Week 09

# SMILES & RNN

Prof. Ph.D. Woo Youn Kim

Chemistry, KAIST

# Goals

| | | |
|---|---|---|
| 7주 | 주제 | Molecular graphs & Graph Neural Network (GNN) |
| | 목표 | Understanding GCN and molecular representation with graphs |
| | 내용 | Molecular graph representation, graph convolutional network<br>Supervised learning of logP and TPSA |
| 9주 | 주제 | Recurrent Neural Network (RNN) |
| | 목표 | Understanding RNN and molecular representation with smiles |
| | 내용 | RNN, LSTM, GRU Feature extraction of molecules using RNN |
| 10주 | 주제 | Message Passing Neural Network (MPNN) |
| | 목표 | Understanding the most general expression of graph neural network |
| | 내용 | MPNN, molecular graph representation, GGNN,<br>supervised learning of logP and TPSA |

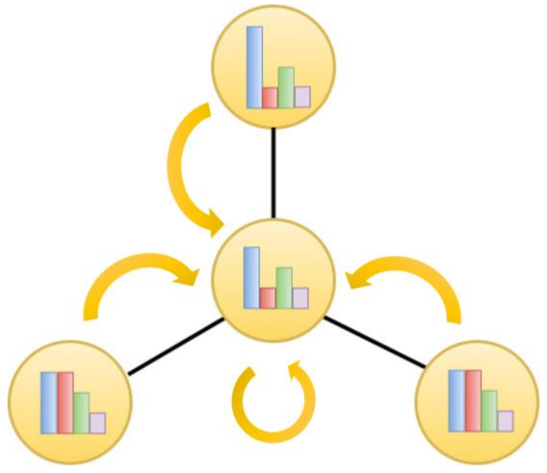
# Contents

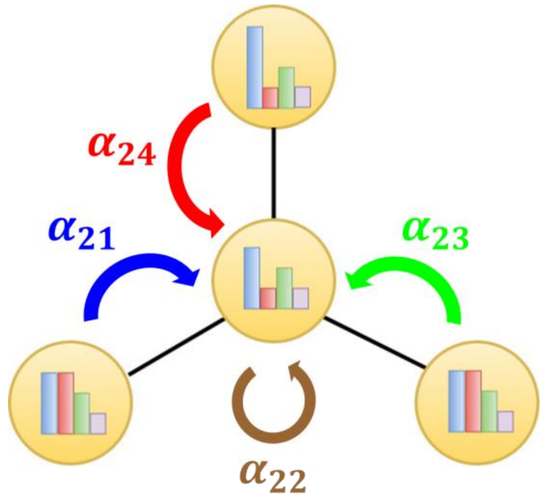
# Review

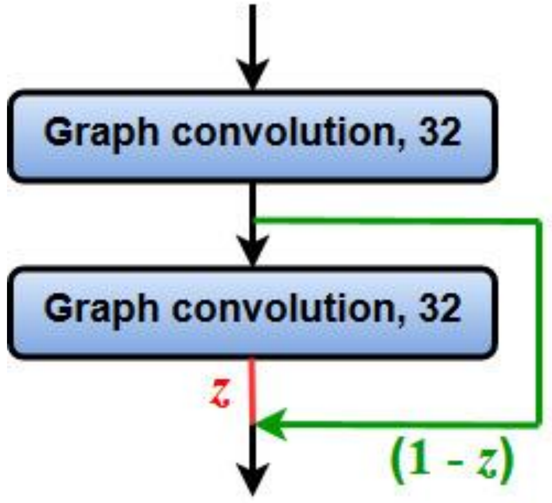
**Graph convolution** shares weights for all nodes in a graph

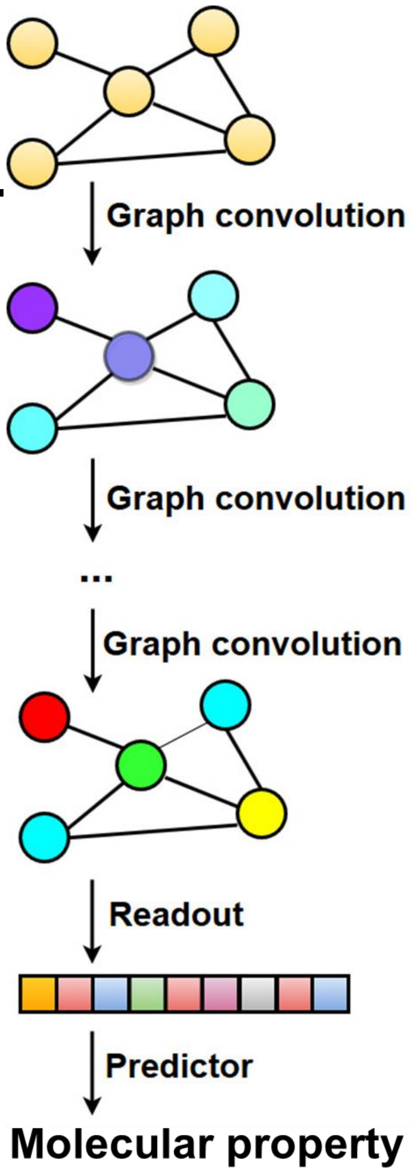
**Attention mechanism** mimics molecular interactions

**Using gated skip-connection** updates atom features much better.

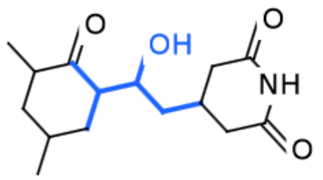


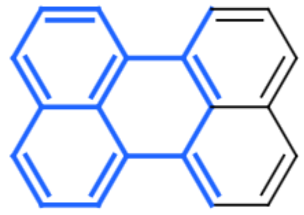


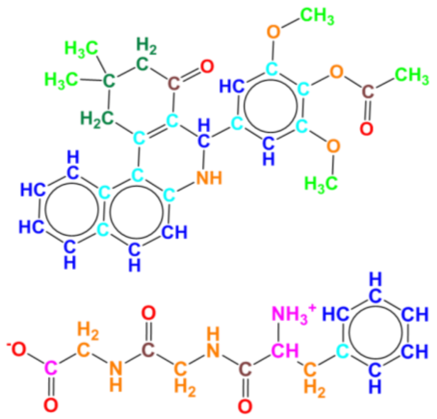


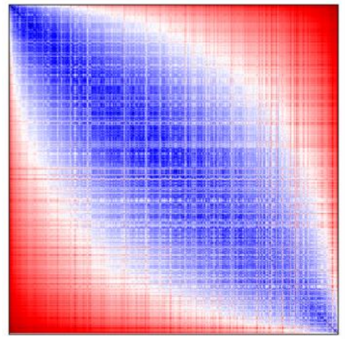


Neural machines can recognize the key features for molecular property predictions. In addition, **A better-developed model** performs **better** in **updating properties** and **predicting properties**.









## Introduction to RNN

## Why RNN?

**Example)** Neural machine translations in natural language processing (NLP)

“J’aime les chats” → Google Translate → “I like cats”

In this case, probability of expected outputs will be like below.

$$p(output = \text{"I like cats"}) > \text{p(output} = \text{"I table cats")}$$

and

$$p(output = \text{"I like cats"}) > \text{p(output} = \text{"like  I cats")}.$$

How can we implement a model suitable for such sequential problems?

# Why RNN?

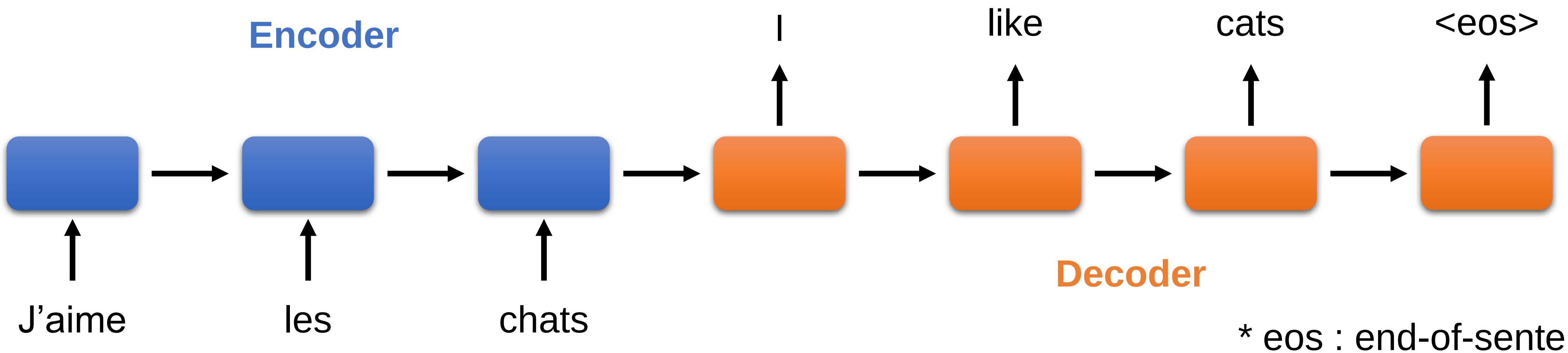
**Example)** Neural machine translations in natural language processing (NLP)

- Source : “J’aime les chats”
- Target : “I like cats”

Predict the output word-by-word (or sequentially):

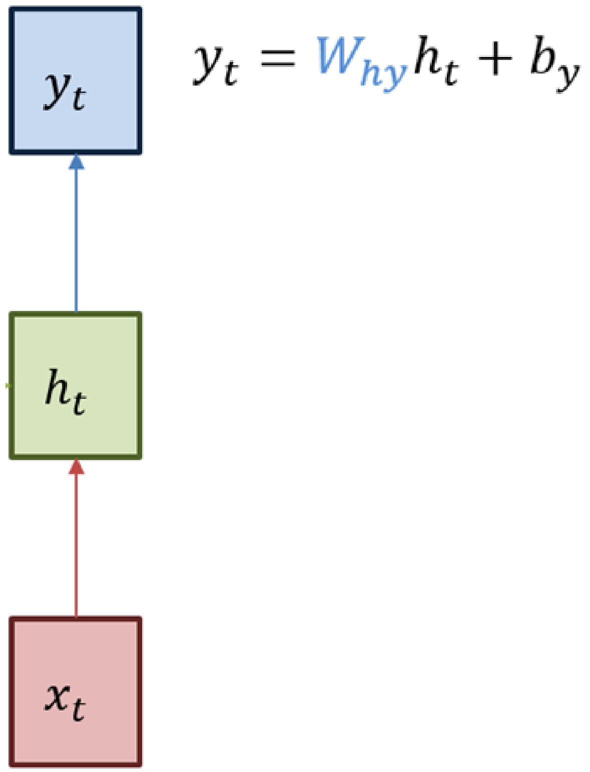
$$p_\theta(w_0|source) \cdot p_\theta(w_1|w_0, source) \cdot \dots \cdot p_\theta(w_n|w_{n-1}, w_{n-2}, \dots, w_0, source)$$

$p_\theta$ is parameterized by a neural network.

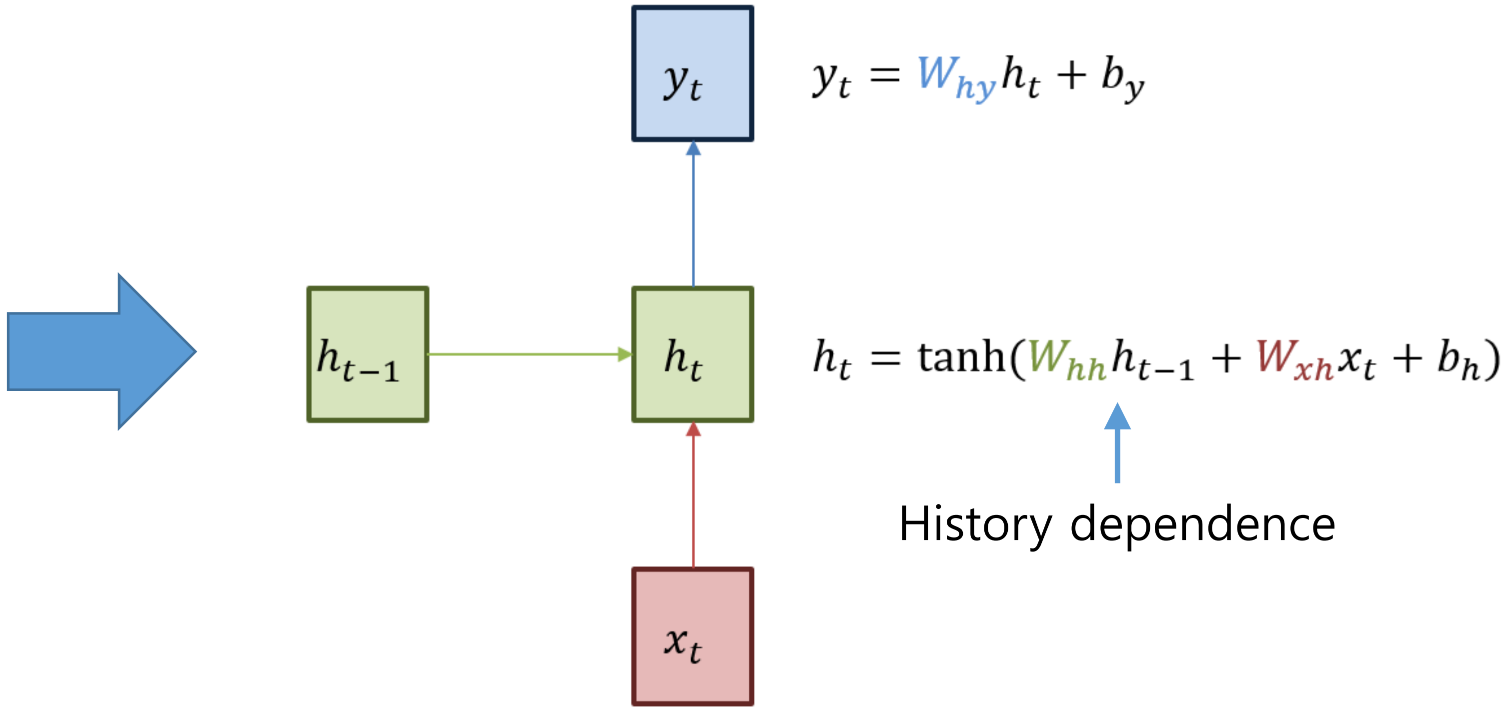


* eos : end-of-sentence

# Why RNN?



$y_t = W_{hy}h_t + b_y$

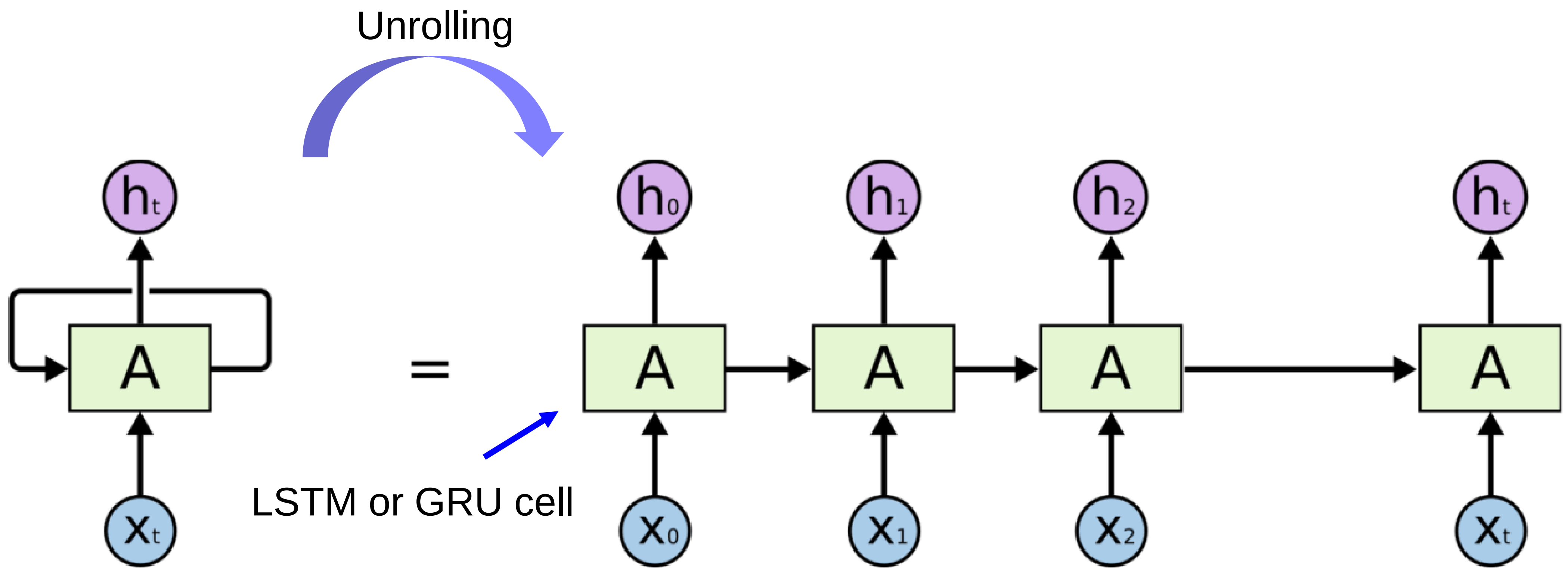


$y_t = W_{hy}h_t + b_y$

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

History dependence

# Why RNN?

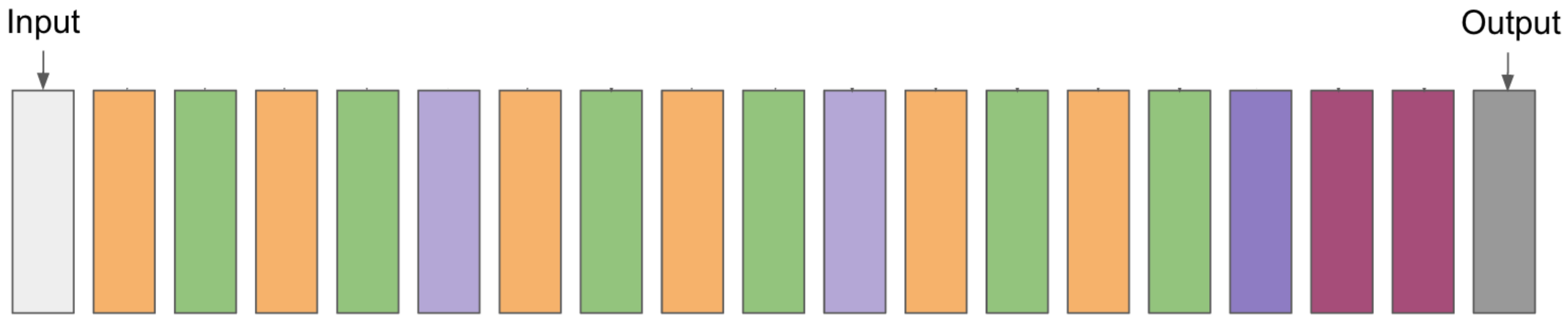


If an expected output is a function of the previous output and a new input, **RNN** is useful.

$$p_\theta(w_0|source) \cdot p_\theta(w_1|w_0, source) \cdot \ldots \cdot p_\theta(w_n|w_{n-1}, w_{n-2}, \ldots, w_0, source)$$

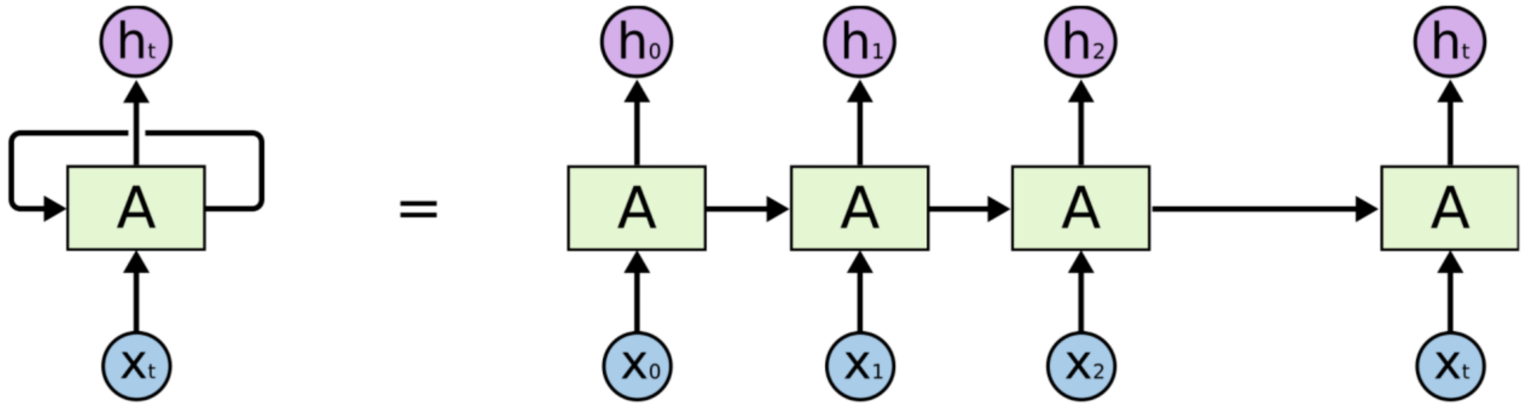
# Comparison to CNN or GCN

CNN: weight sharing within each layer and only one input and output layers

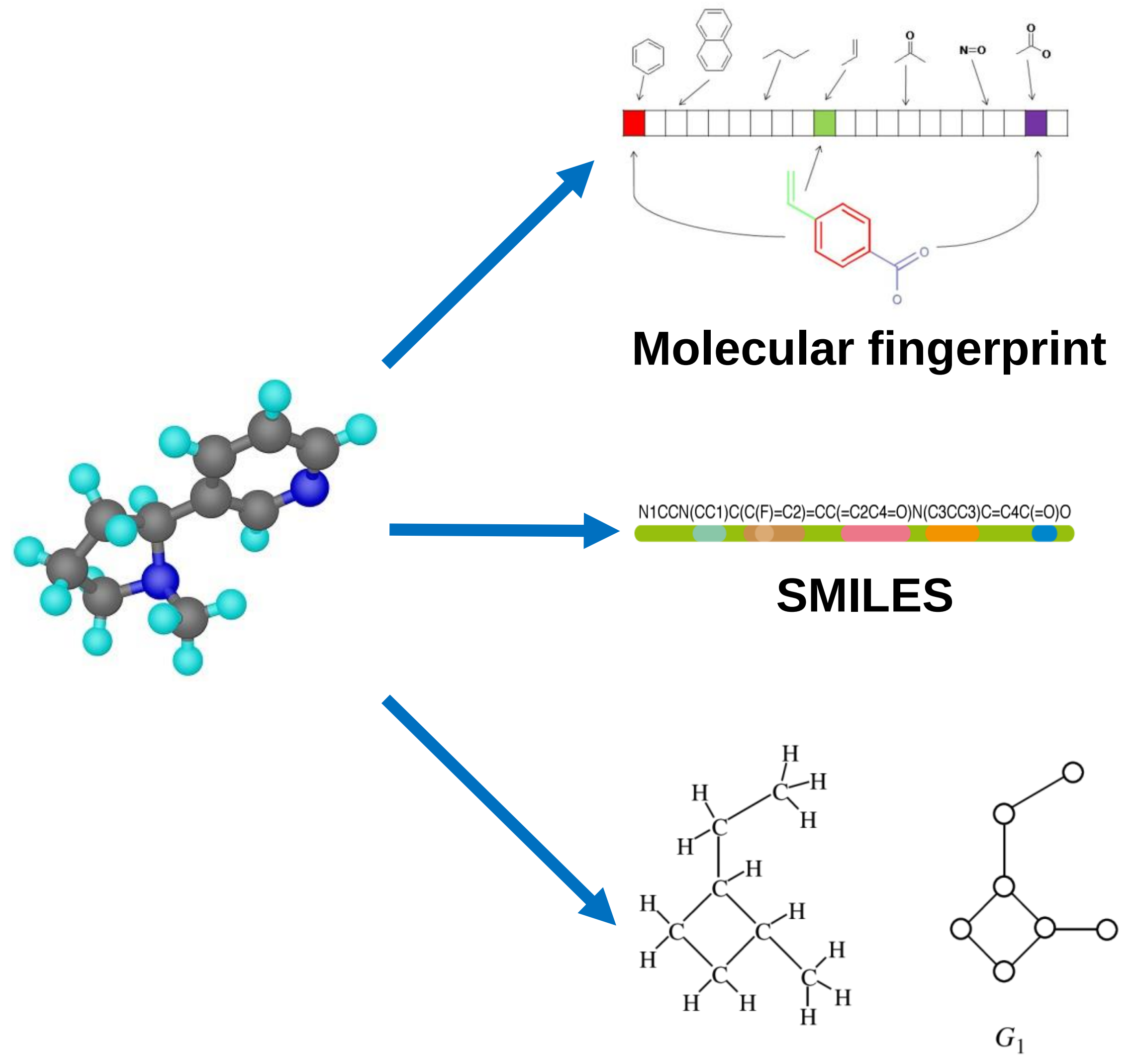


Example of a CNN Architecture

RNN: weight sharing through layers and sequential input and outputs



# Molecular representation



Molecular fingerprint → Multi-layer perceptron (MLP)

SMILES → Convolutional neural network (CNN)

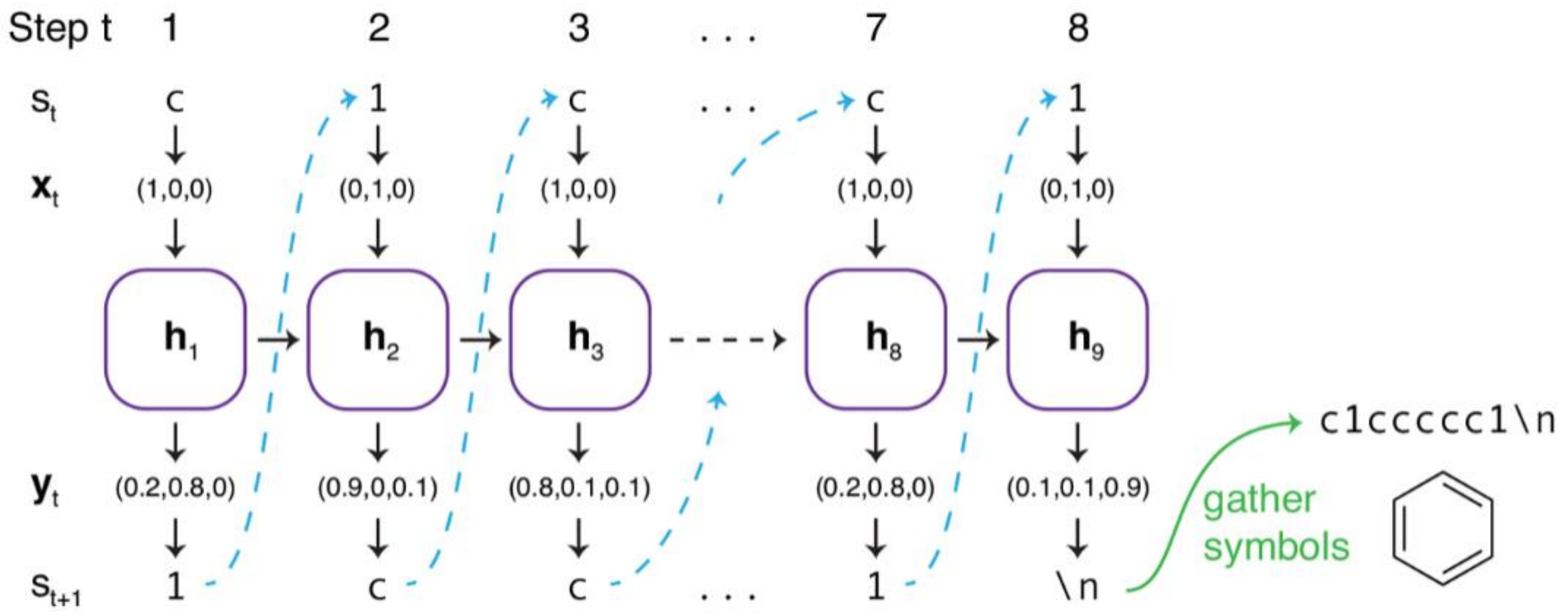
& Recurrent neural network (RNN)

Molecular graph → Graph neural network (GNN)

# Reference paper

## Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks

Marwin H. S. Segler,*[†] Thierry Kogej,[‡] Christian Tyrchan,[§] and Mark P. Waller*[||]



# Reference paper

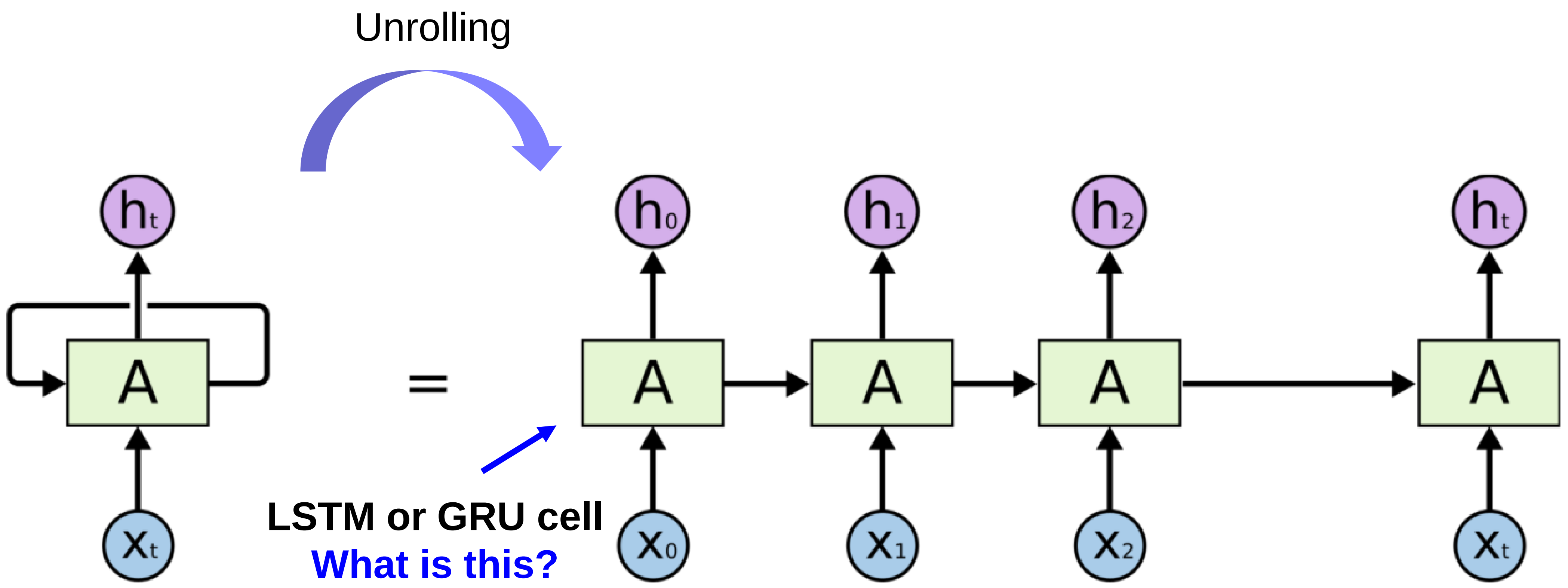
In *de novo* drug design, computational strategies are used to generate novel molecules with good affinity to the desired biological target. **In this work, we show that recurrent neural networks can be trained as generative models for molecular structures, similar to statistical language models in natural language processing.** We demonstrate that the properties of the generated molecules correlate very well with the properties of the molecules used to train the model. In order to enrich libraries with molecules active toward a given biological target, we propose to fine-tune the model with small sets of molecules, which are known to be active against that target. Against *Staphylococcus aureus*, the model reproduced 14% of 6051 hold-out test molecules that medicinal chemists designed, whereas against *Plasmodium falciparum* (Malaria), it reproduced 28% of 1240 test molecules. When coupled with a scoring function, our model can perform the complete *de novo* drug design cycle to generate large sets of novel molecules for drug discovery.

**RNN can be used to featurize the molecular structure represented in SMILES.**

Segler, Marwin HS, et al. "Generating focused molecule libraries for drug discovery with recurrent neural networks." *ACS central science* 4.1 (2017): 120-131.

## Operations in RNN

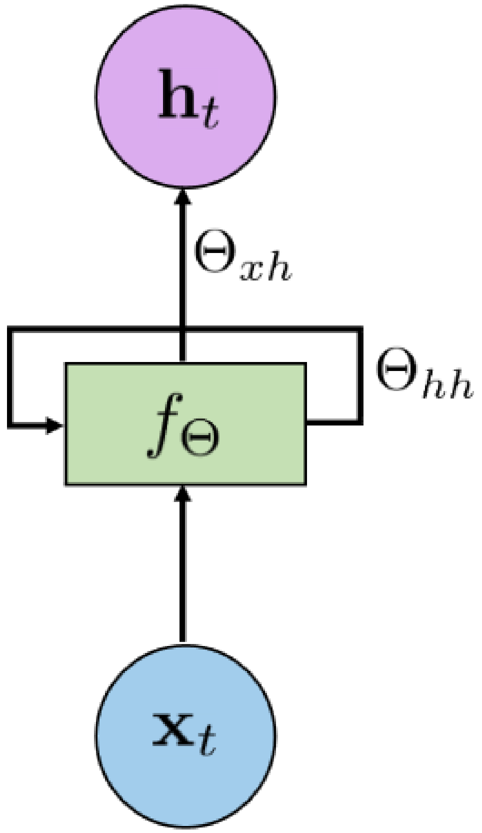
# Operations in RNN



**LSTM or GRU cell**

**What is this?**

# Operations in RNN



Input vector at time step t

$$\underbrace{\mathbf{h}_t}_{\text{New state}} = \underbrace{f}_{\text{Function parametrized by } \boldsymbol{\theta}}(\underbrace{\mathbf{h}_{t-1}}_{\text{Old state}}, \mathbf{x}_t; \boldsymbol{\theta})$$

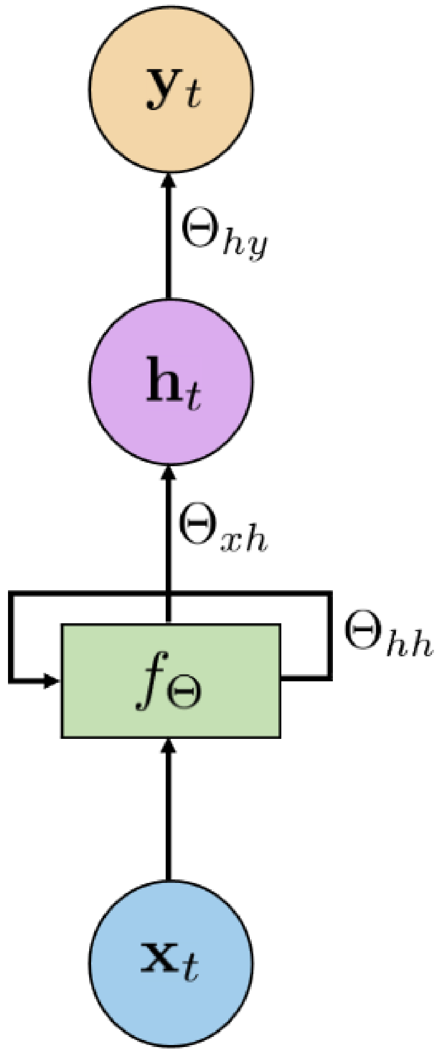
New state

Old state

Function parametrized by $\boldsymbol{\theta}$

Note that the new cell state is a function of the cell state in the previous state and a new input, which is called the autoregressive process in statistics.

# Operations in RNN



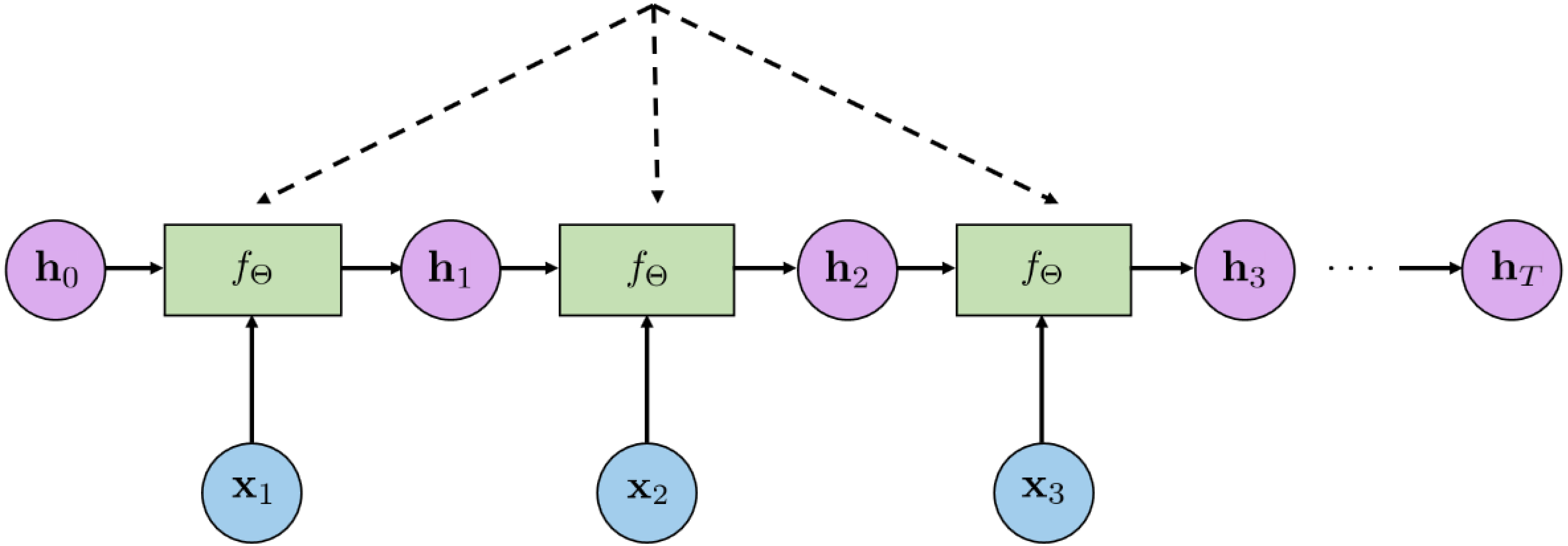
$$\mathbf{y}_t = \boldsymbol{\theta}_{hy}\mathbf{h}_t$$

Cost function in NLP: softmax

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

# Operations in RNN

Re-use the **same** weight matrix $\Theta$ at every time step

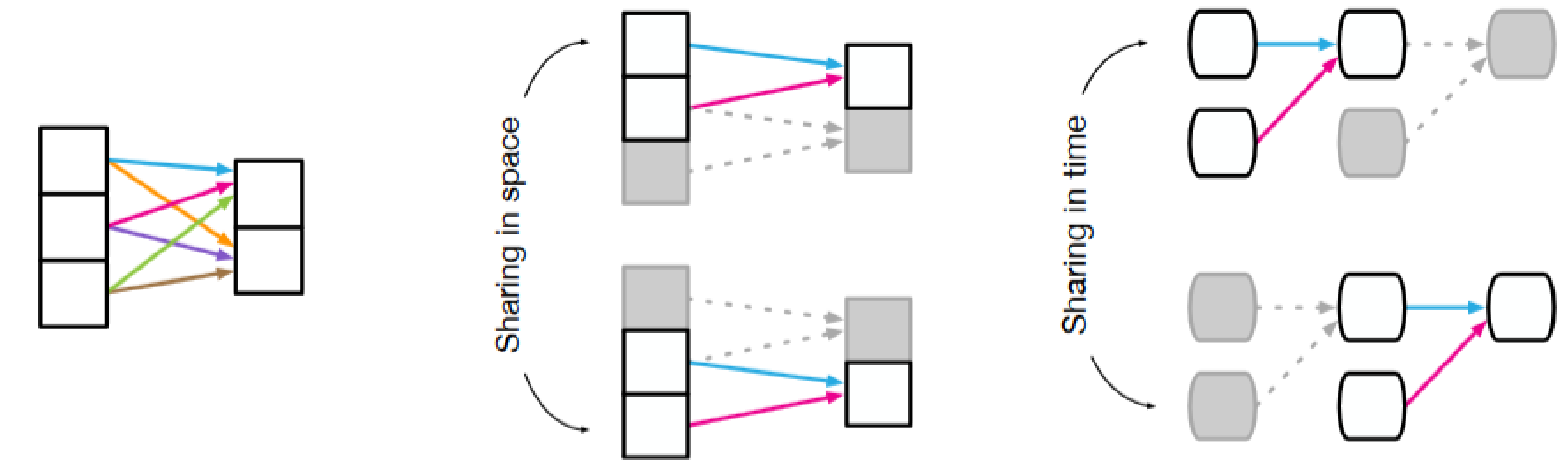


KAIST

http://alinlab.kaist.ac.kr/resource/Lec1_Introduction_to_NN.pdf

# Operations in RNN

Recall that the weight sharing is the key to making efficient neural networks.



(a) Fully connected (b) Convolutional (c) Recurrent

Models with proper **weight sharing**.

→ Capturing key features of data

→ Avoiding over-fitting

→ **Low computational cost**

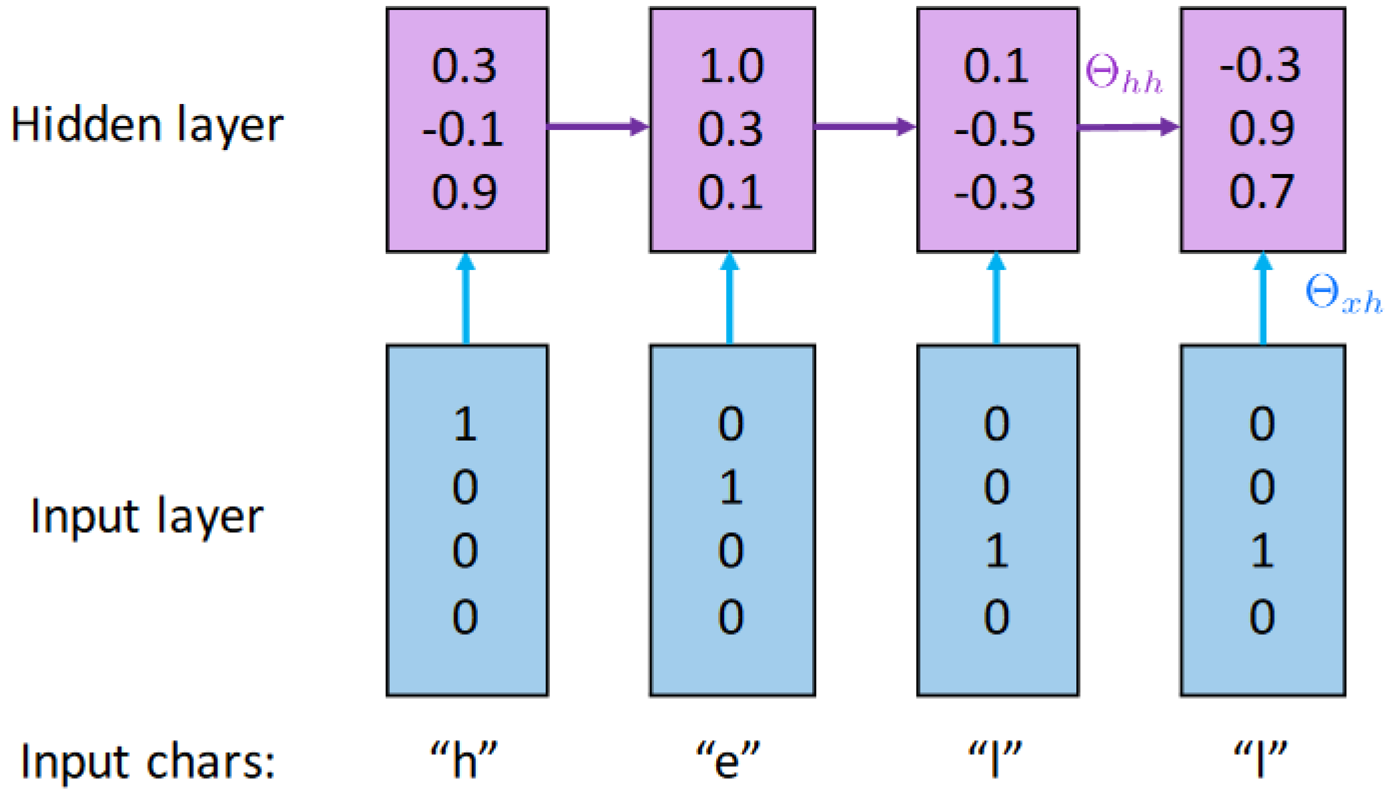
Battaglia, Peter W., et al.

# Operations in RNN

Character-level language model

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

- Vocabulary : [h,e,l,o]
- Example training sequence : “hello”



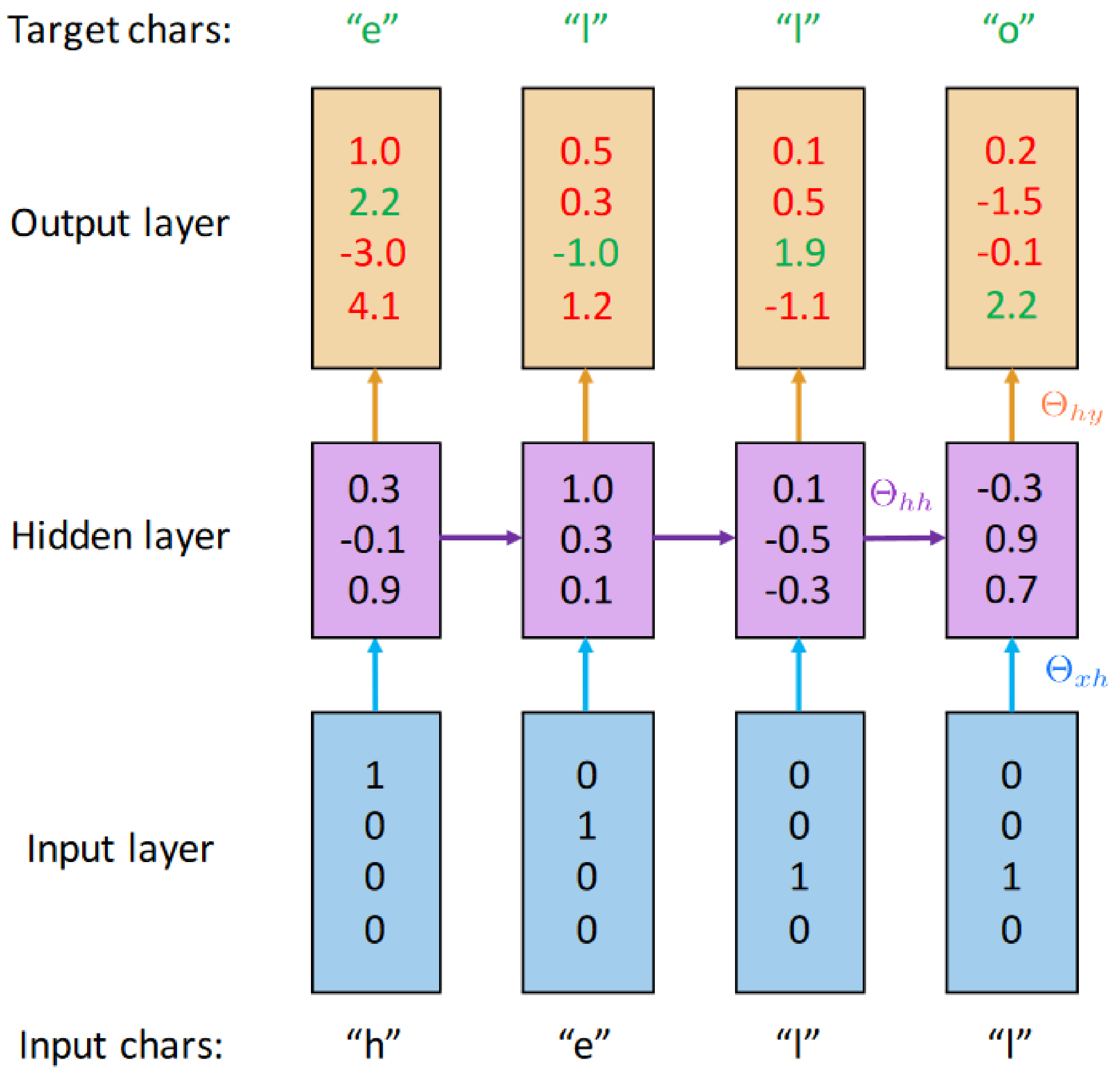
http://alinlab.kaist.ac.kr/resource/Lec1_Introduction_to_NN.pdf

# Operations in RNN

Character-level language model

- Vocabulary : [h,e,l,o]
- Example training sequence : “hello”
- Cost function in NLP: softmax

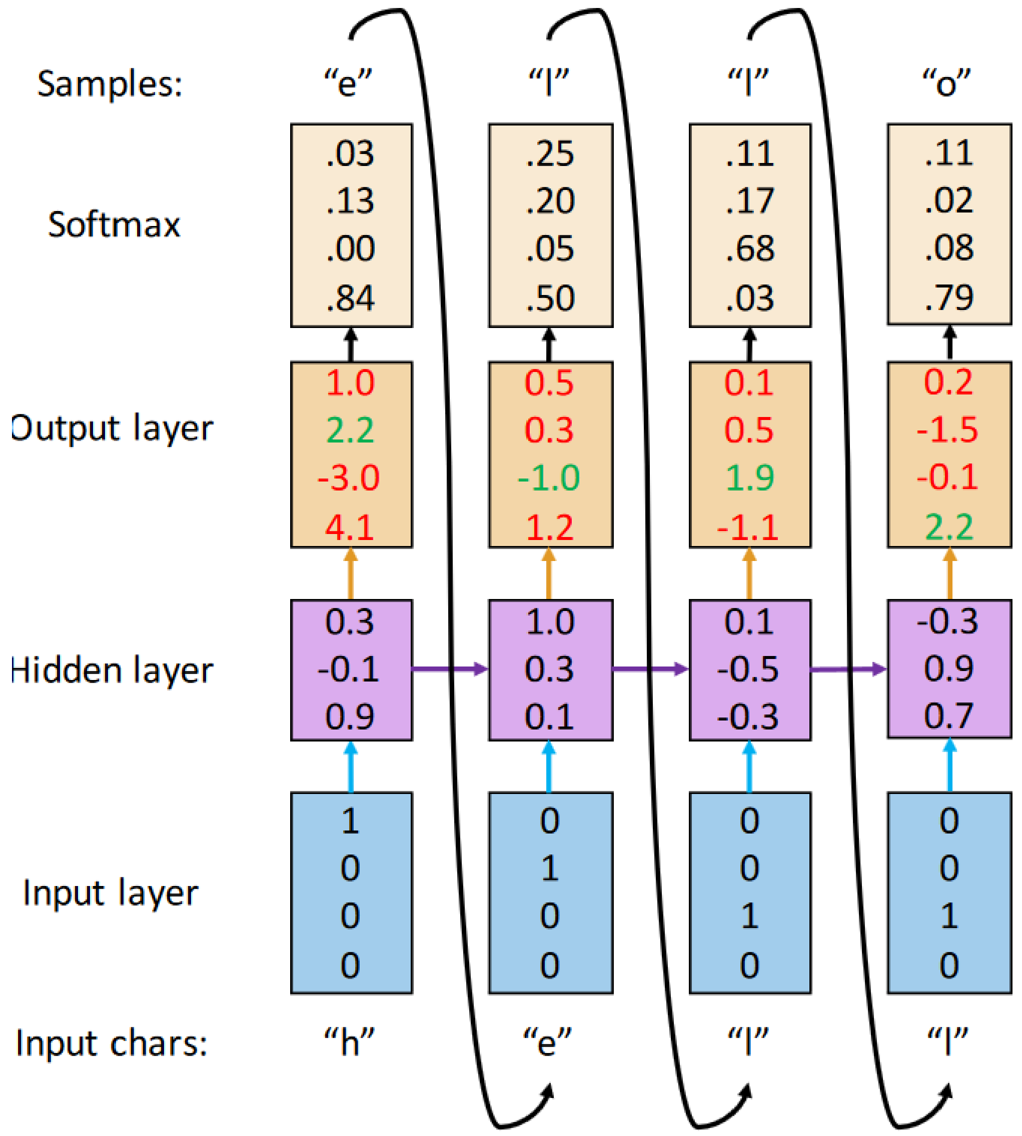
$$\mathbf{y}_t = \boldsymbol{\theta}_{hy}\mathbf{h}_t$$



# Operations in RNN

Character-level language model

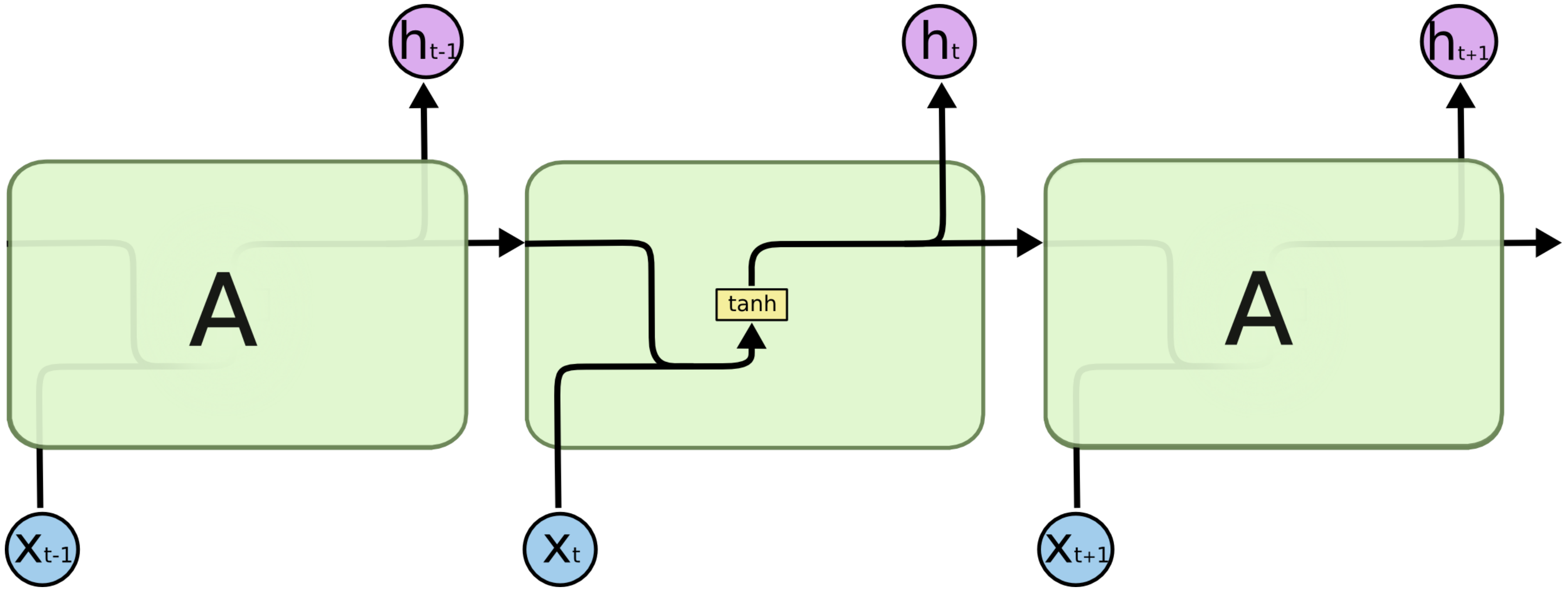
- Vocabulary : [h,e,l,o]
- Example training sequence : “hello”
- Cost function in NLP: softmax
- **At test time**, sample character one at a time and feedback to model



## Vanilla RNN

# Vanilla RNN

**1. Vanilla RNN cell** : the simplest and earliest recurrent unit



$$h_t = \tanh(W[h_{t-1}, x_t])$$

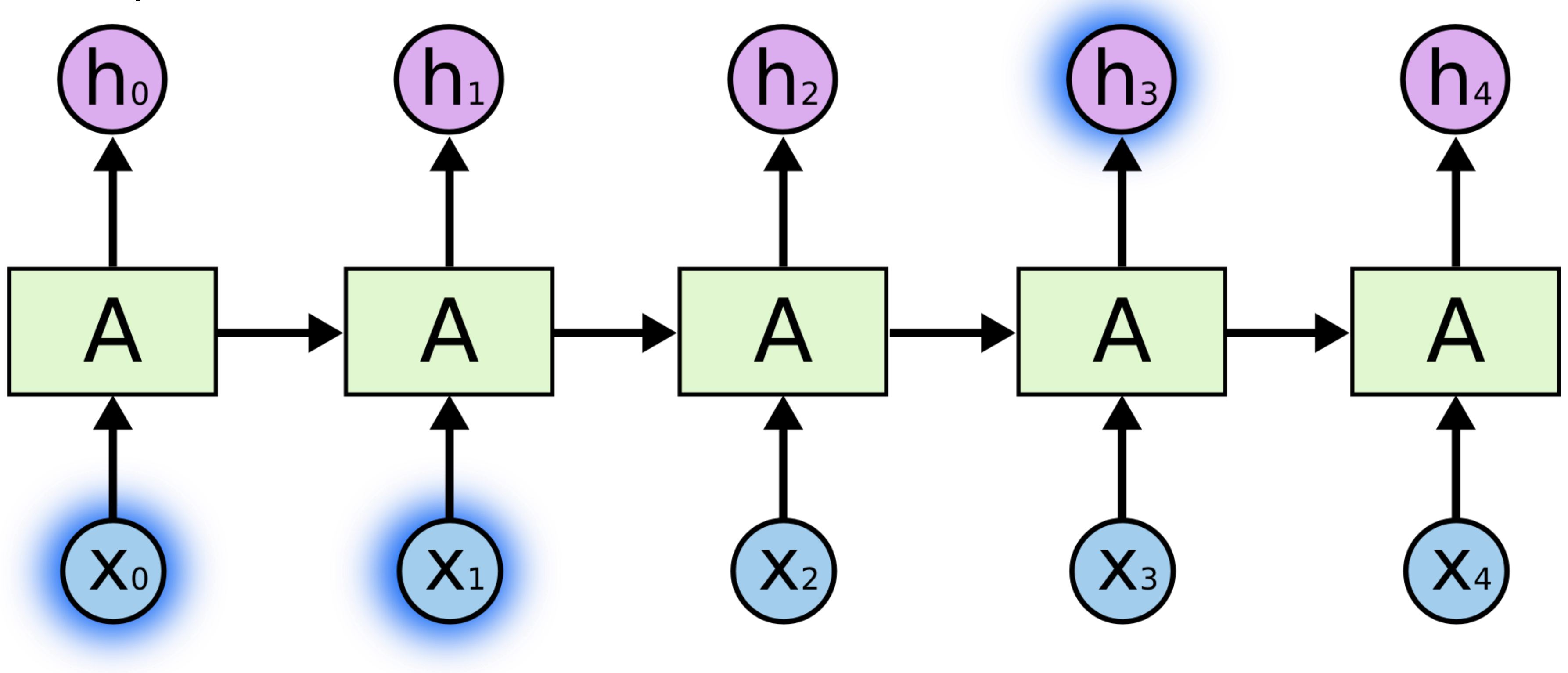
Christopher Olah,
who is a Google research scientist

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

# Vanilla RNN

**1. Vanilla RNN cell** : the simplest and earliest recurrent unit

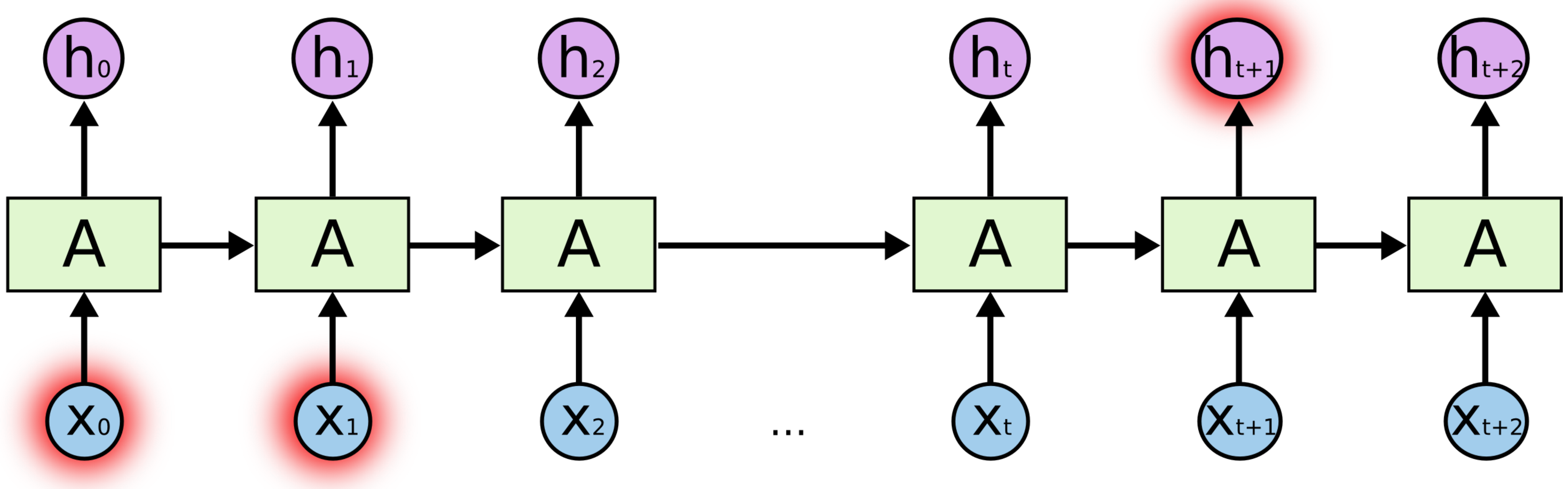
✓ Limitations : cannot reflect long-term dependencies (relation between input and output at a long distance)



# Vanilla RNN

**1. Vanilla RNN cell** : the simplest and earliest recurrent unit

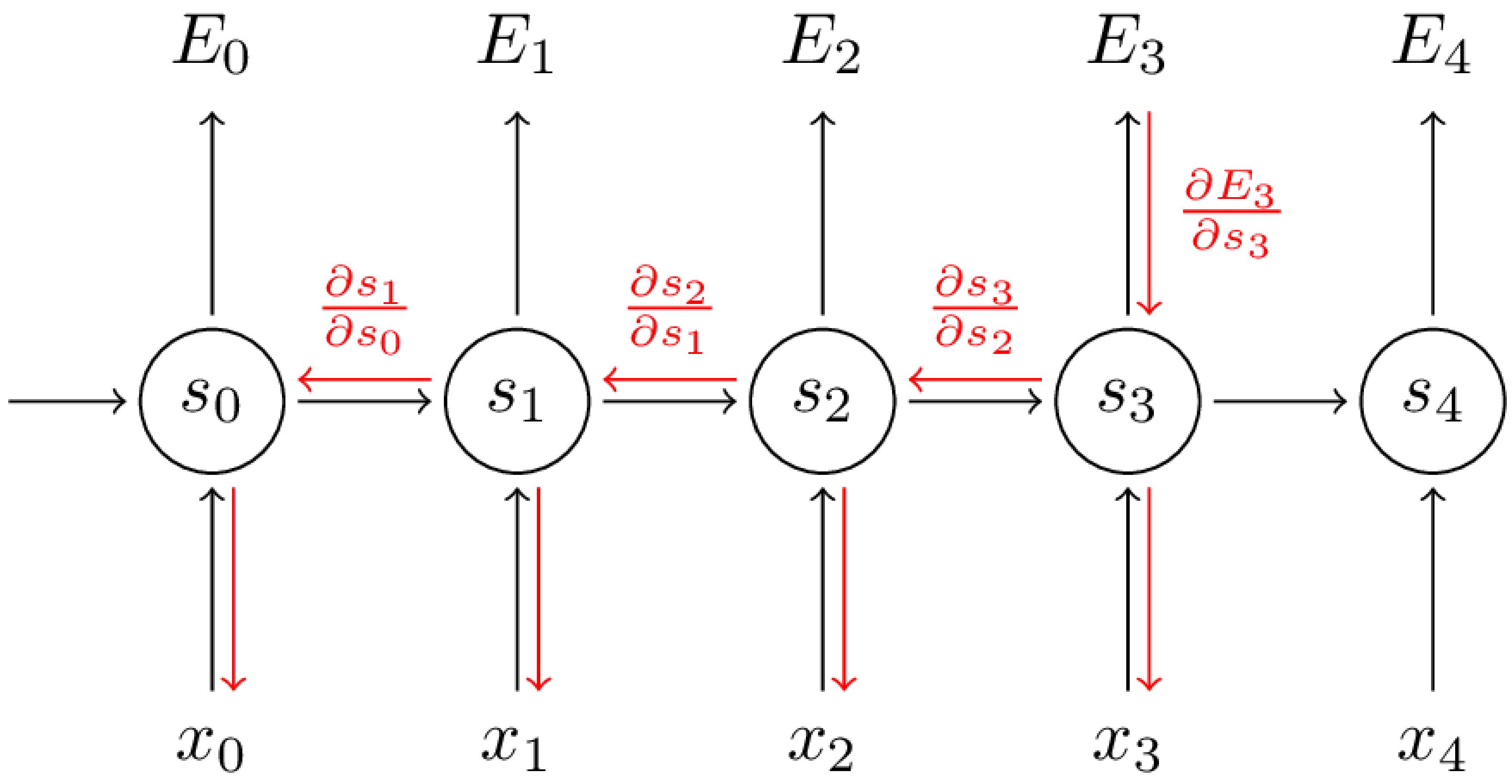
✓ Limitations : cannot reflect long-term dependencies



*Unfortunately, **as distance between input and output grows**, the **vanilla RNN** becomes unable to learn to connect the information. In theory, however, more advanced RNNs are absolutely capable of handling such **"long-term dependencies."***

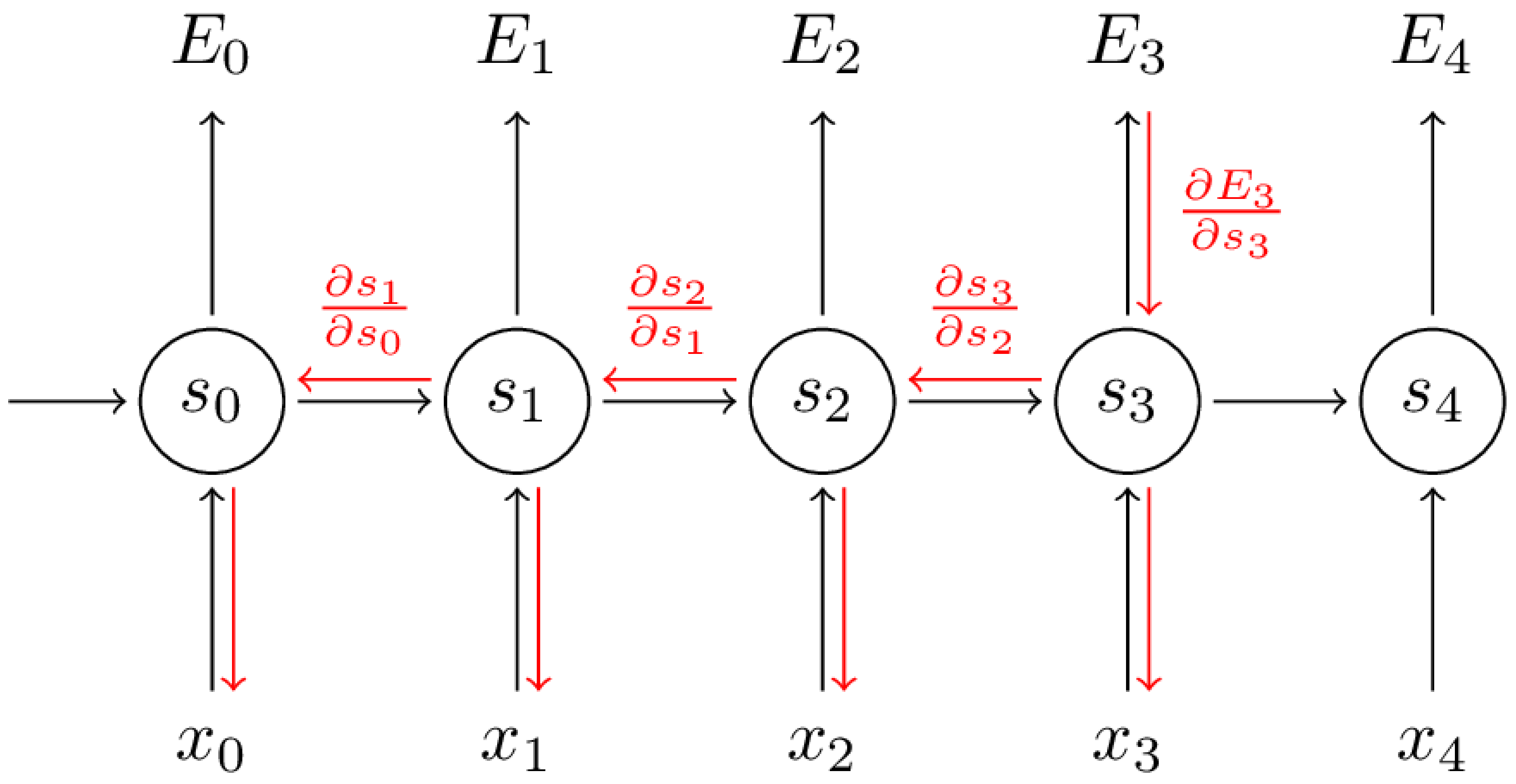
KAIST

# BackPropagation Through Time (BPTT)


$E_0$
$E_1$
$E_2$
$E_3$
$E_4$
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
$s_0$
$s_1$
$s_2$
$s_3$
$s_4$
$x_0$
$x_1$
$x_2$
$x_3$
$x_4$

Vanishing gradient becomes more severe in RNN as it often deals with long time sequential input and so requires as many layers as the number of inputs.

# BackPropagation Through Time (BPTT)



$E_0$
$E_1$
$E_2$
$E_3$
$E_4$
$\frac{\partial E_3}{\partial s_3}$
$\frac{\partial s_1}{\partial s_0}$
$\frac{\partial s_2}{\partial s_1}$
$\frac{\partial s_3}{\partial s_2}$
$s_0$
$s_1$
$s_2$
$s_3$
$s_4$
$x_0$
$x_1$
$x_2$
$x_3$
$x_4$

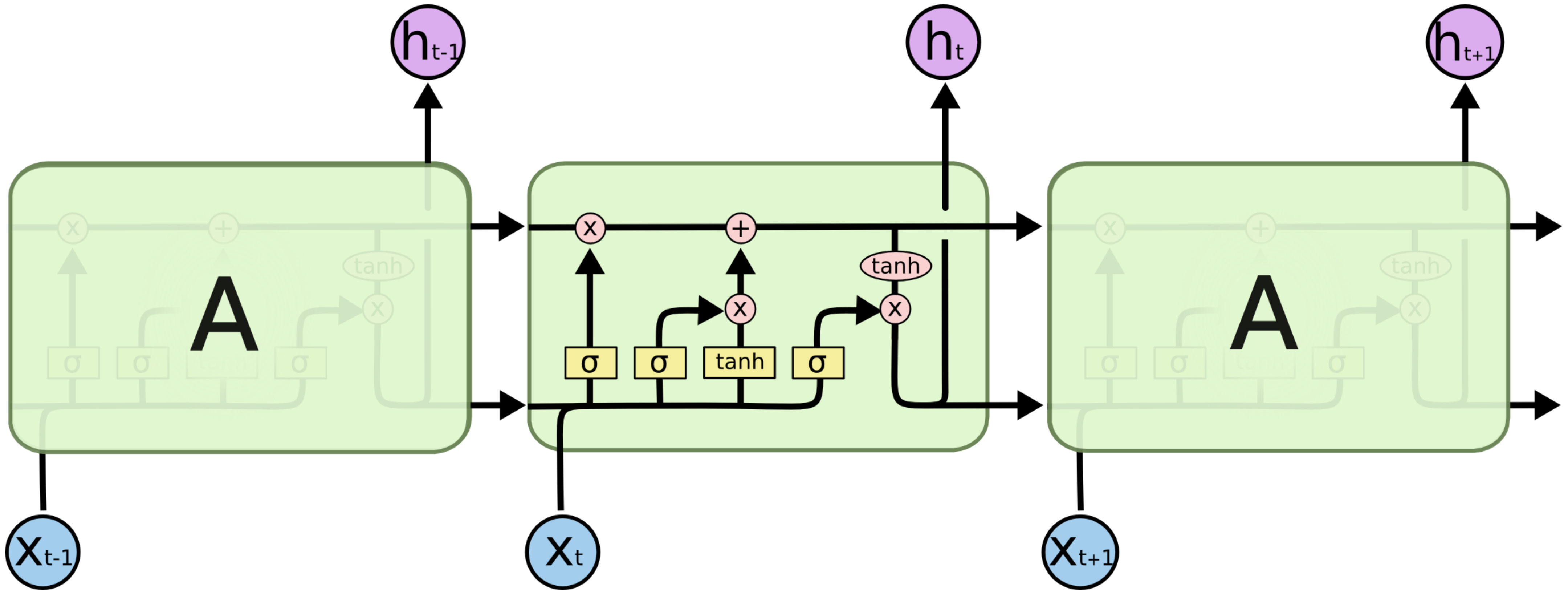
Exploding gradient can be also possible, but it can be easily noted while running a program (NaN). ReLU can be a viable solution, but LSTM and GRU are more popular.

# LSTM & GRU

Source: https://youtu.be/8HyCNIVRbSU

# LSTM

## 2. Long short term memory (LSTM)

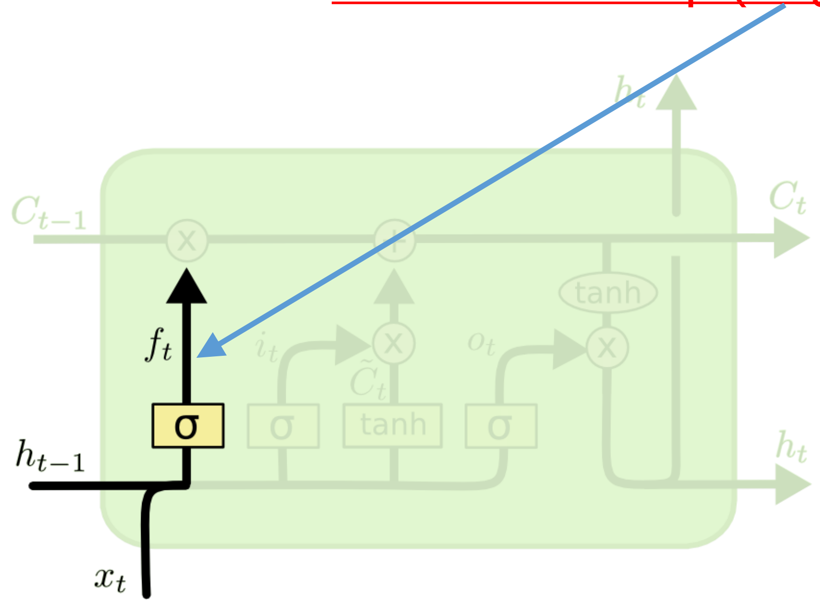


Hochreiter, Sepp, and Jürgen Schmidhuber.
"Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# LSTM

## 2. Long short term memory (LSTM)

STEP1: determine "How much keep (forget) the previous cell state $C_{t-1}$".



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

- $f_t$ : how much keeps the previous information (cell state)

Terminology

- $x_t$ : input vector
- $h_t$ : hidden state
- $C_t$ : cell state
- $f_t, i_t, o_t$ : forget, input, output gate

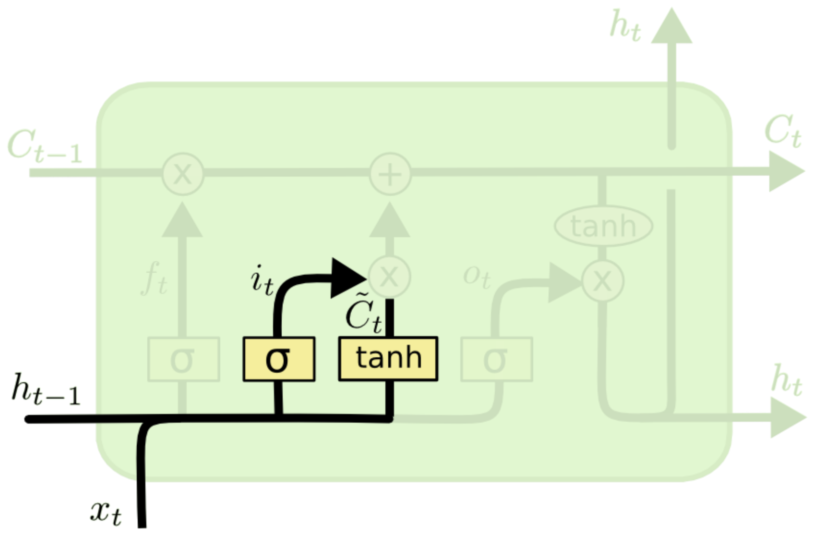
Note that all gates adopt a sigmoid function to bound the output between 0 and 1
While tanh is used to modulate the signal to squeeze it from -1 to 1.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# LSTM

## 2. Long short term memory (LSTM)

STEP2: determine how much new information to store in the cell state.



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

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

- $i_t$ : "input gate" decides which values we'll update.
- $\tilde{C}_t$ : a vector of new candidate values

Terminology

- $x_t$ : input vector
- $h_t$ : hidden state
- $C_t$ : cell state
- $f_t, i_t, o_t$ : forget, input, output gate

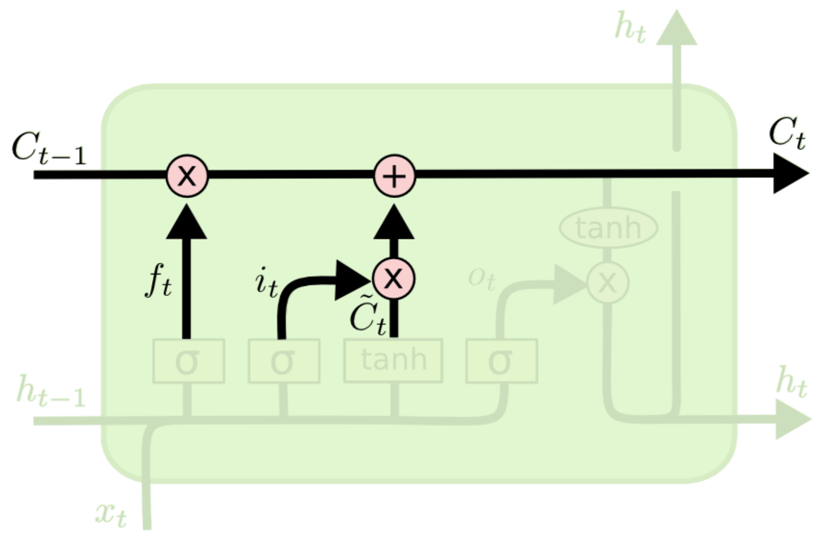
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

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

# LSTM

## 2. Long short term memory (LSTM)

STEP3: update the cell state



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

Terminology

- $x_t$ : input vector
- $h_t$ : hidden state
- $C_t$ : cell state
- $f_t, i_t, o_t$ : forget, input, output gate

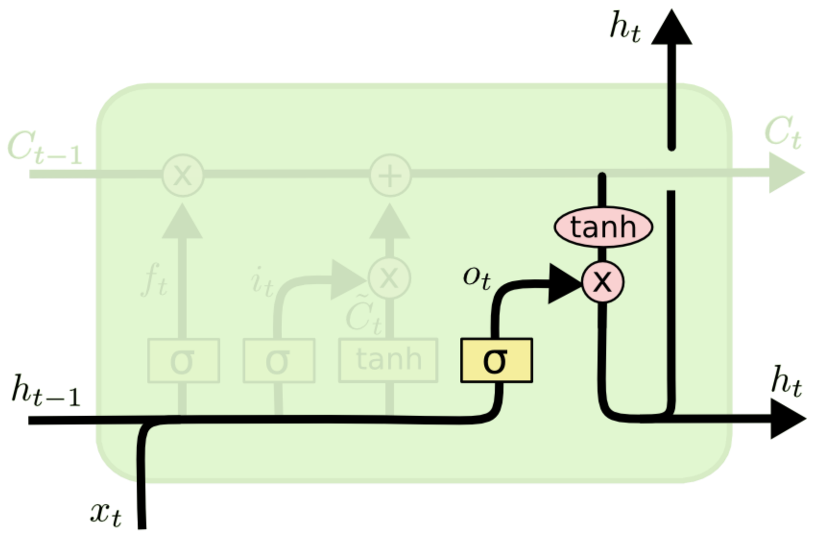
Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

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

# LSTM

## 2. Long short term memory (LSTM)

STEP4: update the hidden state



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

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

Terminology

- $x_t$ : input vector
- $h_t$ : hidden state
- $C_t$ : cell state
- $f_t, i_t, o_t$ : forget, input, output gate

LSTM resolves “the long-term dependencies” by adopting the gate units. However, it is quite computationally expensive.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

# LSTM

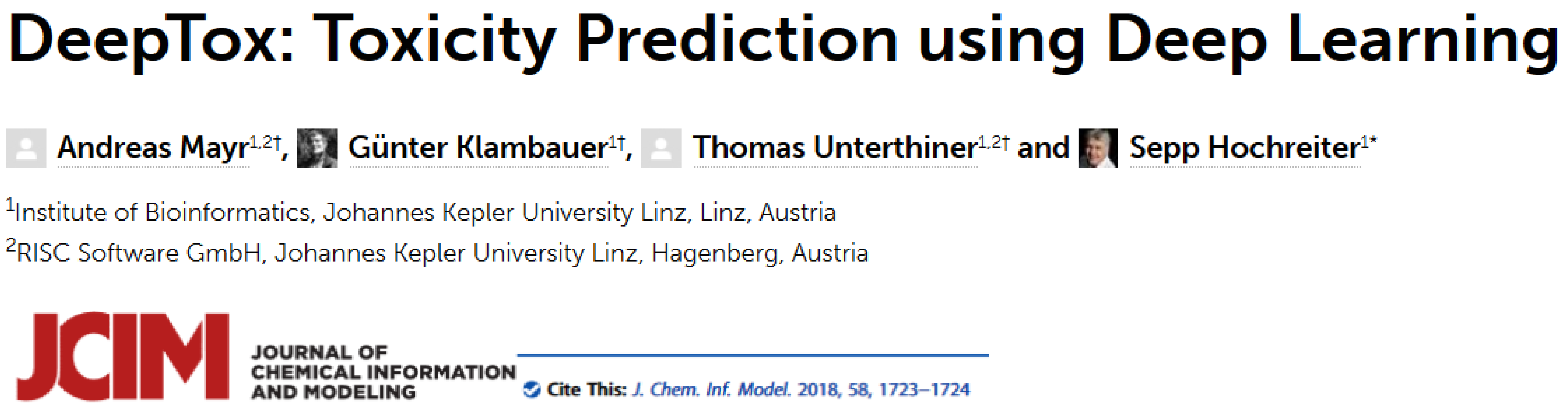
Trivia
: Sepp Hochreiter, who is the developer of LSTM, is also working on bioinformatics.

Sepp Hochreiter

FOLLOW

Professor of Machine Learning, Johannes Kepler University Linz
Verified email at bioinf.jku.at - Homepage

Machine Learning Deep Learning Artificial Intelligence Neural Networks Bioinformatics



DeepTox: Toxicity Prediction using Deep Learning

Andreas Mayr[1,2†], Günter Klambauer[1†], Thomas Unterthiner[1,2†] and Sepp Hochreiter[1*]

[1]Institute of Bioinformatics, Johannes Kepler University Linz, Linz, Austria
[2]RISC Software GmbH, Johannes Kepler University Linz, Hagenberg, Austria

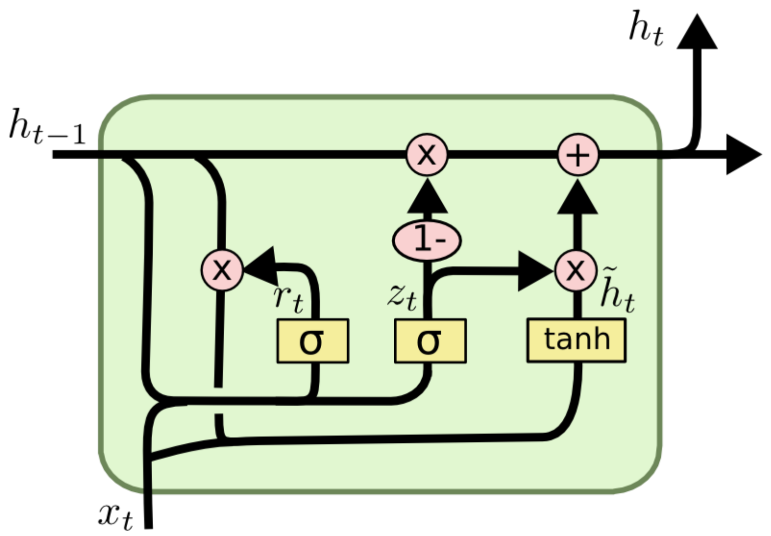
JCIM JOURNAL OF CHEMICAL INFORMATION AND MODELING

Cite This: J. Chem. Inf. Model. 2018, 58, 1723–1724

# GRU

## 3. Gated recurrent unit (GRU)

Recall gated skip-connection



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

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

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

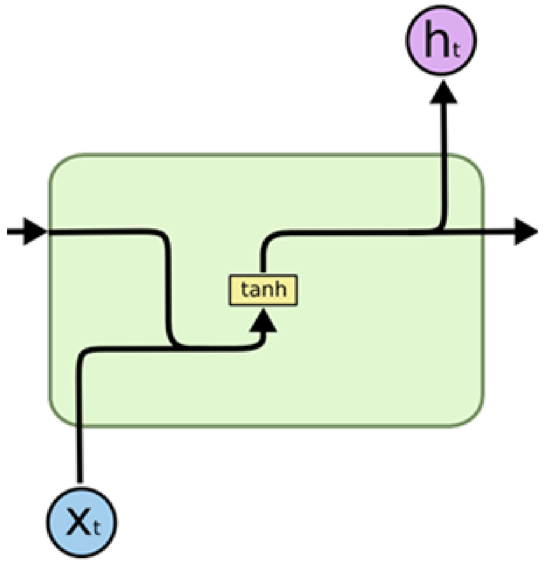
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

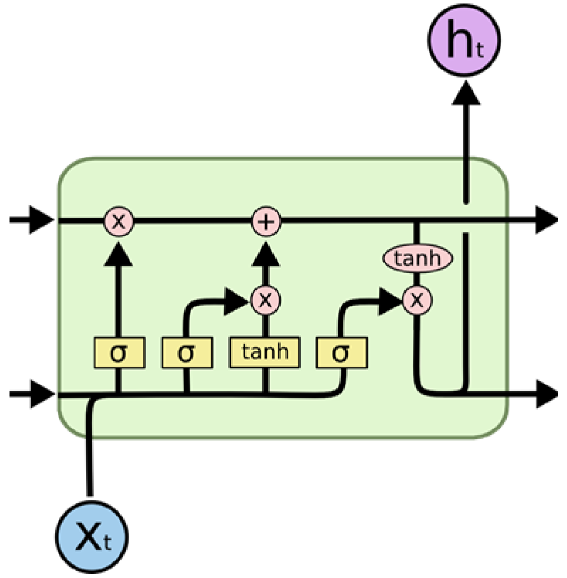
- $z_t$ : How much forgets the previous hidden state $h_{t-1}$ and keeps the temporary state $\tilde{h}_t$
- $r_t$ : How much keeps the previous hidden state $h_{t-1}$ in updating the temporary state $\tilde{h}_t$.
- $\tilde{h}_t$ : Temporary state

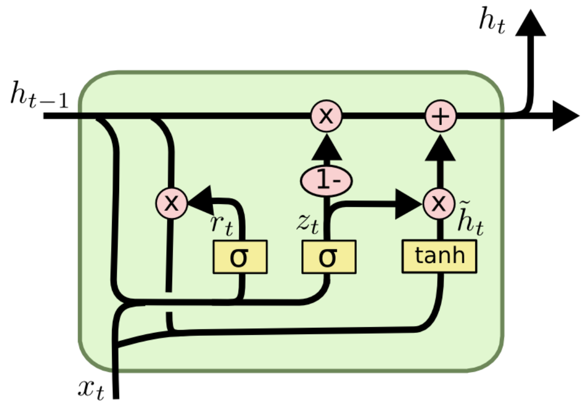
Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).

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

# Vanilla vs LSTM vs GRU







- # gates: vanilla = 0, LSTM = 3 (forget, input, output), GRU = 2 gates (reset and update gate)
- Memory: vanilla = no explicit, LSTM = $C_t$, GRU = combined in $h_t$.
- *i* and *f* gates of LSTM are integrated into z of GRU,
- f gate of LSTM is divided into r and z of GRU
- GRU uses less learning parameters than those of LSTM ➔ more efficient

Note that GRU = vanilla if $r_t = z_t = 1$.

http://aikorea.org/blog/rnn-tutorial-4/

# GRU

Trivia

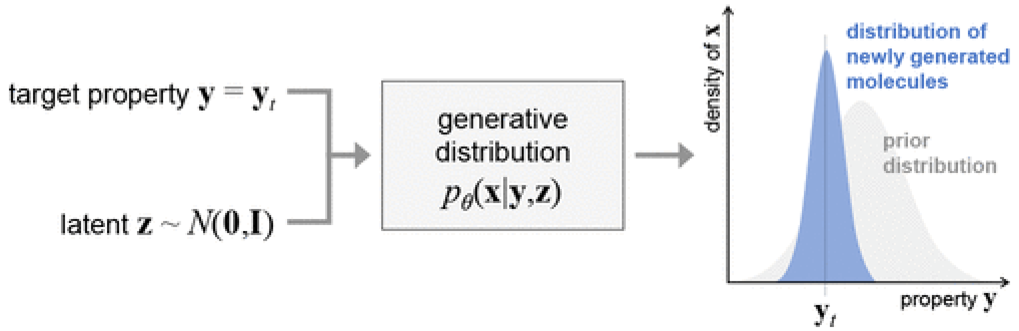
: Kyunghyun Cho, who is the professor of NYU and research scientist at Facebook AI Research (FAIR) and inventor of the GRU, worked on molecular application of deep learning.

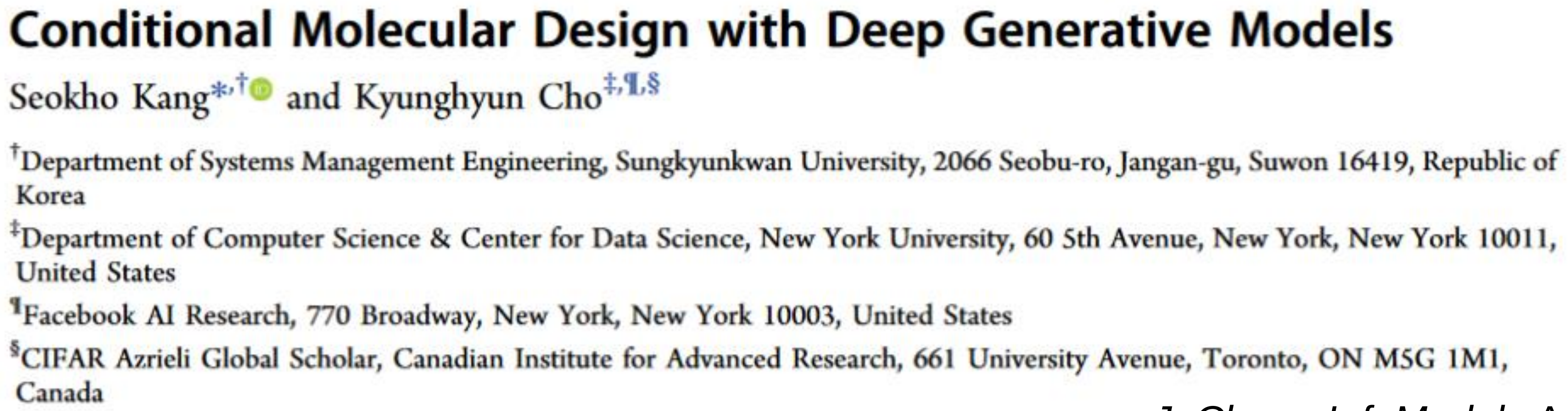
Kyunghyun Cho

New York University, Facebook AI Research

Verified email at nyu.edu - Homepage

Machine Learning Deep Learning


target property $\mathbf{y} = \mathbf{y}_t$
latent $\mathbf{z} \sim \mathcal{N}(\mathbf{0},\mathbf{I})$
generative distribution $p_\theta(\mathbf{x}|\mathbf{y},\mathbf{z})$
density of x
distribution of newly generated molecules
prior distribution
$\mathbf{y}_t$
property $\mathbf{y}$



**Conditional Molecular Design with Deep Generative Models**

Seokho Kang[*,†] and Kyunghyun Cho[‡,¶,§]

[†]Department of Systems Management Engineering, Sungkyunkwan University, 2066 Seobu-ro, Jangan-gu, Suwon 16419, Republic of Korea

[‡]Department of Computer Science & Center for Data Science, New York University, 60 5th Avenue, New York, New York 10011, United States

[¶]Facebook AI Research, 770 Broadway, New York, New York 10003, United States

[§]CIFAR Azrieli Global Scholar, Canadian Institute for Advanced Research, 661 University Avenue, Toronto, ON M5G 1M1, Canada

*J. Chem. Inf. Model.*, Article ASAP

**DOI:** 10.1021/acs.jcim.8b00263

# Discussion

What is memory?

How do you memorize something like apple?

Why has our brain been evolved to the present state based on neural network?

Compare the memory mechanism with electronic devices.

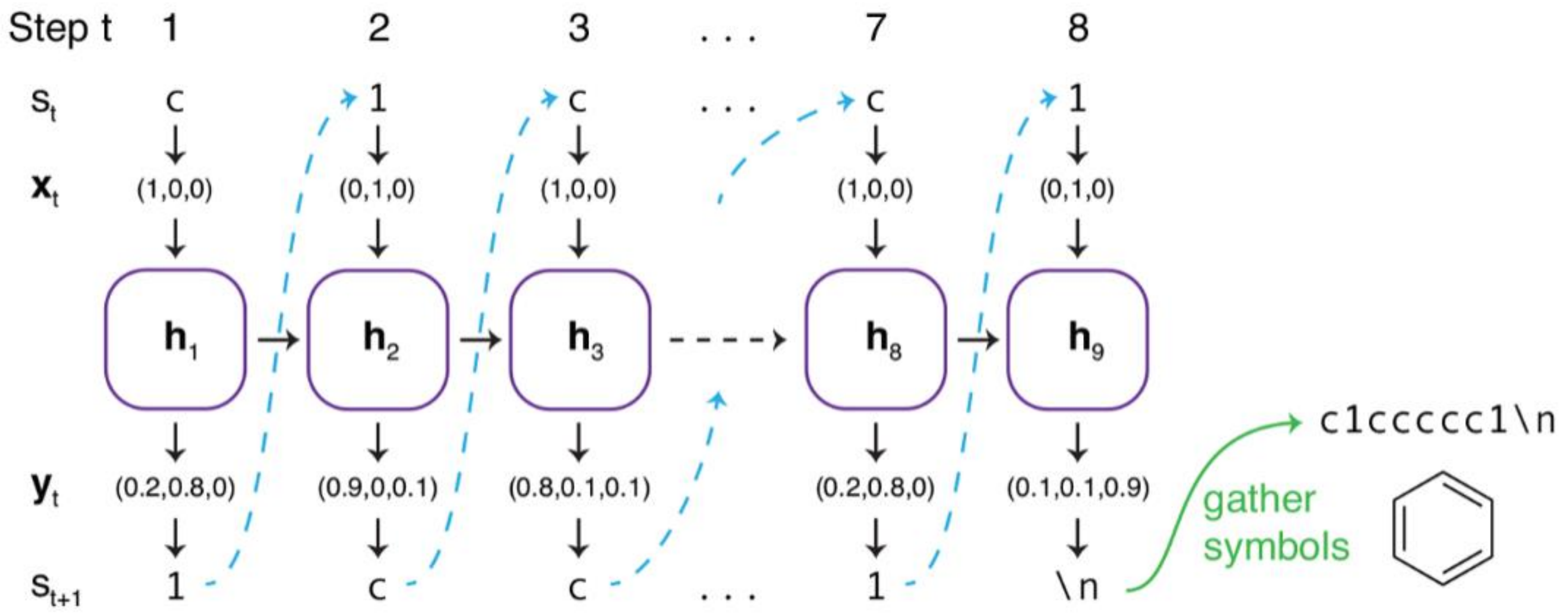
How does ANN memorize things it has seen during training?

# RNN in Chemistry

# Case study

## Generating Focused Molecule Libraries for Drug Discovery with Recurrent Neural Networks

Marwin H. S. Segler,*[†] Thierry Kogej,[‡] Christian Tyrchan,[§] and Mark P. Waller*[||]



$$\mathrm{RNN}(\mathbf{h}_0, \mathbf{x}_{1:n}) = \mathbf{h}_{1:n}, \mathbf{y}_{1:n} \tag{2}$$

$$\mathbf{h}_i = R(\mathbf{h}_{i-1}, \mathbf{x}_i) \tag{3}$$

$$\mathbf{y}_i = O(\mathbf{h}_i) \tag{4}$$

The state vector $\mathbf{h}_i$ stores a representation of the information about all symbols seen in the sequence so far. As an alternative to the recursive definition, the recurrent network can also be *unrolled* for finite sequences (see Figure 2). An unrolled RNN can be seen as a very deep neural network, in which the parameters $\theta$ are shared among the layers, and the hidden state $\mathbf{h}_t$ is passed as an additional input to the next layer. Training the unrolled RNN to fit the parameters $\theta$ can then simply be done by using backpropagation to compute the gradients with respect to the loss function, which is categorical cross-entropy in this work.[55]

As the specific RNN function, in this work, we use the long short-term memory (LSTM), which was introduced by Hochreiter and Schmidhuber.[56] It has been used successfully in many natural language processing tasks,[47] for example in Google's neural machine translation system.[57] For excellent in-depth discussions of the LSTM, we refer to the articles by Goldberg,[55] Graves,[58] Olah,[59] and Greff et al.[60]

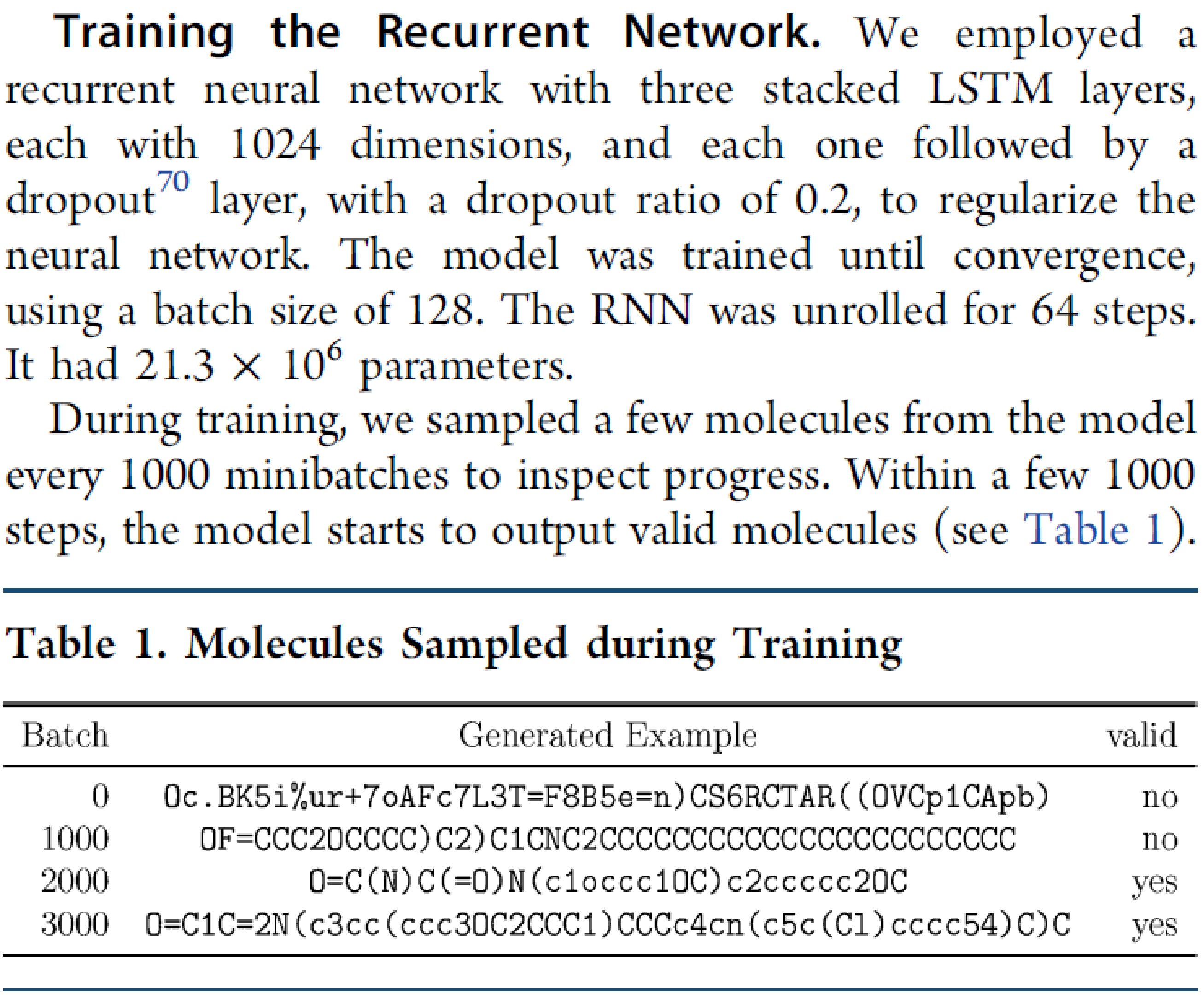
To encode the SMILES symbols as input vectors $\mathbf{x}_t$, we employ the "one-hot" representation.[58] This means that if there are $K$ symbols, and $k$ is the symbol to be input at step $t$, then we can construct an input vector $\mathbf{x}_t$ with length $K$, whose entries are all zero except the $k$th entry, which is one. If we assume a very restricted set of symbols {c, 1, \n}, input c would correspond to $\mathbf{x}_t = (1, 0, 0)$, 1 to $\mathbf{x}_t = (0, 1, 0)$, and \n to $\mathbf{x}_t = (0, 0, 1)$.

The probability distribution $P_\theta(s_{t+1}|s_t, \ldots, s_1)$ of the next symbol given the already seen sequence is thus a multinomial distribution, which is estimated using the output vector $\mathbf{y}_t$ of the recurrent neural network at time step $t$ by

$$P_\theta(s_{t+1} = k|s_t, \ldots, s_1) = \frac{\exp(y_t^k)}{\sum_{k'=1}^{K} \exp(y_t^{k'})} \tag{5}$$

where $y_t^k$ corresponds to the $k$th element of vector $\mathbf{y}_t$.[58] Sampling from this distribution would then allow generating novel molecules: After sampling a SMILES symbol $s_{t+1}$ for the next time step $t + 1$, we can construct a new input vector $\mathbf{x}_{t+1}$, which is fed into the model, and via $\mathbf{y}_{t+1}$ and eq 5 yields $P_\theta(s_{t+2}|s_{t+1}, \ldots, s_1)$. Sampling from the latter generates $s_{t+2}$, which serves again also as the model's input for the next step (see Figure 3). This symbol-by-symbol sampling procedure is repeated until the desired number of characters have been generated.[58]

# Case study



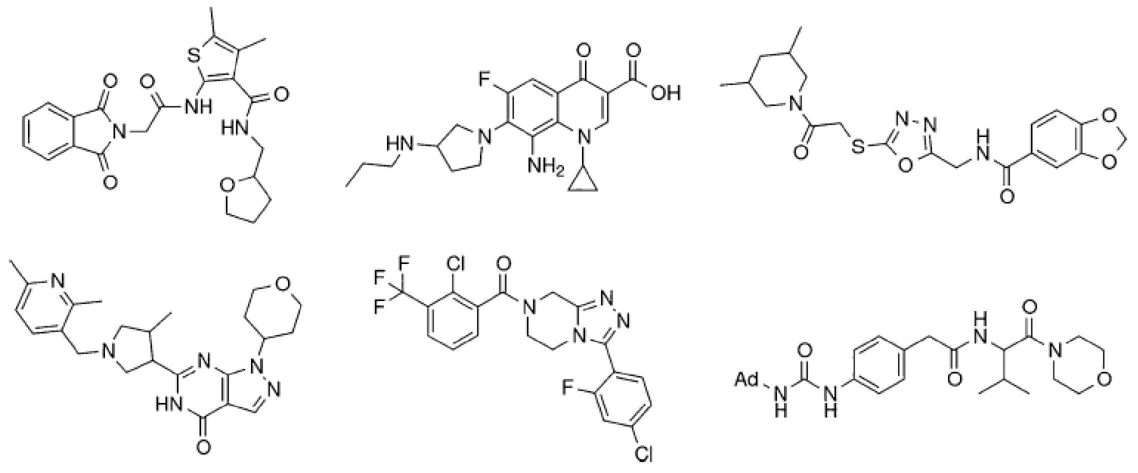
**Training the Recurrent Network.** We employed a recurrent neural network with three stacked LSTM layers, each with 1024 dimensions, and each one followed by a dropout[70] layer, with a dropout ratio of 0.2, to regularize the neural network. The model was trained until convergence, using a batch size of 128. The RNN was unrolled for 64 steps. It had $21.3 \times 10^6$ parameters.

During training, we sampled a few molecules from the model every 1000 minibatches to inspect progress. Within a few 1000 steps, the model starts to output valid molecules (see Table 1).

**Table 1. Molecules Sampled during Training**

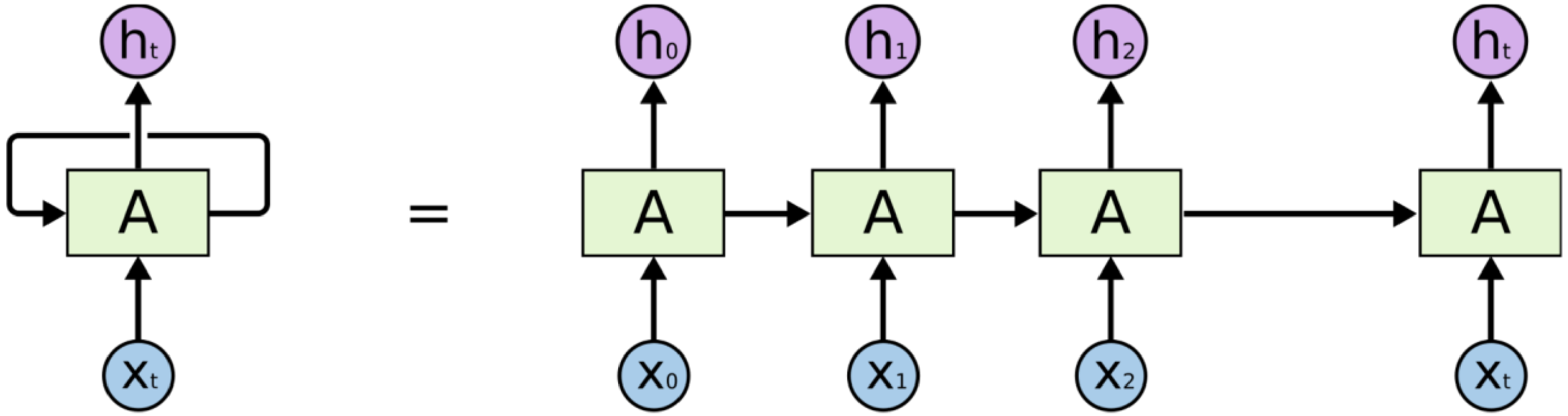
| Batch | Generated Example | valid |
|---|---|---|
| 0 | `Oc.BK5i%ur+7oAFc7L3T=F8B5e=n)CS6RCTAR((OVCp1CApb)` | no |
| 1000 | `OF=CCC2OCCCC)C2)C1CNC2CCCCCCCCCCCCCCCCCCCCCCC` | no |
| 2000 | `O=C(N)C(=O)N(c1occc1OC)c2ccccc2OC` | yes |
| 3000 | `O=C1C=2N(c3cc(ccc3OC2CCC1)CCCc4cn(c5c(Cl)cccc54)C)C` | yes |

The chemical language model was trained on a SMILES file containing 1.4 million molecules from the ChEMBL database, which contains molecules and measured biological activity data. The SMILES strings of the molecules were canonicalized (which means finding a unique representation that is the same for isomorphic molecular graphs)



ıdomly selected, generated molecules. Ad = Adamantyl.

# Summary



- ✓ **Good for sequential problem**
- ✓ **Vanilla RNN:** $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{x}_t; \boldsymbol{\theta})$ **Long-term dependency problem**
- ✓ **LSTM: resolving the long-term dependency problem but very slow**
- ✓ **GRU: more efficient than LSTM**

- $x_t$ : input vector
- $h_t$ : hidden state
- $C_t$ : cell state
- $f_t, i_t, o_t$ : forget, input, output gate

## New terms

- Sequential data structure
- Auto-regressive process
- Recurrent neural network (RNN)
- Long short term memory (LSTM)
- Gated recurrent unit (GRU)
- Backpropagation through time (BPTT)