香 港 中 文 大 學（深 圳）
The Chinese University of Hong Kong, Shenzhen

DDA4220 Deep Learning and Applications

Lecture 6 Recurrent Neural Network

Ruimao Zhang
zhangruimao@cuhk.edu.cn

School of Data Science
The Chinese University of Hong Kong (Shenzhen)

Slides partially credited to Prof. Hongsheng Li at CUHK

Outline

- **Recurrent neural networks**
  - Recurrent neural networks
  - Variants of RNN
- **Advanced RNN Variants**
  - Challenge of long-term dependency
  - Long Short-Term Memory (LSTM) Net
  - Gated Recurrent Unit
- **Applications**

Sequential data

- Sequence of words in an English sentence
- Acoustic features at successive time frames in speech recognition
- Successive frames in video classification
- Rainfall measurements on successive days in Shenzhen
- Daily values of current exchange rate
- Nucleotide base pairs in a strand of DNA
- **Instead of making independent predictions on samples, assume the dependency among samples and make a sequence of decisions for sequential samples**

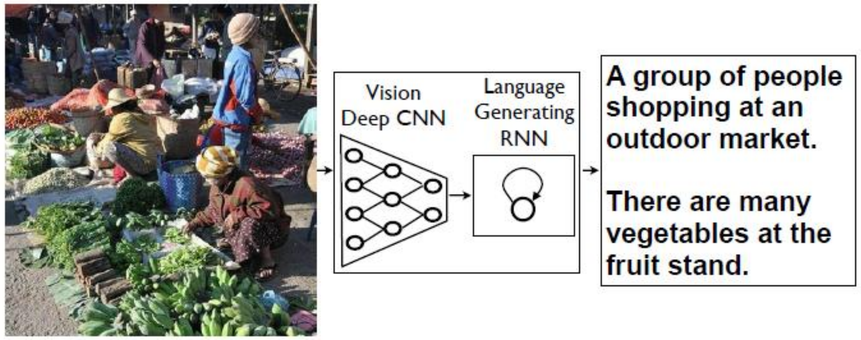
Modeling sequential data

- Sample data sequences from a certain distribution

$$P(x_1, \ldots, x_T)$$

- Generate natural sentences to describe an image

$$P(y_1, \ldots, y_T | I)$$


Vision Deep CNN
Language Generating RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

- Activity recognition from a video sequence

$$P(y | x_1, \ldots, x_T)$$

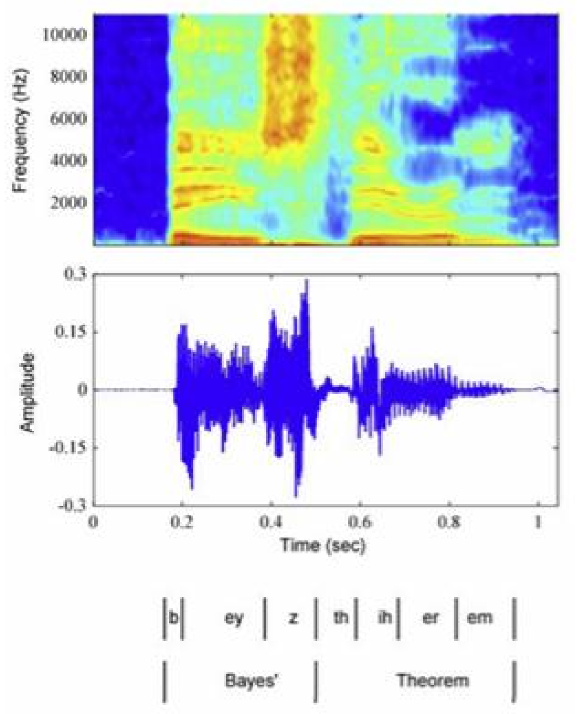
Modeling sequential data

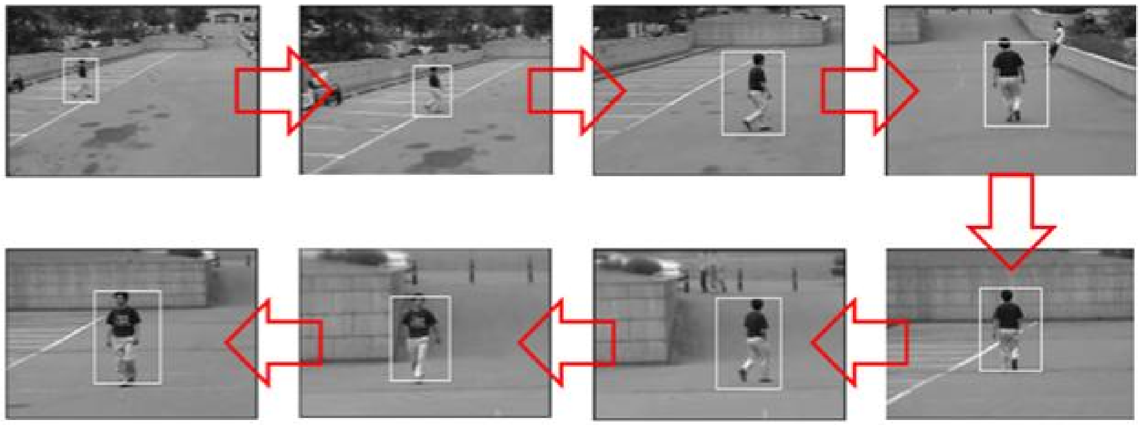
- Speech recognition

$$P(y_1, \ldots, y_T | x_1, \ldots, x_T)$$

- Object tracking

$$P(y_1, \ldots, y_T | x_1, \ldots, x_T)$$





Modeling sequential data

- Use the chain rule to express the joint distribution for a sequence of observations

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

- Impractical to consider general dependence of future dependence on all previous observations

$$p(x_t | x_{t-1}, \ldots, x_0)$$

- Complexity would grow without limit as the number of observations increases
- It is expected that recent observations are more informative than more historical observations in predicting future values

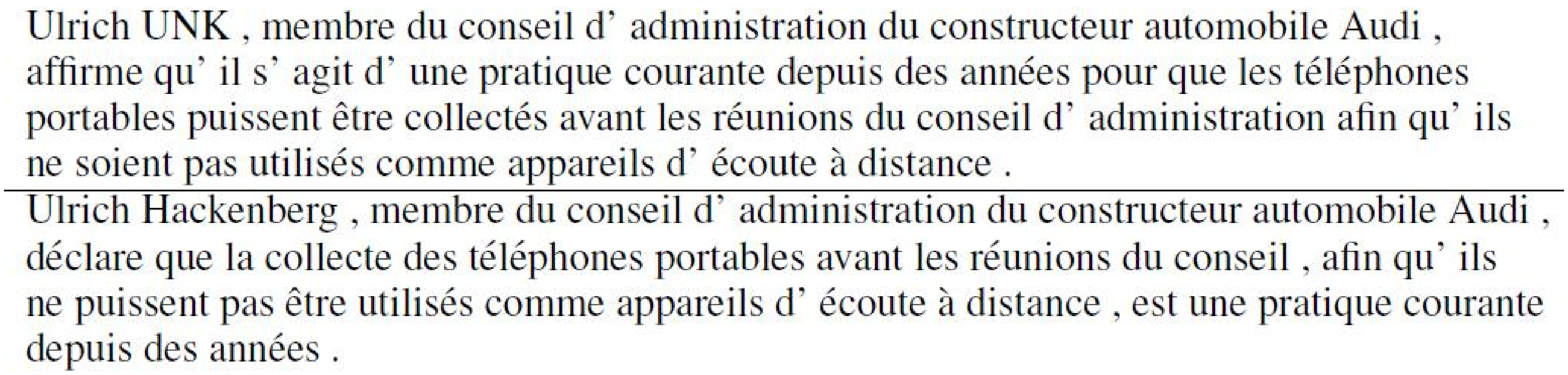
Modeling sequential data

- Generate natural sentences to describe a video

$$P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$$

- Language translation

$$P(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$$



Ulrich UNK , membre du conseil d’ administration du constructeur automobile Audi , affirme qu’ il s’ agit d’ une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d’ administration afin qu’ ils ne soient pas utilisés comme appareils d’ écoute à distance .

Ulrich Hackenberg , membre du conseil d’ administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu’ ils ne puissent pas être utilisés comme appareils d’ écoute à distance , est une pratique courante depuis des années .

Modeling sequential data

- Use the chain rule to express the joint distribution for a sequence of observations

$$p(x_1, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

- Impractical to consider general dependence of future dependence on all previous observations

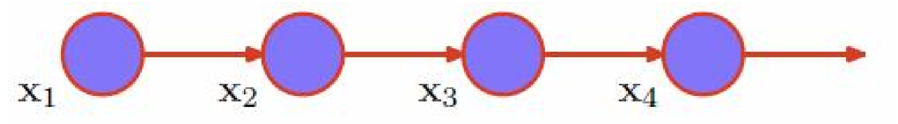
$$p(x_t | x_{t-1}, \ldots, x_0)$$

- Complexity would grow without limit as the number of observations increases
- It is expected that recent observations are more informative than more historical observations in predicting future values

Markov models

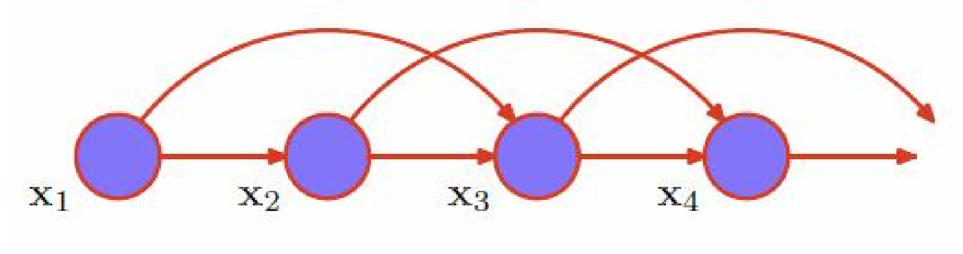
- Markov models assume dependence on most recent observations
- First-order Markov model

$$p(x_1, \dots, x_T) = \prod_{t=1}^{T} p(x_t | x_{t-1})$$



- Second-order Markov model

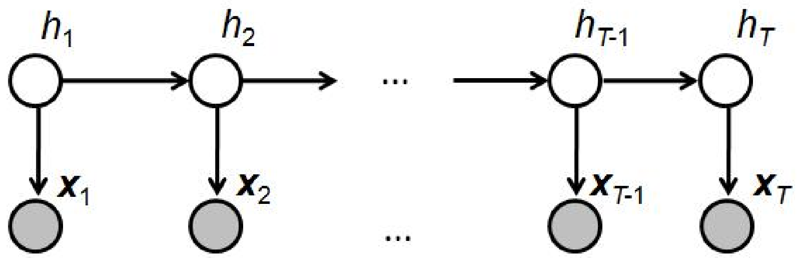
$$p(x_1, \dots, x_T) = \prod_{t=1}^{T} p(x_t | x_{t-1}, x_{t-2})$$



Hidden Markov Model (HMM)

- A classical way to model sequential data
- Sequence pairs $h_1, h_2, \ldots, h_T$ (hidden variables) and $x_1, x_2, \ldots, x_T$ (observations) are generated by the following process
  - Pick h_1 at random from the distribution $P(h_1)$. Pick x_1 from the distribution $p(x_1|h_1)$
  - For *t* = *2* to *T*
    - Choose h_t at random from the distribution $p(h_t|h_{t-1})$
    - Choose x_t at random from the distribution $p(x_t|h_t)$
- The joint distribution is

$$p(x_1, \ldots, x_T, h_1, \ldots, h_T, \theta) = P(h_1)\prod_{t=2}^{T} P(h_t|h_{t-1})\prod_{t=1}^{T} p(x_t|h_t)$$

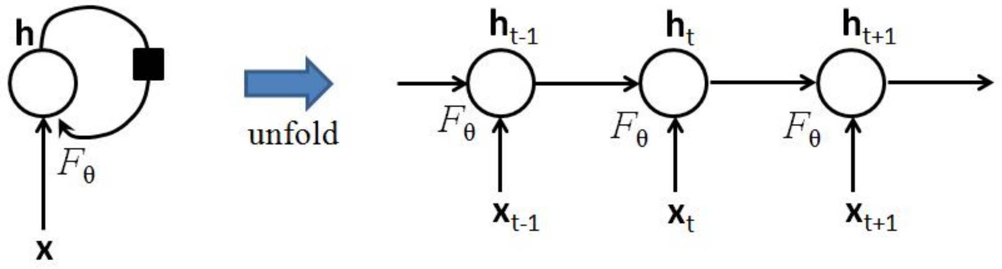


Recurrent neural networks (RNN)

- HMM can be considered as a generative model while RNN is a discriminative model
- Model a dynamic system driven by an external signal x_t

$$h_t = F_\theta(h_{t-1}, x_t)$$

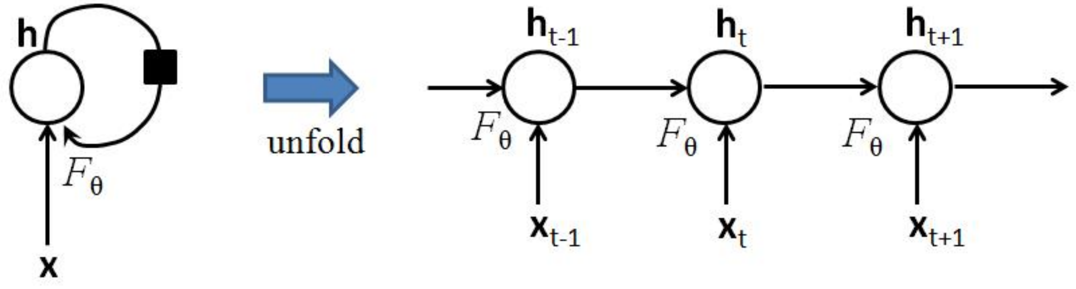
- h_t contains information about the whole past sequence. The equation above implicitly defines a function which maps the whole past sequence $(x_t, \ldots, x_1)$ to the current sate $h_t = G_t(x_t, \ldots, x_1)$



Left: Implementation of RNN, viewed as a circuit. The black square indicates a delay of 1 time step. **Right:** the same RNN viewed as an unfolded flow graph, where each node is now associated with one particular time step.

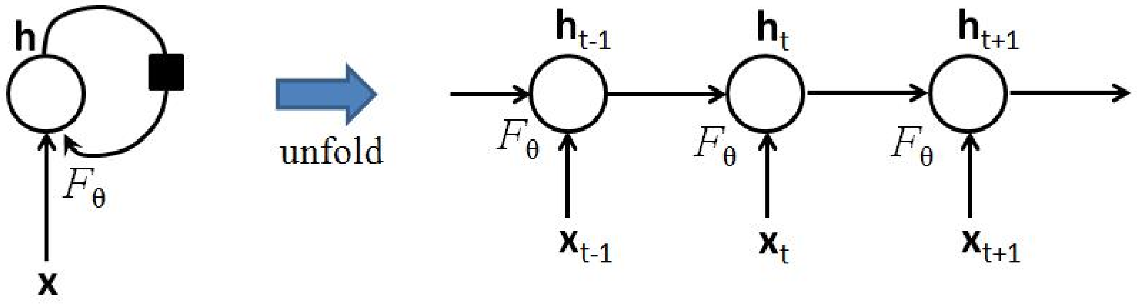
Recurrent neural networks (RNN)

- The summary is lossy, since it maps an arbitrary length sequence $(x_t, \ldots, x_1)$ to a fixed length vector h_t
- Depending on the training criterion, h_t keeps some important aspects of the past sequence.
- **Sharing parameters:** the same weights are used for encoding feature representations (neuron responses) at different time steps
- A similar idea with CNN: replacing a fully connected network with local connections with parameter sharing
- It allows to apply the network to input sequences of different lengths and predicts sequences of different lengths



Recurrent neural networks (RNN)

- **Sharing parameters for any sequence length allows more better generalization properties.** If we have to define a different function G_t for each possible sequence length, each with its own parameters, we would not get any generalization to sequences of a size not seen in the training set
- One would need to see a lot more training examples, because a separate model would have to be trained for each sequence length

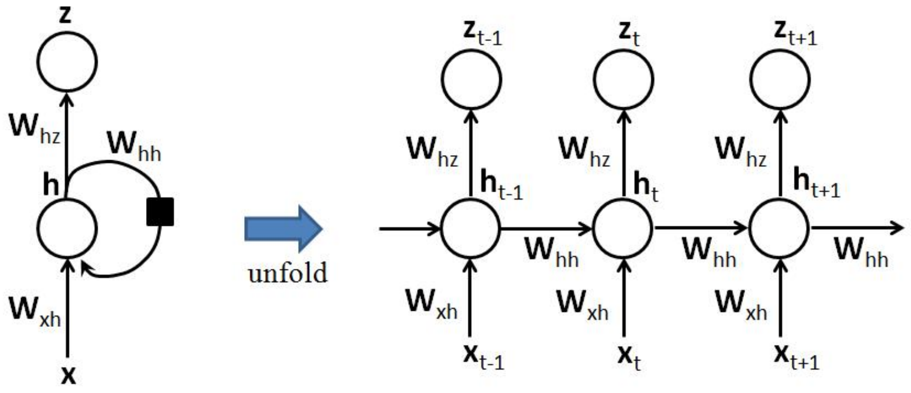


A vanilla RNN to predict sequences from input

$$P(y_1, \ldots, y_T | x_1, \ldots, x_T)$$

- Forward propagation equations, assuming that hyperbolic tangent non-linearities are used in the hidden units and softmax is used in output for classification problems

$$h_t = \tanh(W_{xh} x_t + W_{hh} h_{t-1} + b_h)$$
$$z_t = \text{softmax}(W_{hz} h_t + b_z)$$
$$p(y_t = c) = z_{t,c}$$

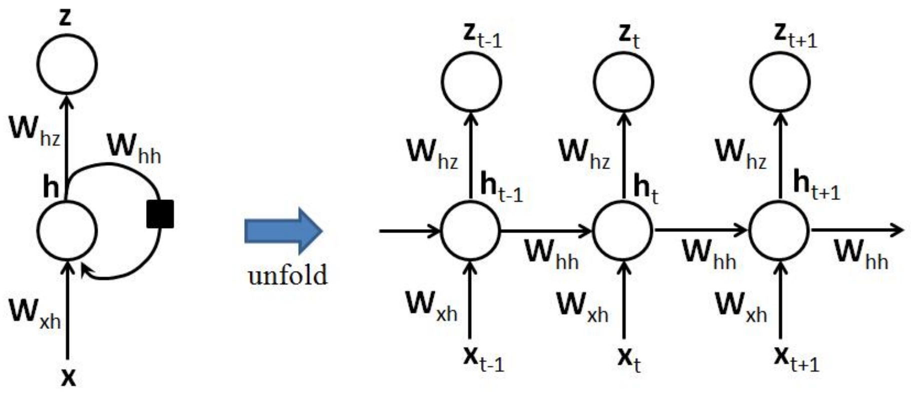


A vanilla RNN to predict sequences from input

$$P(y_1, \ldots, y_T | x_1, \ldots, x_T)$$

- Forward propagation equations, assuming that hyperbolic tangent non-linearities are used in the hidden units and softmax is used in output for classification problems

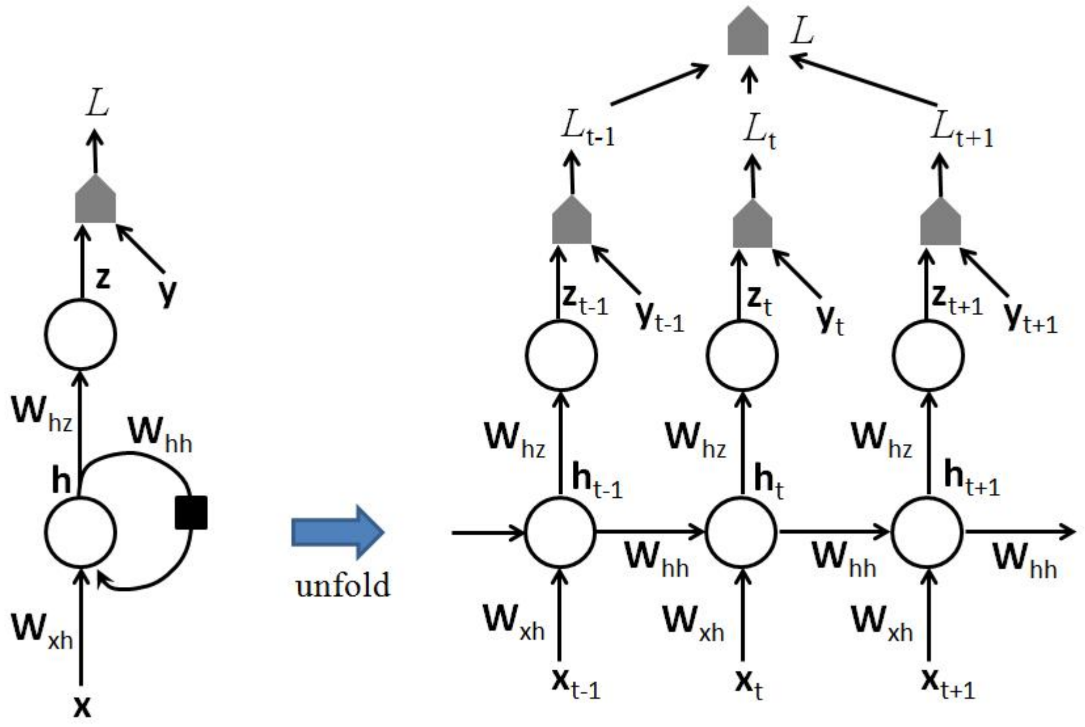
$$h_t = \tanh(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$
$$z_t = \text{softmax}(W_{hz}h_t + b_z)$$
$$p(y_t = c) = z_{t,c}$$



Overall Loss Function

- The total loss for a given input/target sequence pair (x, y), , measured by the cross entropy, is formulated as

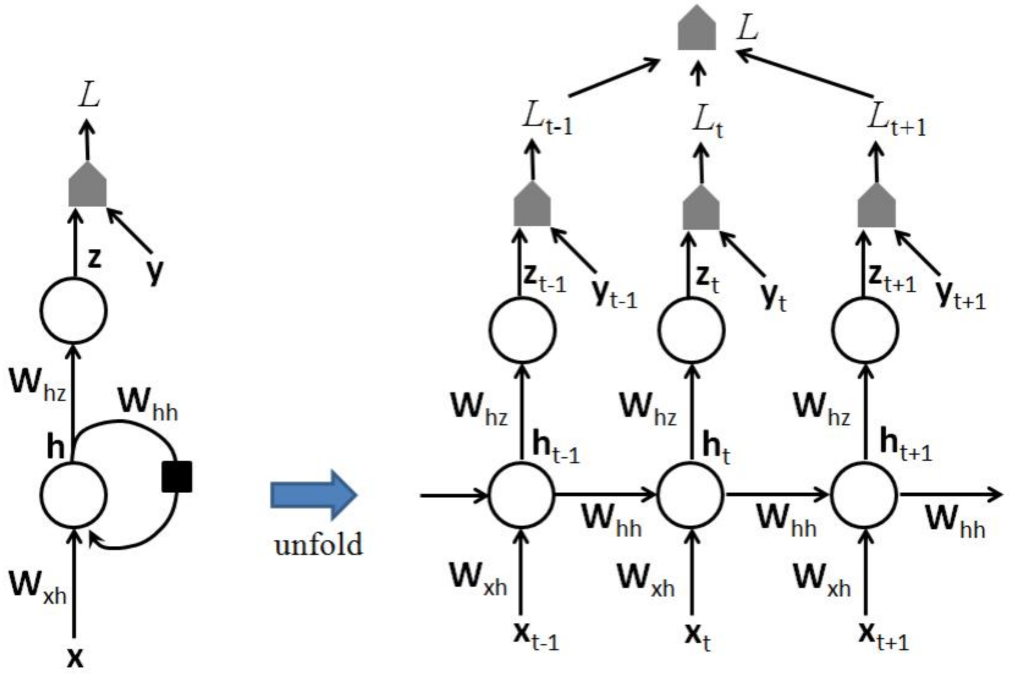
$$L(x,y) = \sum_{t=1}^{T} L_t = \sum_{t=1}^{T} -y_t \log z_t$$



Gradients on W_{hh} and W_{xh}

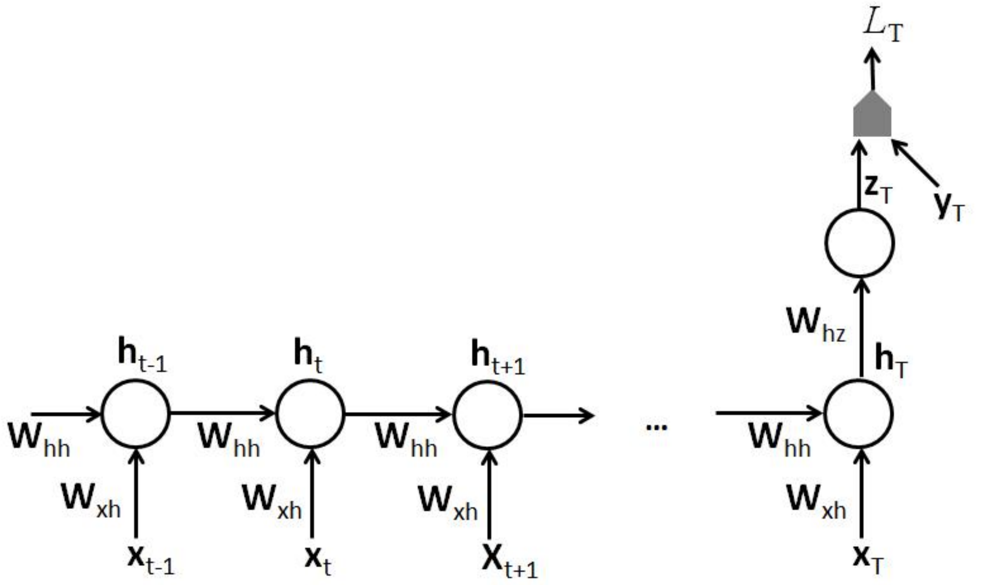
$$\frac{\partial L}{\partial W_{hh}} = \sum_{t=1}^{T} \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial W_{hh}}$$

$$\frac{\partial L}{\partial h_t} = \frac{\partial L}{\partial h_{t+1}} \frac{\partial h_{t+1}}{\partial h_t} + \frac{\partial L}{\partial z_t} \frac{\partial z_t}{\partial h_t}$$



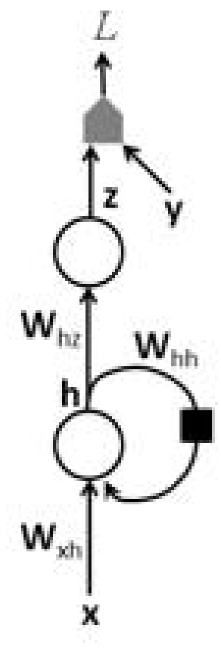
Predict a single output at the end of the sequence

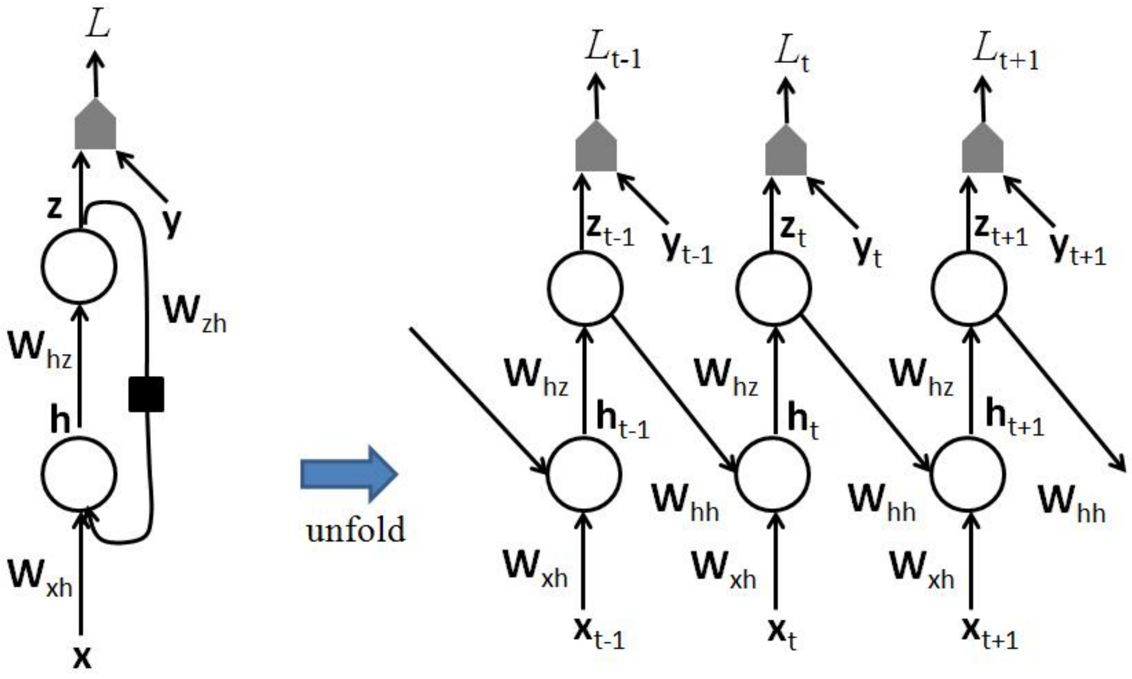
- Such a network can be used to summarize a sequence and produce a fixed-size representation used as input for further processing
- There might be a target right at the end or the gradient on the output z_t can be obtained by back-propagation from further downsteam modules



Network with output recurrence

- The output can be from the prediction of the previous steps, which limits its expressive power but makes it easier to train



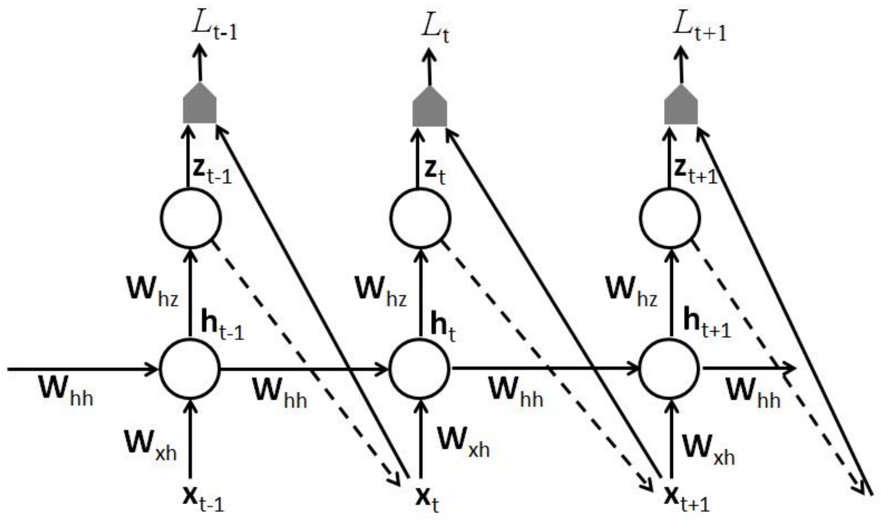


Generative RNN modeling $P(x_1,...,x_T)$

- It can generate sequences from this distribution
- At the training stage, each x_t of the observed sequence serves both as input (for the current time step) and as target (for the previous time step)
- The output z_t encodes the parameters of a conditional distribution

$$P(x_{t+1}|x_1,\ldots,x_t) = P(x_{t+1}|z_t)$$

for x_{t+1} given the past sequence $x_1,\ldots,x_t$



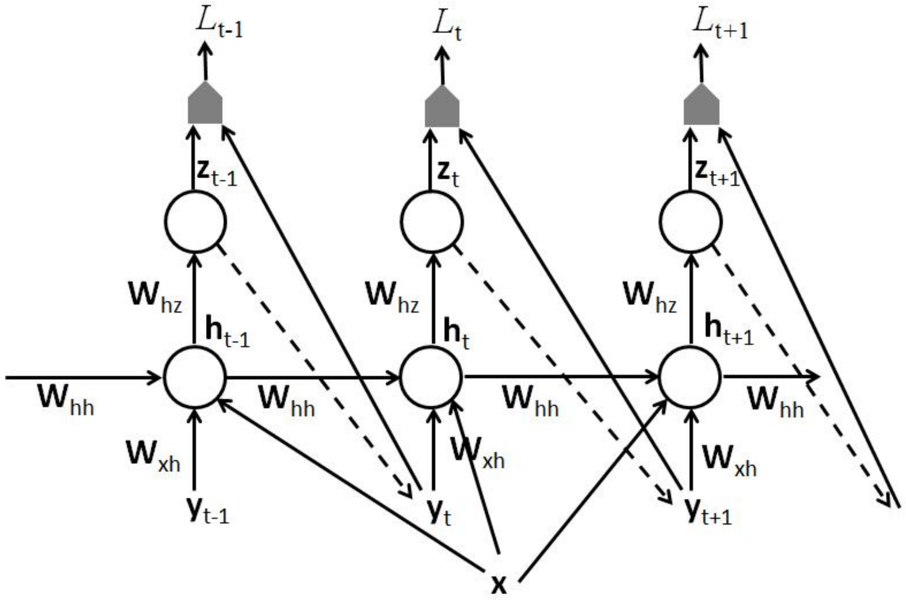
Generative RNN modeling $P(x_1,...,x_T)$

- If RNN is used to generate sequences, one must also incorporate in the output information allowing to stochastically decide when to stop generating new output elements
- In the case when the output is a symbol taken from a vocabulary, one can add a special symbol corresponding to the end of a sequence
- One could also directly directly model the length *T* of the sequence through some parametric distribution. $P(x_1,...,x_T)$ is decomposed into

$$P(x_1,\ldots,x_T) = P(x_1,\ldots,x_T|T)P(T)$$

RNNs to represent conditional distributions $P(y \mid x)$

- If *x* is a fixed-sized vector, we can simply make it an extra input of the RNN that generates the *y* sequence. Some common ways of providing the extra input
  - as an extra input at each time step, or
  - as the initial state h_0 , or
  - both

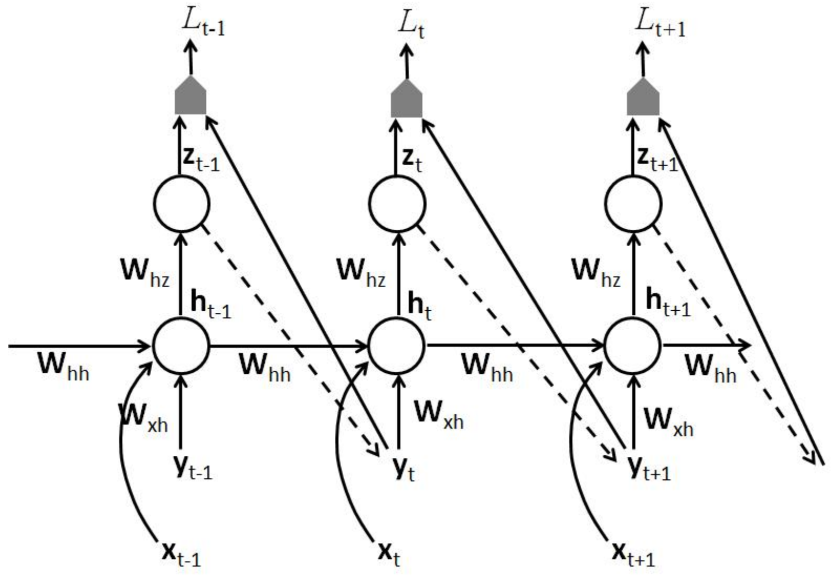


RNNs to represent conditional distributions $P(y \mid x)$

- The input *x* is a sequence of the same length as the output sequence *y*
- Removing the dash lines, it assumes y_t are independent of each other when the past input sequence is given,

$$P(y_t|y_{t-1},\ldots,y_1,x_t,\ldots,x_1) = P(y_t|x_t,\ldots,x_1)$$

- Without the conditional independence assumption, add the dash lines and the prediction of y_{t+1} is based on both the past *x*'s and past *y*'s

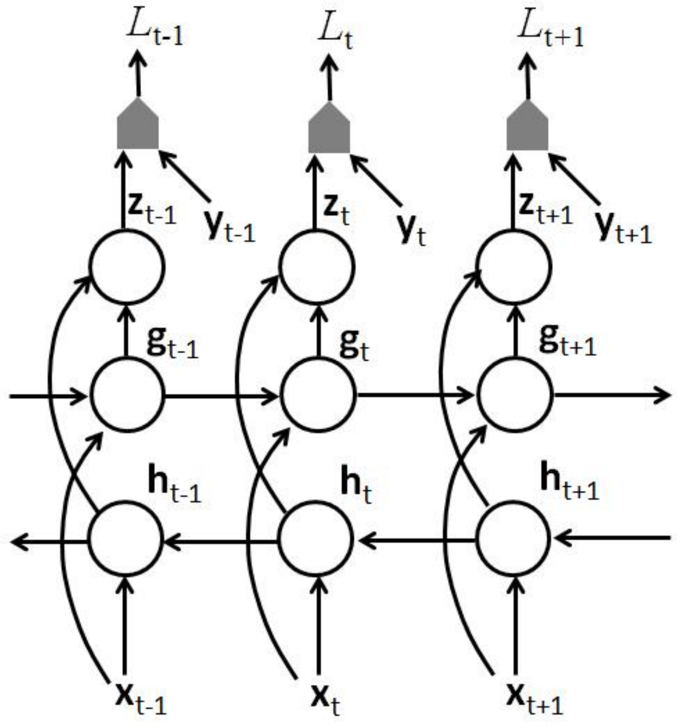


Bidirectional RNNs

- In some applications, we want to output at time *t* a prediction regarding an output which may depend on the whole input sequence
  - In speech recognition, the correct interpretation of the current sound as a phoneme may depend on the next few phonemes because co-articulation and may depend on the next few words because of the linguistic dependencies between words
- Bidirectional recurrent neural network was proposed to address such requirement
- It combines a forward-going RNN and a backward-going RNN
- The idea can be extended to 2D input with four RNN going in four directions

Bidirectional RNNs

- g_t summaries the information from the past sequence, and h_t summaries the information from the future sequence



Difficulty of Learning Long-Term Dependencies

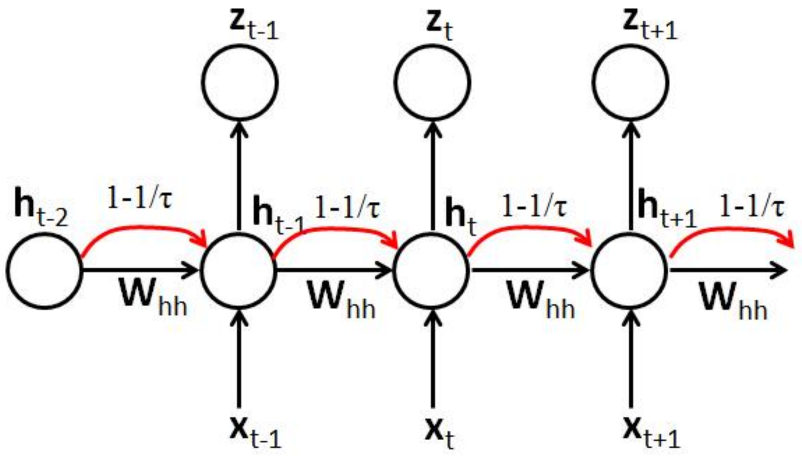
$$\frac{\partial h_T}{\partial h_t} = \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial h_{T-2}} \cdots \frac{\partial h_{t+1}}{\partial h_t}$$

- Each layer-wise Jacobian $\frac{\partial h_{t+1}}{\partial h_t}$ is the product of two matrices: (a) the recurrent matrix *W* and (b) the diagonal matrix whose entries are the derivatives of the non-linearities associated with the hidden units, which variy depending on the time step. This makes it likely that successive Jacobians have simliar eigenvectors, making the product of these Jacobians explode or vanish even faster
- $\frac{\partial L_T}{\partial \theta}$ is a weighted sum of terms over spans [*T, t*], with weights that are exponentially smaller (or larger) for long-term dependencies relating the state at *t* to the state at *T*
- The signal about long term dependecies will tend to be hidden by the smallest fluctuations arising from short-term dependenties

Leaky units with self-connections

$$h_{t+1} = (1 - \frac{1}{\tau_i})h_t + \frac{1}{\tau_i}\tanh(W_{xh}x_t + W_{hh}h_t + b_h)$$

- The new value of the state h_{t+1} is a combination of linear and non-linear parts of h_t
- The errors are easier to be back propagated through the paths of red lines, which are linear



Leaky units with self-connections

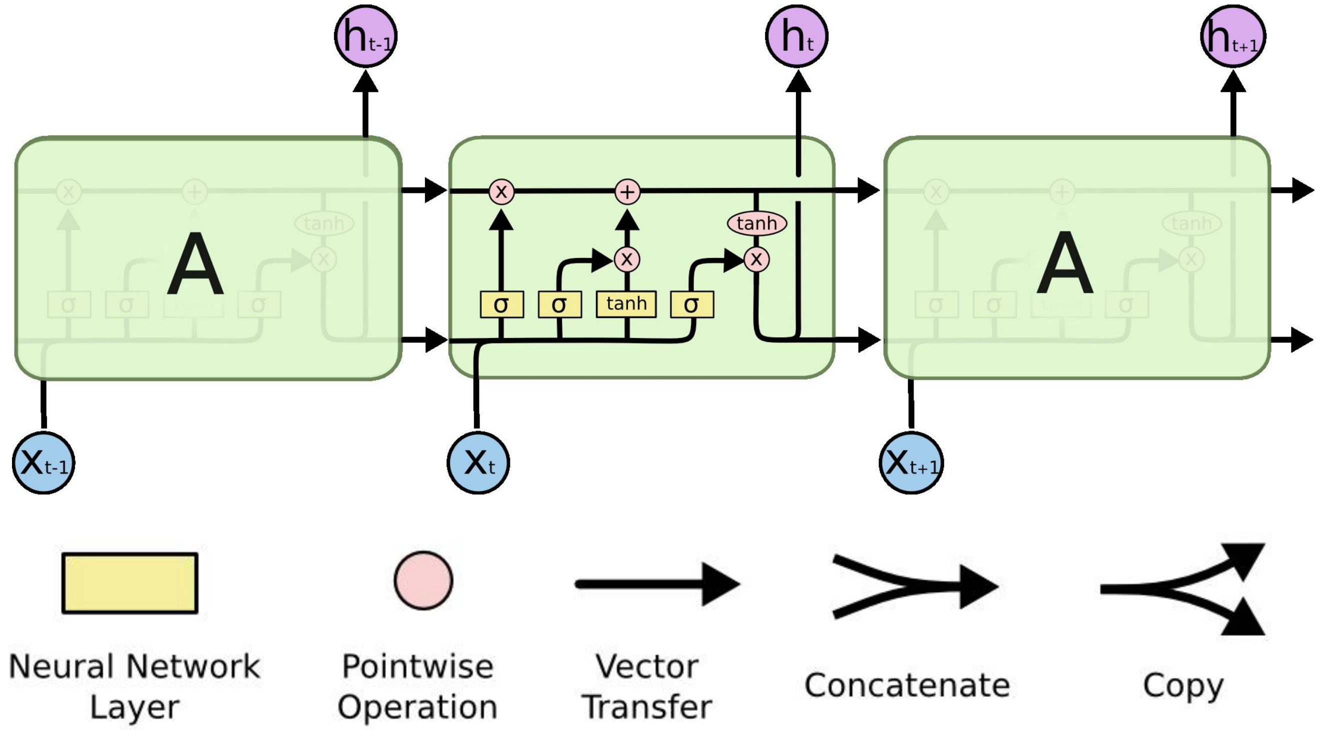
- When $\tau = 1$, there is no linear self-recurrence, only the nonlinear update which we can find in ordinary recurrent networks
- When $\tau > 1$, this linear recurrence allows gradients to propagate more easily. When τ is large, the sate changes very slowly, integrating the past values associated with the input sequence
- τ controls the rate of forgetting old states. It can be viewed as a smooth variant of the idea of the previous model
- By associating different time scales τ with different units, one obtains different paths corresponding to different forgetting rate
- Those time constants can be fixed manually or can be learned as free parameters

Long Short-Term Memory (LSTM) net

- In the leaky units with self-connections, the forgetting rate is constant during the whole sequence.
- The role of leaky units is to accumulate information over a long duration. However, once that information gets used, it might be useful for the neural network to forget the old state
  - For example, if a video sequence is composed as subsequences corresponding to different actions, we want a leaky unit to accumulate evidence inside each subsequnece, and we need a mechanism to forget the old state by setting it to zero and starting to count from fresh when starting to process the next subsequence
- The forgetting rates are expected to be different at different time steps, depending on their previous hidden states and current input (conditioning the forgetting on the context)
- Parameters controlling the forgetting rates (not forgetting gates themselves) are learned from train data

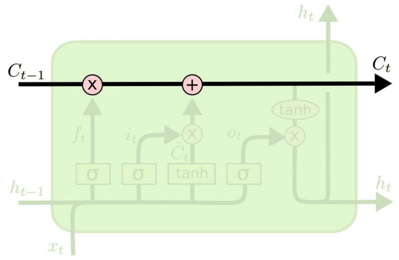
Long Short-Term Memory (LSTM) net

- LSTMs also have this chain like structure, but the repeating module has a different structure.
- Instead of having a single neural network layer, there are four, interacting in a very special way

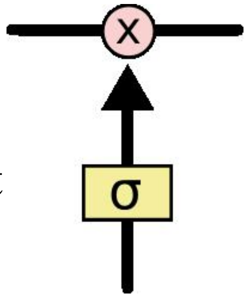


Long Short-Term Memory (LSTM) net

- The key to LSTMs is the cell state. It's very easy for information to just flow along it unchange

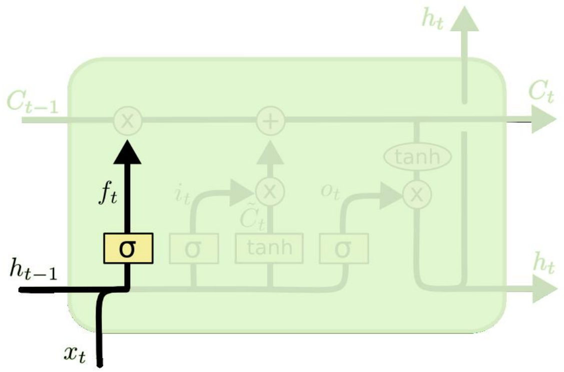


- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates
- Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation
- The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through"



Long Short-Term Memory (LSTM) net

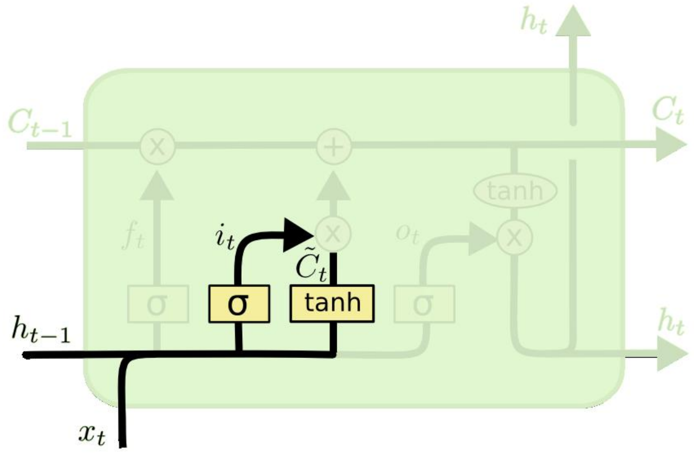
- The first step in our LSTM is to decide what information we are going to throw away from the cell state
- This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}
- A 1 represents "completely keep this" while a 0 represents "completely get rid of this"



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

Long Short-Term Memory (LSTM) net

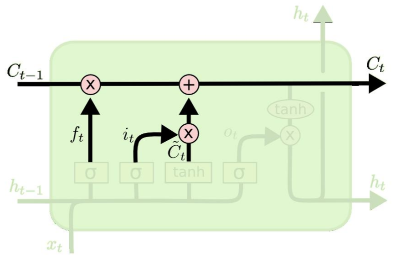
- The next step is to decide what new information we are going to store in the cell state
- First, a sigmoid layer called the "input gate layer" decides which values we'll update
- Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state



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

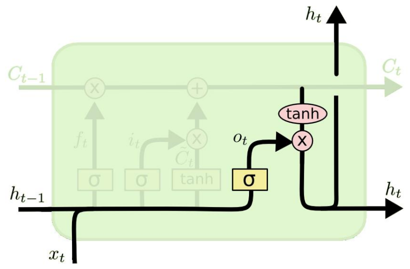
Long Short-Term Memory (LSTM) net

- Now, it is time to update the old cell state, C_{t-1} , into the new cell state C_t



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

- Finally, decide what are going to be output. This output will be based on our cell state, but will be a filtered version
- Run a sigmoid layer which decides what parts of the cell state we are going to output
- Put the cell state through tanh (to push the values to be between -1 and 1 and multiply it by the output of the sigmoid gate to output only a part

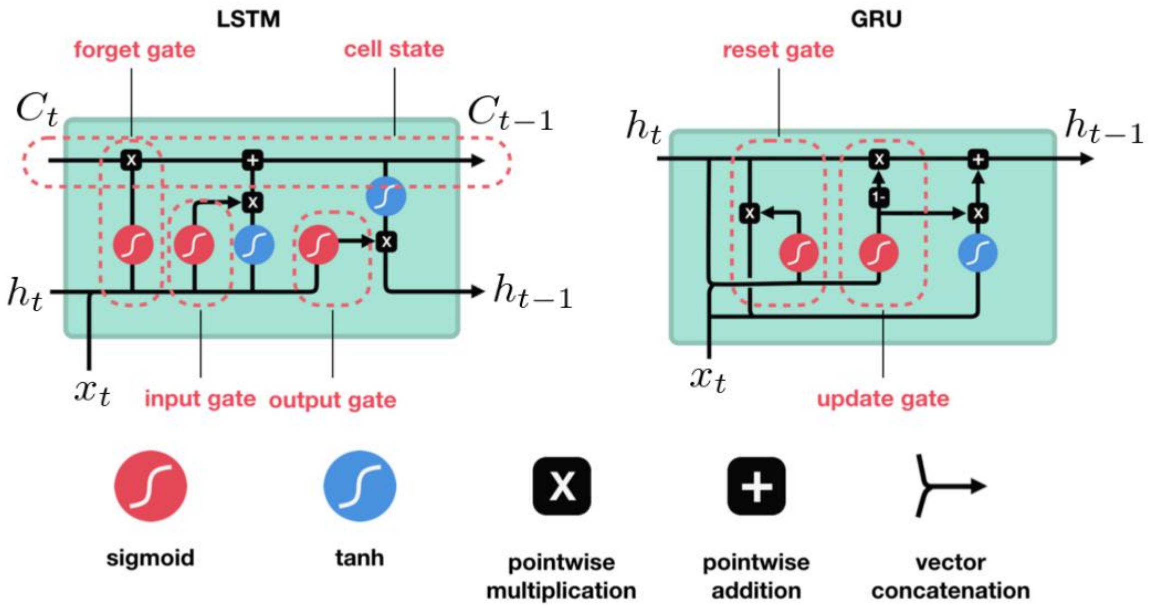


$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Gated Recurrent Unit (GRU) network

- GRU abandons the cell state and used hidden state to transfer information. It also only has two gates, a reset gate and update gate
- **Update gate:** it helps the model to determine how much of the past information (from previous time steps) needs to be passed along to the future

$$z_t = \sigma(W_z x_t + U_z h_{t-1})$$



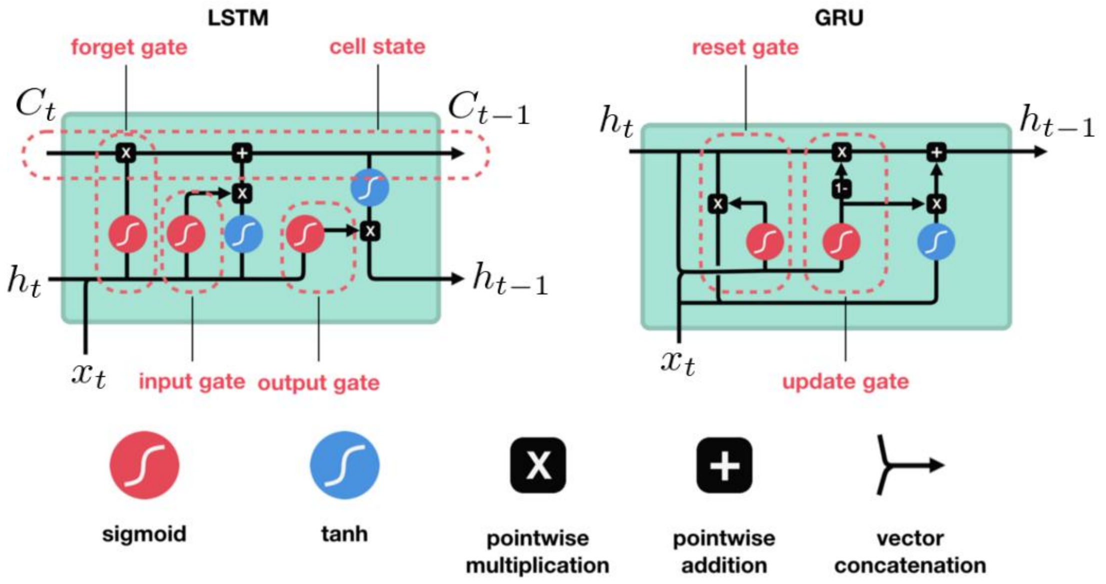
Gated Recurrent Unit (GRU) network

- **Reset gate:** it decides how much of the past information to forget

$$r_t = \sigma(W_r x_t + U_r h_{t-1})$$

- New memory content is calculated as $h'_t = \tanh(W x_t + r_t \odot U h_{t-1})$
- Final memory at current time step: it determines what to collect from the current memory content h'_t and what from the previous steps h_{t-1} ?

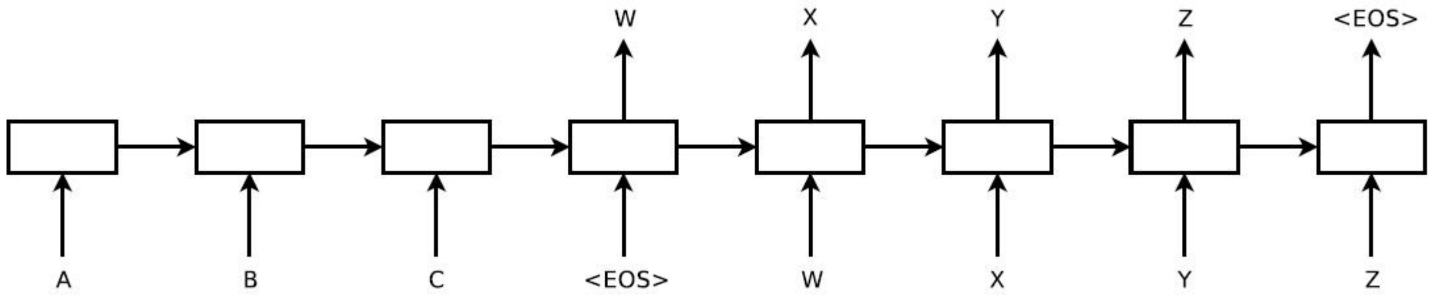
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t$$



Sequence-to-sequence language translation

- Model $P(\mathbf{y}_1, \ldots, \mathbf{y}_{T'} | \mathbf{x}_1, \ldots, \mathbf{x}_T)$. The input and output sequences have different lengths, are not aligned, and even do not have monotonic relationship
- Use one LSTM to read the input sequence $(\mathbf{x}_1, \ldots, \mathbf{x}_T)$, one timestep at a time, to obtain a large fixed-dimensional vector representation **v**, which is given by the last hidden state of the LSTM
- Then conditioned on v, a second LSTM generates the output sequence $(\mathbf{y}_1, \ldots, \mathbf{y}_{T'})$ and computes its probability

$$p(\mathbf{y}_1, \ldots, \mathbf{y}_{T'} | \mathbf{v}) = \prod_{t=1}^{T'} p(\mathbf{y}_t | v, \mathbf{y}_1, \ldots, \mathbf{y}_{t-1})$$

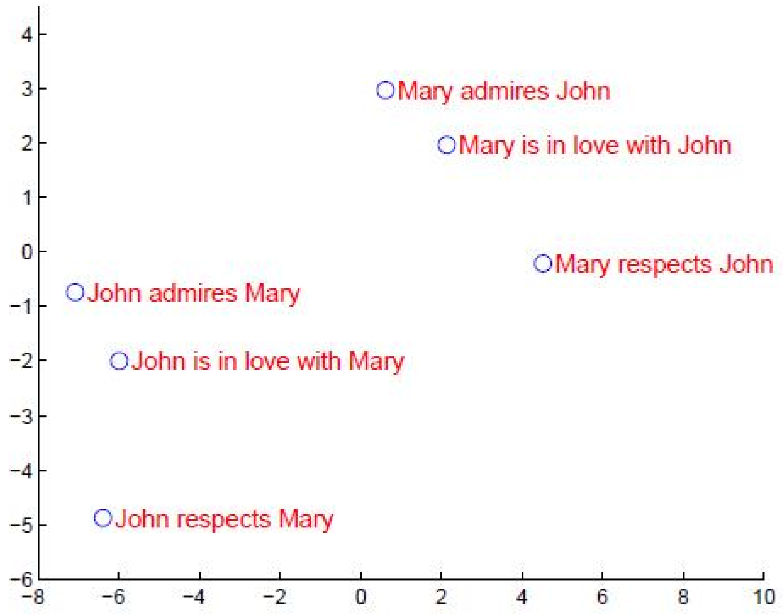


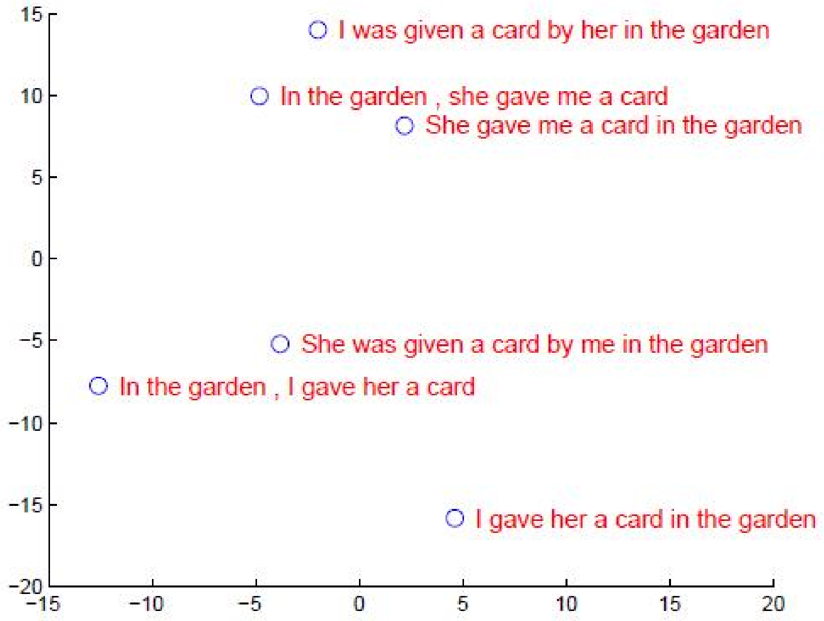
The model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token.

Sequence-to-sequence language translation

- It requires each sentence ends with a special end-of-sequence symbol "<EOS>", which enables the model to define a distribution over sequences of all possible lengths
- It is valuable to reverse the order of the words of the input sequence for decoding. For example, instead of mapping the sentence a, b, c to the sentence α, β, γ, the LSTM is asked to map c, b, a to α, β, γ, where α, β, γ is the translation of a, b, c.
- This way, a is in close proximity to α, b is fairly close to β, and so on, a fact that makes it easy for stochastic gradient descent to "establish communication" between the input and the output. It introduces many short term dependencies in the data that make the optimization problem much easier.

Sequence-to-sequence language translation

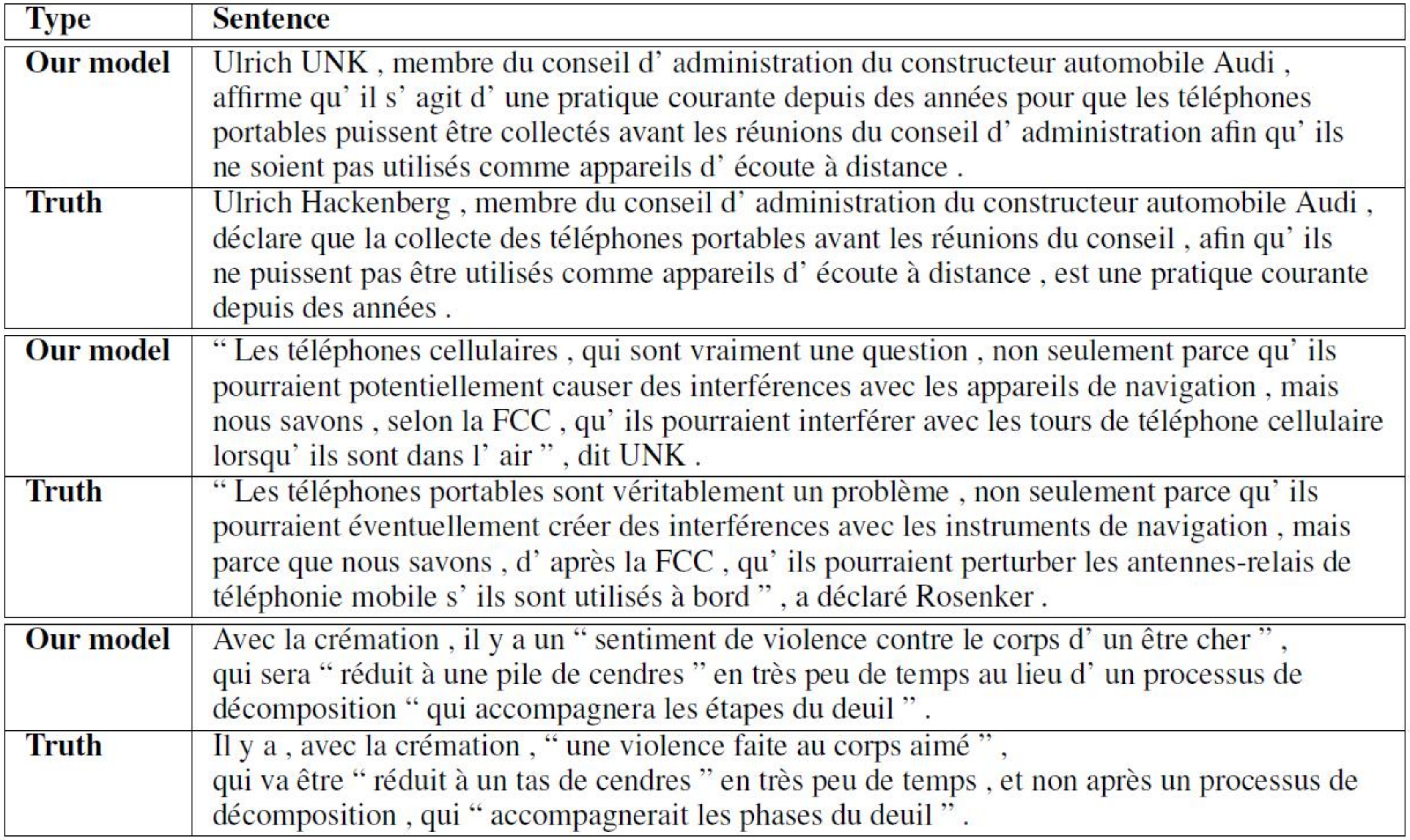




The figure shows a 2-dimensional PCA projection of the LSTM hidden states that are obtained after processing the phrases in the figures. The phrases are clustered by meaning, which in these examples is primarily a function of word order, which would be difficult to capture with a bag-of-words model. The figure clearly shows that the representations are sensitive to the order of words, while being fairly insensitive to the replacement of an active voice with a passive voice.

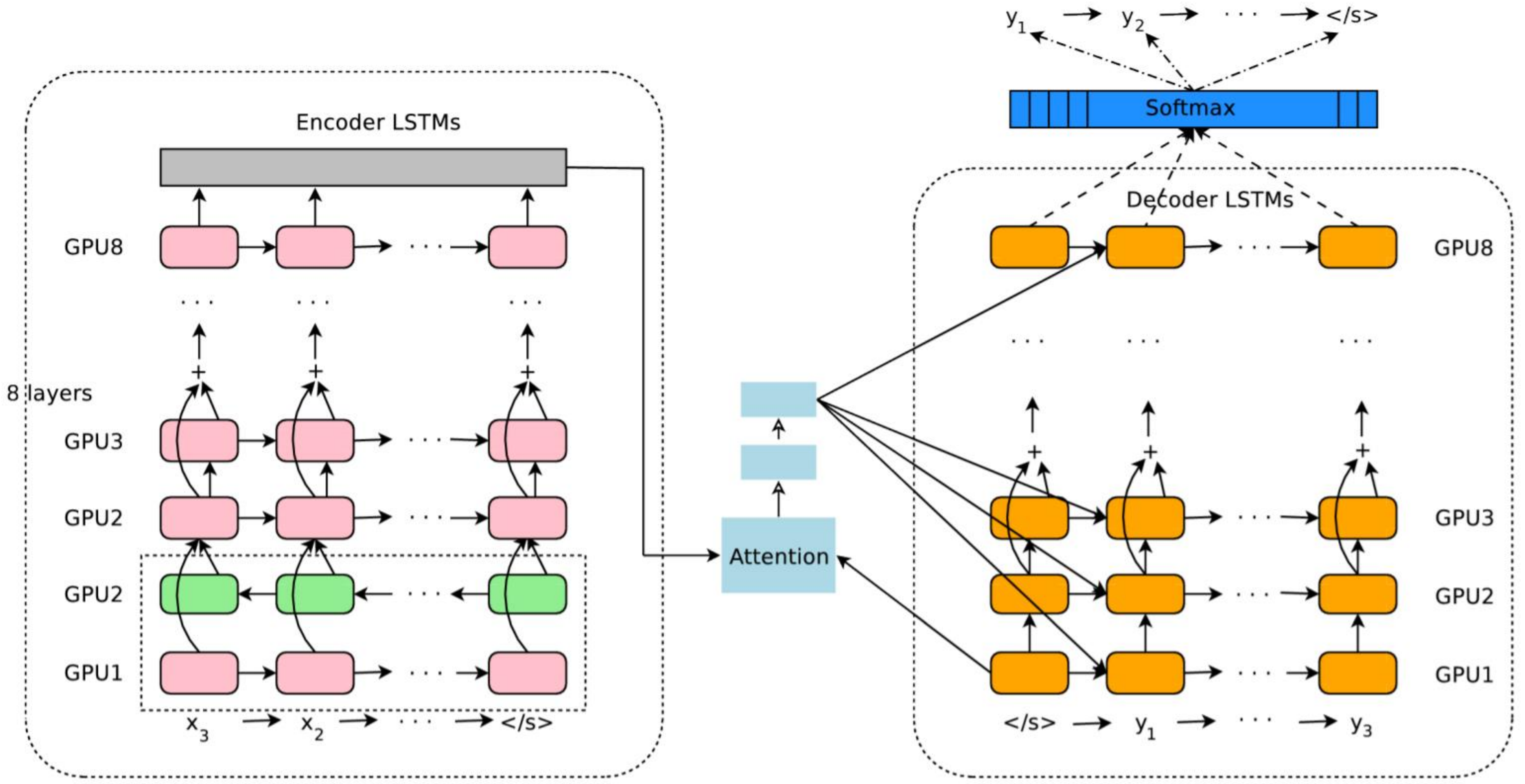
Sequence-to-sequence language translation

- LSTM can correctly translate very long sentences



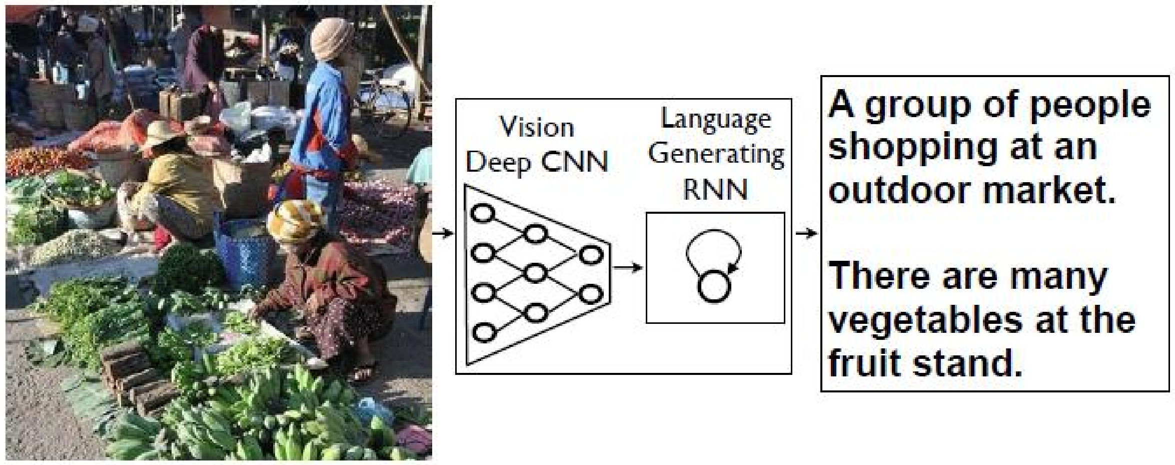
Type	Sentence
Our model	Ulrich UNK , membre du conseil d’ administration du constructeur automobile Audi , affirme qu’ il s’ agit d’ une pratique courante depuis des années pour que les téléphones portables puissent être collectés avant les réunions du conseil d’ administration afin qu’ ils ne soient pas utilisés comme appareils d’ écoute à distance .
Truth	Ulrich Hackenberg , membre du conseil d’ administration du constructeur automobile Audi , déclare que la collecte des téléphones portables avant les réunions du conseil , afin qu’ ils ne puissent pas être utilisés comme appareils d’ écoute à distance , est une pratique courante depuis des années .
Our model	“ Les téléphones cellulaires , qui sont vraiment une question , non seulement parce qu’ ils pourraient potentiellement causer des interférences avec les appareils de navigation , mais nous savons , selon la FCC , qu’ ils pourraient interférer avec les tours de téléphone cellulaire lorsqu’ ils sont dans l’ air ” , dit UNK .
Truth	“ Les téléphones portables sont véritablement un problème , non seulement parce qu’ ils pourraient éventuellement créer des interférences avec les instruments de navigation , mais parce que nous savons , d’ après la FCC , qu’ ils pourraient perturber les antennes-relais de téléphonie mobile s’ ils sont utilisés à bord ” , a déclaré Rosenker .
Our model	Avec la crémation , il y a un “ sentiment de violence contre le corps d’ un être cher ” , qui sera “ réduit à une pile de cendres ” en très peu de temps au lieu d’ un processus de décomposition “ qui accompagnera les étapes du deuil ” .
Truth	Il y a , avec la crémation , “ une violence faite au corps aimé ” , qui va être “ réduit à un tas de cendres ” en très peu de temps , et non après un processus de décomposition , qui “ accompagnerait les phases du deuil ” .

Google's Neural Machine Translation System in 2016



Generate image captions (Vinyals et al. arXiv 2014)

- Use a CNN as an image encoder and transform it to a fixed-length vector
- It is used as the initial hidden state of a “decoder” RNN that generates the target sequence


Vision Deep CNN
Language Generating RNN
A group of people shopping at an outdoor market.
There are many vegetables at the fruit stand.

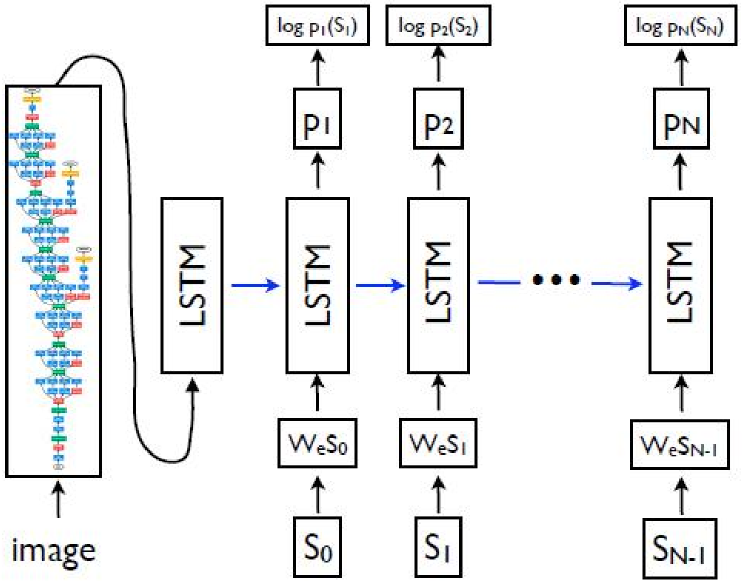
Generate image captions

- The learning process is to maximize the probability of the correct description given the image

$$\theta^* = \arg\max_{\theta} \sum_{(I,S)} \log P(S|I;\theta)$$

$$\log P(S|I) = \sum_{t=0}^{N} \log P(S_t|I, S_0, \ldots, S_{t-1})$$

I is an image and **S** is its correct description



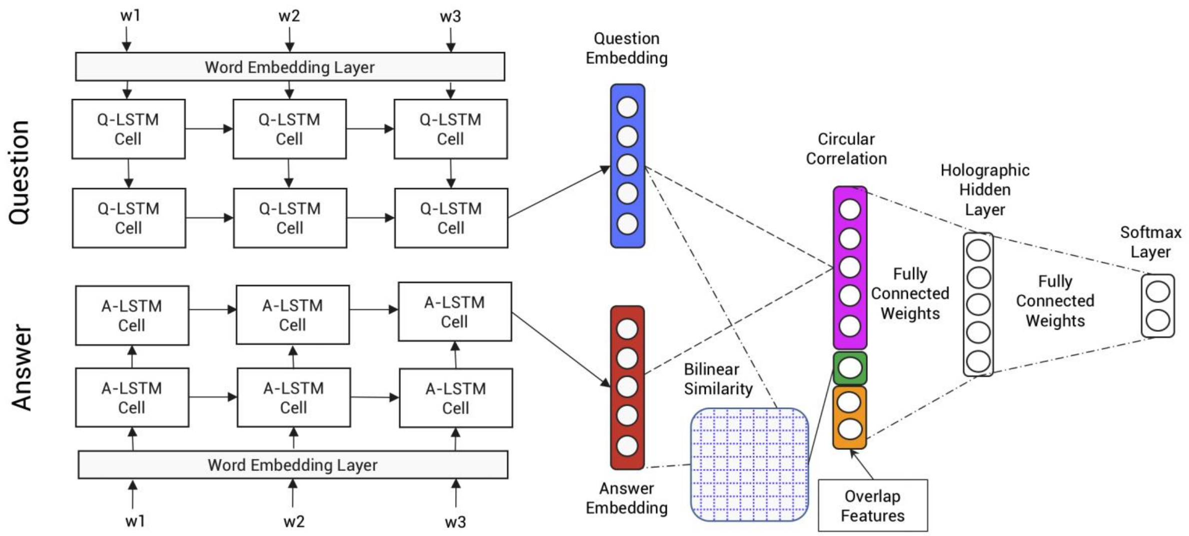
Generate image captions

- Denote by S_0 a special start work and by S_N a special stop word
- Both the image and the words are mapped to the same space, the image by using CNN, the words by using word embedding W_e
- The image ***I*** is only input once to inform the LSTM about the image contents
- Sampling: sample the first word according to P_1 , then provide the corresponding embedding as input and sample P_2 , continuing like this until it samples the special end-of-sentence token

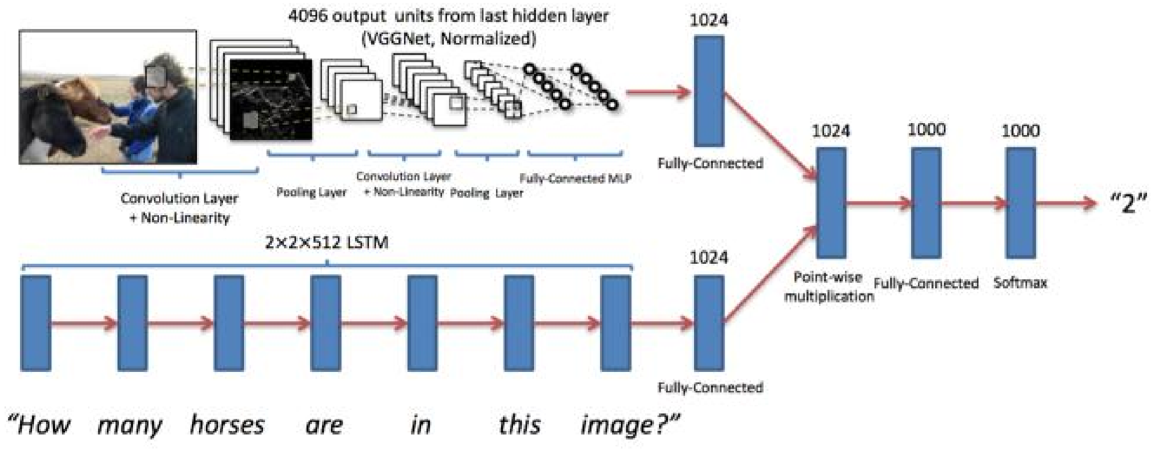
$$x_{-1} = \text{CNN}(\mathbf{I})$$
$$x_t = W_e S_t, t \in \{0, \ldots, N-1\}$$
$$P_{t+1} = \text{LSTM}(x_t), t \in \{0, \ldots, N-1\}$$
$$L(I, S) = -\sum_{t=1}^{N} \log P_t(S_t)$$

Question Answering and Visual Question Answering

- Question Answering



- Visual Question Answering


4096 output units from last hidden layer (VGGNet, Normalized)
1024
Fully-Connected
Convolution Layer + Non-Linearity
Pooling Layer
Convolution Layer + Non-Linearity
Pooling Layer
Fully-Connected MLP
2×2×512 LSTM
1024
1024
1000
1000
Point-wise multiplication
Fully-Connected
Softmax
"2"
Fully-Connected
"How many horses are in this image?"

Reading Materials

- R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification," Chapter 6, 2000.
- Y. Bengio, I. J. Goodfellow and A. Courville, "Sequence Modeling: Recurrent and Recursive Nets" in "Deep Learning", Book in preparation for MIT Press, 2014.
- I. Sutskever, O. Vinyals, and Q. Le, "Sequence to Sequence Learning with Neural Networks," NIPS 2014.
- S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, K. Saenko, "Translating Videos to Natural Language Using Deep Recurrent Neural Networks," arXiv: 1412.4729, 2014.
- J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term Recurrent Convolutional Networks for Visual Recognition and Description," arXiv:1411.4389, 2014.
- O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and Tell: A Neural Image Caption Generator," arXiv: 1411.4555, 2014.

香 港 中 文 大 學（深 圳）
The Chinese University of Hong Kong, Shenzhen

Thanks for Listening !

Ruimao Zhang

Room 517, Daoyuan Building, The Chinese Univeristy of Hong Kong, Shenzhen
zhangruimao@cuhk.edu.cn
ruimao.zhang@ieee.org