Recurrent Neural Network

October 30-- November 7

Outline

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 Hong Kong
- 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**

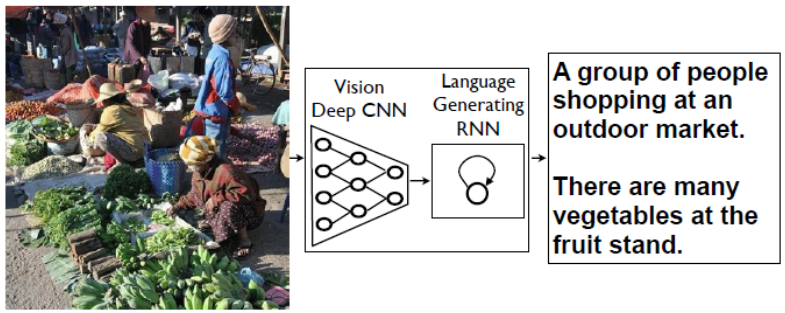
Modeling sequential data

- Sample data sequences from a certain distribution

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_T)$$

- Generate natural sentences to describe an image

$$P(\mathbf{y}_1, \ldots, \mathbf{y}_T | I)$$



- Activity recognition from a video sequence

$$P(\mathbf{y} | \mathbf{x}_1, \ldots, \mathbf{x}_T)$$

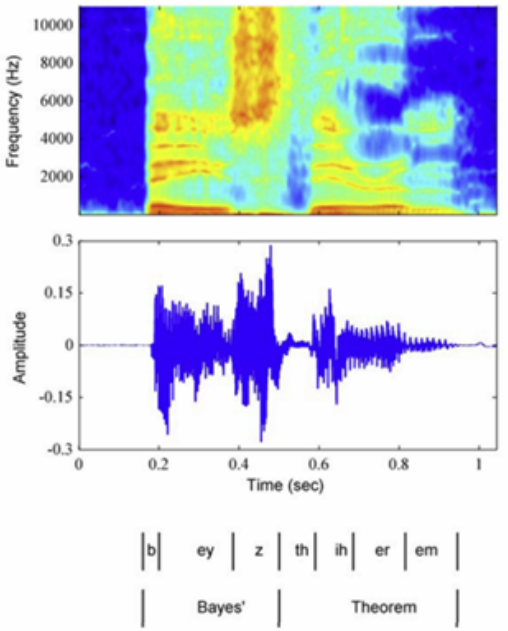
Modeling sequential data

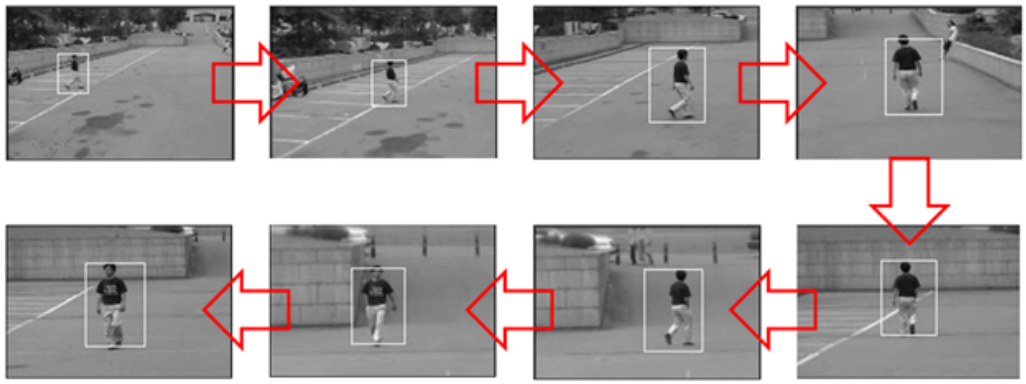
- Speech recognition

$$P(\mathbf{y}_1, \ldots, \mathbf{y}_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$$

- Object tracking

$$P(\mathbf{y}_1, \ldots, \mathbf{y}_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$$





Modeling sequential data

- Generate natural sentences to describe a video

$$P(\mathbf{y}_1, \ldots, \mathbf{y}_{T'} | \mathbf{x}_1, \ldots, \mathbf{x}_T)$$

- Language translation

$$P(\mathbf{y}_1, \ldots, \mathbf{y}_{T'} | \mathbf{x}_1, \ldots, \mathbf{x}_T)$$

Modeling sequential data

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

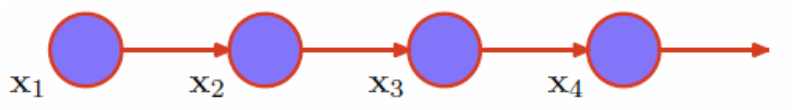
$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_t | \mathbf{x}_1, \ldots, \mathbf{x}_{t-1})$$

- Impractical to consider general dependence of future dependence on all previous observations $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots, \mathbf{x}_0)$
  - 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

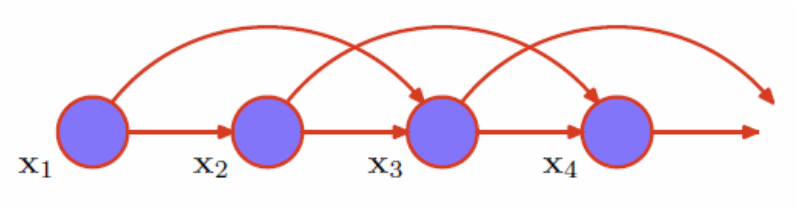
- Markov models assume dependence on most recent observations
- First-order Markov model

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_t | \mathbf{x}_{t-1})$$



- Second-order Markov model

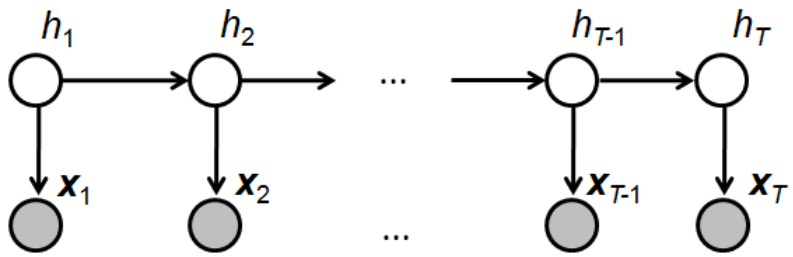
$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T} p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2})$$



Hidden Markov Model (HMM)

- A classical way to model sequential data
- Sequence pairs $h_1, h_2, \ldots, h_T$ (hidden variables) and $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T$ (observations) are generated by the following process
  - Pick h_1 at random from the distribution $P(h_1)$. Pick $\mathbf{x}_1$ from the distribution $p(\mathbf{x}_1|h_1)$
  - For $t = 2$ to T
    - ★ Choose h_t at random from the distribution $p(h_t|h_{t-1})$
    - ★ Choose $\mathbf{x}_t$ at random from the distribution $p(\mathbf{x}_t|h_t)$
- The joint distribution is

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_T, h_1, \ldots, h_T, \boldsymbol{\theta}) = P(h_1) \prod_{t=2}^{T} P(h_t|h_{t-1}) \prod_{t=1}^{T} p(\mathbf{x}_t|h_t)$$

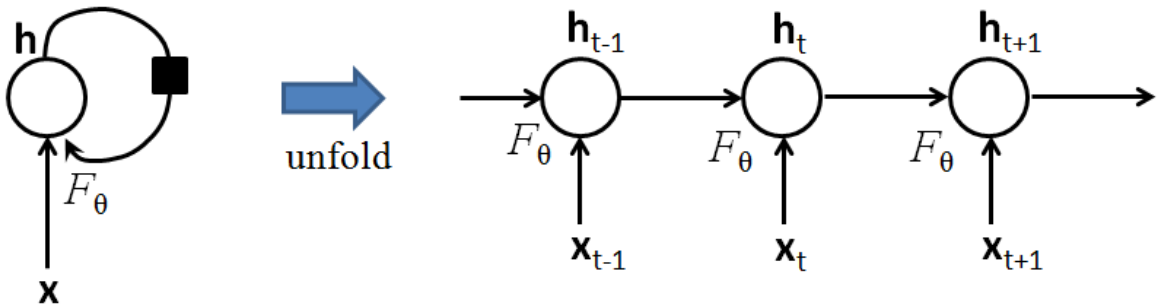


Recurrent neural networks (RNN)

- While HMM is a generative model RNN is a discriminative model
- Model a dynamic system driven by an external signal $\mathbf{x}_t$

$$\mathbf{h}_t = F_\theta(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

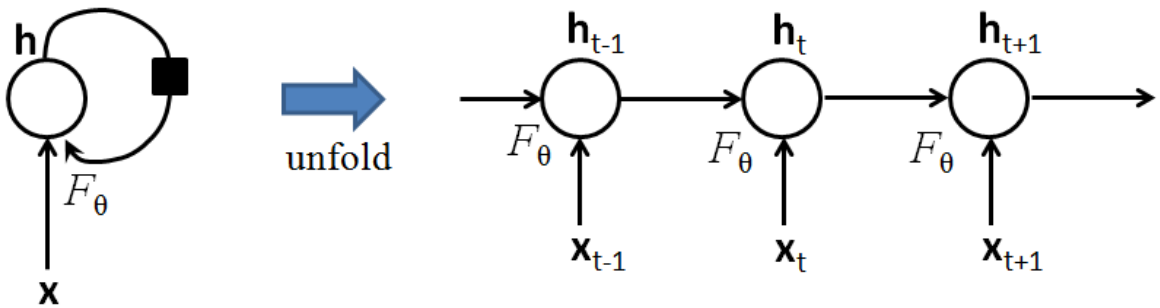
- $\mathbf{h}_t$ contains information about the whole past sequence. The equation above implicitly defines a function which maps the whole past sequence $(\mathbf{x}_t, \ldots, \mathbf{x}_1)$ to the current sate $\mathbf{h}_t = G_t(\mathbf{x}_t, \ldots, \mathbf{x}_1)$



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

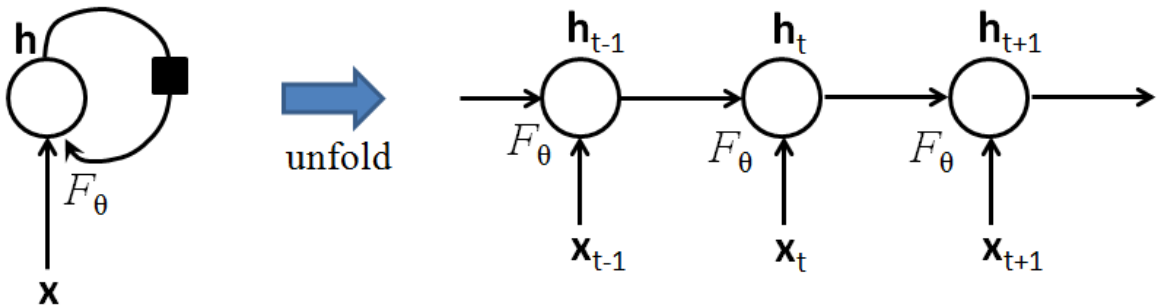
Recurrent neural networks (RNN)

- The summary is lossy, since it maps an arbitrary length sequence $(\mathbf{x}_t, \ldots, \mathbf{x}_1)$ to a fixed length vector $\mathbf{h}_t$. Depending on the training criterion, $\mathbf{h}_t$ keeps some important aspects of the past sequence.
- Sharing parameters: the same weights are used for different instances of the artificial neurons at different time steps
- Share 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 predict 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

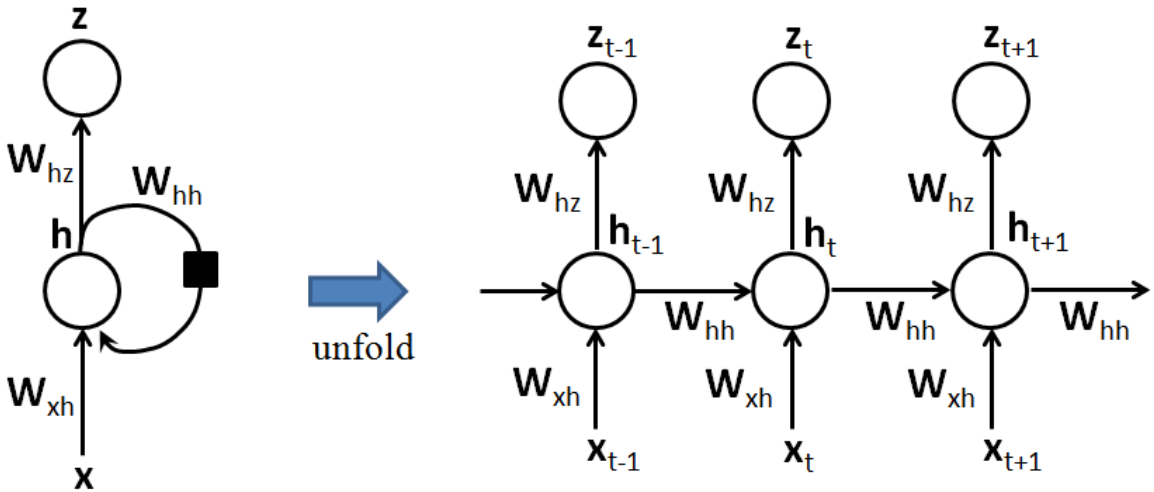
$$P(y_1, \ldots, y_T | \mathbf{x}_1, \ldots, \mathbf{x}_T)$$

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

$$\mathbf{h}_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$$

$$\mathbf{z}_t = \text{softmax}(\mathbf{W}_{hz}\mathbf{h}_t + \mathbf{b}_z)$$

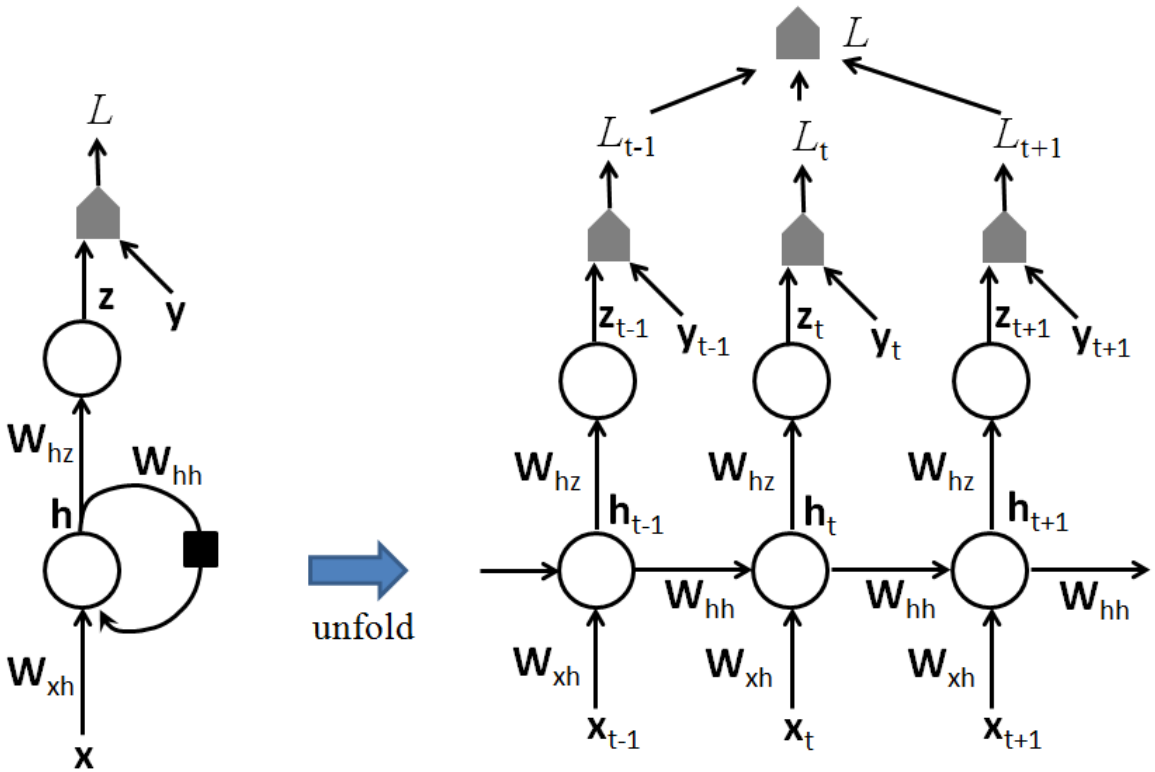
$$p(y_t = c) = z_{t,c}$$



Cost function

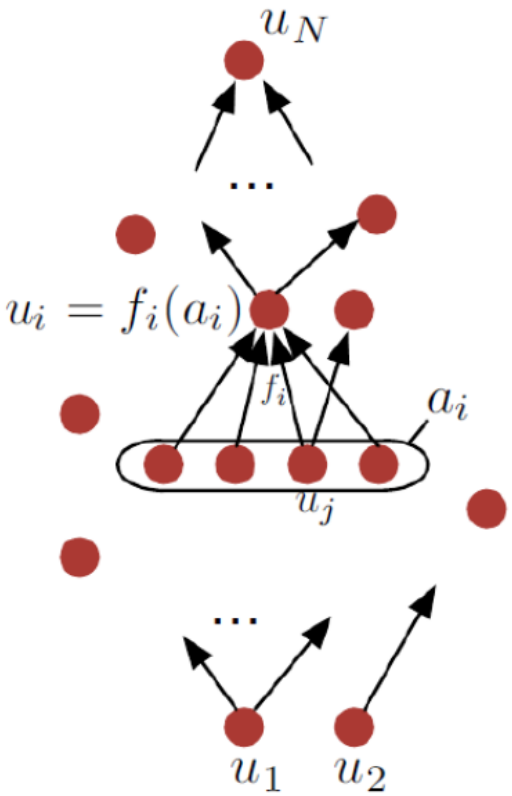
- The total loss for a given input/target sequence pair (**x**, **y**), measured in cross entropy

$$L(\mathbf{x}, \mathbf{y}) = \sum_t L_t = \sum_t -\log z_{t,y_t}$$



Backpropagation on RNN

- Review BP on flow graph



$\frac{\partial u_N}{\partial u_N} \leftarrow 1$
for $j = N-1$ down to 1 **do**
  $\frac{\partial u_N}{\partial u_j} \leftarrow \sum_{i:j\in \text{parents}(i)} \frac{\partial u_N}{\partial u_i}\frac{\partial u_i}{\partial u_j}$
end for
return $\left(\frac{\partial u_N}{\partial u_i}\right)_{i=1}^{M}$

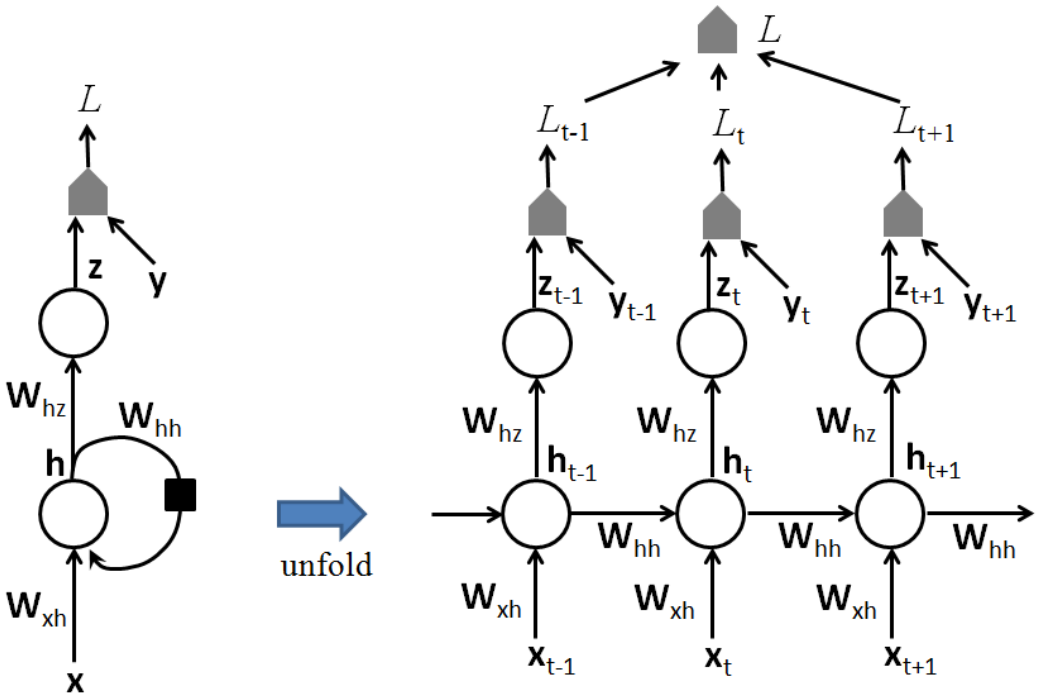
(Bengio et al. Deep Learning 2014)

$$\frac{\partial u_N}{\partial w_{ji}} = \frac{\partial u_N}{\partial u_i}\frac{\partial u_i}{\partial net_i}\frac{\partial net_i}{\partial w_{ji}}$$

Gradients on $\mathbf{W}_{hz}$ and $\mathbf{b}_z$

$$\frac{\partial L}{\partial L_t} = 1, \qquad \frac{\partial L}{\partial \mathbf{z}_t} = \frac{\partial L}{\partial L_t}\frac{\partial L_t}{\partial \mathbf{z}_t} = \frac{\partial L_t}{\partial \mathbf{z}_t}$$

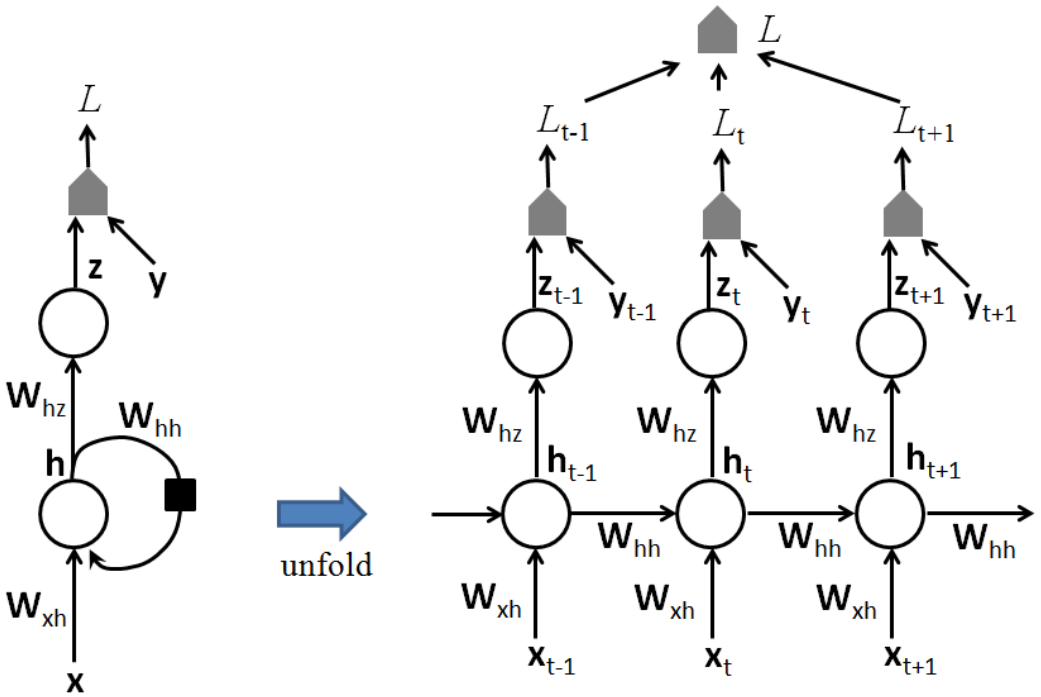
$$\frac{\partial L}{\partial \mathbf{W}_{hz}} = \sum_t \frac{\partial L_t}{\partial \mathbf{z}_t}\frac{\partial \mathbf{z}_t}{\partial \mathbf{W}_{hz}}, \qquad \frac{\partial L}{\partial \mathbf{b}_z} = \sum_t \frac{\partial L_t}{\partial \mathbf{z}_t}\frac{\partial \mathbf{z}_t}{\partial \mathbf{b}_z}$$



Gradients on $\mathbf{W}_{hh}$ and $\mathbf{W}_{xh}$

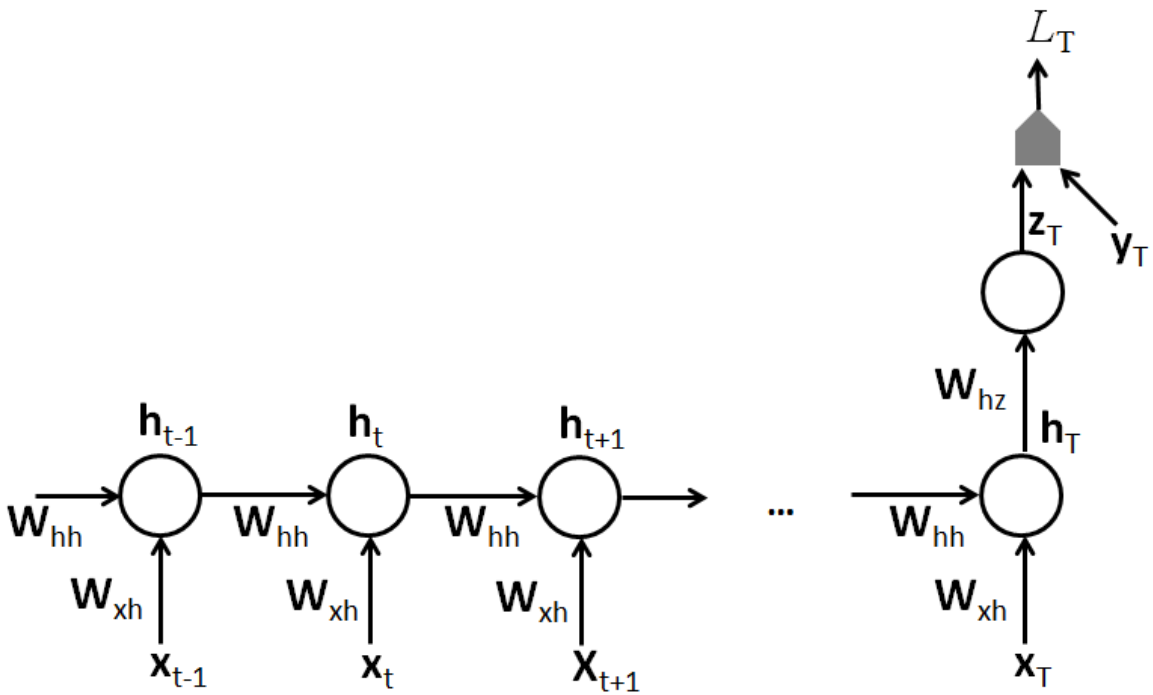
$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \sum_t \frac{\partial L}{\partial \mathbf{h}_t} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hh}}$$

$$\frac{\partial L}{\partial \mathbf{h}_t} = \frac{\partial L}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t} + \frac{\partial L}{\partial \mathbf{z}_t} \frac{\partial \mathbf{z}_t}{\partial \mathbf{h}_t}$$



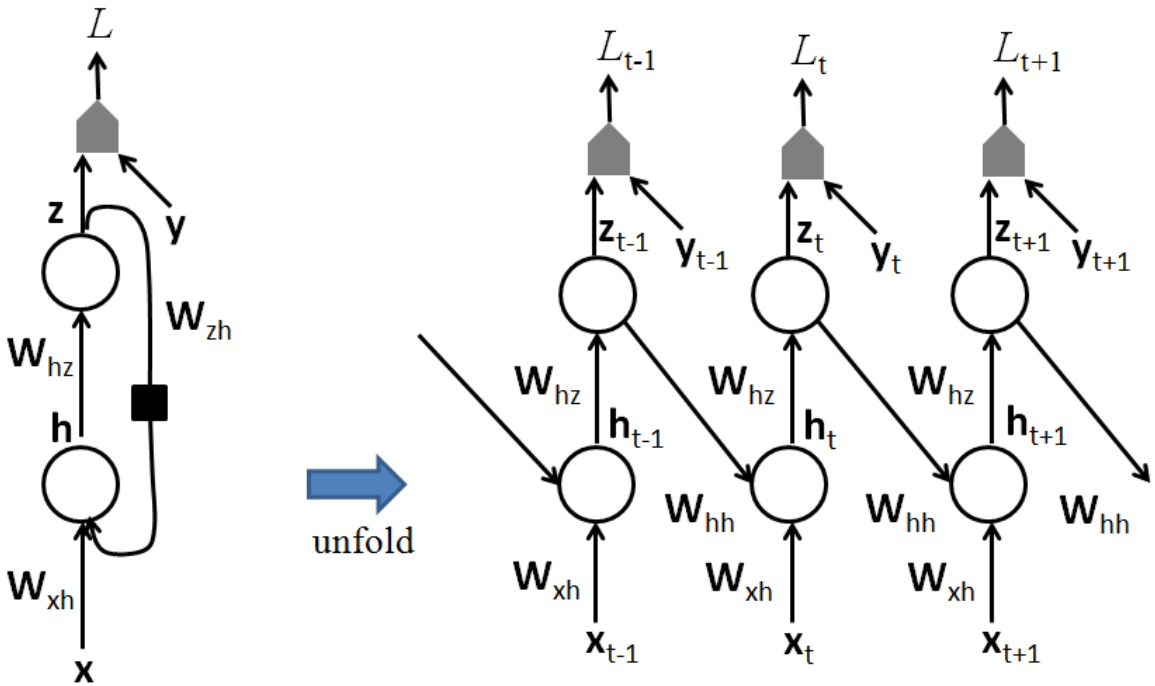
Predict a single output at the end of the sequence

- 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 $\mathbf{z}_t$ can be obtained by backpropagation from further downsteam modules



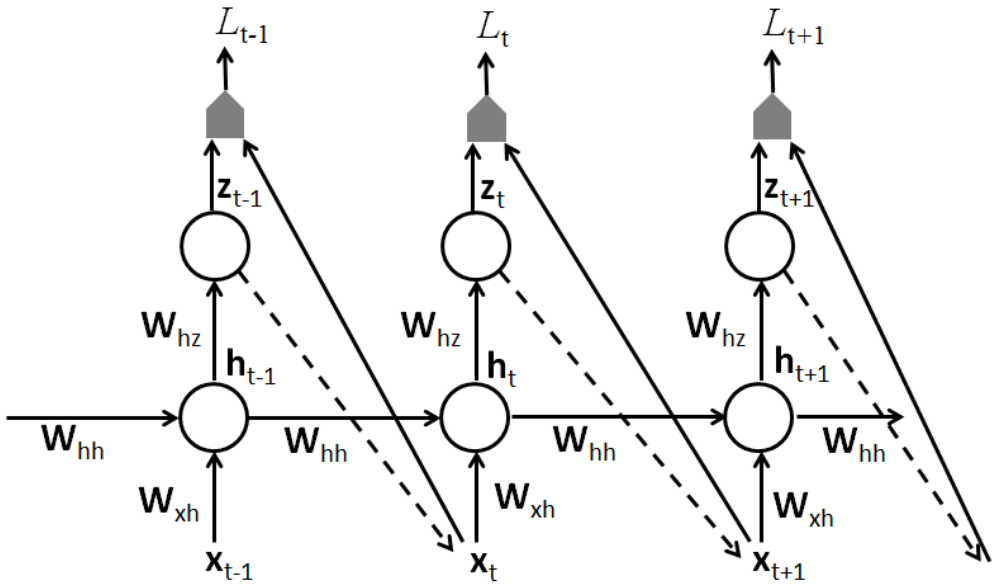
Network with output recurrence

- Memory is from the prediction of the previous target, which limits its expressive power but makes it easier to train



Generative RNN modeling $P(\mathbf{x}_1, \ldots, \mathbf{x}_T)$

- It can generate sequences from this distribution
- At the training stage, each $\mathbf{x}_t$ of the observed sequence serves both as input (for the current time step) and as target (for the previous time step)
- The output $\mathbf{z}_t$ encodes the parameters of a conditional distribution $P(\mathbf{x}_{t+1}|\mathbf{x}_1, \ldots, \mathbf{x}_t) = P(\mathbf{x}_{t+1}|\mathbf{z}_t)$ for $\mathbf{x}_{t+1}$ given the past sequence $\mathbf{x}_1, \ldots, \mathbf{x}_t$



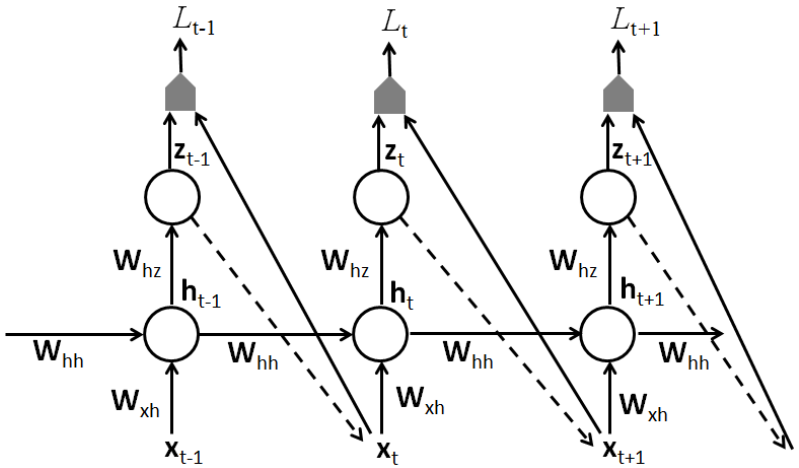
Generative RNN modeling $P(\mathbf{x}_1, \ldots, \mathbf{x}_T)$

- Cost function: negative log-likelihood of $\mathbf{x}$, $L = \sum_t L_t$

$$P(\mathbf{x}) = P(\mathbf{x}_1, \ldots, \mathbf{x}_T) = \prod_{t=1}^{T} P(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots, \mathbf{x}_1)$$

$$L_t = -\log P(\mathbf{x}_t | \mathbf{x}_{t-1}, \ldots, \mathbf{x}_1)$$

- In generative mode, $\mathbf{x}_{t+1}$ is sampled from the conditional distribution $P(\mathbf{x}_{t+1} | \mathbf{x}_1, \ldots, \mathbf{x}_t) = P(\mathbf{x}_{t+1} | \mathbf{z}_t)$ (dashed arrows) and then that generated sample $\mathbf{x}_{t+1}$ is fed back as input for computing the next state $\mathbf{h}_{t+1}$



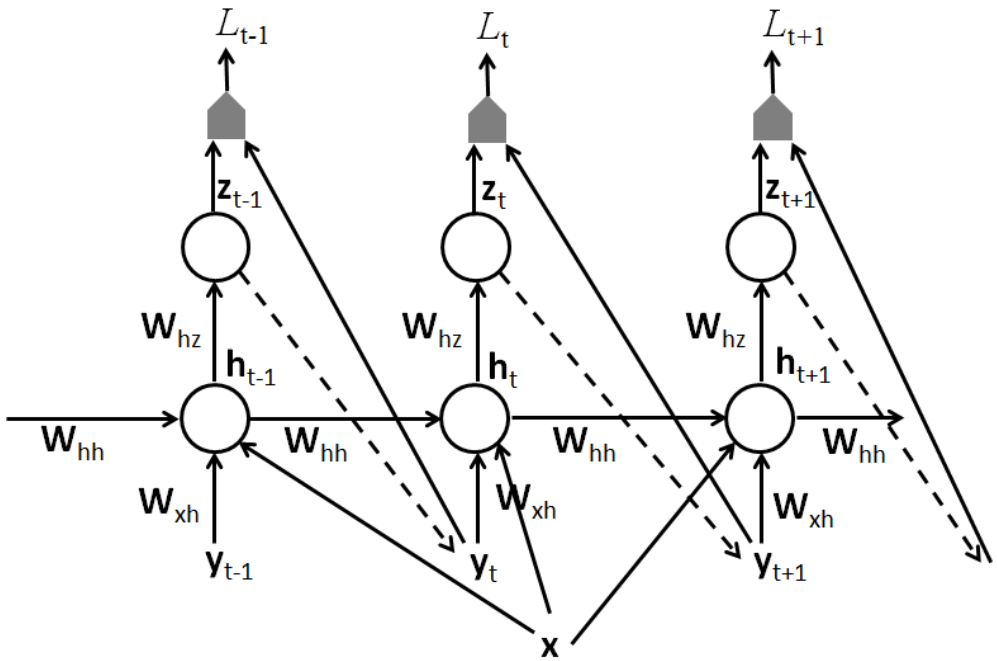
Generative RNN modeling $P(\mathbf{x}_1, \ldots, \mathbf{x}_T)$

- 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 model the length T of the sequence through some parametric distribution. $P(\mathbf{x}_1, \ldots, \mathbf{x}_T)$ is decomposed into

$$P(\mathbf{x}_1, \ldots, \mathbf{x}_T) = P(\mathbf{x}_1, \ldots, \mathbf{x}_T | T) P(T)$$

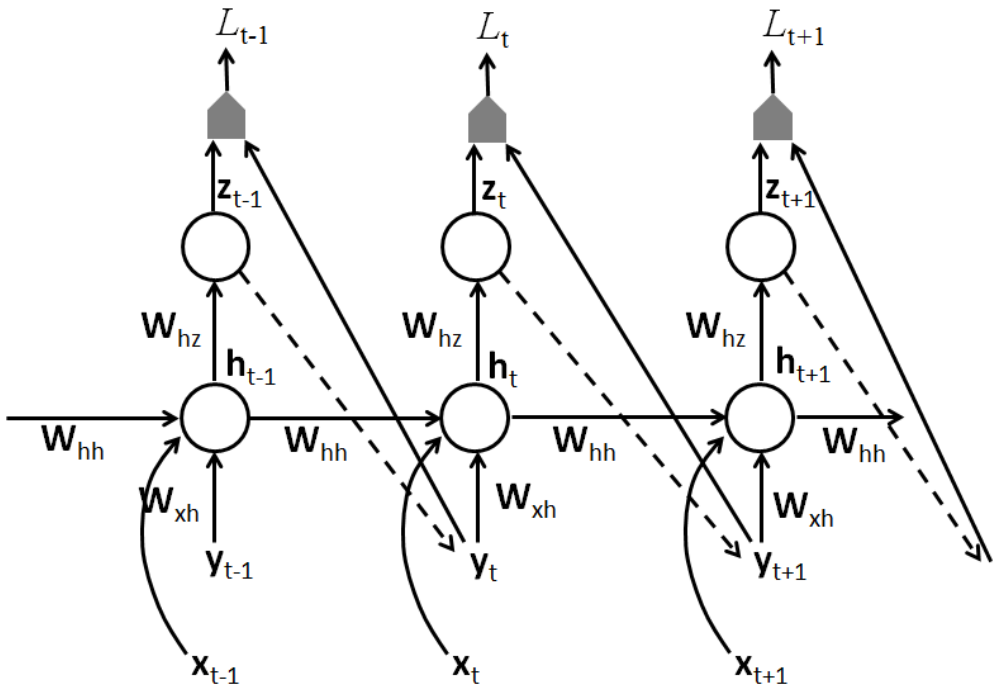
RCNNs to represent conditional distributions $P(\mathbf{y}|\mathbf{x})$

- If $\mathbf{x}$ is a fixed-sized vector, we can simply make it an extra input of the RNN that generates the $\mathbf{y}$ sequence. Some common ways of providing the extra input
  - as an extra input at each time step, or
  - as the initial state $\mathbf{h}_0$, or
  - both
- Example: generate caption for an image



RCNNs to represent conditional distributions $P(\mathbf{y}|\mathbf{x})$

- The input $\mathbf{x}$ is a sequence of the same length as the output sequence $\mathbf{y}$
- Removing the dash lines, it assumes $\mathbf{y}_t$'s are independent of each other when the past input sequence is given, i.e. $P(\mathbf{y}_t|\mathbf{y}_{t-1}, \ldots, \mathbf{y}_1, \mathbf{x}_t, \ldots, \mathbf{x}_1) = P(\mathbf{y}_t|\mathbf{x}_t, \ldots, \mathbf{x}_1)$
- Without the conditional independence assumption, add the dash lines and the prediction of $\mathbf{y}_{t+1}$ is based on both the past $\mathbf{x}$'s and past $\mathbf{y}$'s

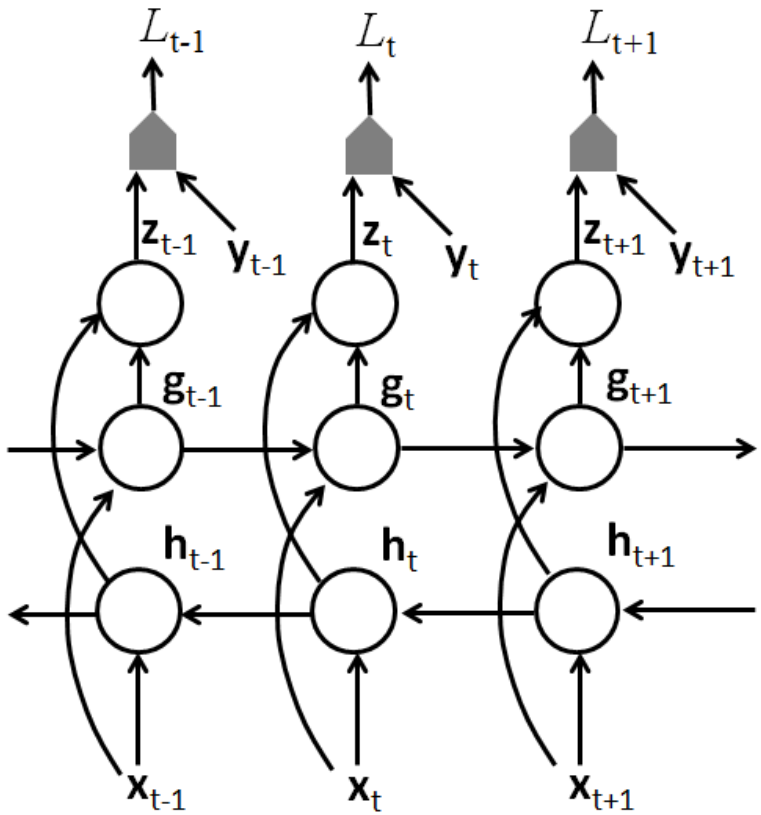


Bidirectional RNNs

- 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 need
- 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

- $\mathbf{g}_t$ summaries the information from the past sequence, and $\mathbf{h}_t$ summaries the information from the future sequence



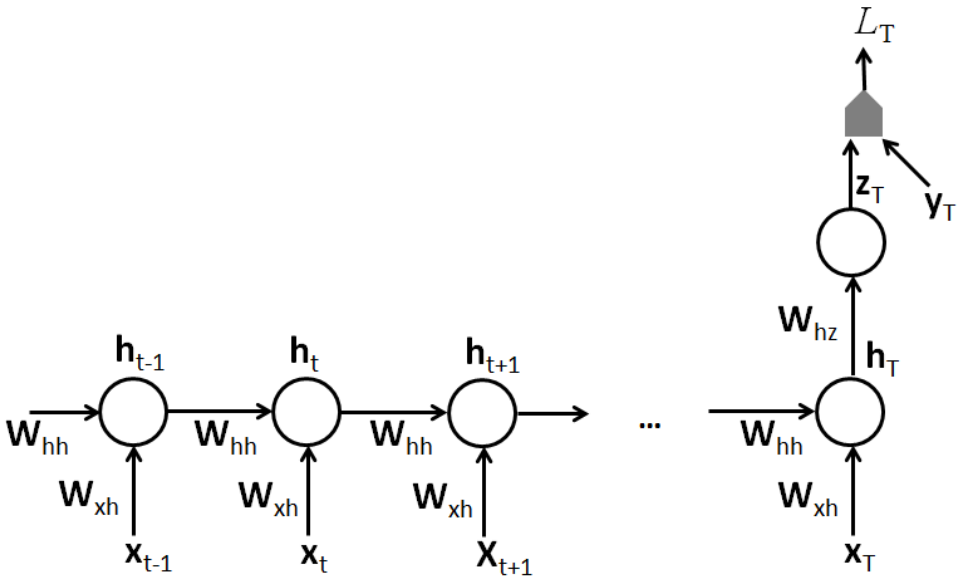
Difficulty of Learning Long-Term Dependencies

- Consider the gradient of a loss L_T at time T with respect to the parameter θ of the recurrent function F_θ

$$\mathbf{h}_t = F_\theta(\mathbf{h}_{t-1}, \mathbf{x}_t)$$

$$\frac{\partial L_T}{\partial \theta} = \sum_{t \leq T} \frac{\partial L_T}{\partial \mathbf{h}_T} \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} \frac{\partial F_\theta(\mathbf{h}_{t-1}, \mathbf{x}_t)}{\partial \theta}$$

$\frac{\partial L_T}{\partial \mathbf{h}_T} \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} \frac{\partial F_\theta(\mathbf{h}_{t-1}, \mathbf{x}_t)}{\partial \theta}$ encodes long-term dependency when $T - t$ is large



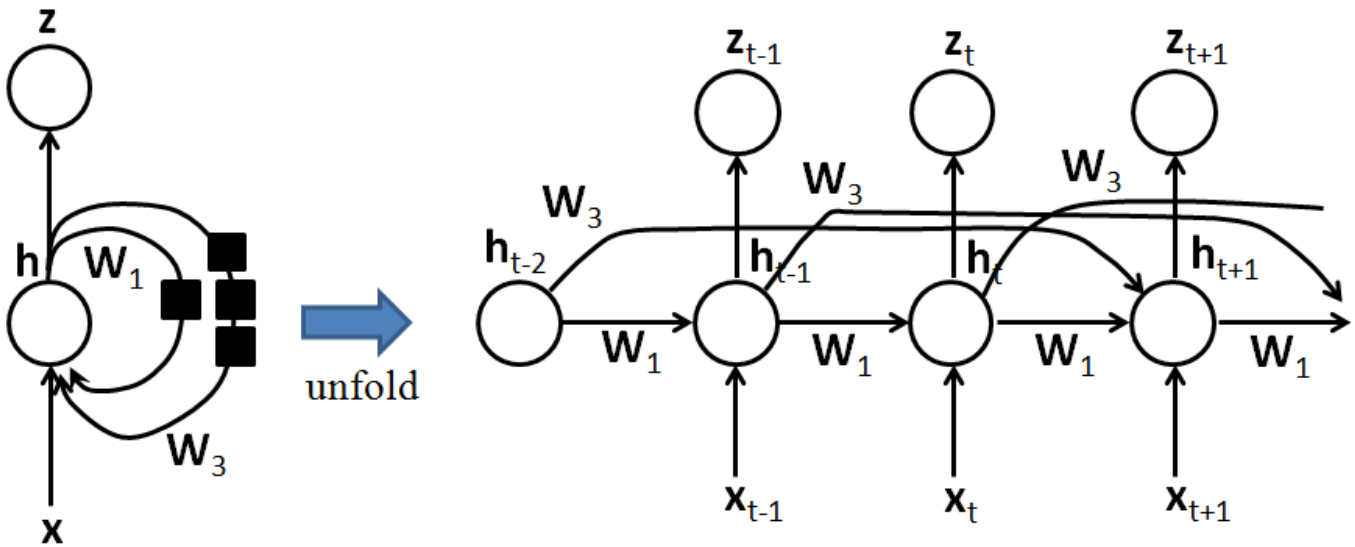
Difficulty of Learning Long-Term Dependencies

$$\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_t} = \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_{T-1}} \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{h}_{T-2}} \cdots \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}$$

- Each layer-wise Jacobian $\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}$ is the product of two matrices: (a) the recurrent matrix $\mathbf{W}$ and (b) the diagonal matrix whose entries are the derivatives of the non-linearities associated with the hidden units, which vary 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

Combine short and long paths in unfolded flow graph

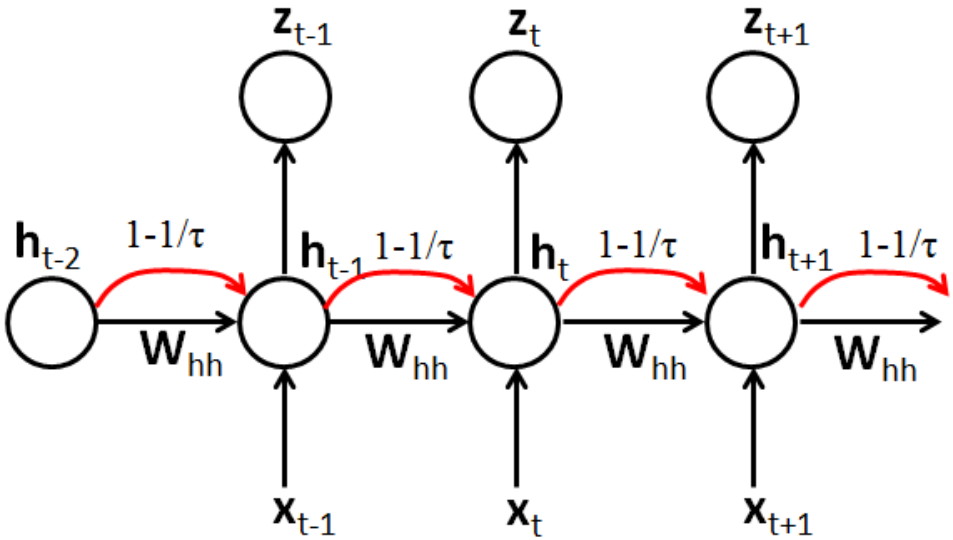
- Longer-delay connections allow to connect the past states to future states through short paths
- Gradients will vanish exponentially with respect to the number of time steps
- If we have recurrent connections with a time-delay of D, the instead of the vanishing or explosion going as $O(\lambda^T)$ over T steps (where λ is largest eigenvalue of the Jacobians $\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}$), the unfolded recurrent network now has paths through which gradients grow as $O(\lambda^{T/D})$ because the number of effective steps is T/D



Leaky units with self-connections

$$\mathbf{h}_{t+1} = (1 - \frac{1}{\tau_i})\mathbf{h}_t + \frac{1}{\tau_i}\tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_t + \mathbf{b}_h)$$

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



Leaky units with self-connections

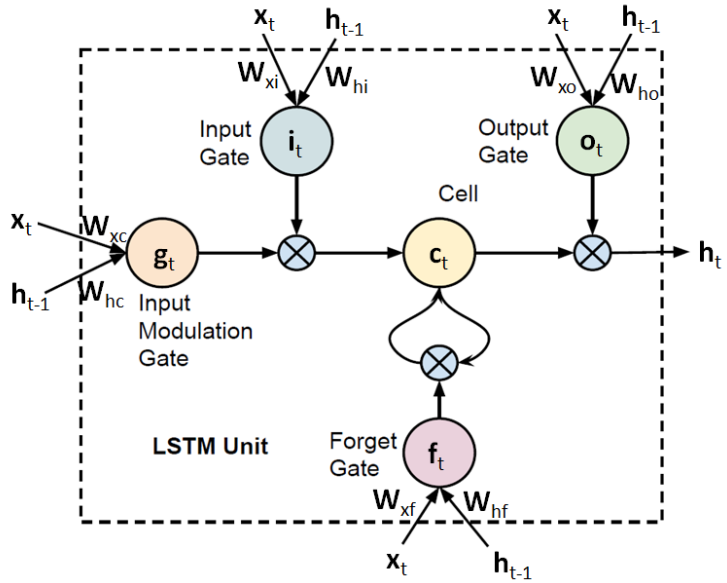
- When $\tau = 1$, there is no linear self-recurrence, only the nonlinear update which we can find in ordinary recurrent networks
- When $\tau > 1$, this linear recurrence allows gradients to propagate more easily. When τ is large, the sate changes very slowly, integrating the past values associated 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 rates
- Those time constants can be fixed manually or can be learned as free parameters

Long Short-Term Memory (LSTM) net

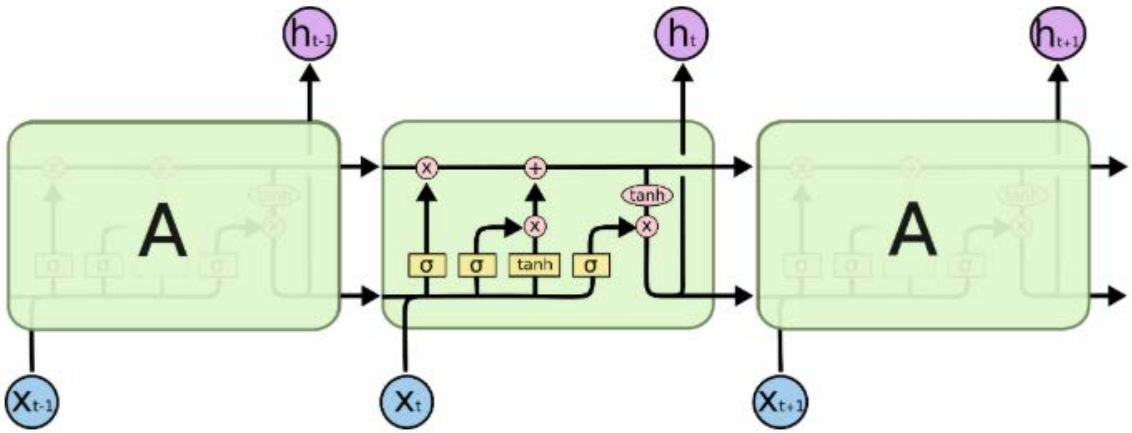
- In the leaky units with self-connections, the forgetting rate is constant during the whole sequence.
- The role of leaky units is to accumulate information over a long duration. However, once that information gets used, it might be useful for the neural network to forget the old state.
  - For example, 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 sates and current input (conditioning the forgetting on the context)
- Parameters controlling the forgetting rates are learned from train data

Long Short-Term Memory (LSTM) net

$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f),\ \mathbf{i}_t = \sigma(\mathbf{W}_{xi}\mathbf{x}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{b}_i),\ \mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)$$
$$\mathbf{g}_t = \tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c),\ \mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \mathbf{g}_t$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t),\ \mathbf{z}_t = \mathrm{softmax}(\mathbf{W}_{hz}\mathbf{h}_t + \mathbf{b}_z)$$

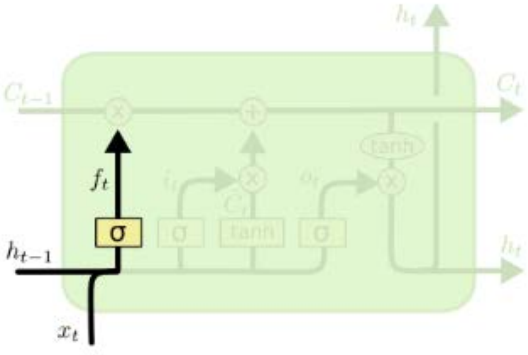


http://colah.github.io/posts/2015-08-Understanding-LSTMs/



The repeating module in an LSTM contains four interacting layers.

Long Short-Term Memory (LSTM) net

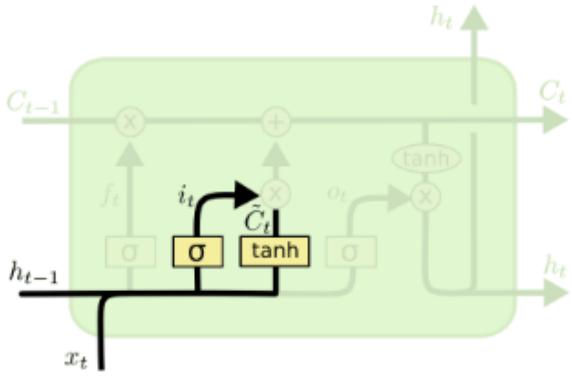


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Forget gate: 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.”

Long Short-Term Memory (LSTM) net

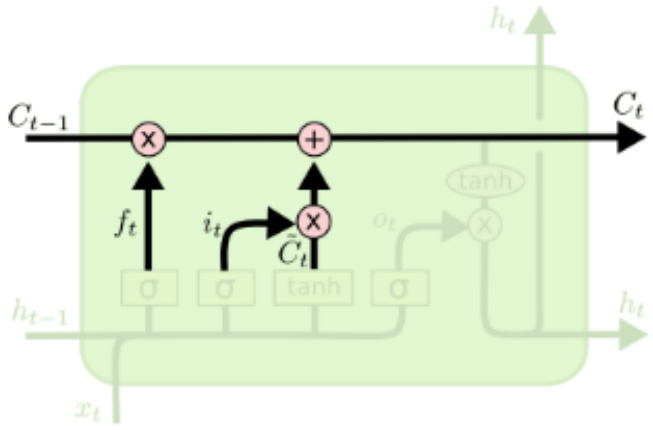


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Input gate: decides which values we'll update.

New candidate values, $\tilde{C}_t$

Long Short-Term Memory (LSTM) net

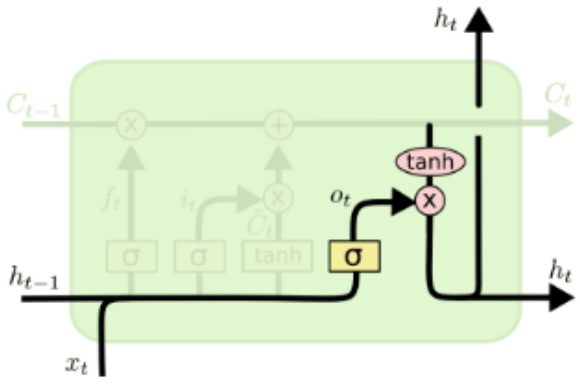


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Update the old cell state, C_{t-1}, into the new cell state C_t with forget gate.

Add the new candidate $\tilde{C}_t$ to C_t with input gate

Long Short-Term Memory (LSTM) net



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

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

Put the cell state through **tanh** (to push the values to be between -1 and 1)

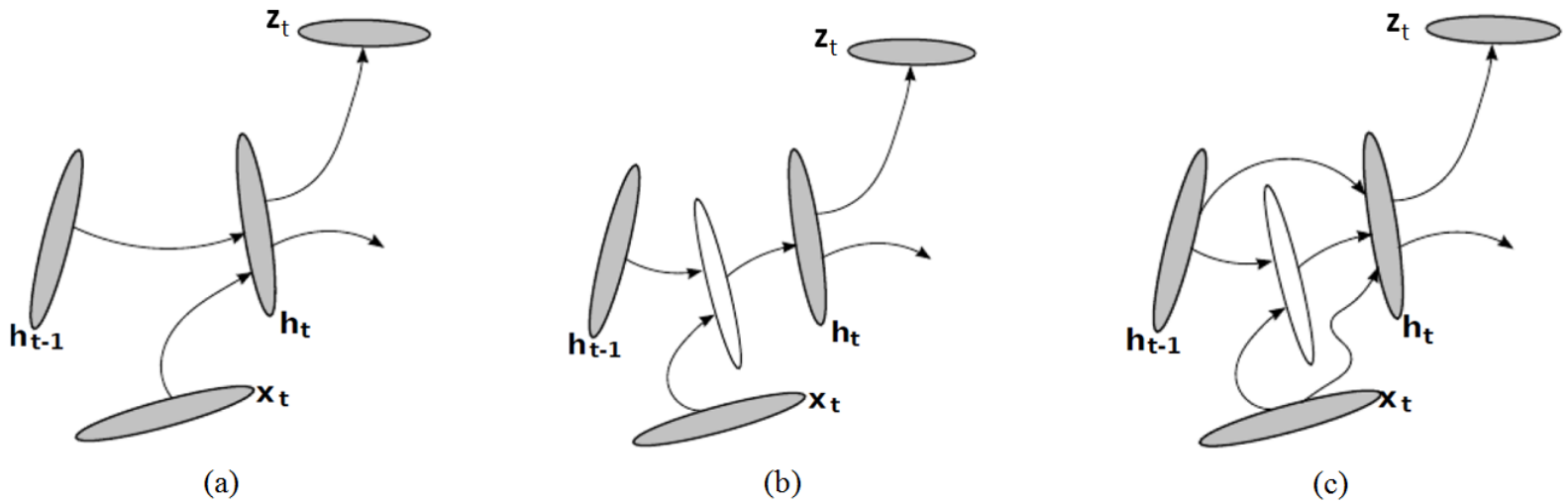
output h_t through a learned output gate

Long Short-Term Memory (LSTM) net

- The core of LSTM is a memory cell $\mathbf{c}_t$ which encodes, at every time step, the knowledge of the inputs that have been observed up to that step.
- The memory cell $\mathbf{c}_t$ has the same inputs ($\mathbf{h}_{t-1}$ and $\mathbf{x}_t$) and outputs ($\mathbf{h}_t$) as a normal recurrent network, but has more parameters and a system of gating units that controls the flow of information
- $\mathbf{c}_t$ has a linear self-connection similar to the leaky units, but the self-connection weight is controlled by a forget gate unit $\mathbf{f}_t$, that sets this weight to a value between 0 and 1 via a sigmoid unit $\mathbf{f}_t = \sigma(\mathbf{W}_{xf}\mathbf{x}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{b}_f)$
- The input gate unit $\mathbf{i}_t$ is computed similarly to the forget gate, but with its own parameters
- The output $\mathbf{h}_t$ of the LSTM cell can also be shut off , via the output gate $\mathbf{o}_t$ ($\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$), which is also a sigmoid unit for gating $\mathbf{o}_t = \sigma(\mathbf{W}_{xo}\mathbf{x}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{b}_o)$

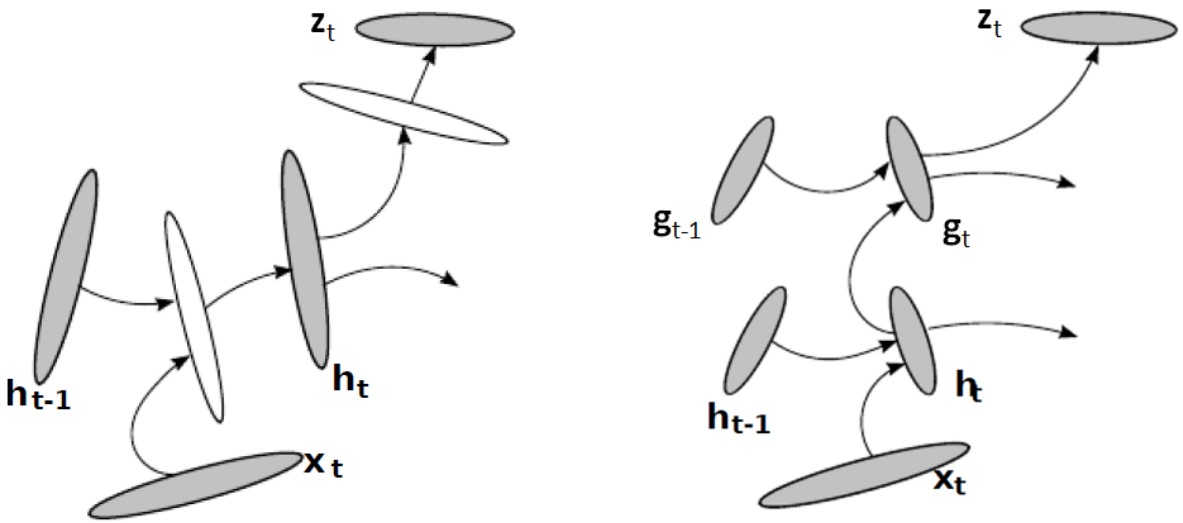
Long Short-Term Memory (LSTM) net

- (a): A vanilla RNN with input sequence and an output sequence
- (b): Add a deep hidden-to-hidden transformation
- (c): Skip connections and allow gradients to flow more easily backwards in spite of the extra non-linearity due to the intermediate hidden layer



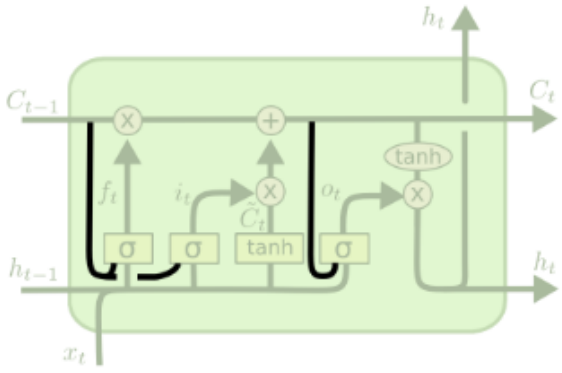
Long Short-Term Memory (LSTM) net

- (c): Depth can also be added in the hidden-to-output transform
- (d): A hierarchy of RNNs, which can be stacked on top of each other



(c) (d)

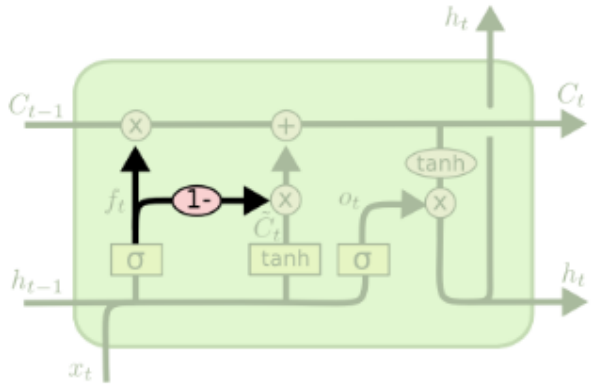
Variants on LSTM (LSTM with peephole)



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

The above diagram adds peepholes to **all the gates**, but many papers will give some peepholes and not others.

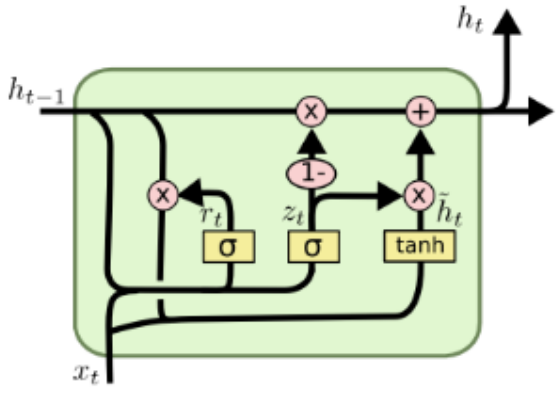
Variants on LSTM (coupled forget and input gates)



$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

Coupled forget and input gates: Instead of separately deciding what to forget and what we should add new information to, we make those decisions together.

Variants on LSTM (GRU)



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

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

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

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

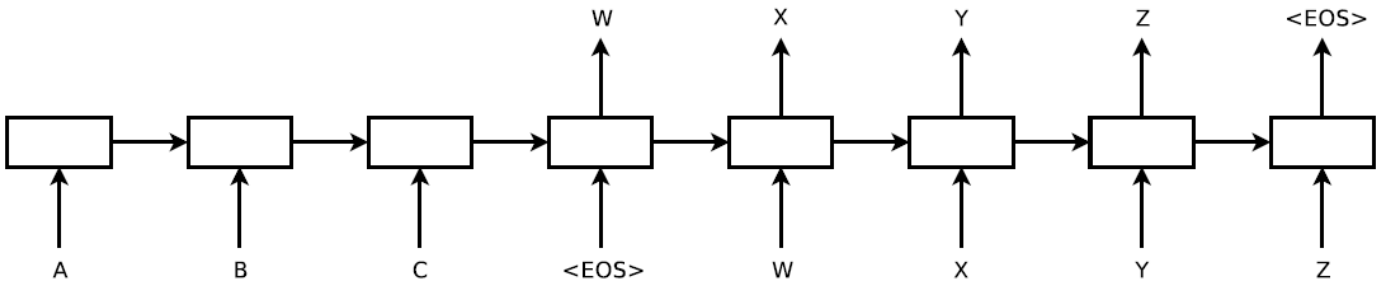
1. Combines the forget and input gates into a single “update gate.”

2. Merges the cell state and hidden state, and makes some other changes.

Sequence-to-sequence language translation

- Sutskever, Vinyals, and Le NIPS 2014
- 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 $\mathbf{v}$, which is given by the last hidden sate of the LSTM
- Then conditioned on $\mathbf{v}$, a second LSTM generates the output sequence $(\mathbf{y}_1, \ldots, \mathbf{y}_{T'})$ and computes its probability

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

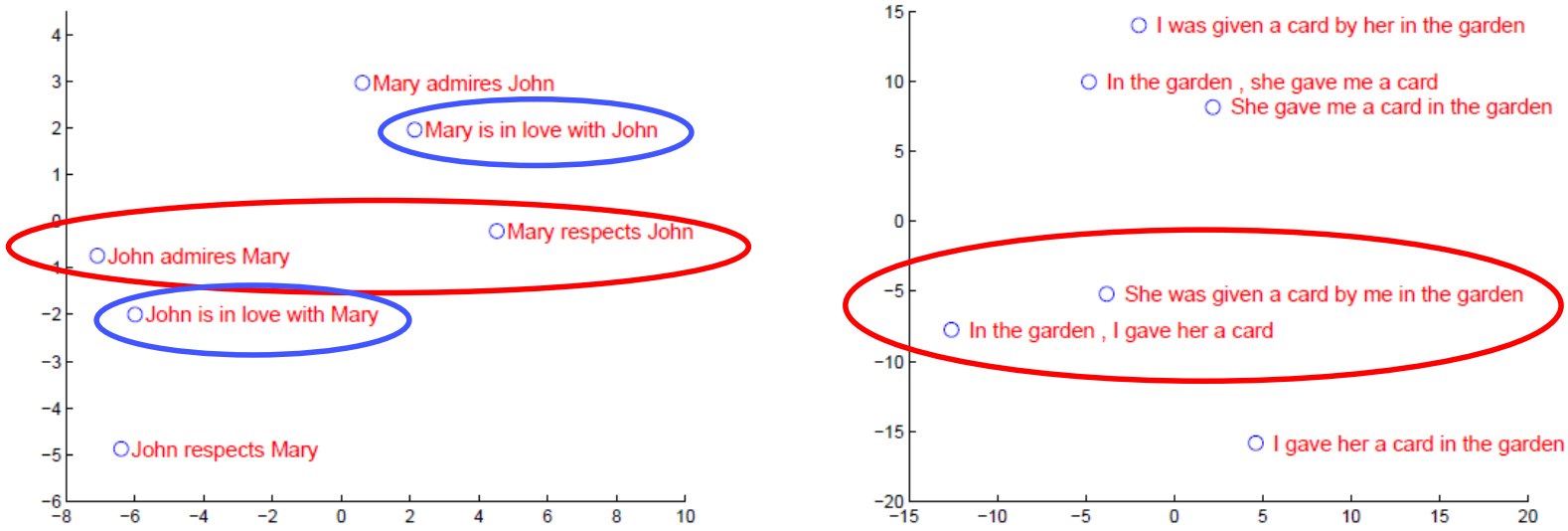


The model reads an input sentence “ABC” and produces “WXYZ” as the output sentence. The model stops making predictions after outputting the end-of-sentence token.

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 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.

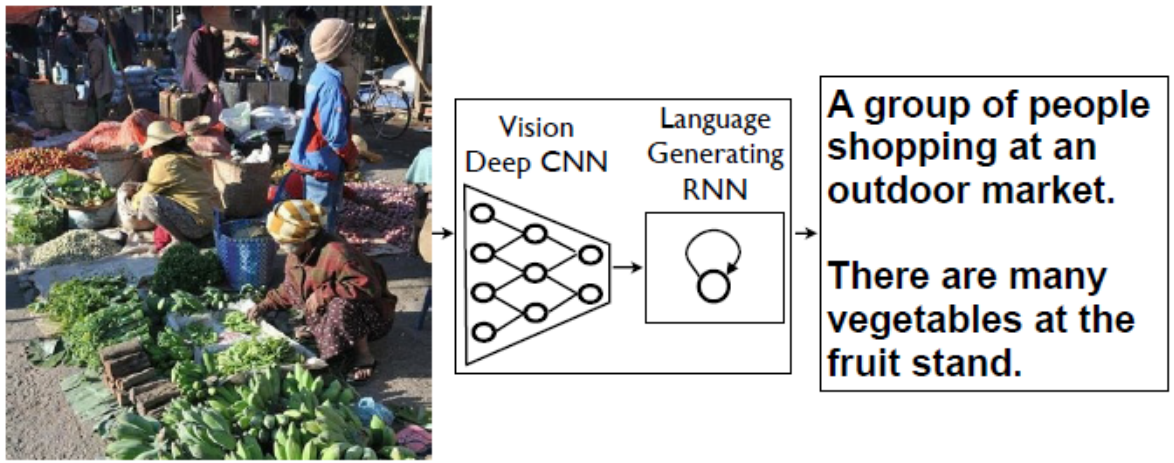
Sequence-to-sequence language translation

- 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 " .

Generate image caption

- 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



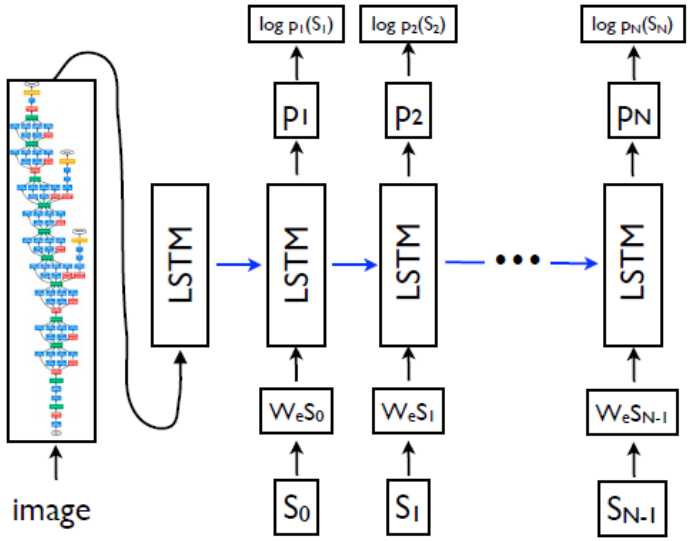
Generate image caption

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

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

$$\log P(\mathbf{S}|\mathbf{I}) = \sum_{t=0}^{N} \log P(\mathbf{S}_t|I,\mathbf{S}_0,\ldots,\mathbf{S}_{t-1})$$

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



Generate image caption

- Denote by $\mathbf{S}_0$ a special start work and by $\mathbf{S}_N$ 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 $\mathbf{W}_e$
- The image $\mathbf{I}$ is only input once at $t - 1$ 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

$$\mathbf{x}_{-1} = \text{CNN}(\mathbf{I})$$

$$\mathbf{x}_t = \mathbf{W}_e \mathbf{S}_t, t \in \{0, \ldots, N-1\}$$

$$P_{t+1} = \text{LSTM}(\mathbf{x}_t), t \in \{0, \ldots, N-1\}$$

$$L(\mathbf{I}, \mathbf{S}) = -\sum_{t=1}^{N} \log P_t(\mathbf{S}_t)$$

Translate videos to sentences

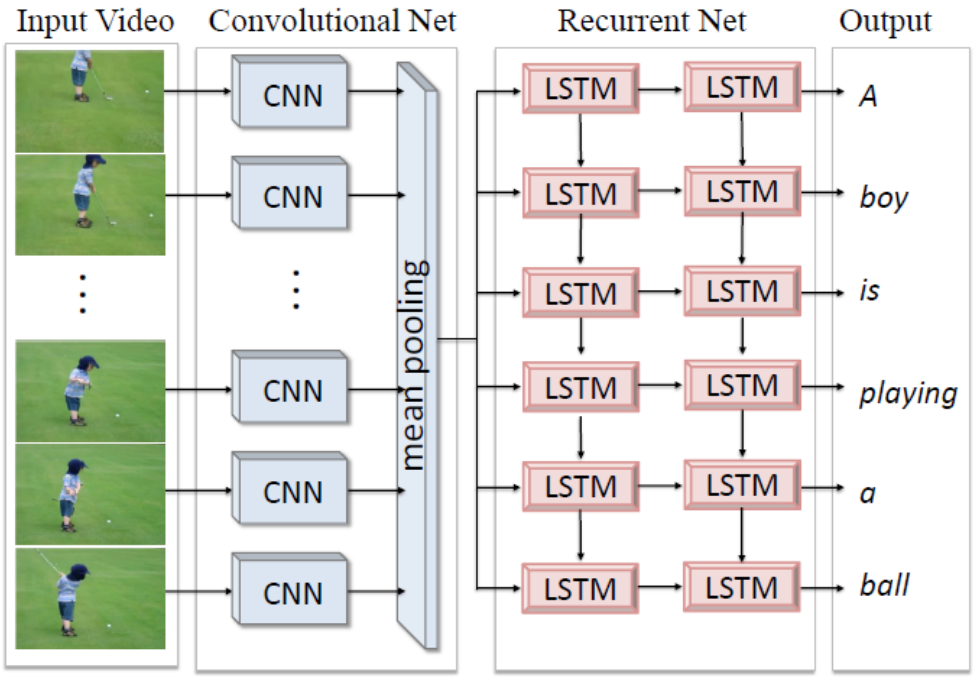
- Venugopalan et al. arXiv 2014
- The challenge is to capture the joint dependencies of a sequence of frames and a corresponding sequence of words
- Previous works simplified the problem by detecting a fixed set of semantic roles, such as subject, verb, and object, as an intermediate representation and adopted oversimplified rigid sentence templates.

Input video:

Machine output: *A cat is playing with toy.*
Humans: *A Ferret and cat fighting with each other. / A cat and a ferret are playing. / A kitten and a ferret are playfully wresting.*

Translate videos to sentences

- Each frame is modeled as CNN pre-trained on ImageNet
- The meaning sate and sequence of words is modeled by a RNN pre-trained on images with associated with sentence captions



Translate videos to sentences

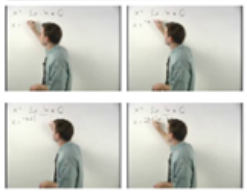
- Use CNN to convert a video to a fixed length representation vector $\mathbf{v}$
- Use RNN to decode the vector into a sentence just like language translation

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

- Use two layers of LSTMs (one LSTM stacked on top of another)

Translate videos to sentences

FGM: A person is playing a guitar in the house.
YT: A group of performing on stage.
YT_C: A man is doing a trick.
YT_CF: **A man is jumping on a pole.**
GT: Two men working on a high building.



FGM: A person is playing a guitar in the house.
YT: A boy is walking.
YT_C: A man is doing a women.
YT_CF: **A man is talking on a wall.**
GT: A man is doing algebraic equations on a white board.

FGM: A person is riding the horse
YT: A group of running.
YT_C: **A group of elephants.**
YT_CF: A group of elephants are walking on a horse.
GT: An elephant leads it's young.

FGM: A person playing the goal of the road.
YT: A player player in a goal.
YT_C: **A man playing a soccer ball.**
YT_CF: **A soccer player is running.**
GT: Two teams are playing soccer.

FGM: A person is running a race on the road.
YT: A group of running.
YT_C: **A group of people are running.**
YT_CF: A man is running.
GT: Eight men are running a race on a track.

FGM: factor graph model, using templates to generate sentences

YT: LSTM trained on the YouTube video dataset
YT_C: LSTM with pre-training on the Coco image dataset
YT_CF: LSTM with pre-training on the CoCo and Flicker image datasets
GT: Ground truth from human description

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.