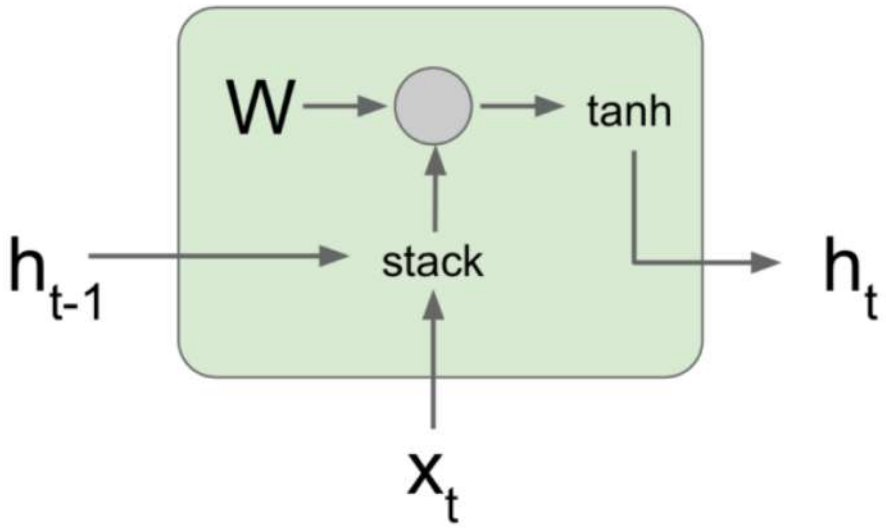
→ **Gradient clipping**: Scale gradient if its norm is too big

```
grad_norm = np.sum(grad * grad)
if grad_norm > threshold:
  grad *= (threshold / grad_norm)
```

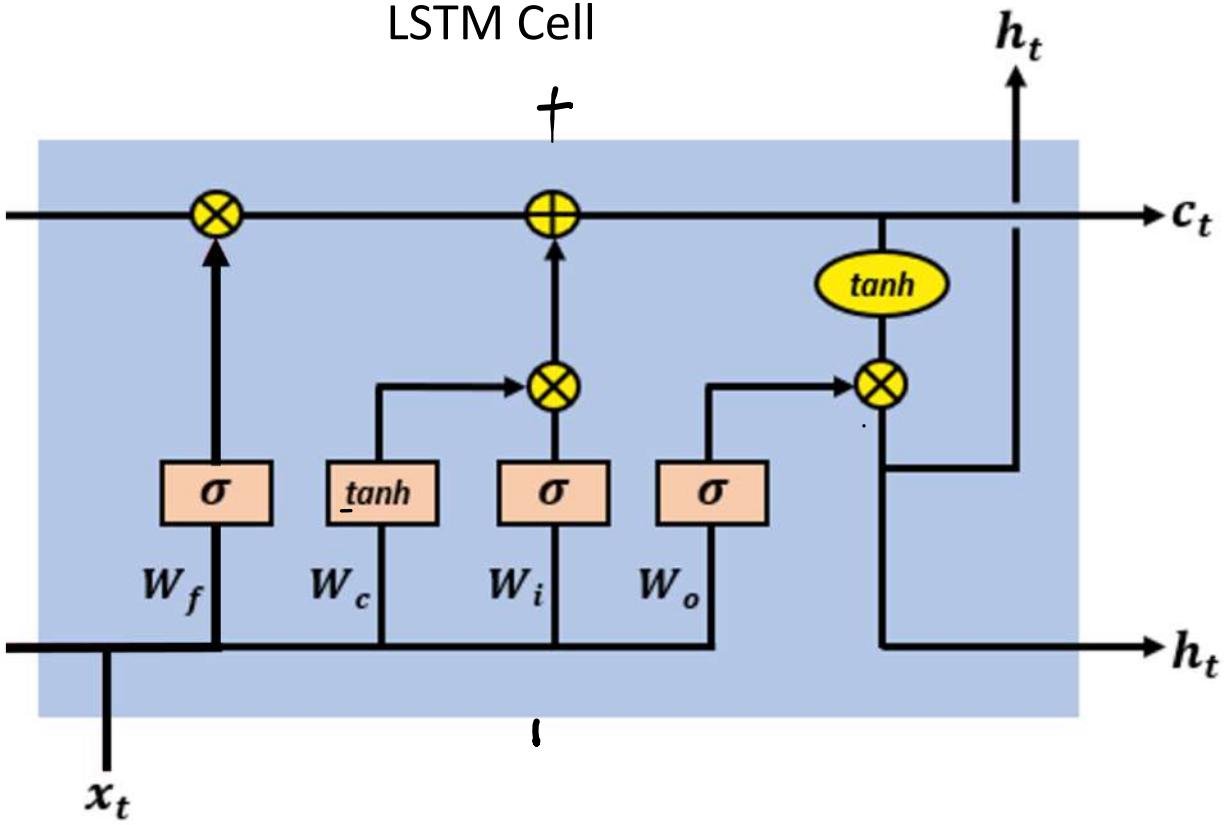
Largest singular value < 1:
Vanishing gradients

LSTM

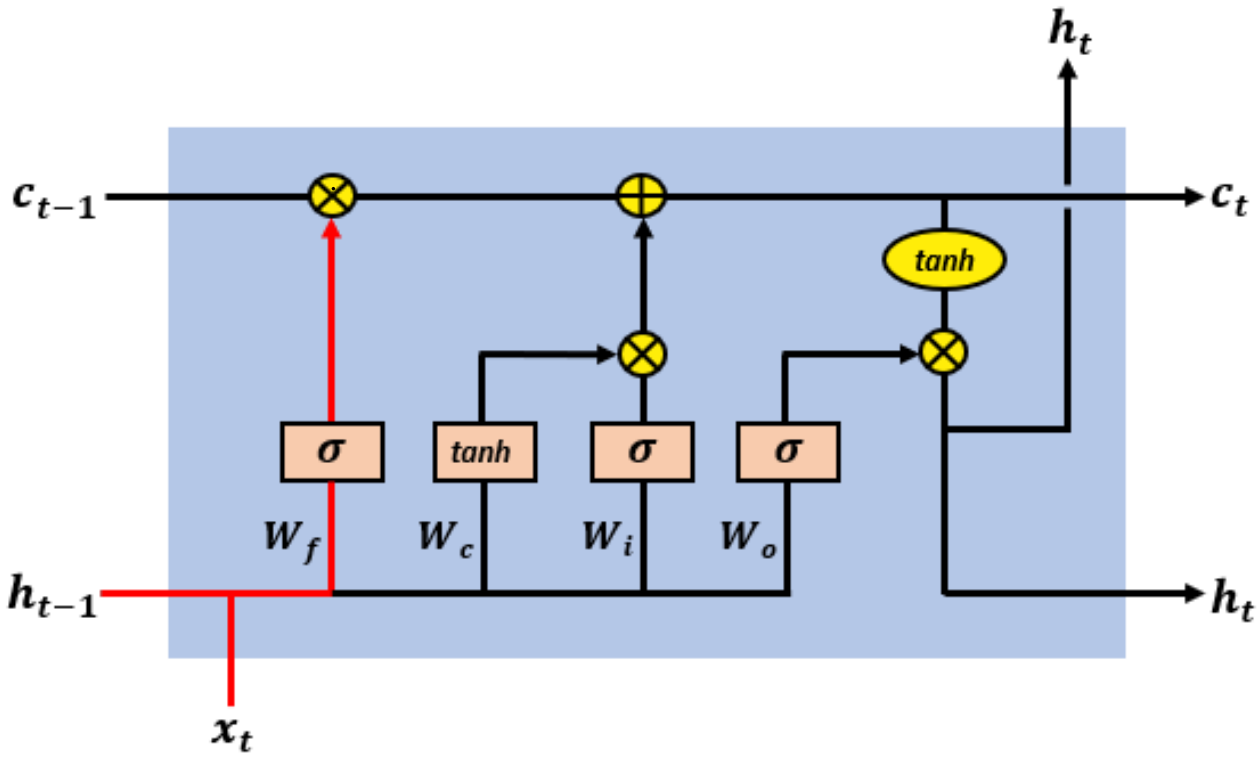
RNN



LSTM Cell



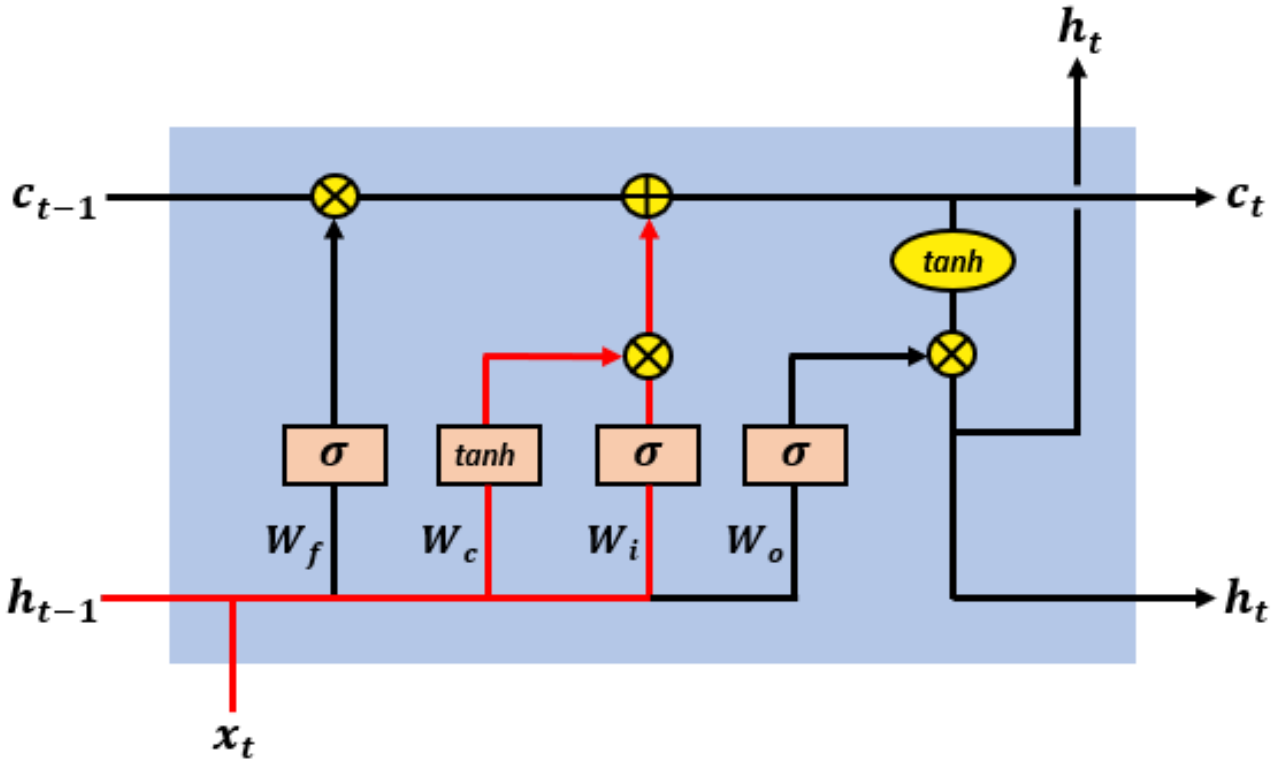
LSTM



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

forget gate: regulates what information to throw away from the cell.

LSTM



$$\tilde{c}_t \otimes i_t$$

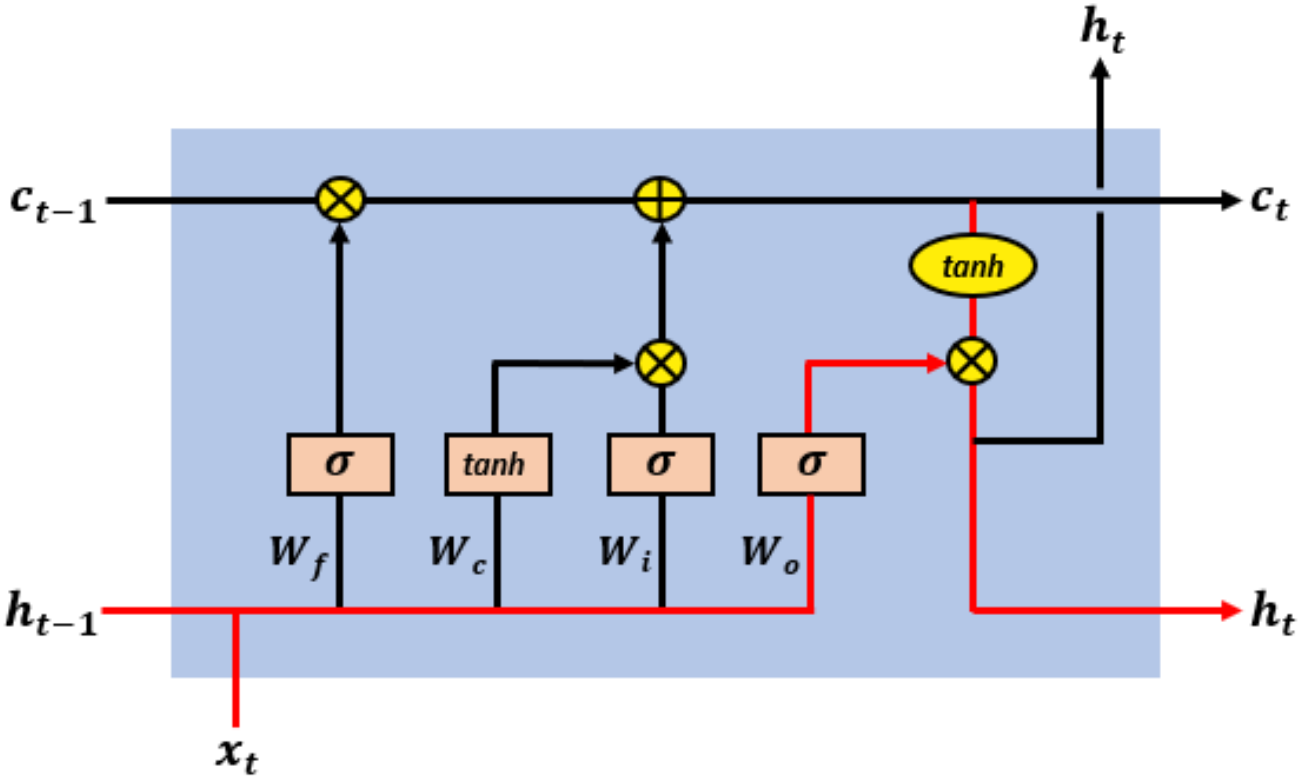
$$\tilde{c}_t = tanh\ (W_c \cdot [h_{t-1}, x_t])$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$

input gate: decides what information is relevant to add from the current step.

If the gate's value is zero, the flow from another node is cut off, whereas if its value is one, all flow is passed.

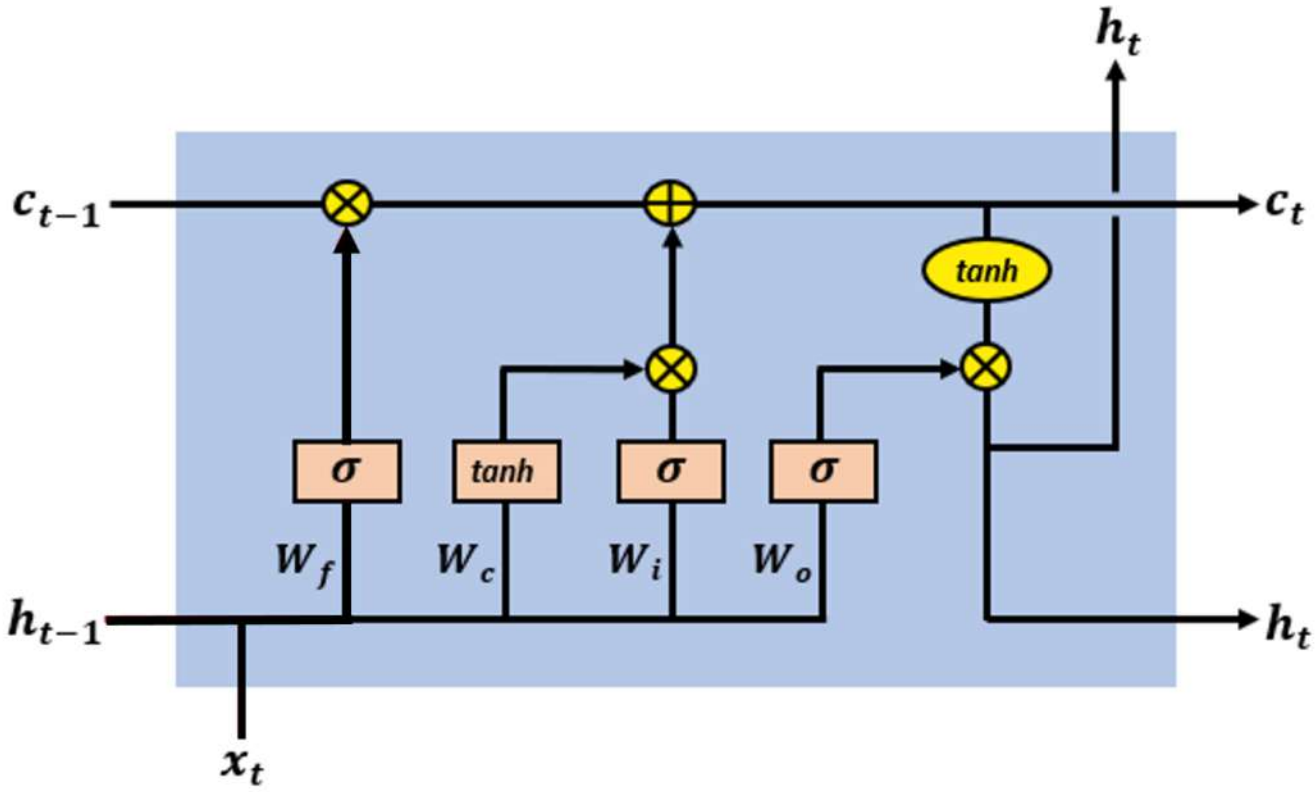
LSTM



$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t])$$

output gate: decides what to reveal to next hidden state.

LSTM



$$c_t = c_{t-1} \otimes f_t \oplus \tilde{c}_t \otimes i_t$$

$$\tilde{c}_t = tanh\,(W_c \cdot [h_{t-1}, x_t])$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t])$$

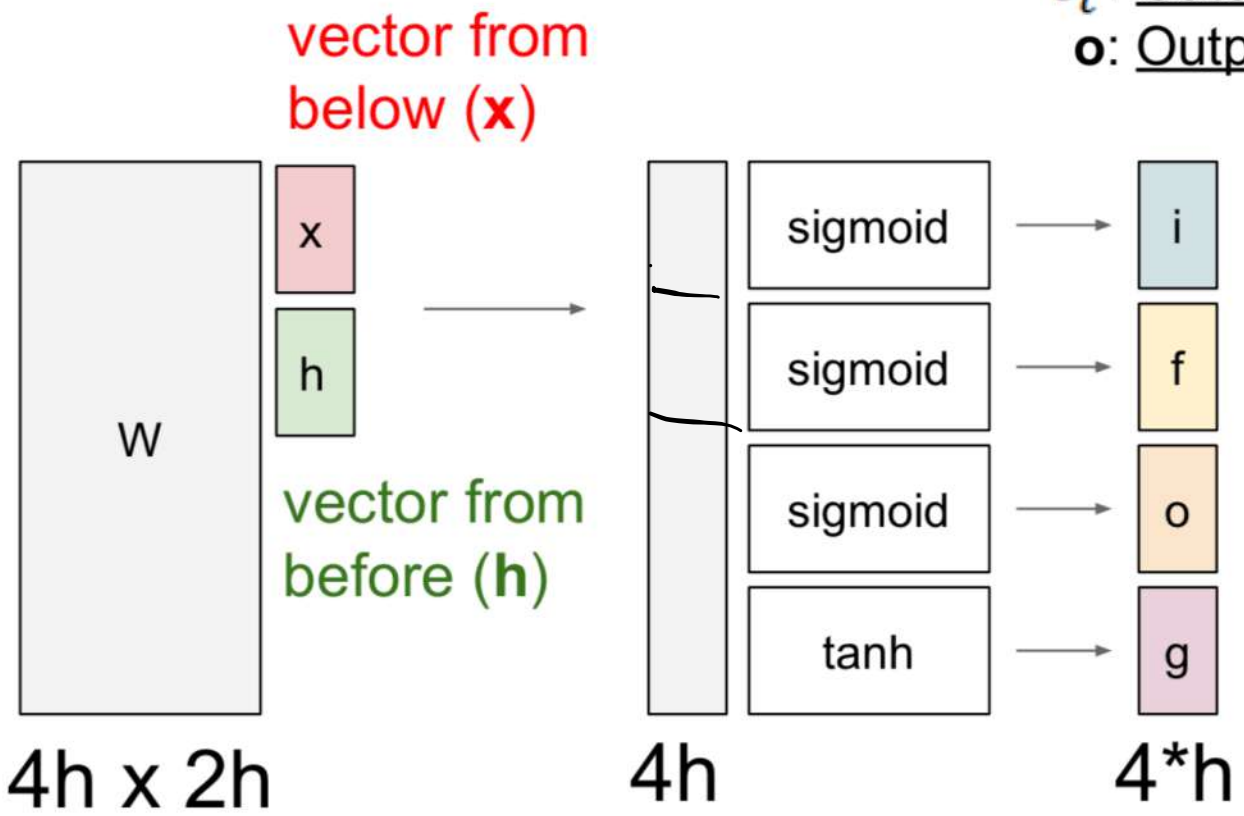
$$h_t = o_t \otimes tanh(c_t)$$

cell state: an updated cell state (c_t) and a new hidden state (h_t) will be passed to the next cell.

Long Short Term Memory (LSTM)

[Hochreiter et al., 1997]

f: Forget gate, Whether to erase cell
i: Input gate, whether to write to cell
$\tilde{c}_t$: Gate gate (?), How much to write to cell
o: Output gate, How much to reveal cell



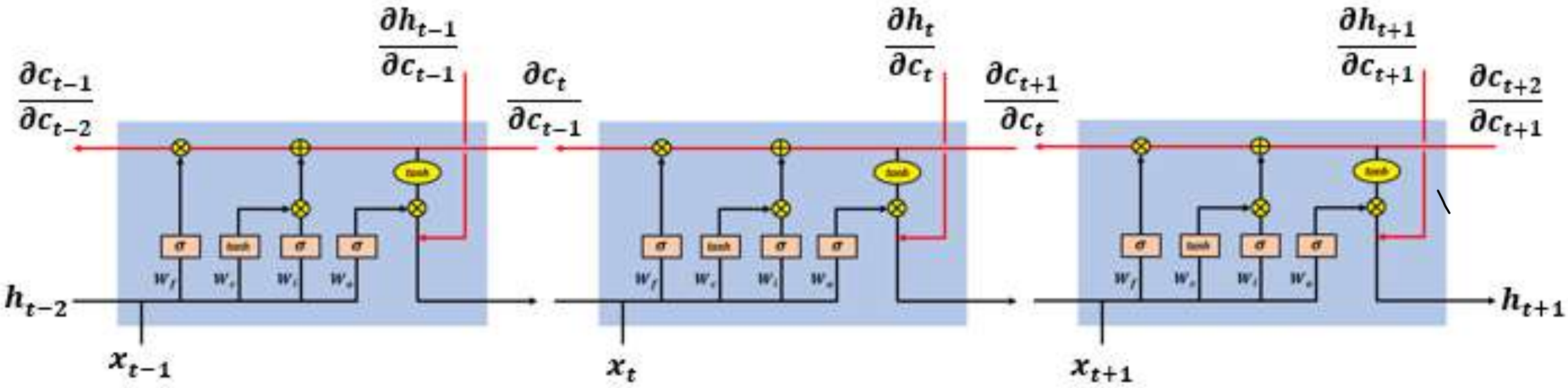
$$\begin{pmatrix} i \\ f \\ o \\ \widetilde{C}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot$$

$$h_t = o \odot \tanh(c_t)$$

LSTM

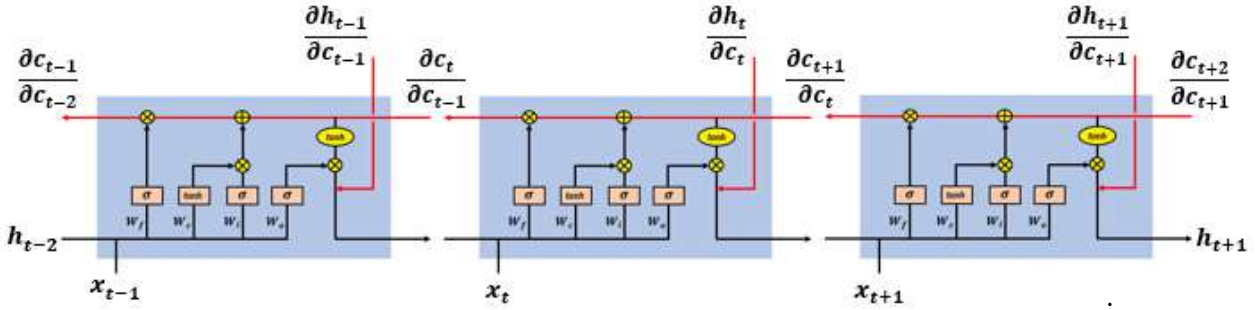
Backpropagation through time in LSTMs



LSTM

After T time steps, the error term gradient is given by the following sum of T gradients

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$



So if we don't want the gradient to vanish, our network needs to increase the likelihood that at least some of these sub gradients will not vanish. Note that

$$\frac{\partial E_k}{\partial W} = \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\cdots\frac{\partial c_2}{\partial c_1}\frac{\partial c_1}{\partial W}$$

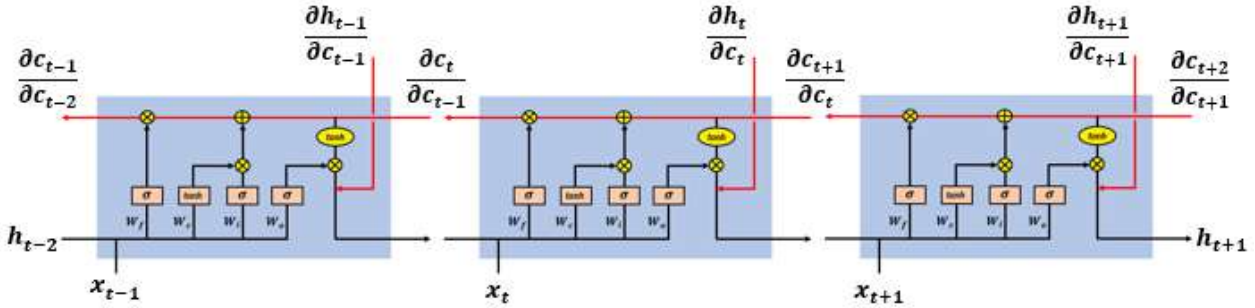
product causes the gradients to vanish

$$= \frac{\partial E_k}{\partial h_k}\frac{\partial h_k}{\partial c_k}\left(\prod_{t=2}^{k}\frac{\partial c_t}{\partial c_{t-1}}\right)\frac{\partial c_1}{\partial W} \qquad (4)$$

LSTM

Backpropagation through time in LSTMs

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t$$



$$A_t = \sigma'\left(W_f \cdot [h_{t-1}, x_t]\right) \cdot W_f \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot c_{t-1}$$

$$B_t = f_t$$

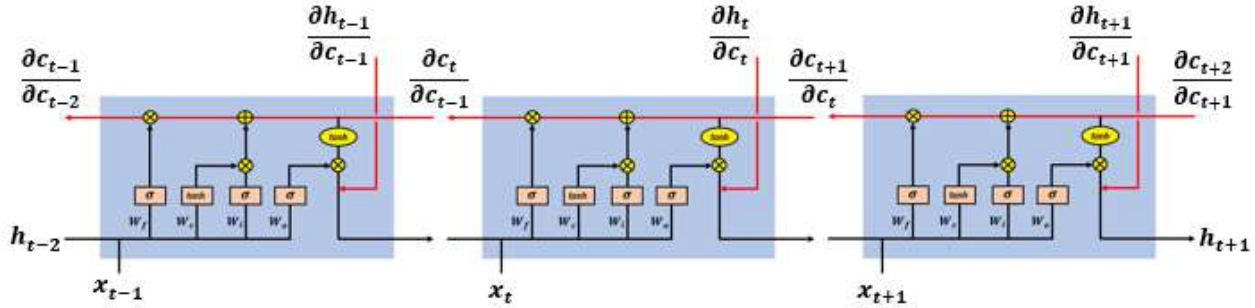
$$C_t = \sigma'(W_i \cdot [h_{t-1}, x_t]) \cdot W_i \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot \tilde{c}_t$$

$$D_t = \sigma'(W_c \cdot [h_{t-1}, x_t]) \cdot W_c \cdot o_{t-1} \otimes tanh'(c_{t-1}) \cdot i_t$$

LSTM

Backpropagation through time in LSTM

$$\frac{\partial c_t}{\partial c_{t-1}} = A_t + B_t + C_t + D_t$$



$$A_t \approx \overrightarrow{0.1},\ B_t = f_t \approx \overrightarrow{0.7},\ \mathrm{C_t} \approx \overrightarrow{0.1},\ \mathrm{D_t} \approx \overrightarrow{0.1}$$

$$\frac{\partial c_t}{\partial c_{t-1}} = \left[\overrightarrow{0.1} + \overrightarrow{0.7} + \overrightarrow{0.1} + \overrightarrow{0.1}\right] \approx \vec{1} \nrightarrow 0$$

An Intuitive LSTM Example

Write a children's book

Doug saw Jane.

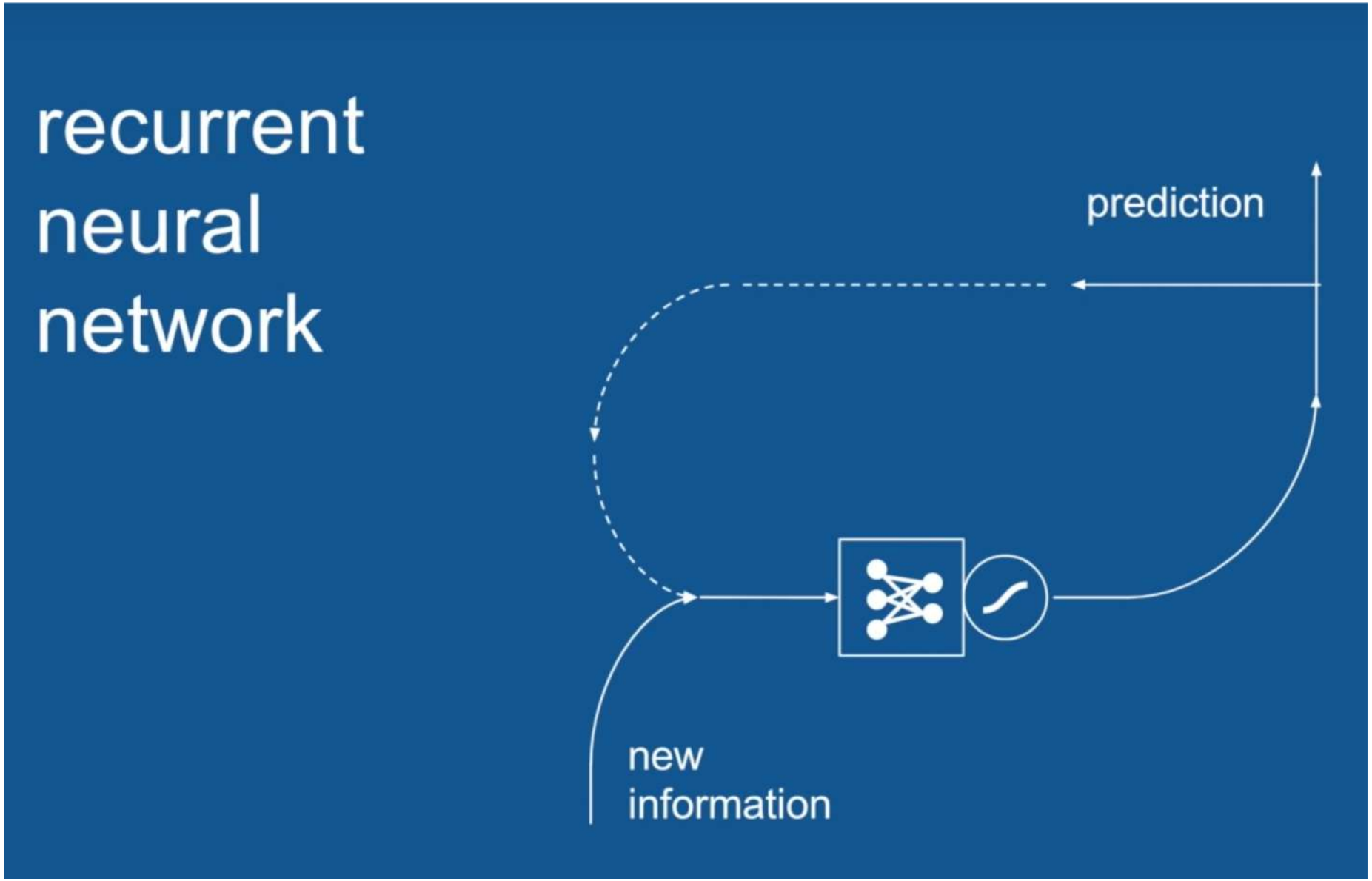
Jane saw Spot.

Spot saw Doug.

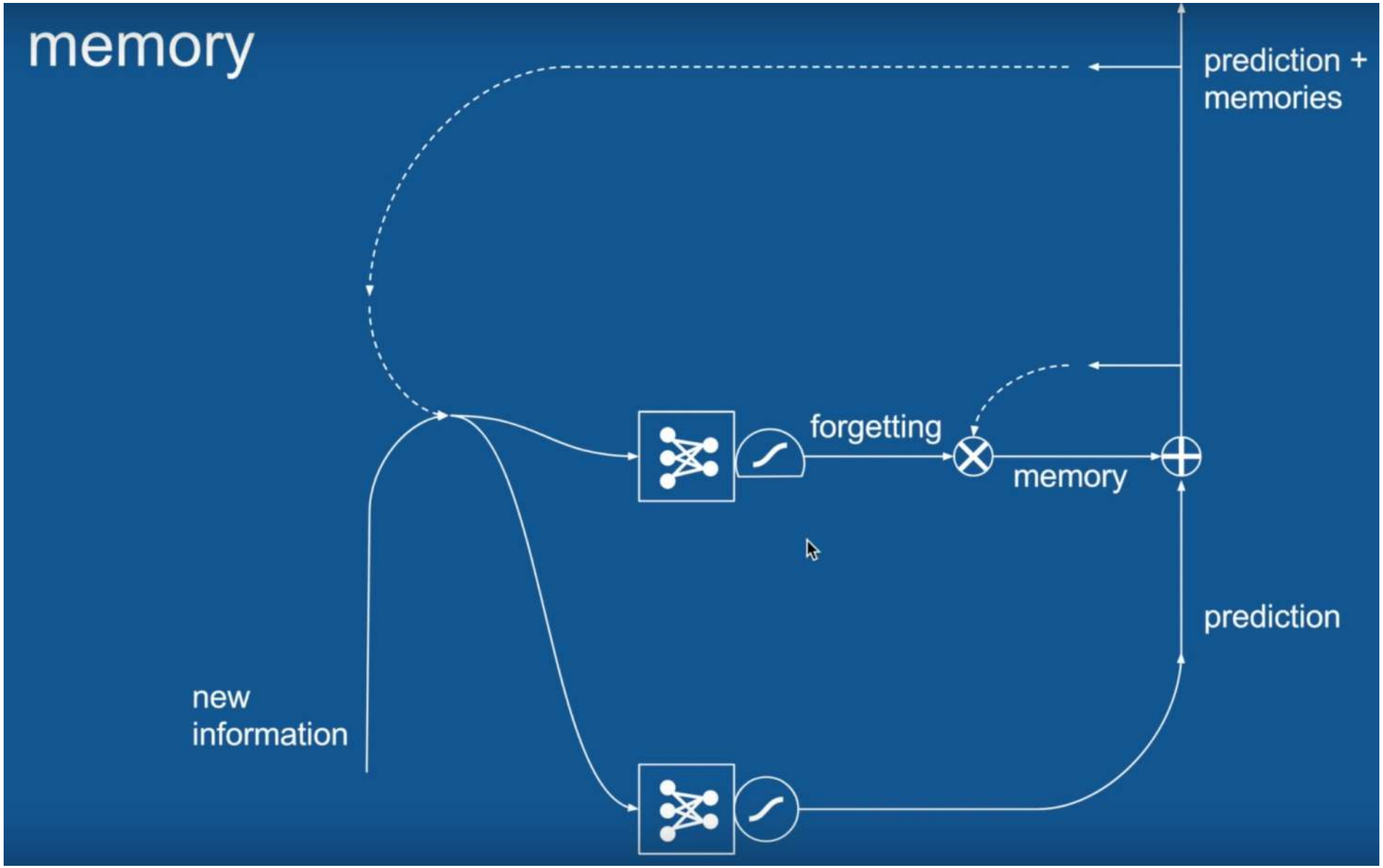
...

Your dictionary is small: {Doug, Jane, Spot, saw, .}

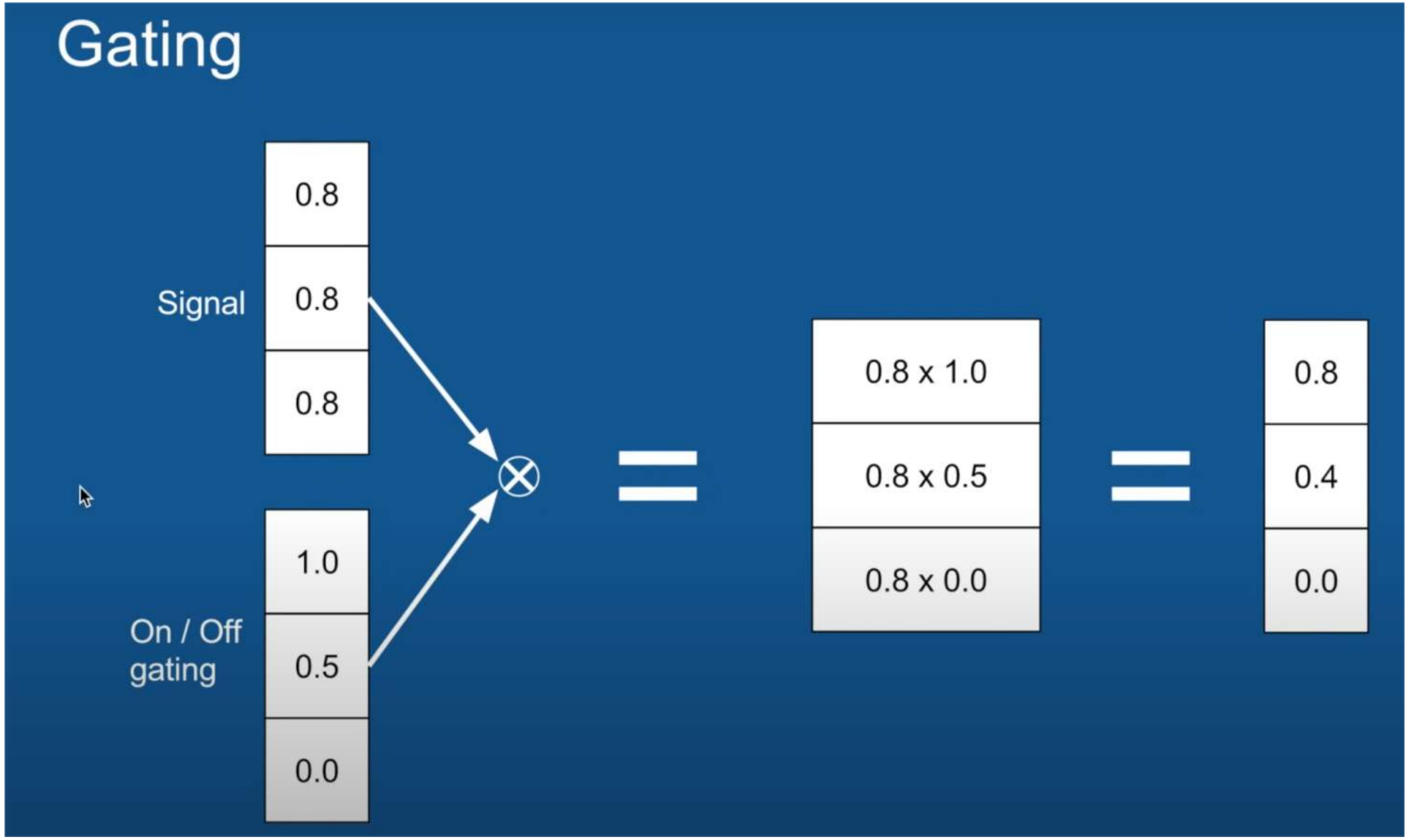
An Intuitive LSTM Example



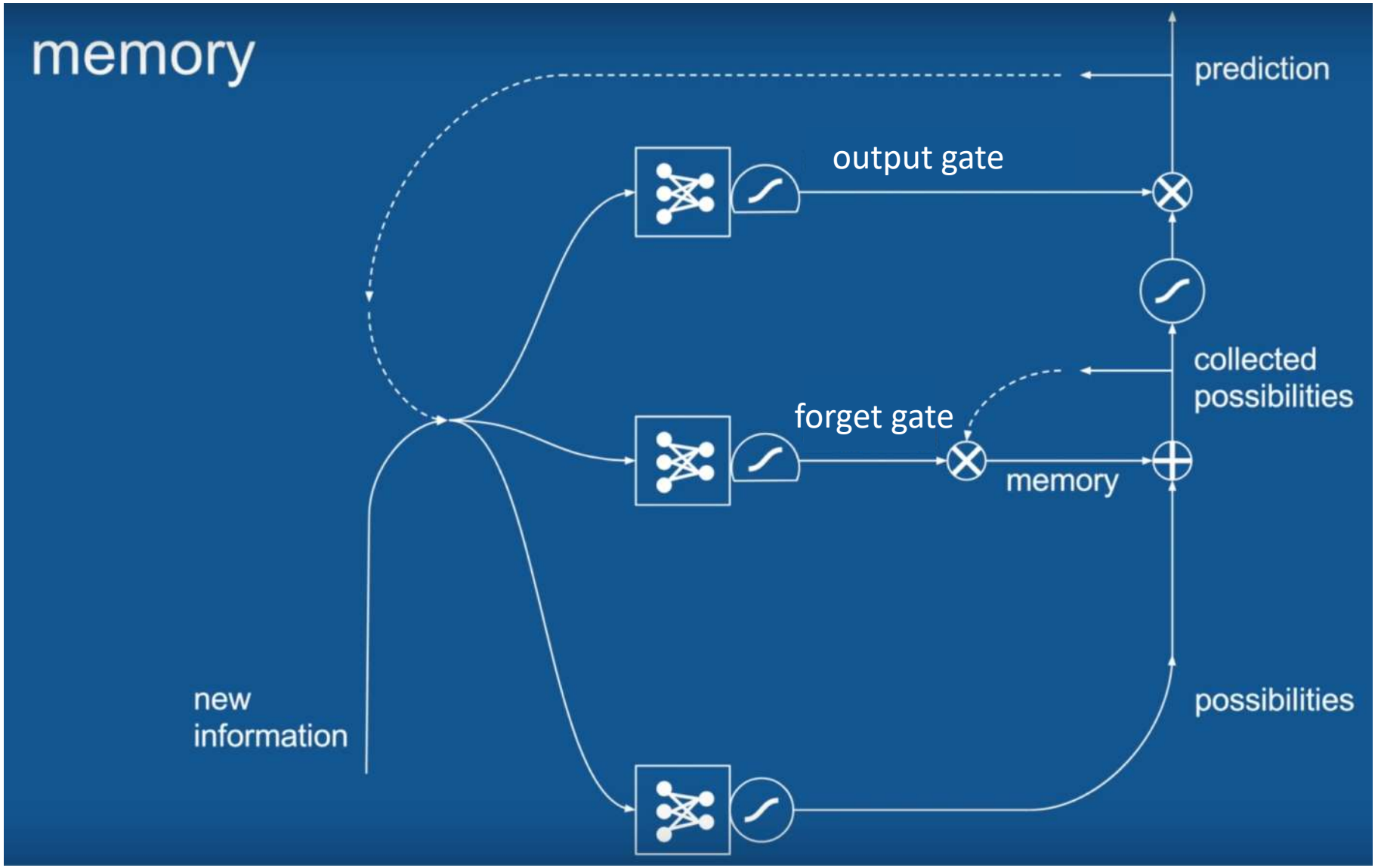
An Intuitive LSTM Example



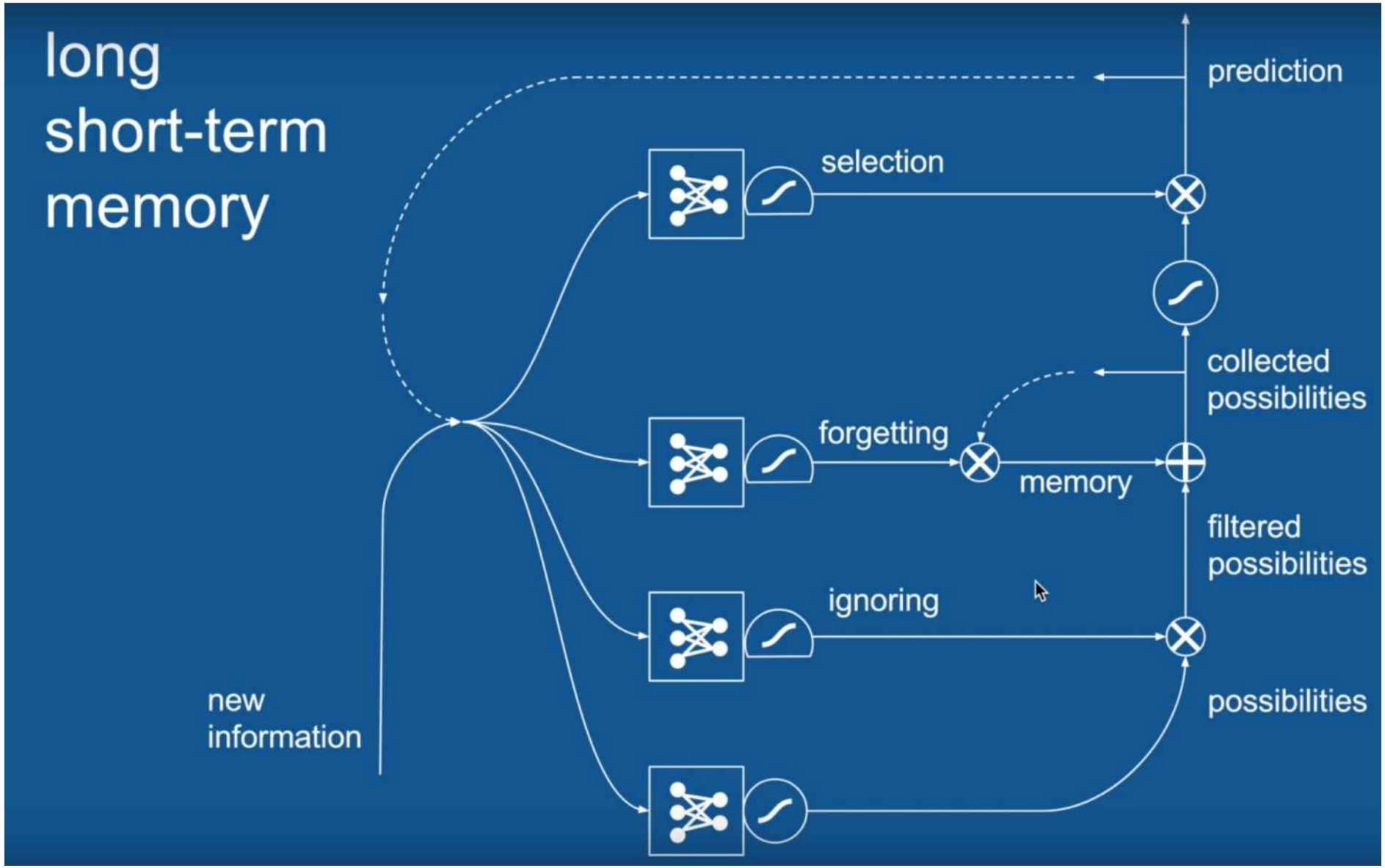
An Intuitive LSTM Example



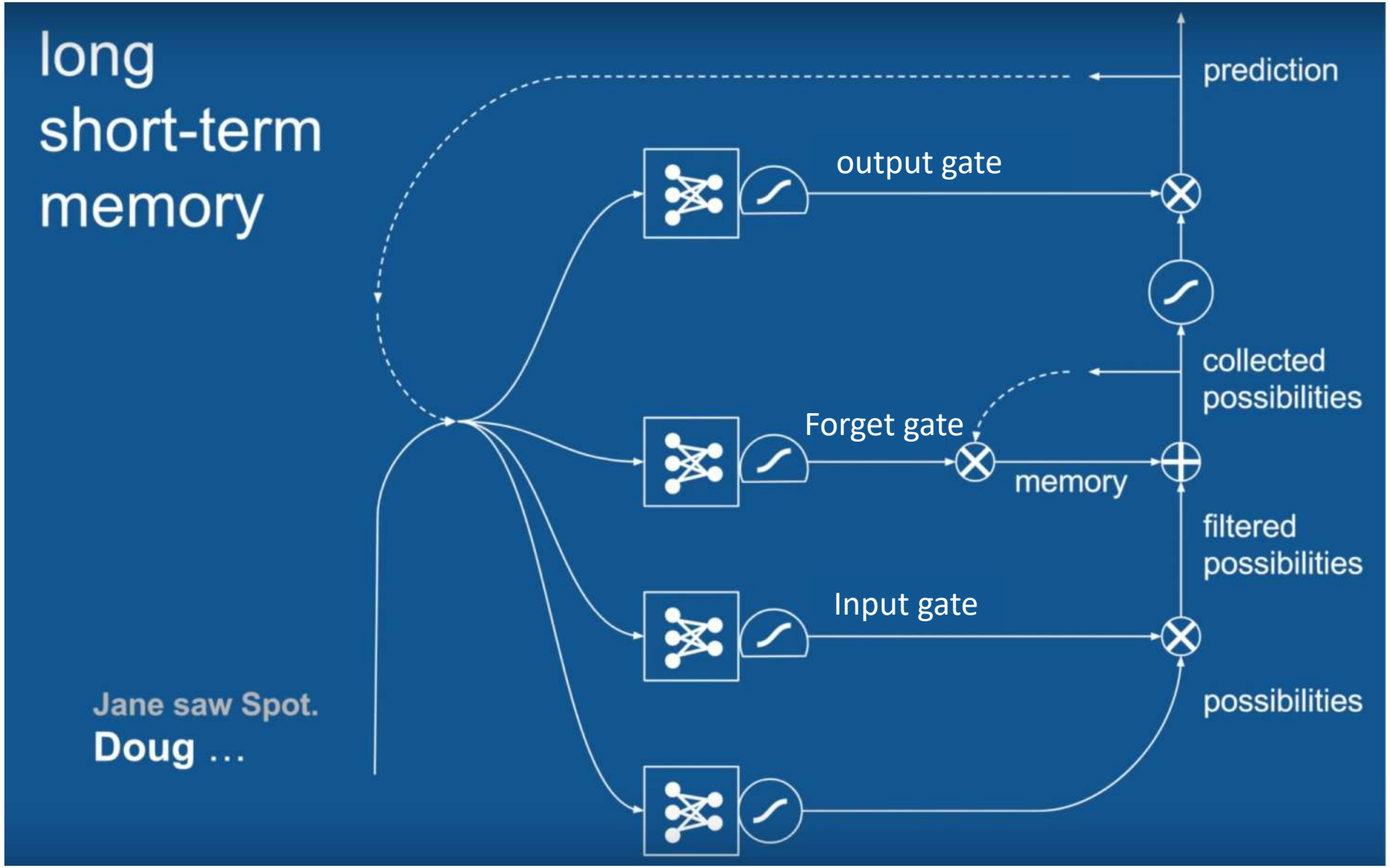
An Intuitive LSTM Example



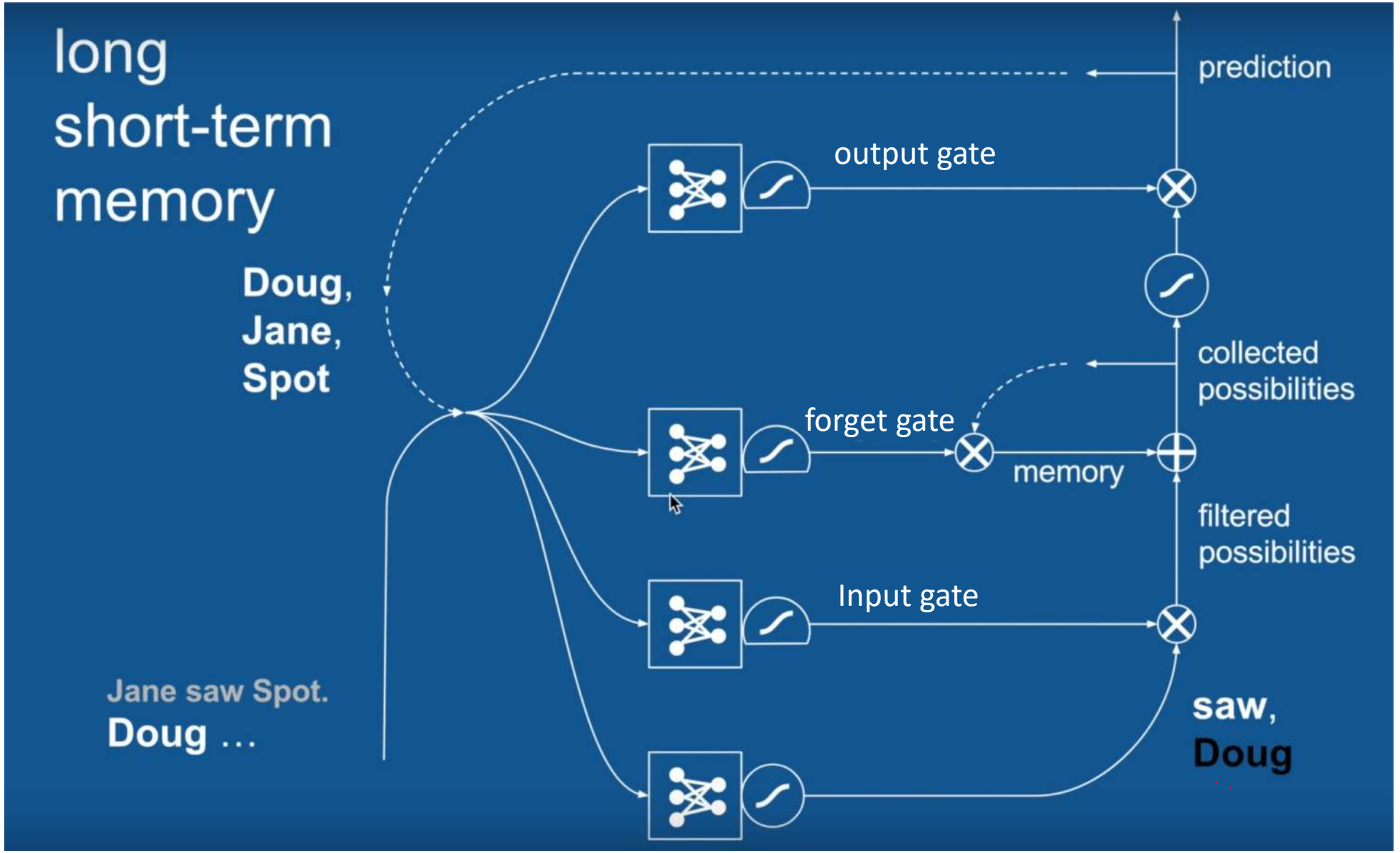
An Intuitive LSTM Example



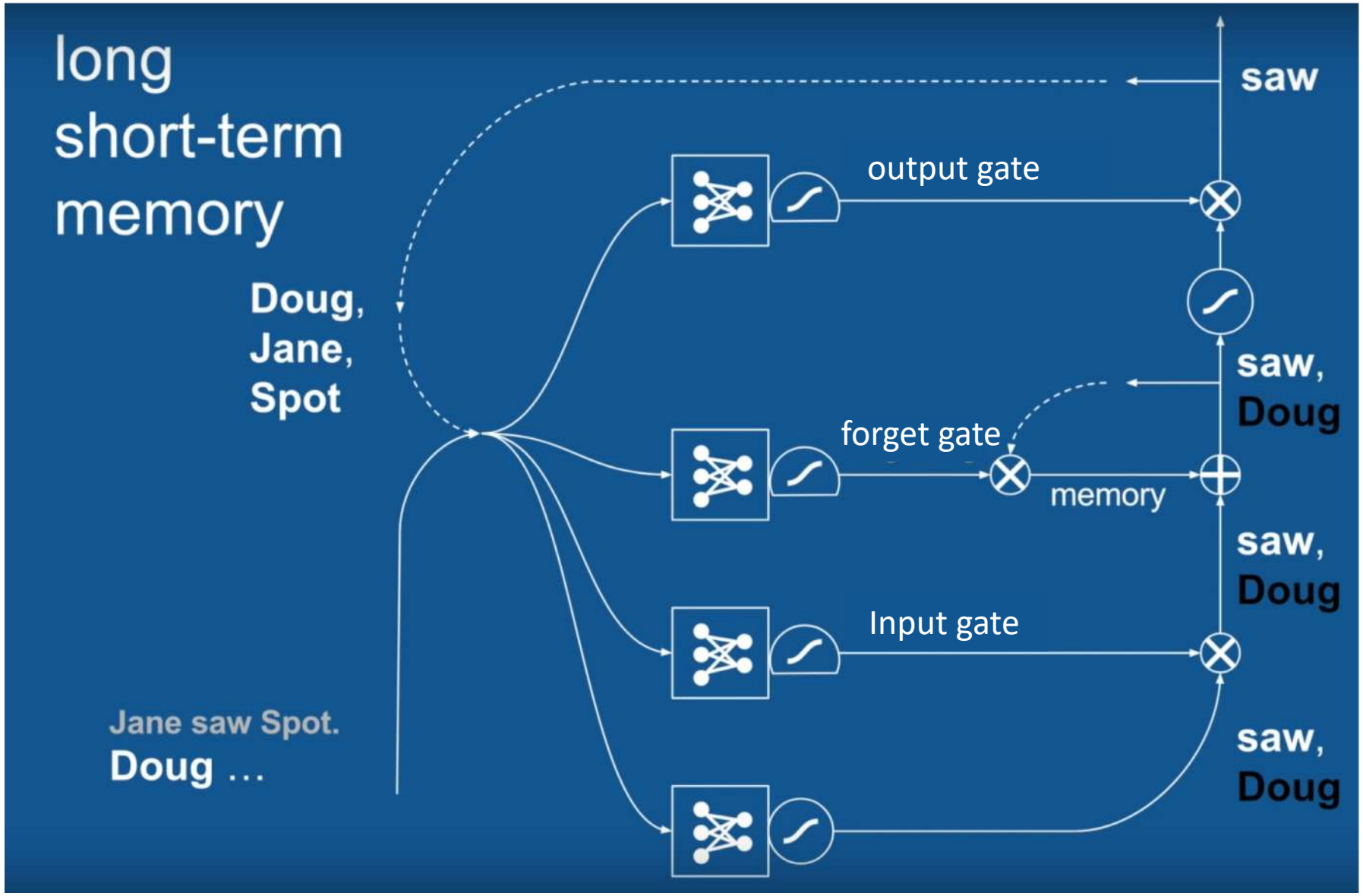
An Intuitive LSTM Example



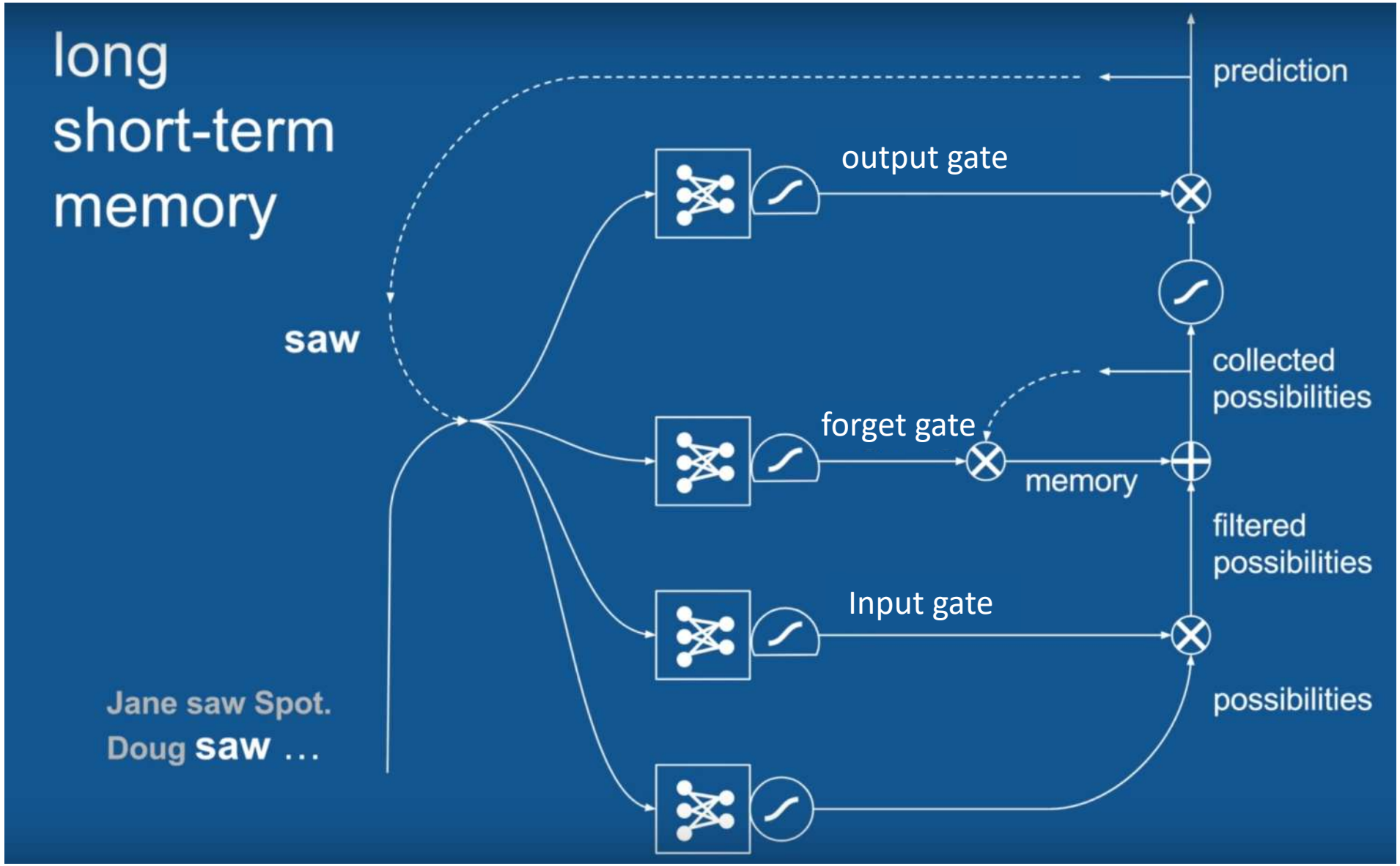
An Intuitive LSTM Example



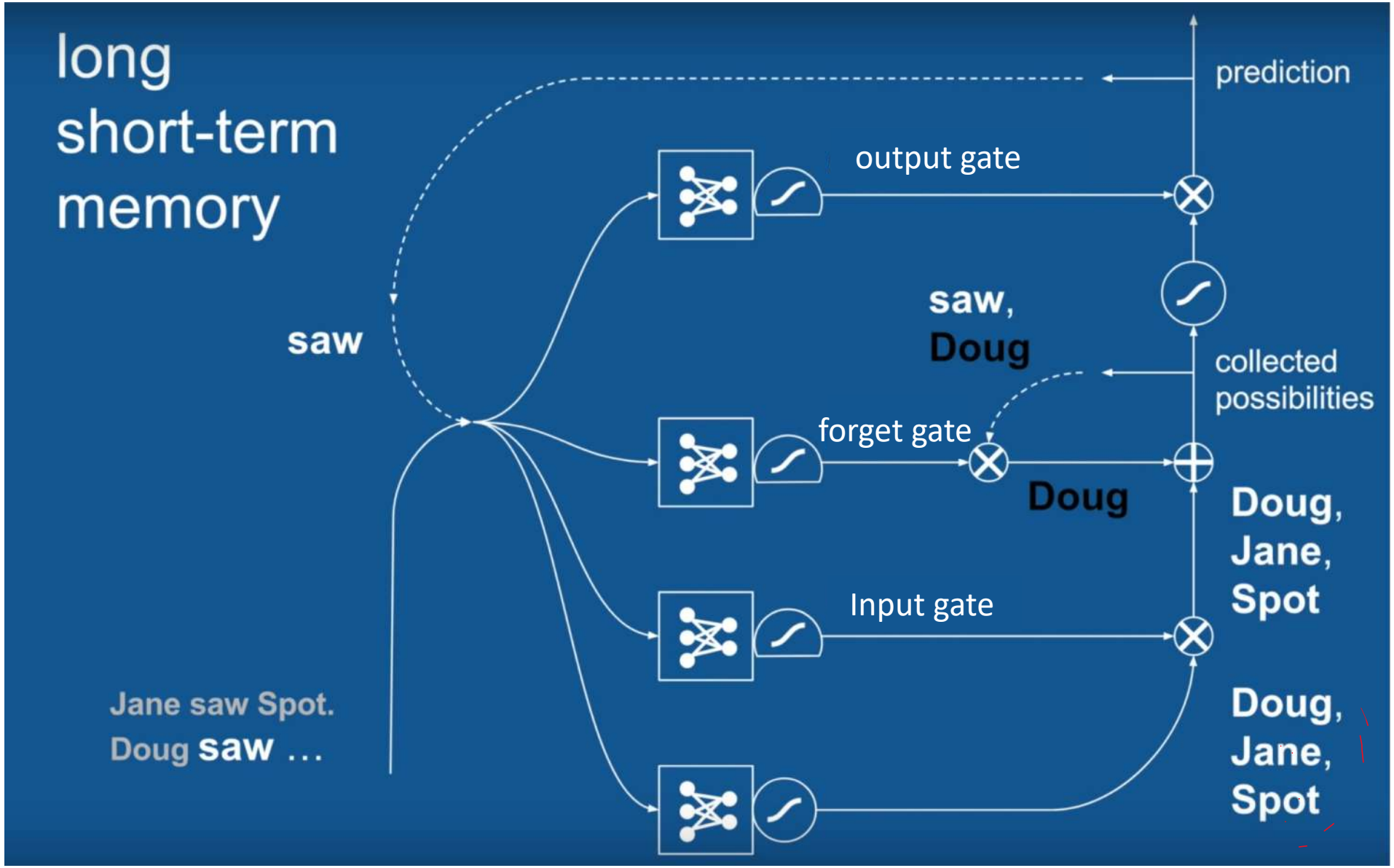
An Intuitive LSTM Example



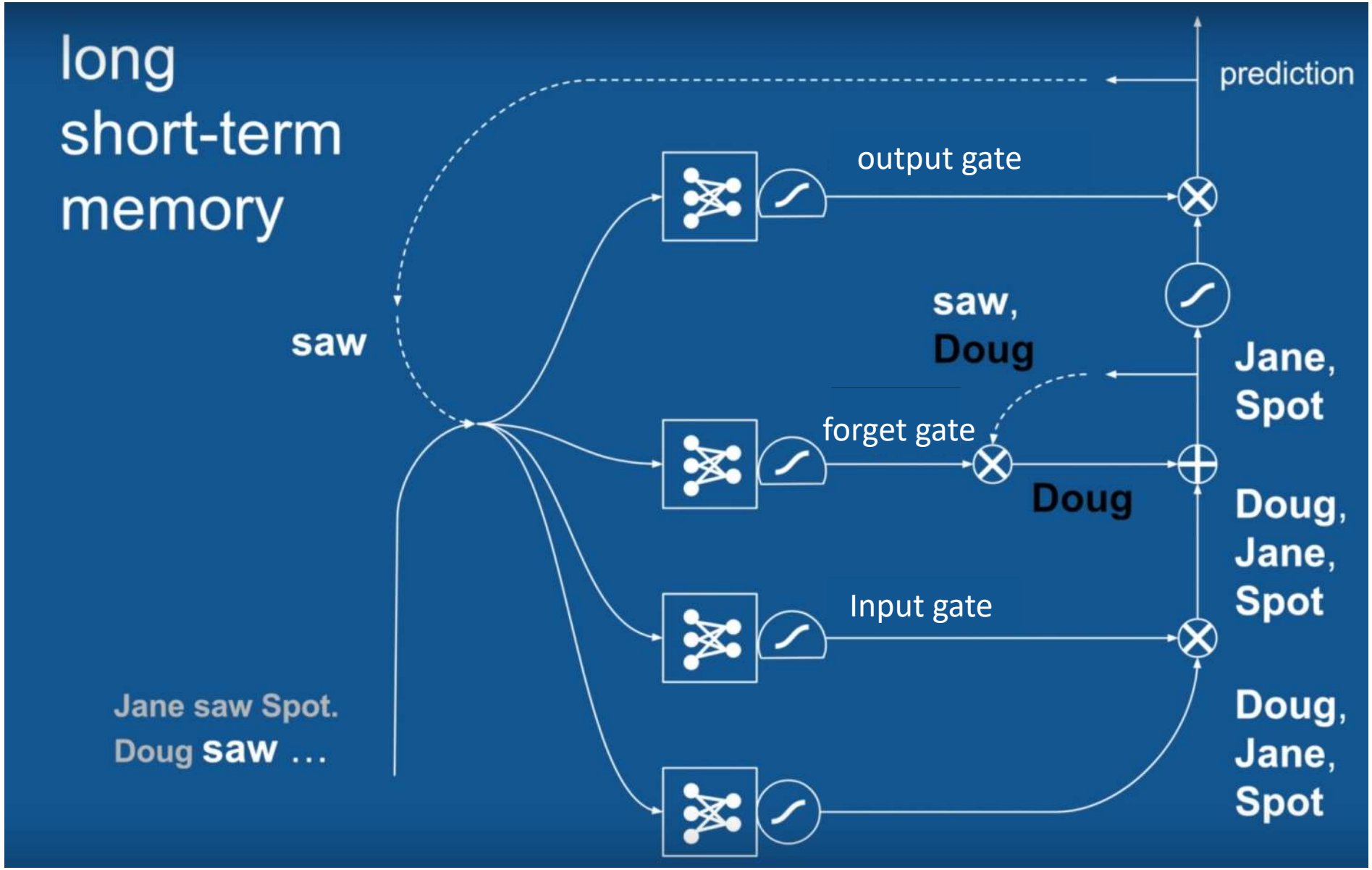
An Intuitive LSTM Example



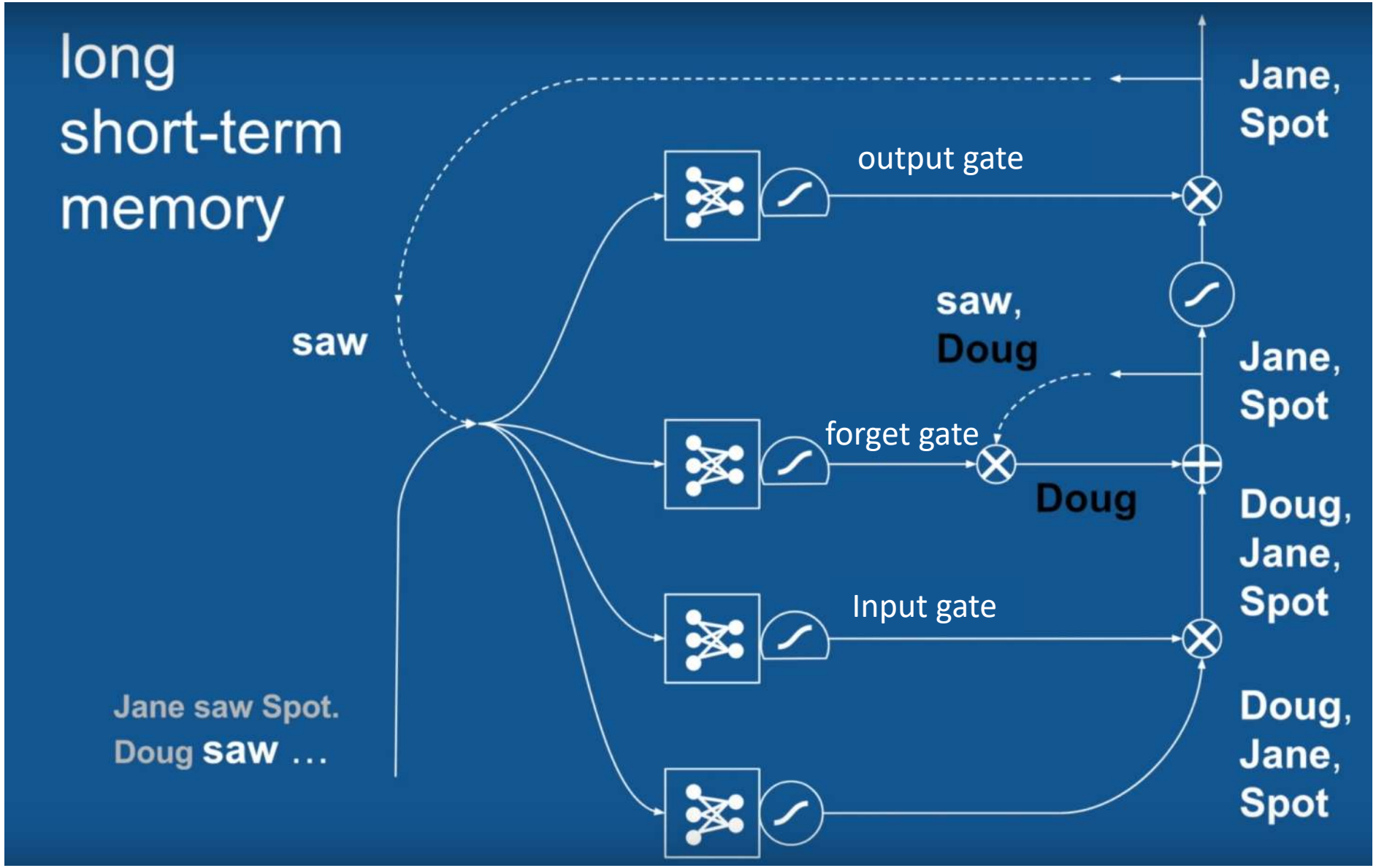
An Intuitive LSTM Example



An Intuitive LSTM Example



An Intuitive LSTM Example



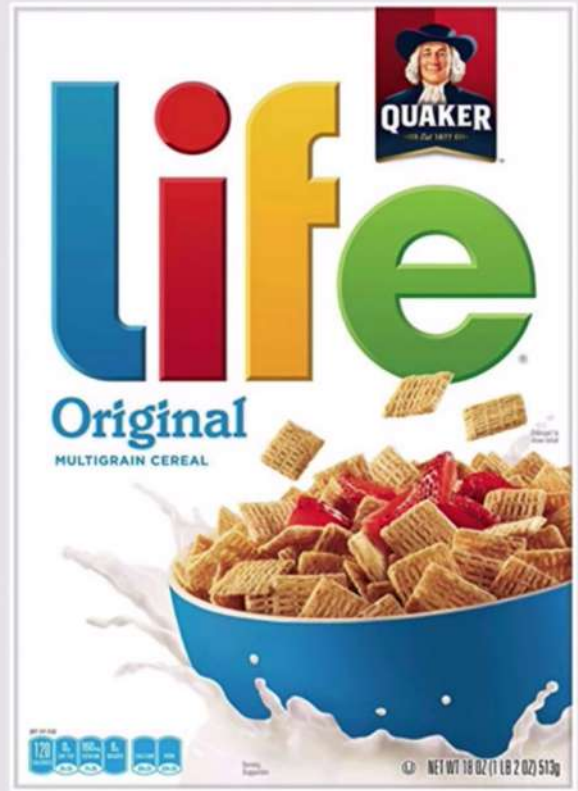
Customers Review 2,491

Thanos

September 2018

Verified Purchase

Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!



A Box of Cereal

$3.99

Sentiment Prediction

Customers Review 2,491

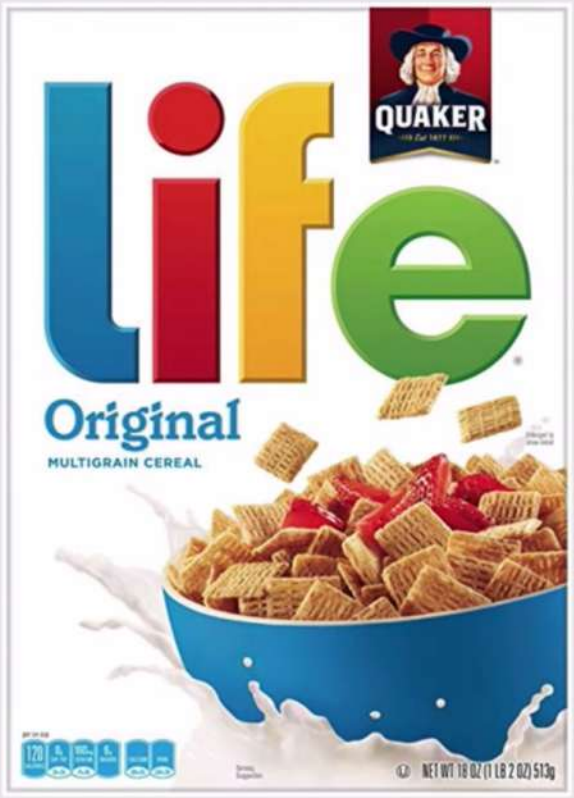
Thanos

September 2018

Verified Purchase

Amazing! This box of cereal gave me a **perfectly balanced breakfast**, as all things should be. I only ate half of it but **will definitely be buying again!**

LSTM can learn to keep only relevant information to make predictions, and forget irrelevant data.



A Box of Cereal

$3.99

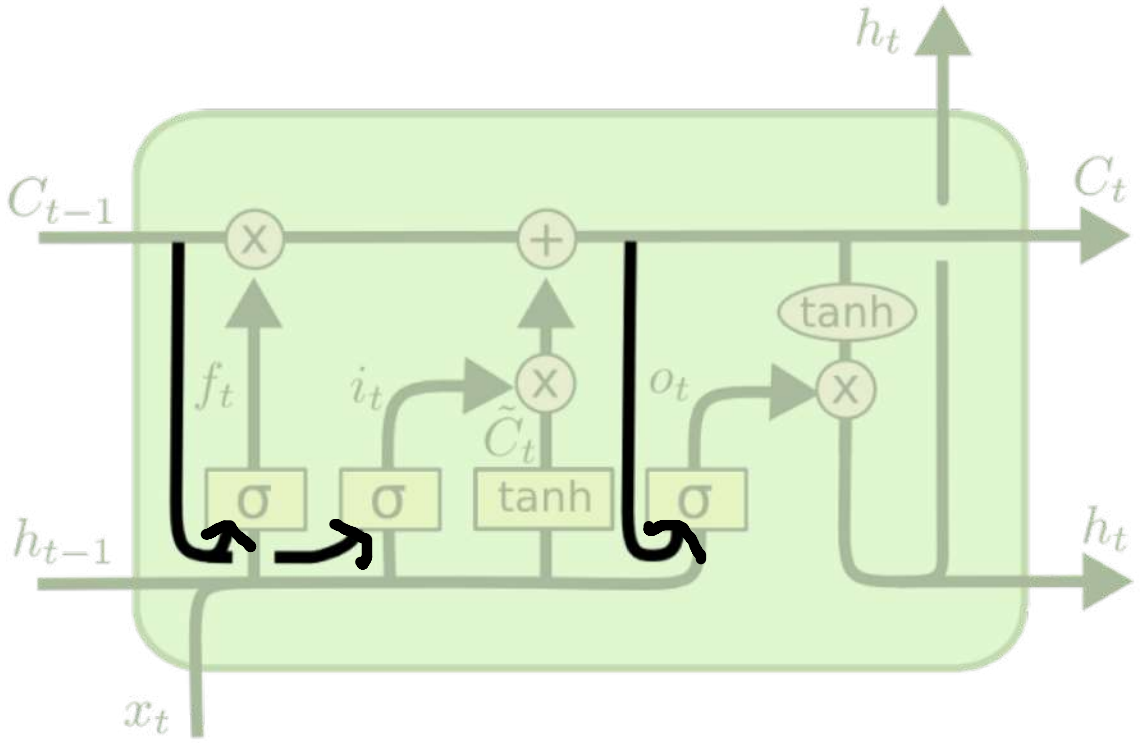
LSTM Variants

LSTM: A search space **odyssey**

K Greff, RK Srivastava, J Koutník… - IEEE transactions on …, 2016 - ieeexplore.ieee.org

In this paper, we present the first large-scale analysis of eight LSTM variants on three representative tasks: speech recognition, handwriting recognition, and polyphonic music modeling. The hyperparameters of all LSTM variants for each task were optimized separately using random search, and their importance was assessed using the powerful functional ANalysis Of VAriance framework. In total, we summarize the results of 5400 experimental runs (≈15 years of CPU time), which makes our study the largest of its kind on LSTM networks. Our results show that none of the variants can improve upon the standard LSTM architecture significantly, and demonstrate the forget gate and the output activation function to be its most critical components. We further observe that the studied hyperparameters are virtually independent and derive guidelines for their efficient adjustment.

Peephole LSTM



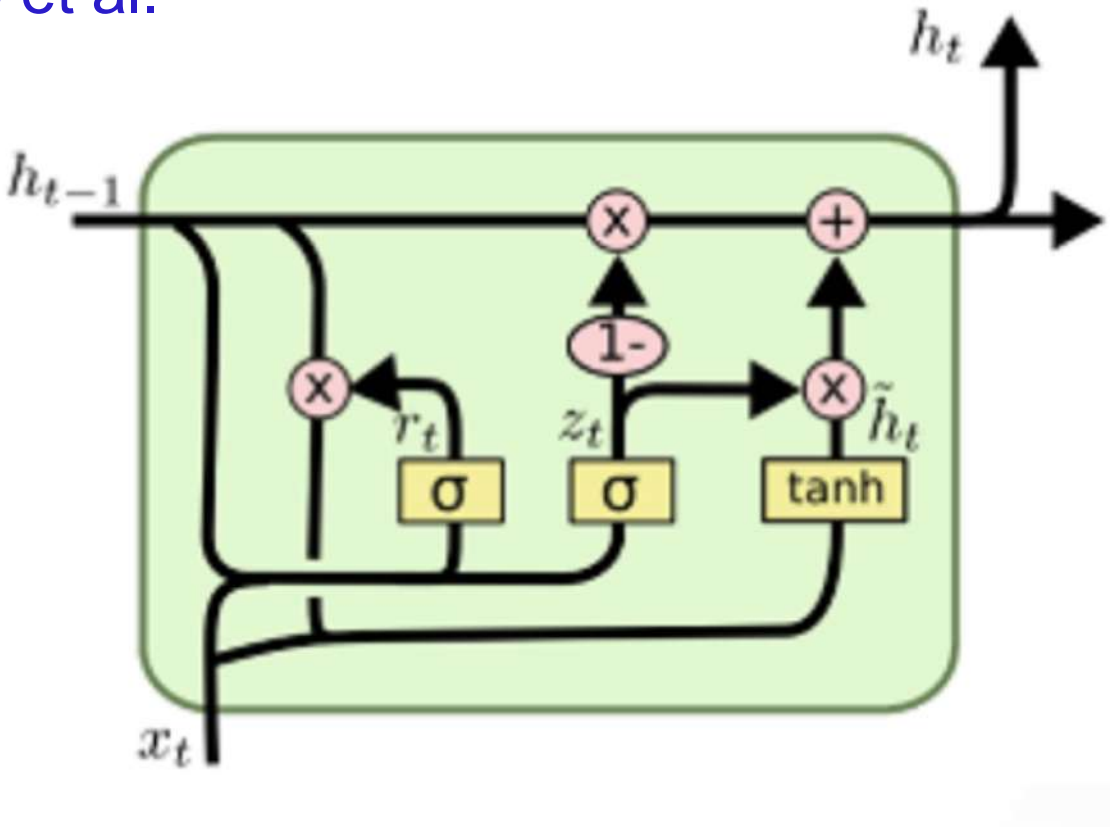
$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \; + \; b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \; + \; b_o\right)$$

The 3 lines from the memory cell *C* to the 3 gates *f, i,* and *o* represent the *peephole* connections. All the gates are having an input along with the cell state

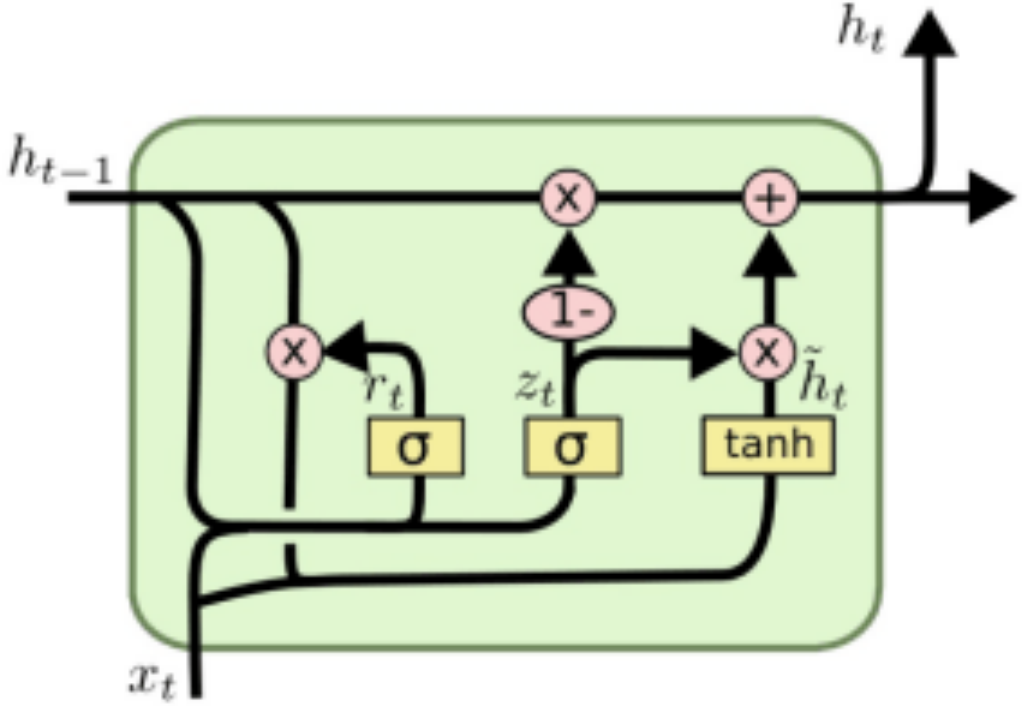
GRU

Introduced in 2014 by Kyunghyun Cho et al.

- **GRU got rid of the cell state and used the hidden state to transfer information.**
- **It also only has two gates, a reset gate and update gate.**



GRU



Update gate: controls what parts of hidden state are updated vs preserved

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

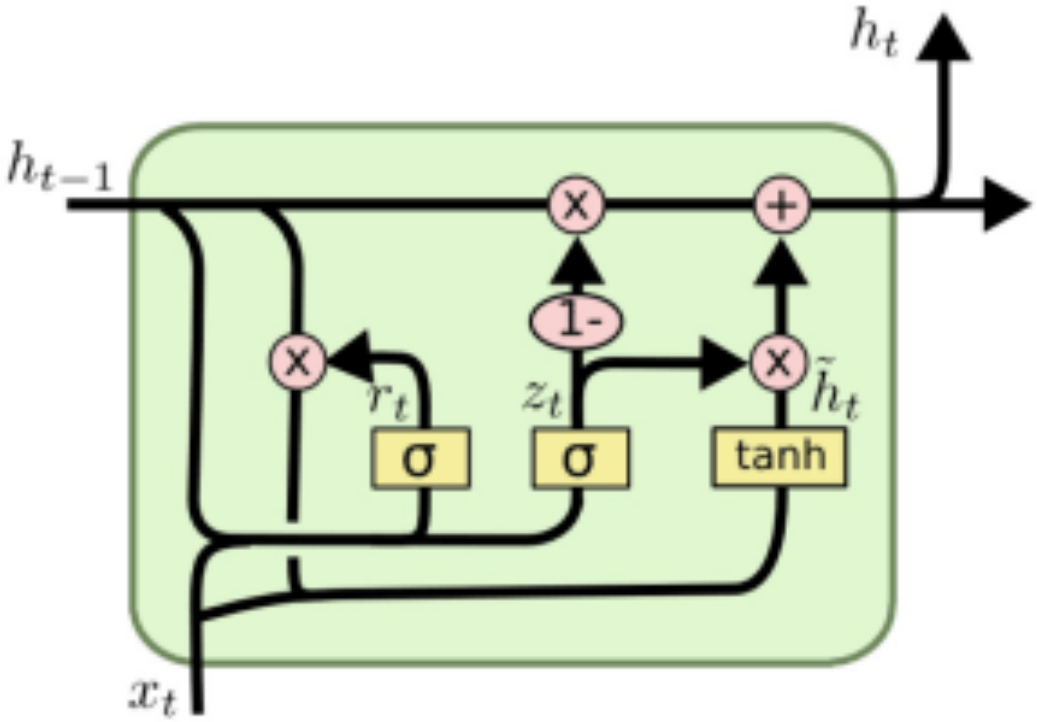
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

Acts similar to the forget plus input gate of an LSTM. It decides what information to throw away and what new information to add.

GRU



Reset gate: controls what parts of previous hidden state are used to compute new content.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

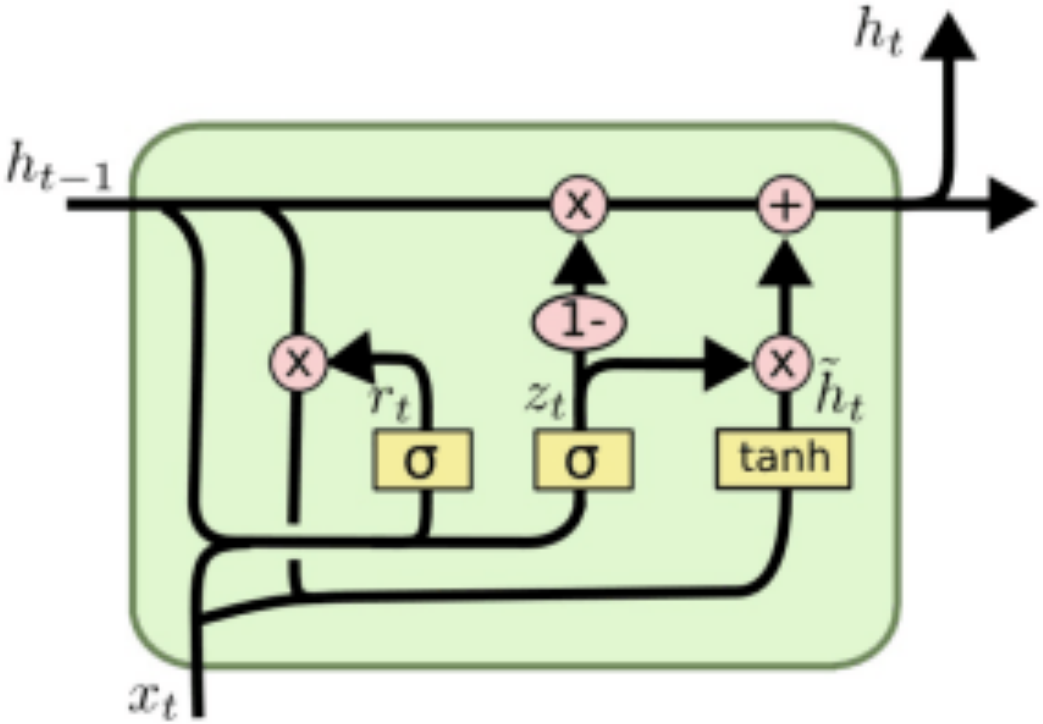
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

Works like the output gate in LSTM.

GRU



New hidden state content: reset gate selects useful parts of prev hidden state. Use this and current input to compute new hidden content.

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

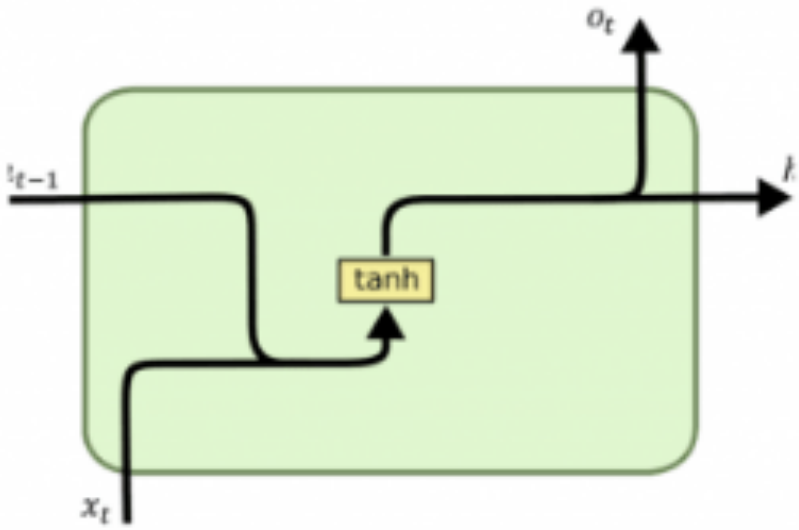
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

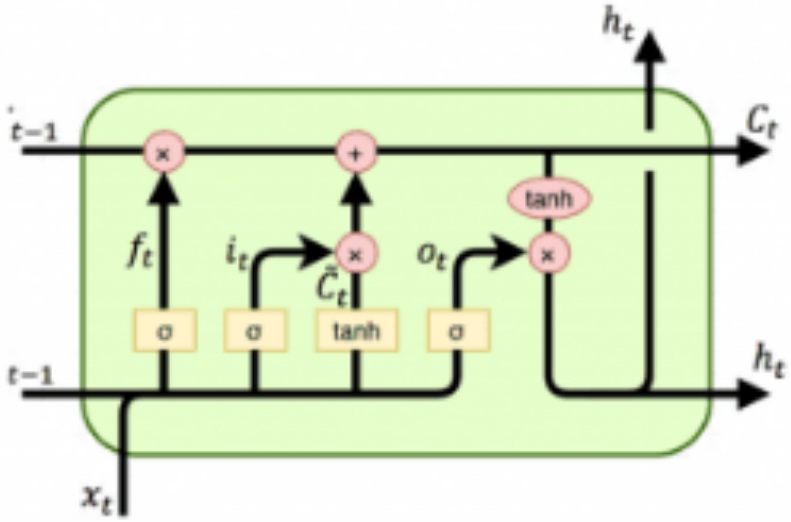
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU

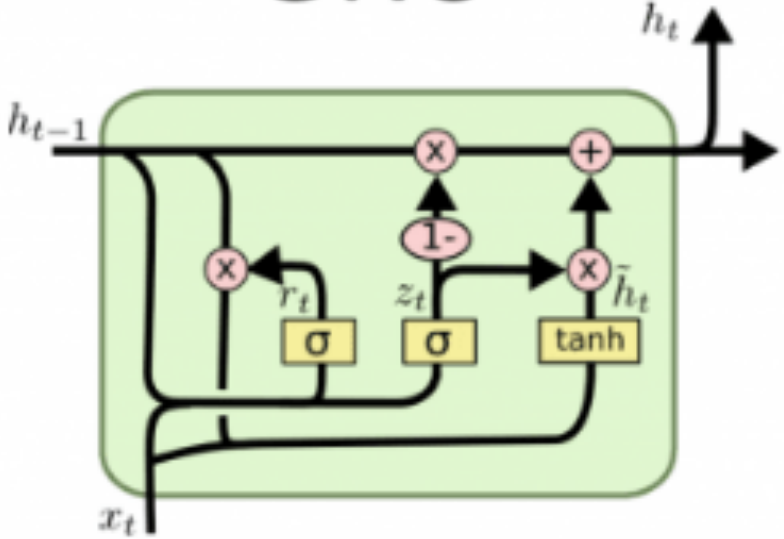
RNN



LSTM



GRU



LSTM vs. GRU

- GRU's performance on certain tasks of polyphonic music modeling and speech signal modeling was found to be similar to that of LSTM.
- GRUs have been shown to exhibit even better performance on certain smaller datasets. (2014)
- Shown by Gail Weiss & Yoav Goldberg & Eran Yahav, the LSTM is "strictly stronger" than the GRU. (2018)
- GRU fails to learn simple languages that are learnable by the LSTM. (2018)
- Shown by Denny Britz & Anna Goldie & Minh-Thang Luong & Quoc Le of Google Brain, LSTM consistently outperform GRU in "the first large-scale analysis of architecture variations for Neural Machine Translation." (2018)

Multi-layer RNN and LSTM

Multilayer RNNs

$$h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$h \in \mathbb{R}^n$ $\quad$ $W^l \ [n \times 2n]$

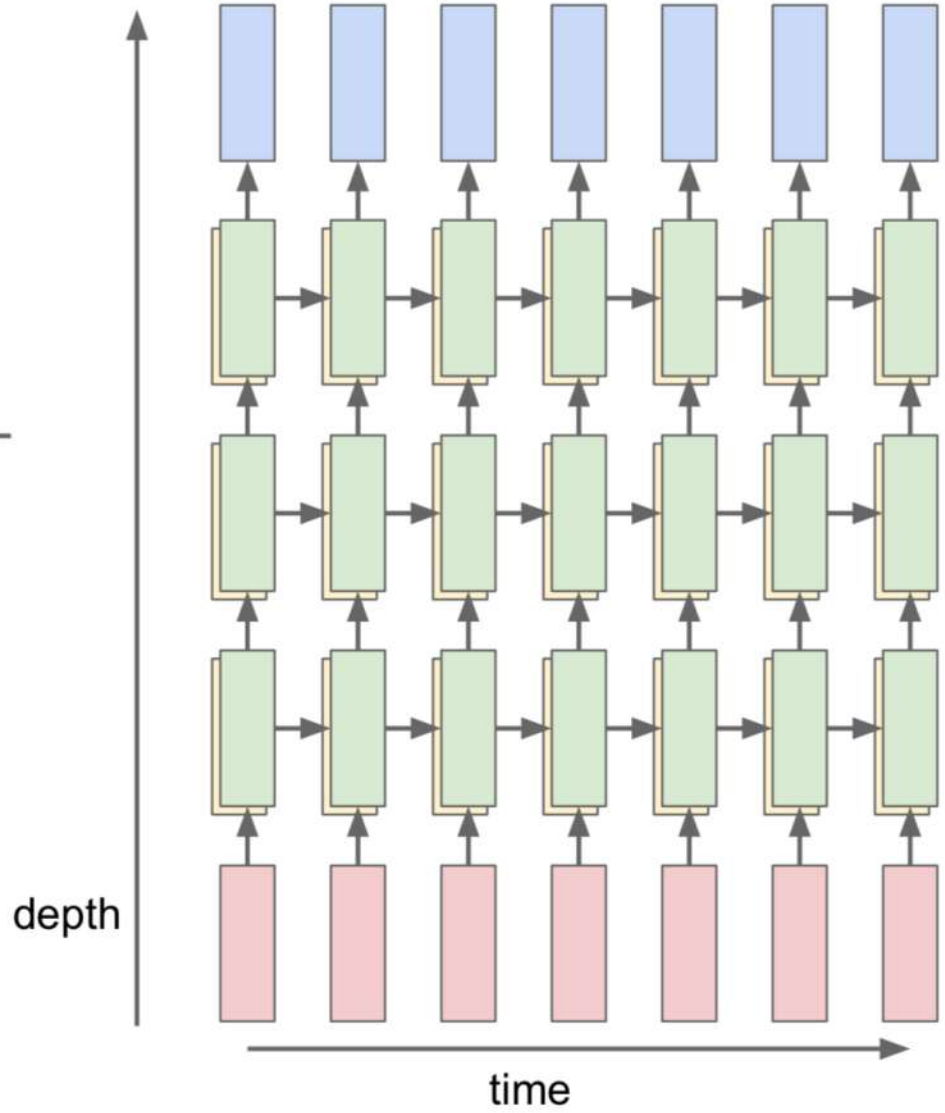
LSTM:

$W^l \ [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ \tilde{c}_t \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

$$c_t^l = f \odot c_{t-1}^l + i \odot \tilde{c}_t$$

$$h_t^l = o \odot \tanh(c_t^l)$$

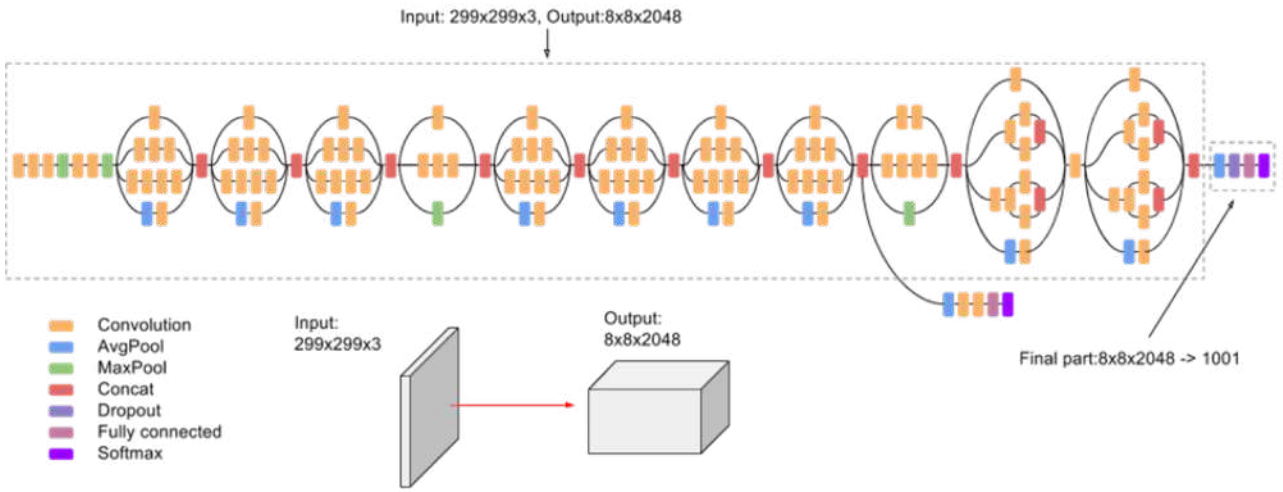


Some Previous Business

Quality Measure of VAE

Inception Score (IS)

- Apply an Inception-v3 network pre-trained on ImageNet to generated samples
- Compare the conditional label distribution with the marginal label distribution:



$$\mathrm{IS} = \exp\left(\mathbb{E}_{x \sim p_g} D_{KL}(p(y|x)||p(y))\right)$$

Higher scores are better, corresponding to a larger KL-divergence between the two distributions.

Quality Measure of VAE

Fréchet Inception Distance (FID)

Improve the IS by *comparing* the statistics of generated samples to real samples (Heusel, Ramsauer, Unterthiner, Nessler, & Hochreiter, 2017)

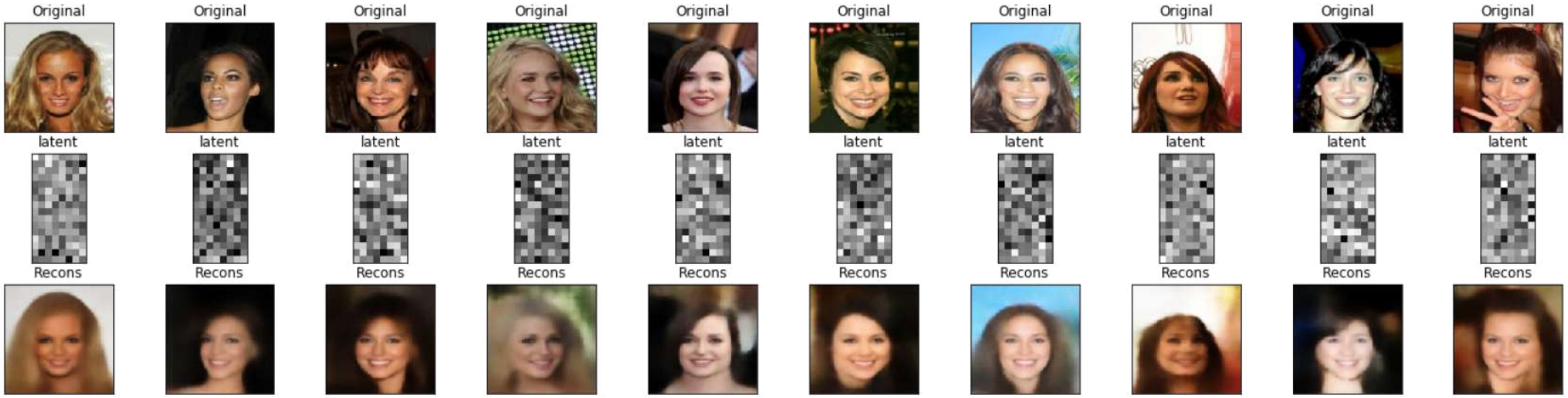
$$\text{FID} = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}),$$

where $X_r \sim \mathcal{N}(\mu_r, \Sigma_r)$ and $X_g \sim \mathcal{N}(\mu_g, \Sigma_g)$ are the 2048-dimensional activations of the Inception-v3 pool3 layer for real and generated samples respectively.[2]

Lower FID is better, corresponding to more similar real and generated samples as measured by the distance between their activation distributions.

What's expected in HW3

- Reconstruction



- Generation

- Compute FID scores

```
from utils import getFID
getFID(real_images, generated_images)
```

Quality Measure of Language Model

Perplexity

- In information theory, **perplexity** is a measurement of how well a probability distribution or probability model predicts a test data
- The perplexity of a discrete probability distribution *p* is defined as

$$2^{H(p)} = 2^{-\sum_x p(x) \log_2 p(x)}$$

Perplexity is just an *exponentiation of the entropy*

- A low **perplexity** indicates the probability distribution is good at predicting the sample.

Quality Measure of Language Model

- In natural language processing, **perplexity** is a way of evaluating language models.
- A language model is a probability distribution over entire sentences or texts.
- A Language Model (LM) score (metric is **perplexity**) represents the degree of 'uncertainty' a model has in predicting some text.

Quality Measure of Language Model

```
import math
from pytorch_pretrained_bert import OpenAIGPTTokenizer, OpenAIGPTModel, OpenAIGPTLMHeadModel
# Load pre-trained model (weights)
model = OpenAIGPTLMHeadModel.from_pretrained('openai-gpt')
model.eval()
# Load pre-trained model tokenizer (vocabulary)
tokenizer = OpenAIGPTTokenizer.from_pretrained('openai-gpt')

def score(sentence):
    tokenize_input = tokenizer.tokenize(sentence)
    tensor_input = torch.tensor([tokenizer.convert_tokens_to_ids(tokenize_input)])
    loss=model(tensor_input, lm_labels=tensor_input)
    return math.exp(loss)


a=['there is a book on the desk',
                'there is a plane on the desk',
                        'there is a book in the desk']
print([score(i) for i in a])
21.31652459381952, 61.45907380241148, 26.24923942649312
```

- HW3 Compute LM scores

```
from utils import lmScore
lmscore(text)
```

- Next class: Exam
- Covers everything up to this lecture
- Two hours
- 30%: T/F, Multiple Choices
- 20%: Easy
- 20%: Moderate
- 30%: Needs thinking