자연어 처리 DAY 2

Jaegul Choo
Associate Professor, Graduate School of AI, KAIST

boostcamp AI Tech

DAVIAN
Data and Visual Analytics Lab

1.

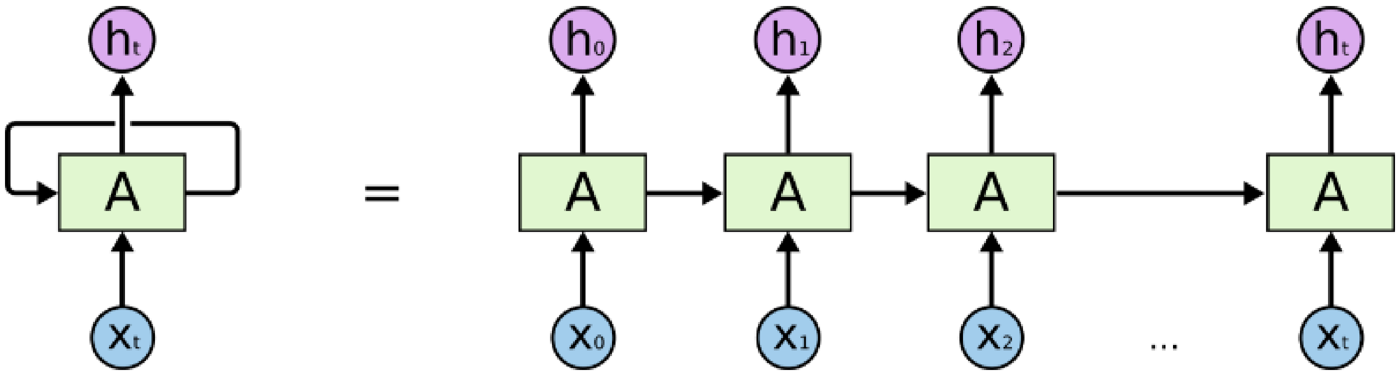
Basics of Recurrent Neural Networks (RNNs)

Basic problem settings

Model architecture and how it works

Types of RNNs

- Basic structure

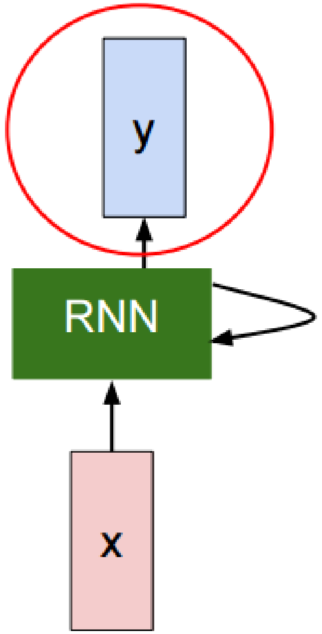


An unrolled recurrent neural network.

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

Recurrent Neural Network

- **Inputs and outputs of RNNs (rolled version)**
  - We usually want to predict a vector at some time steps

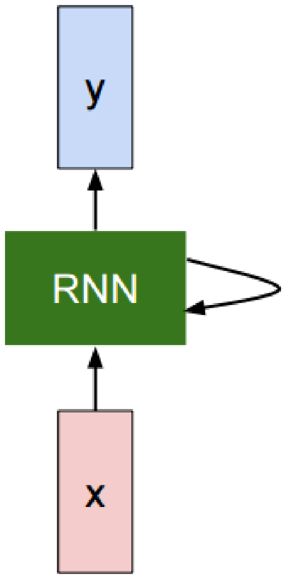


http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Recurrent Neural Network

- **How to calculate the hidden state of RNNs**
  - We can process a sequence of vectors by applying a recurrence formula at every time step
  - h_{t-1}: old hidden-state vector
  - x_t: input vector at some time step
  - h_t: new hidden-state vector
  - f_W: RNN function with parameters W
  - y_t: output vector at time step t

$$h_t = f_W(h_{t-1}, x_t)$$

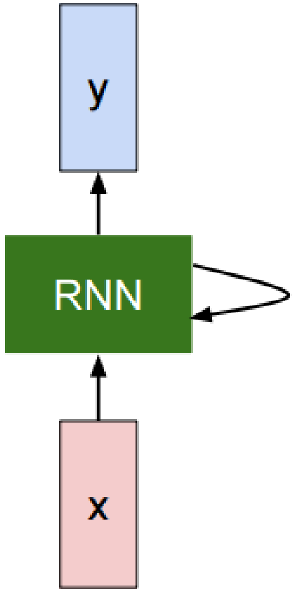


http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Recurrent Neural Network

- How to calculate the hidden state of RNNs
  - Notice: The same function and the same set of parameters are used at every time step

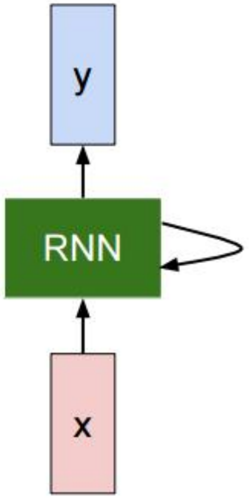
$$h_t = f_W(h_{t-1}, x_t)$$



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Recurrent Neural Network

- **How to calculate the hidden state of RNNs**
  - The state consists of a single "hidden" vector h



$$h_t = f_W(h_{t-1}, x_t)$$

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

$$y_t = W_{hy}h_t$$

End of Document
Thank You.

2.
Types of RNNs

one-to-one

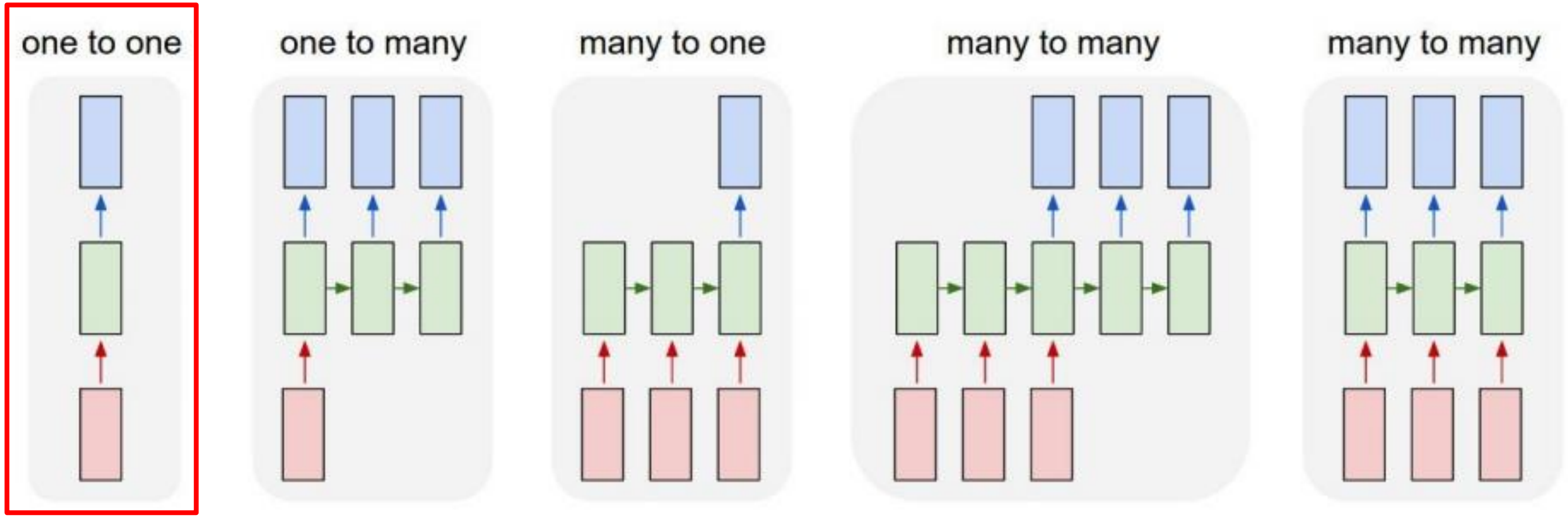
one-to-many

many-to-one

many-to-many

Types of RNNs

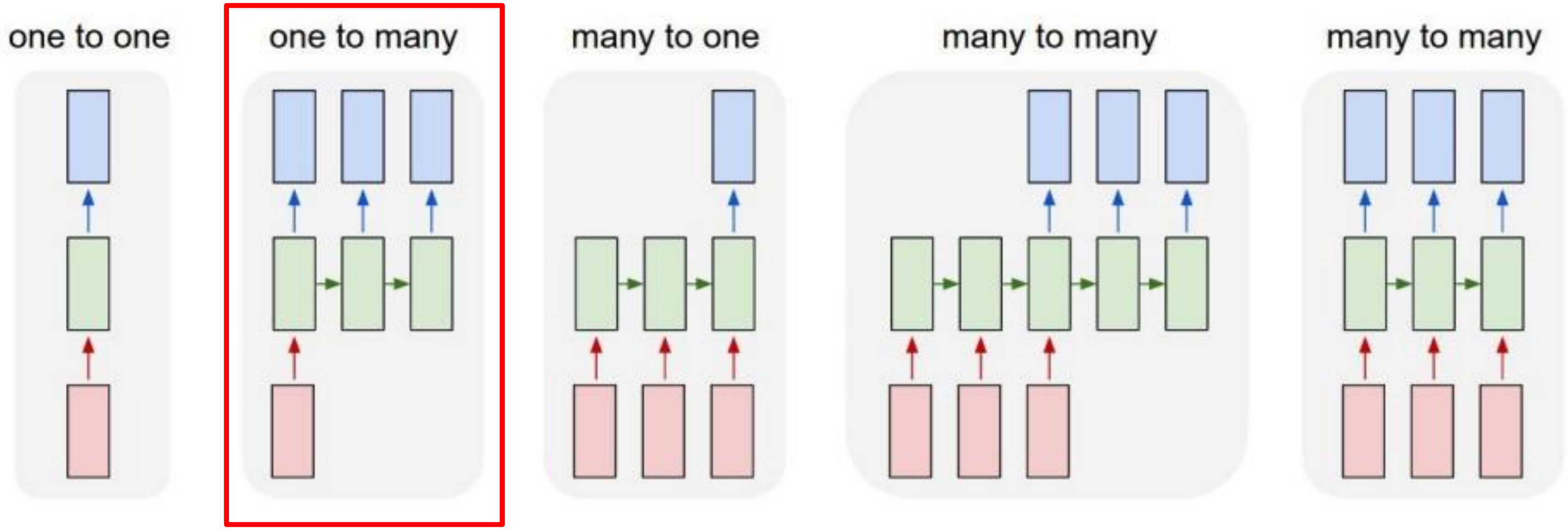
- **One-to-one**
  - Standard Neural Networks



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Types of RNNs

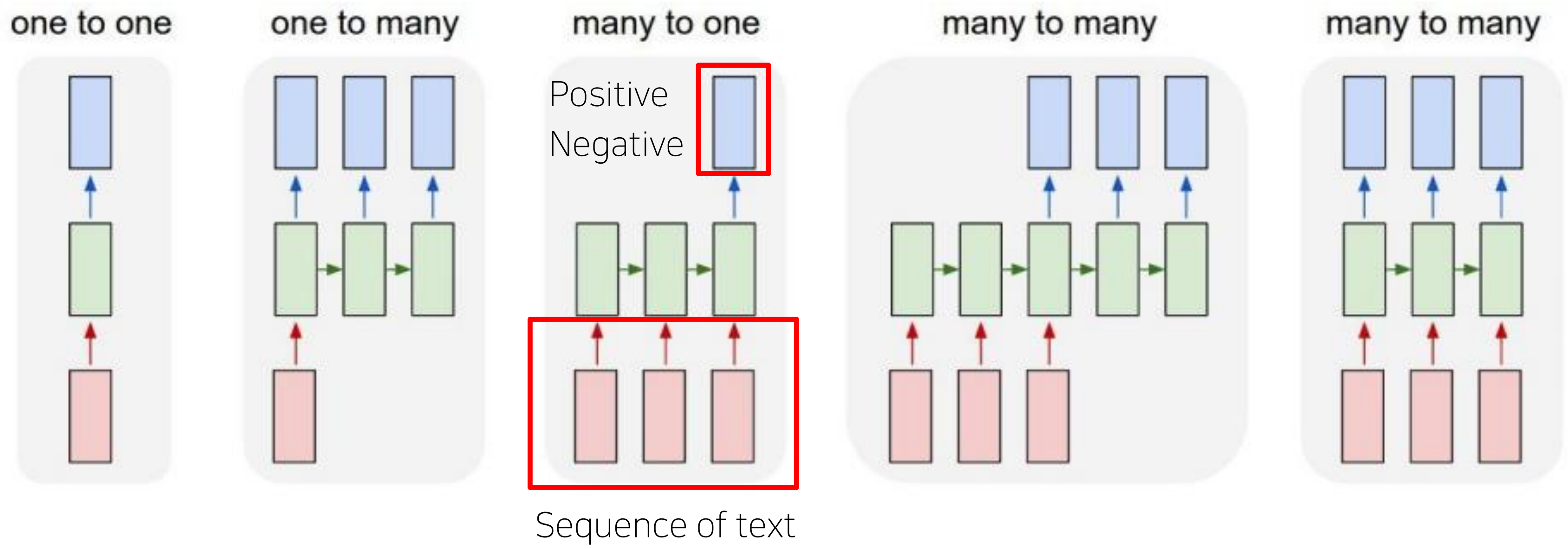
- **One-to-many**
  - Image Captioning



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Types of RNNs

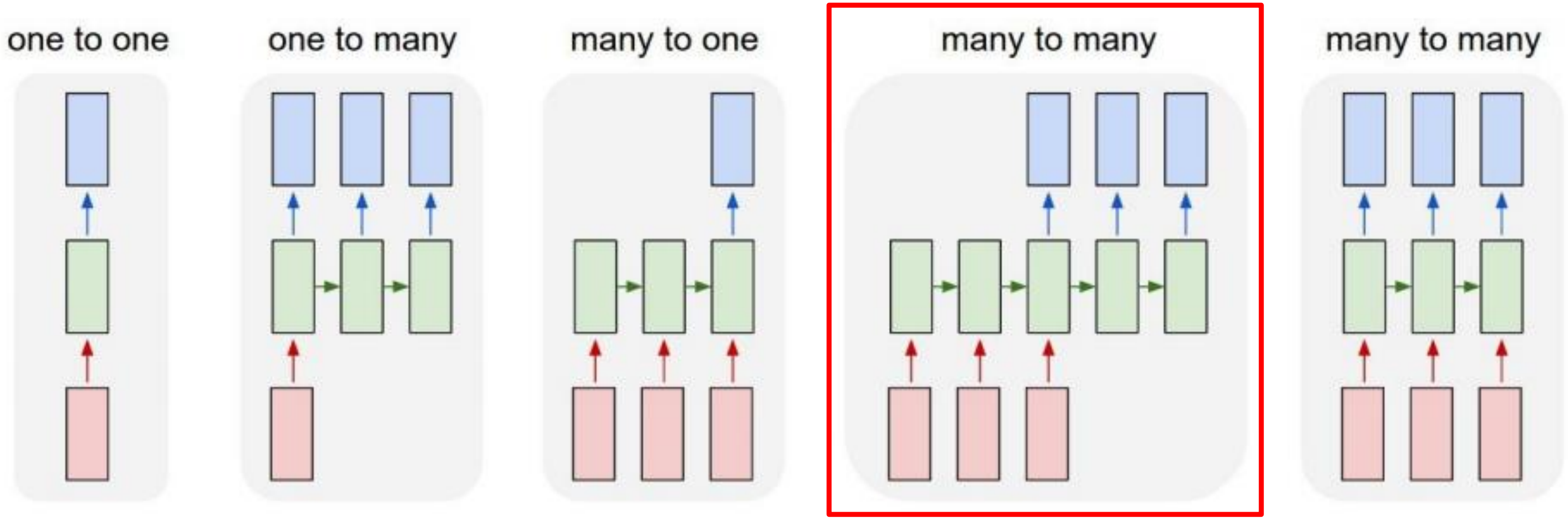
- **Many-to-one**
  - Sentiment Classification



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Types of RNNs

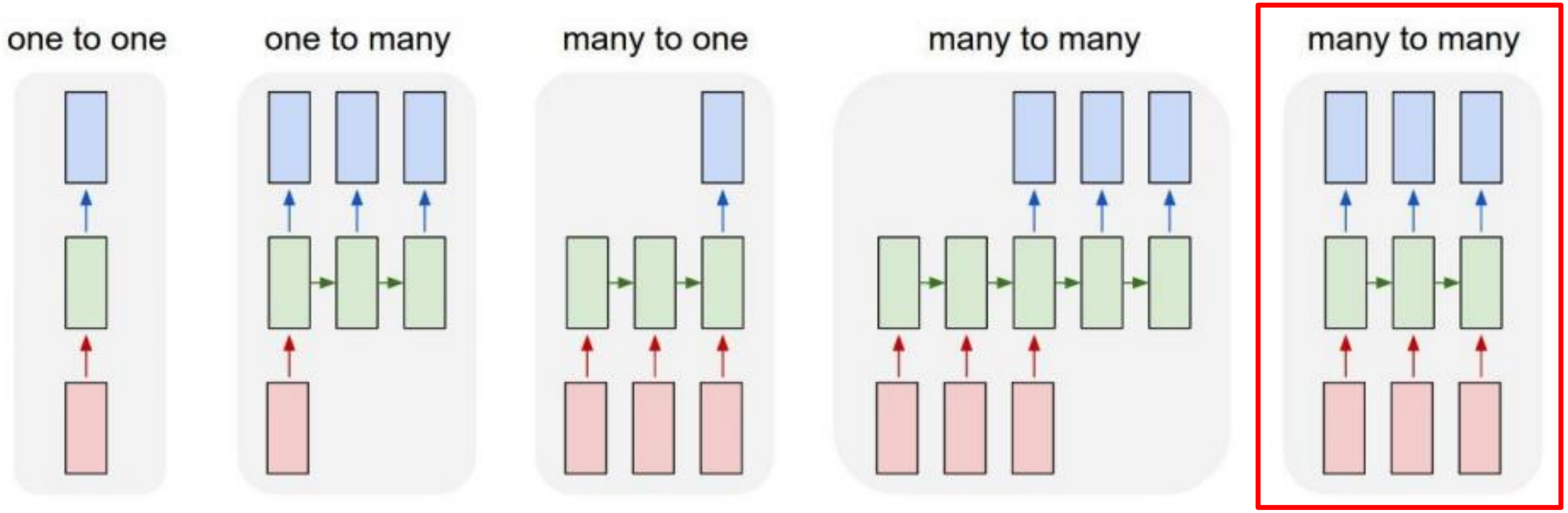
- **Sequence-to-sequence**
  - Machine Translation



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Types of RNNs

- **Sequence-to-sequence**
  - Video classification on frame level



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

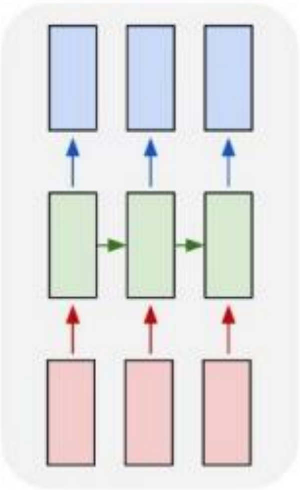
End of Document
Thank You.

3.

Character-level Language Model

Character-level Language Model

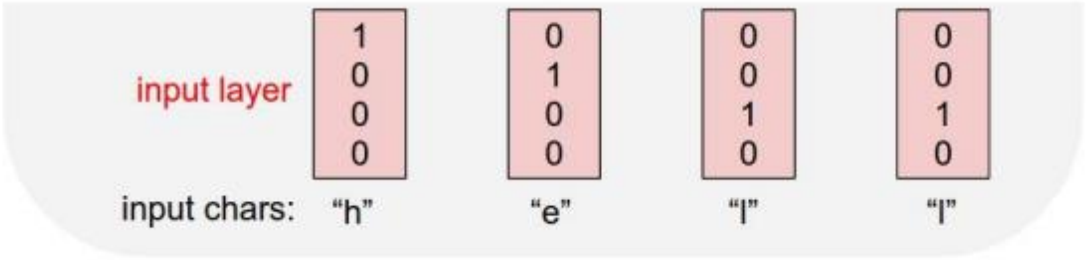
- **Example of training sequence "hello"**
  - Vocabulary: [h, e, l, o]
  - Example training sequence: "hello"



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

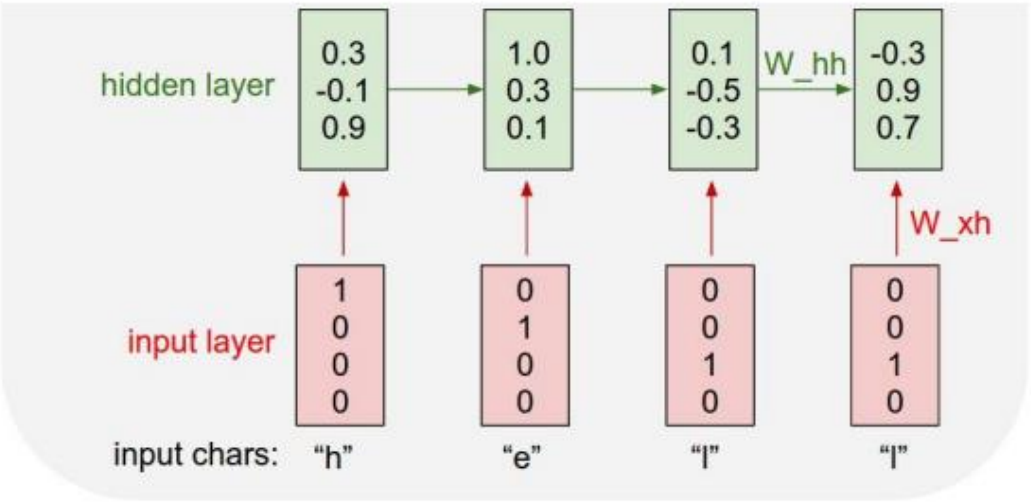
- **Example of training sequence "hello"**
  - Vocabulary: [h, e, l, o]
  - Example training sequence: "hello"



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

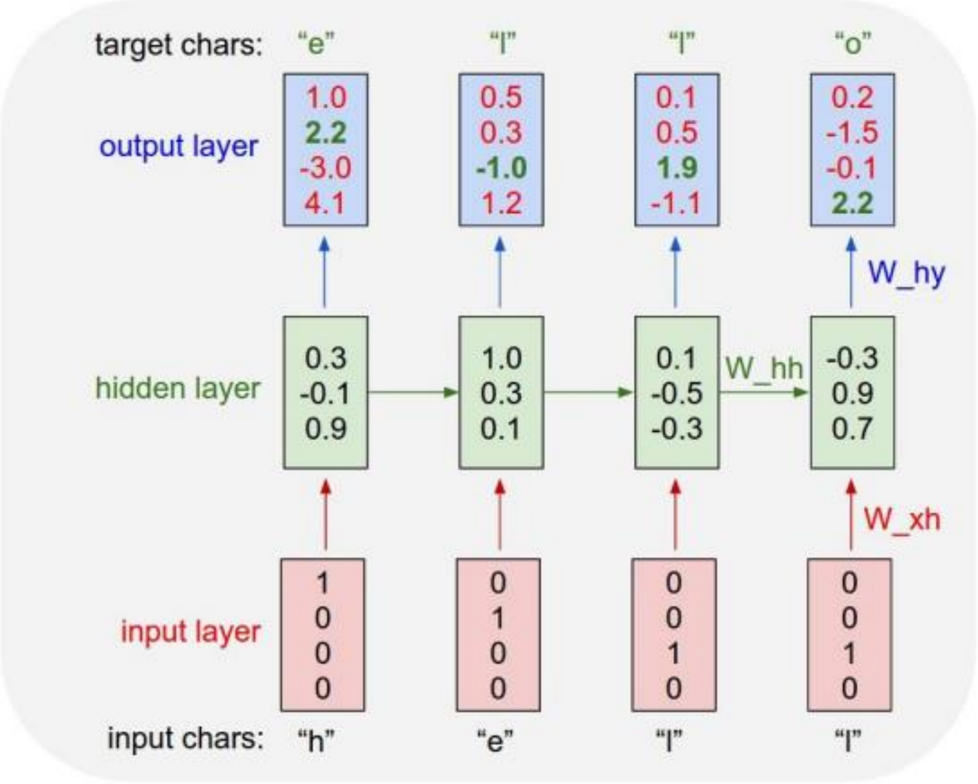
- Example of training sequence "hello"
  - $h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t + b)$



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

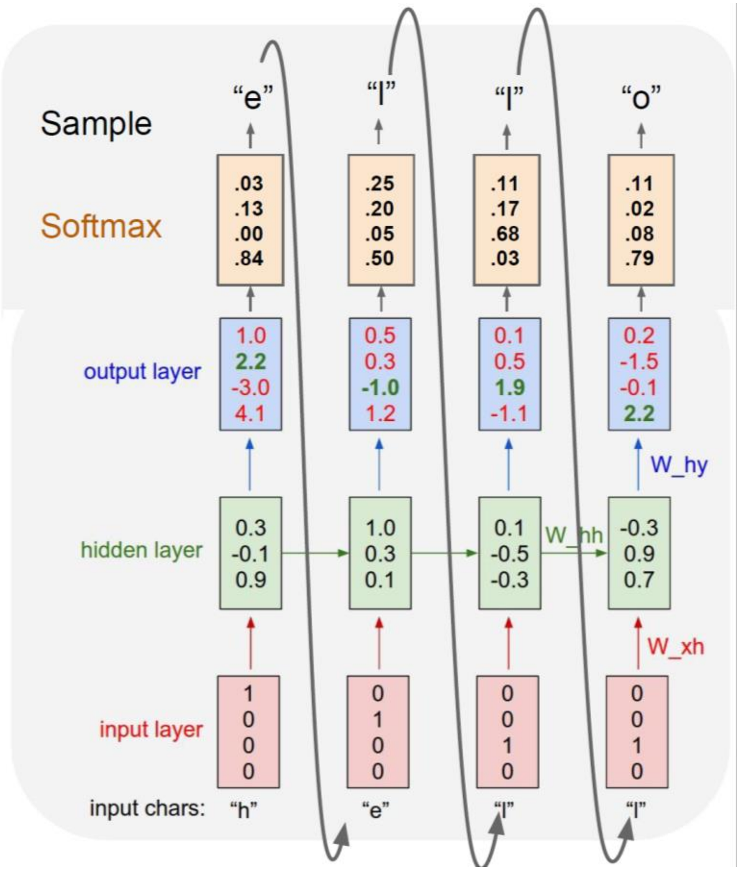
- Example of training sequence "hello"
  - $\text{Logit} = W_{hy}h_t + b$



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

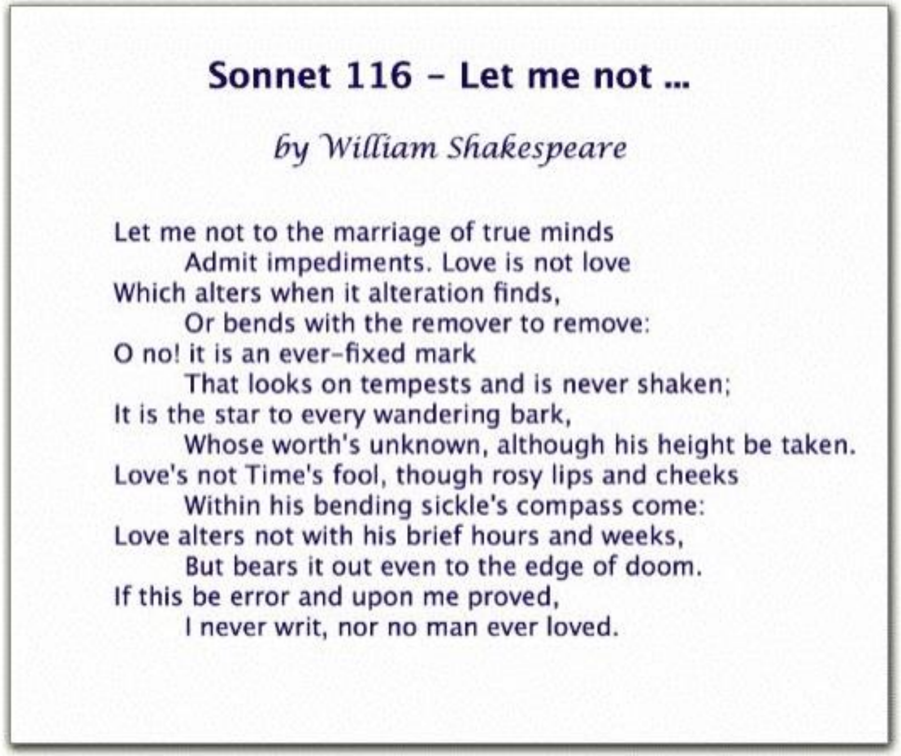
- **Example of training sequence "hello"**
  - At test-time, sample characters one at a time, feed back to model



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

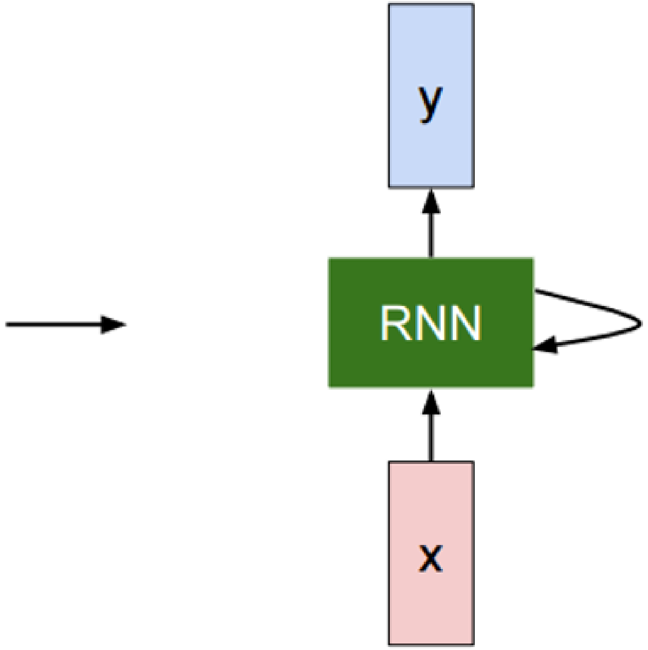
- Training a RNN on Shakespeare's plays



Sonnet 116 – Let me not ...

by William Shakespeare

Let me not to the marriage of true minds
Admit impediments. Love is not love
Which alters when it alteration finds,
Or bends with the remover to remove:
O no! it is an ever-fixed mark
That looks on tempests and is never shaken;
It is the star to every wandering bark,
Whose worth's unknown, although his height be taken.
Love's not Time's fool, though rosy lips and cheeks
Within his bending sickle's compass come:
Love alters not with his brief hours and weeks,
But bears it out even to the edge of doom.
If this be error and upon me proved,
I never writ, nor no man ever loved.



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Character-level Language Model

- Training process of RNN

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

- Results of trained RNN

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

```
VIOLA:
Why, Salisbury must find his flesh and thought
That which I am not aps, not a man and in fire,
To show the reining of the raven and the wars
To grace my hand reproach within, and not a fair are hand,
That Caesar and my goodly father's world;
When I was heaven of presence and our fleets,
We spare with hours, but cut thy council I am great,
Murdered and by thy master's ready there
My power to give thee but so much as hell:
Some service in the noble bondman here,
Would show him to her wine.

KING LEAR:
O, if you were a feeble sight, the courtesy of your law,
Your sight and several breath, will wear the gods
With his heads, and my hands are wonder'd at the deeds,
So drop upon your lordship's head, and your opinion
Shall be against your honour.
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

- A paper written by RNN

Proof. Omitted. □

Lemma 0.1. *Let $\mathcal{C}$ be a set of the construction.*

Let $\mathcal{C}$ be a gerber covering. Let $\mathcal{F}$ be a quasi-coherent sheaves of $\mathcal{O}$-modules. We have to show that

$$\mathcal{O}_{\mathcal{O}_X} = \mathcal{O}_X(\mathcal{L})$$

.

Proof. This is an algebraic space with the composition of sheaves $\mathcal{F}$ on $X_{\acute{e}tale}$ we have

$$\mathcal{O}_X(\mathcal{F}) = \{morph_1 \times_{\mathcal{O}_X} (\mathcal{G}, \mathcal{F})\}$$

where $\mathcal{G}$ defines an isomorphism $\mathcal{F} \to \mathcal{F}$ of $\mathcal{O}$-modules. □

Lemma 0.2. *This is an integer $\mathcal{Z}$ is injective.*

Proof. See Spaces, Lemma ??. □

Lemma 0.3. *Let S be a scheme. Let X be a scheme and X is an affine open covering. Let $\mathcal{U} \subset \mathcal{X}$ be a canonical and locally of finite type. Let X be a scheme. Let X be a scheme which is equal to the formal complex.*

The following to the construction of the lemma follows.

Let X be a scheme. Let X be a scheme covering. Let

$$b : X \to Y' \to Y \to Y \to Y' \times_X Y \to X.$$

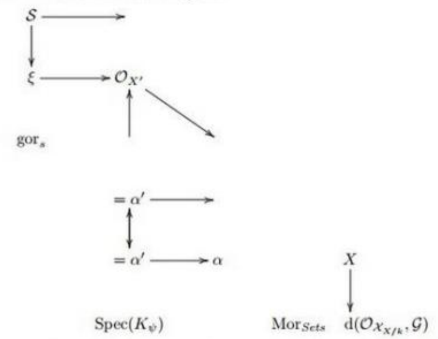
be a morphism of algebraic spaces over S and Y.

Proof. Let X be a nonzero scheme of X. Let X be an algebraic space. Let $\mathcal{F}$ be a quasi-coherent sheaf of $\mathcal{O}_X$-modules. The following are equivalent

(1) $\mathcal{F}$ is an algebraic space over S.
(2) If X is an affine open covering.

Consider a common structure on X and X the functor $\mathcal{O}_X(U)$ which is locally of finite type. □

This since $\mathcal{F} \in \mathcal{F}$ and $x \in \mathcal{G}$ the diagram



is a limit. Then $\mathcal{G}$ is a finite type and assume S is a flat and $\mathcal{F}$ and $\mathcal{G}$ is a finite type f_*. This is of finite type diagrams, and

- the composition of $\mathcal{G}$ is a regular sequence,
- $\mathcal{O}_{X'}$ is a sheaf of rings.

□

Proof. We have see that $X = \mathrm{Spec}(R)$ and $\mathcal{F}$ is a finite type representable by algebraic space. The property $\mathcal{F}$ is a finite morphism of algebraic stacks. Then the cohomology of X is an open neighbourhood of U. □

Proof. This is clear that $\mathcal{G}$ is a finite presentation, see Lemmas ??.
A *reduced above* we conclude that U is an open covering of $\mathcal{C}$. The functor $\mathcal{F}$ is a "field

$$\mathcal{O}_{X,x} \longrightarrow \mathcal{F}_{\bar{x}} \;\; \text{-1}(\mathcal{O}_{X_{\acute{e}tale}}) \longrightarrow \mathcal{O}_{X_\ell}^{-1}\mathcal{O}_{X_\lambda}(\mathcal{O}_{X_\eta}^{\overline{v}})$$

is an isomorphism of covering of $\mathcal{O}_{X_i}$. If $\mathcal{F}$ is the unique element of $\mathcal{F}$ such that X is an isomorphism.
The property $\mathcal{F}$ is a disjoint union of Proposition ?? and we can filtered set of presentations of a scheme $\mathcal{O}_X$-algebra with $\mathcal{F}$ are opens of finite type over S.
If $\mathcal{F}$ is a scheme theoretic image points. □

If $\mathcal{F}$ is a finite direct sum $\mathcal{O}_{X_\lambda}$ is a closed immersion, see Lemma ??. This is a sequence of $\mathcal{F}$ is a similar morphism.

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

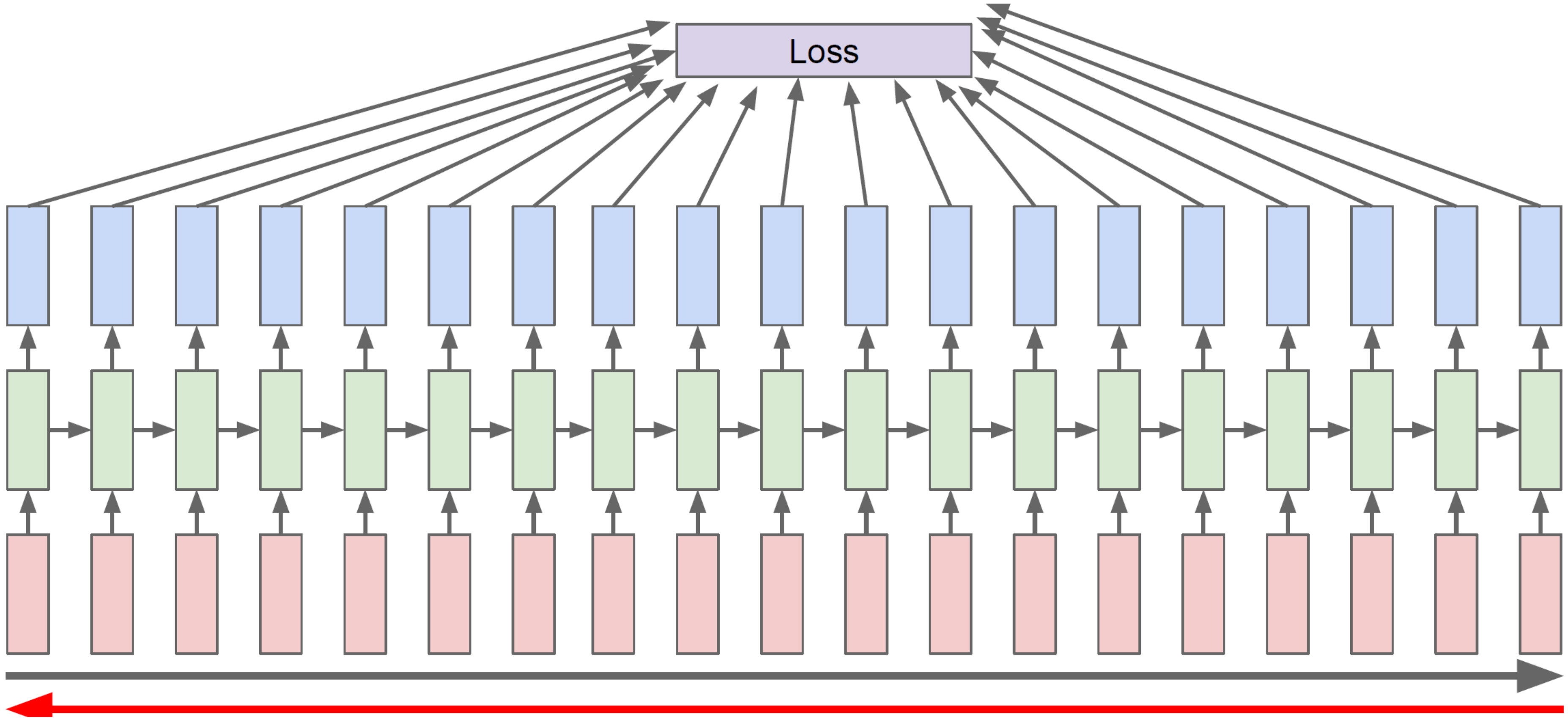
- C code generated by RNN

```
static void do_command(struct seq_file *m, void *v)
{
  int column = 32 << (cmd[2] & 0x80);
  if (state)
    cmd = (int)(int_state ^ (in_8(&ch->ch_flags) & Cmd) ? 2 : 1);
  else
    seq = 1;
  for (i = 0; i < 16; i++) {
    if (k & (1 << 1))
      pipe = (in_use & UMXTHREAD_UNCCA) +
        ((count & 0x00000000fffffff8) & 0x000000f) << 8;
    if (count == 0)
      sub(pid, ppc_md.kexec_handle, 0x20000000);
    pipe_set_bytes(i, 0);
  }
  /* Free our user pages pointer to place camera if all dash */
  subsystem_info = &of_changes[PAGE_SIZE];
  rek_controls(offset, idx, &soffset);
  /* Now we want to deliberately put it to device */
  control_check_polarity(&context, val, 0);
  for (i = 0; i < COUNTER; i++)
    seq_puts(s, "policy ");
}
```

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Backpropagation through time (BPTT)

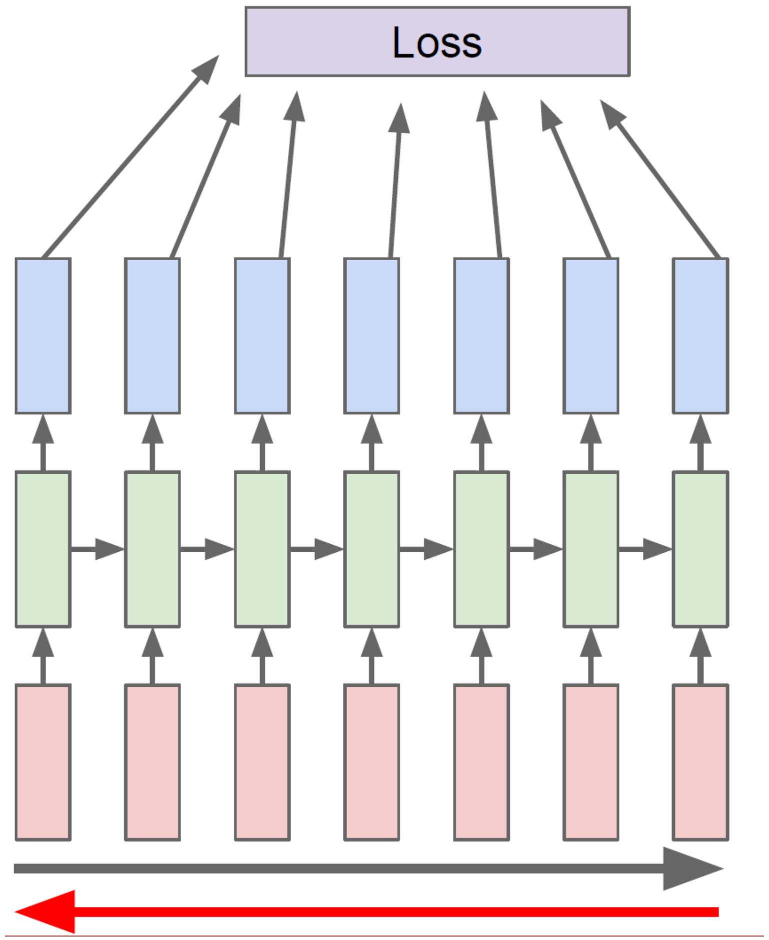
- Forward through entire sequence to compute loss, then backward through entire sequence to comput gradient



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Backpropagation through time (BPTT)

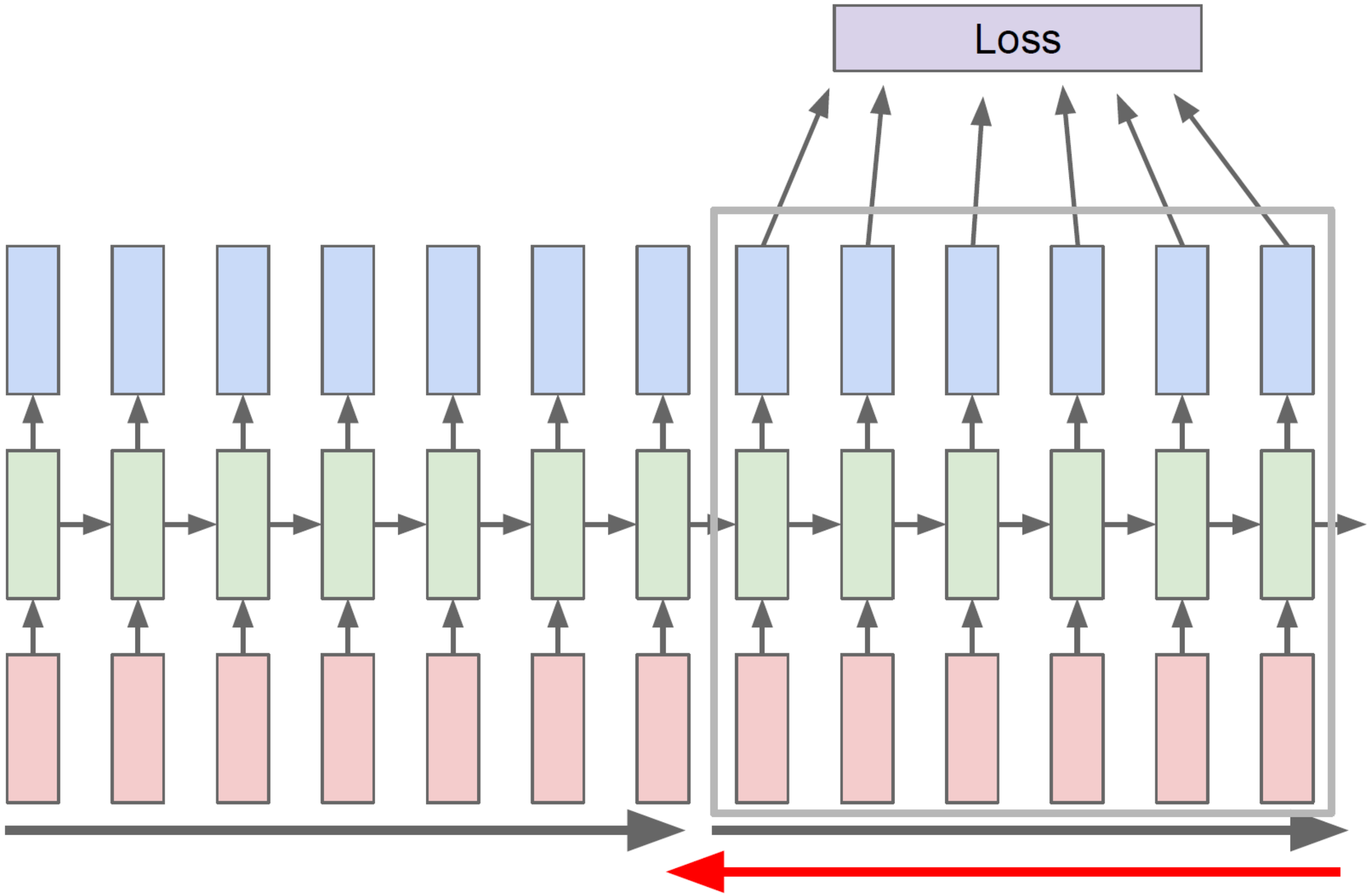
- Run forward and backward through chunks of the sequence instead of whole sequence



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

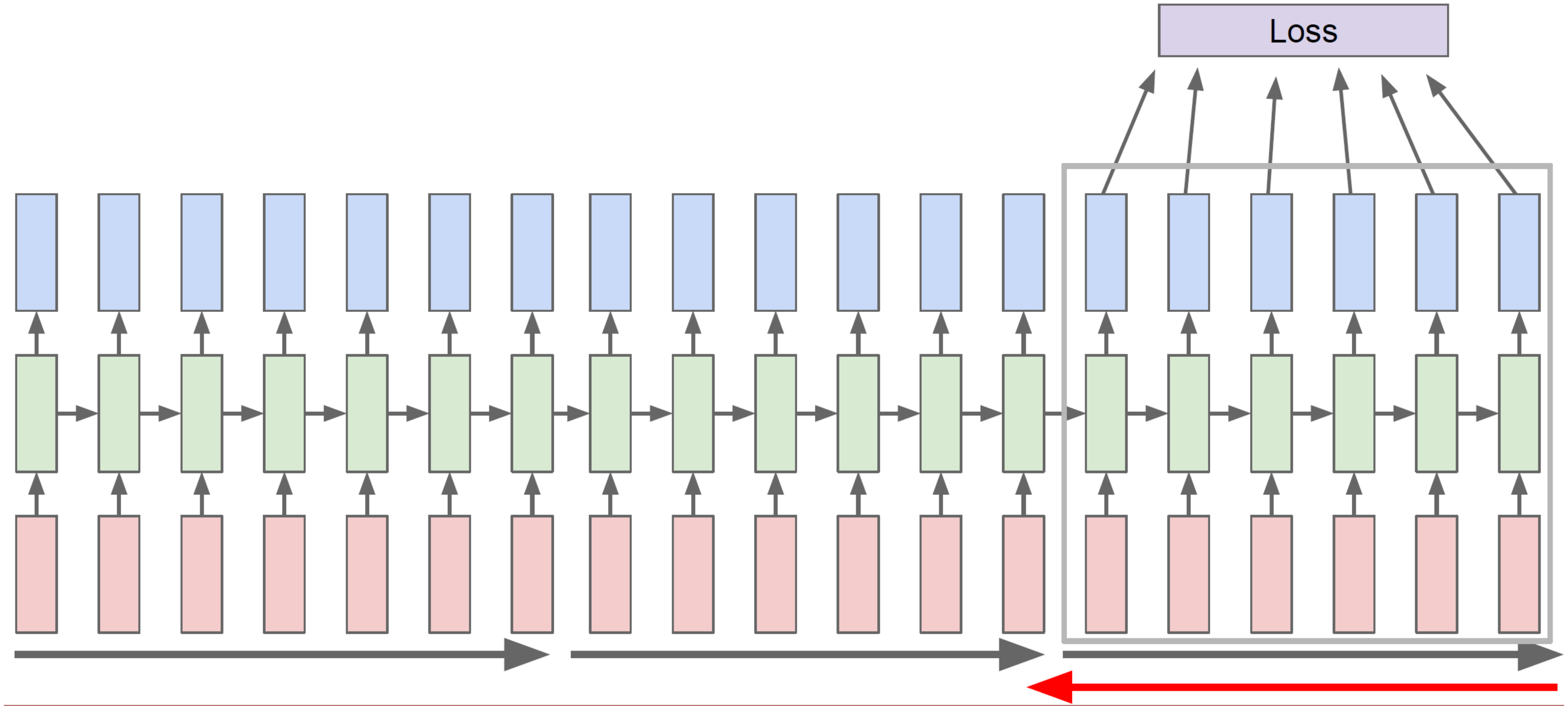
Backpropagation through time (BPTT)

- Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps



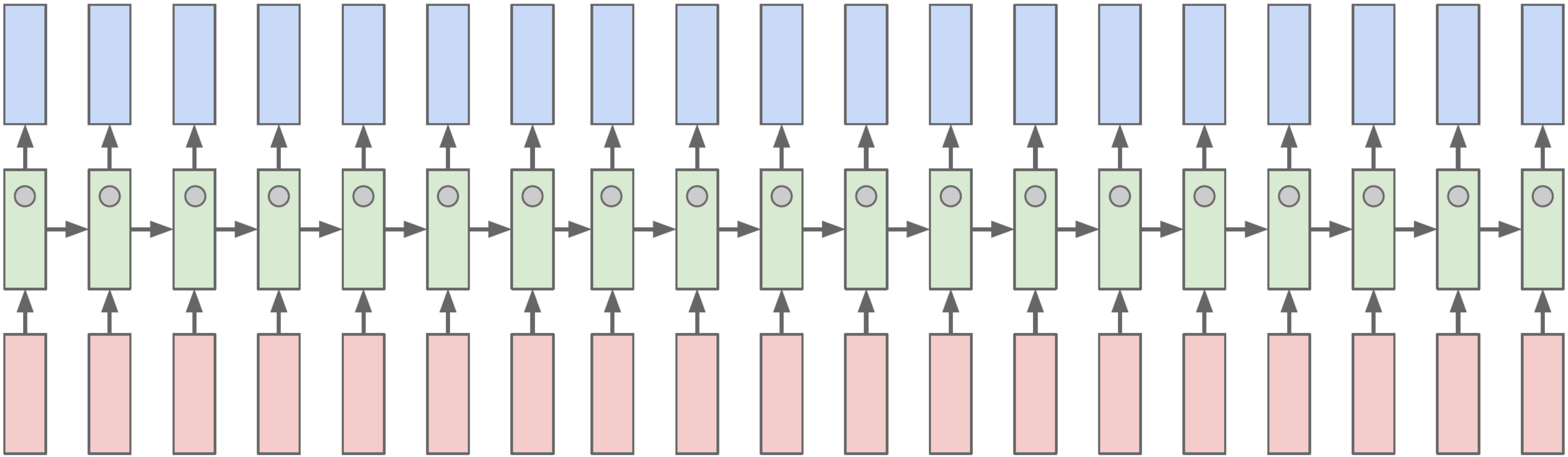
http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Backpropagation through time (BPTT)



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

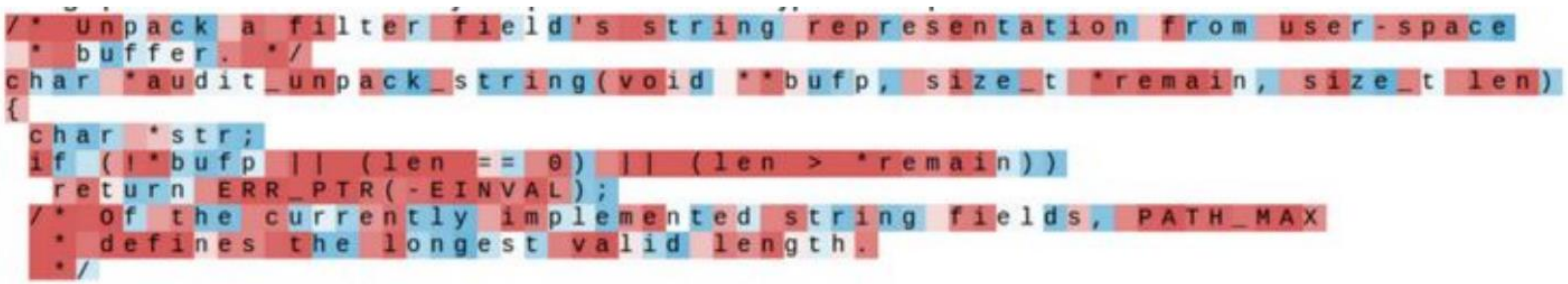
Searching for Interpretable Cells



http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

Character-level Language Model

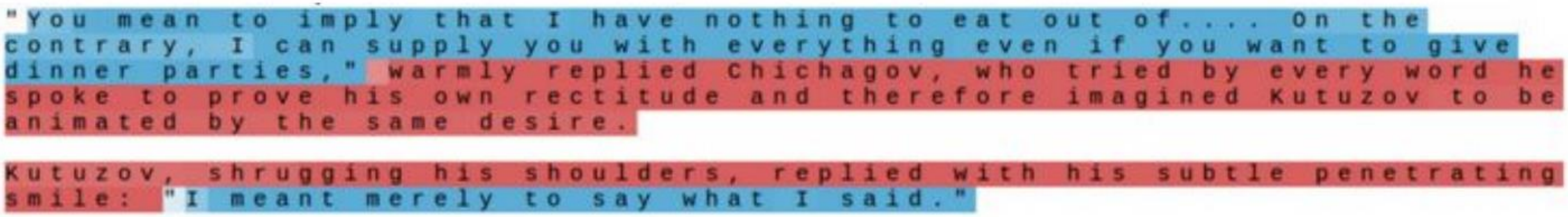
- How RNN works



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

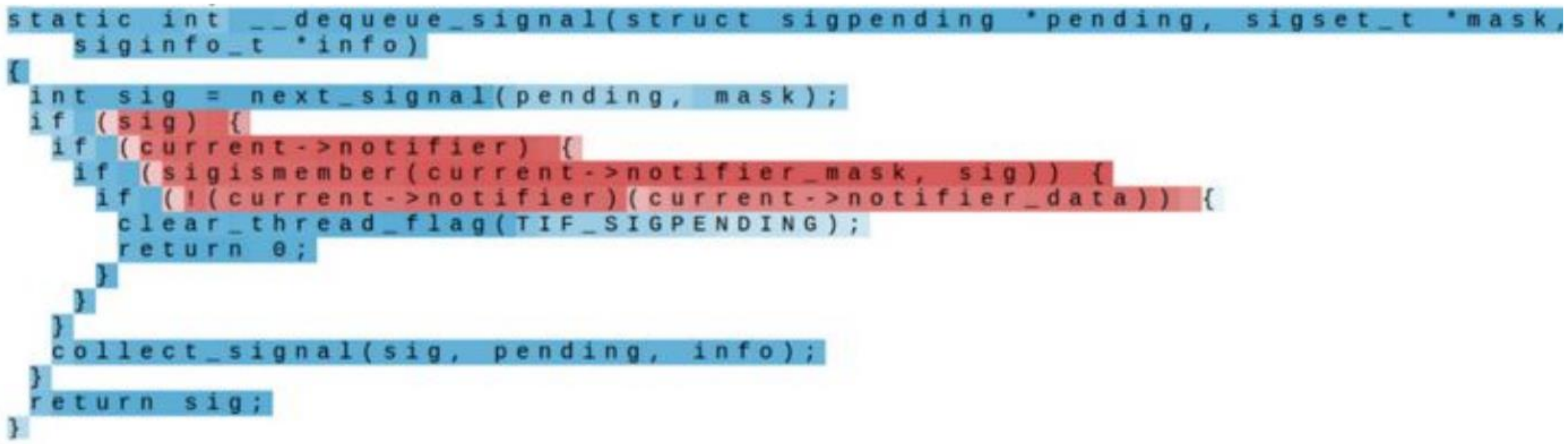
- **How RNN works**
  - Quote detection cell



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Character-level Language Model

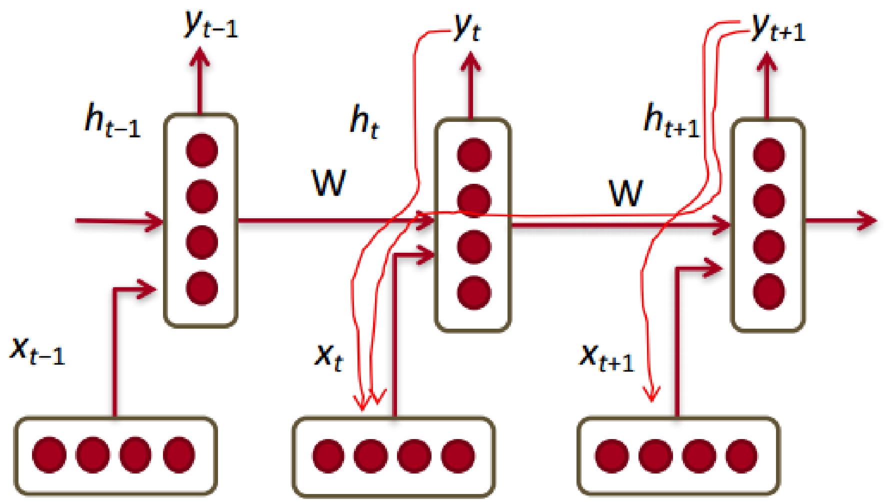
- **How RNN works**
  - If statement cell



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Vanishing/Exploding Gradient Problem in RNN

- RNN is excellent, but…
  - Multiplying the same matrix at each time step during backpropagation causes gradient vanishing or exploding



Vanishing/Exploding Gradient Problem in RNN

- Toy Example
  - $h_t = \tanh(w_{xh}x_t + w_{hh}h_{t-1} + b), t = 1,2,3$
  - For $w_{hh} = 3, w_{xh} = 2, b = 1$

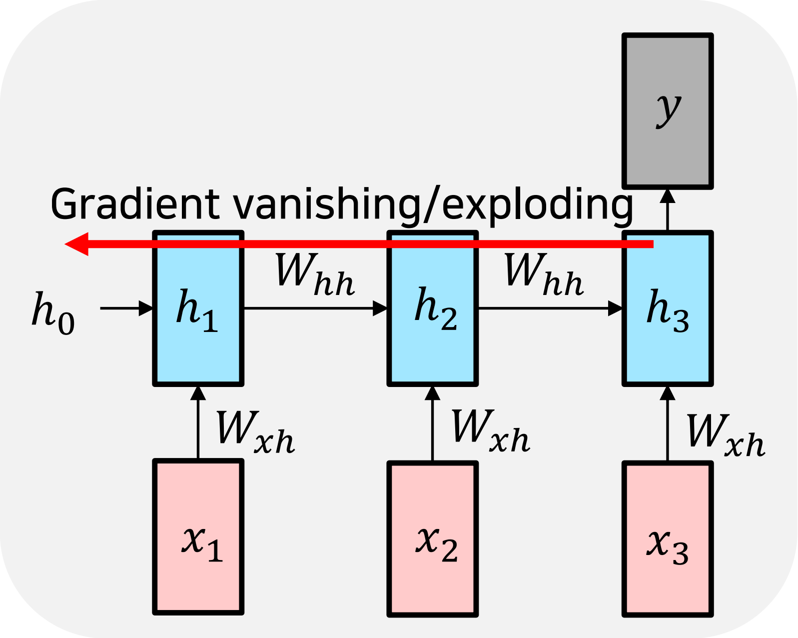
$$h_3 = \tanh(2x_3 + 3h_2 + 1)$$

$$h_2 = \tanh(2x_2 + 3h_1 + 1))$$

$$h_1 = \tanh(2x_1 + 3h_0 + 1))$$

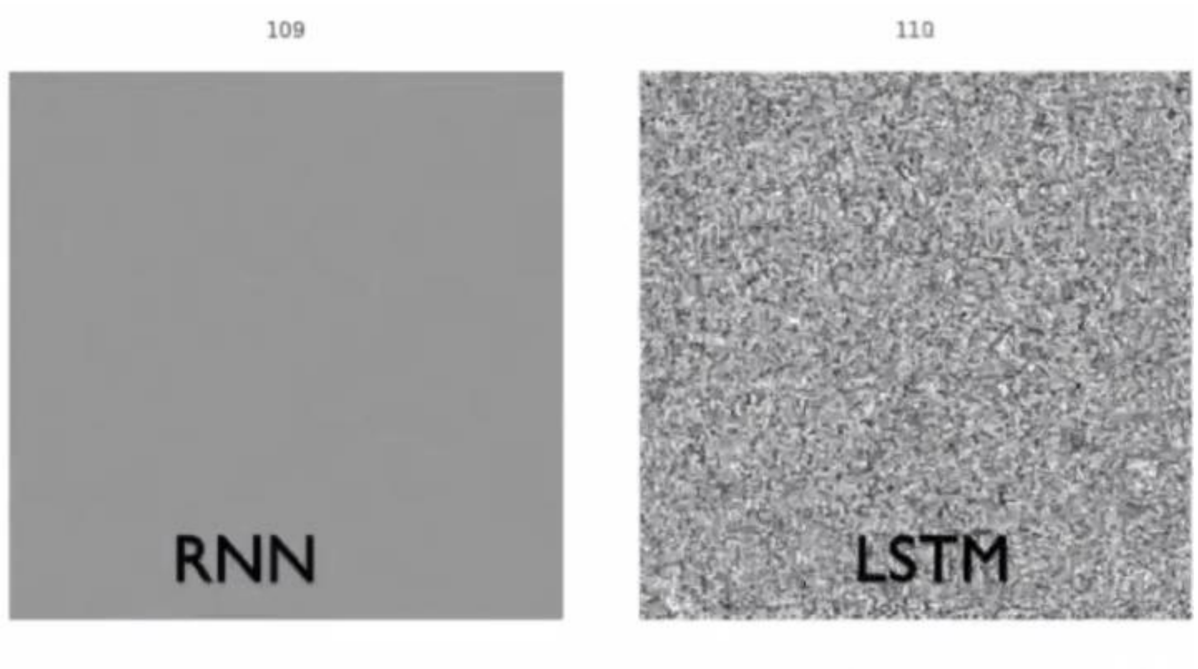
$$\dots$$

$$h_3 = \tanh(2x_3 + 3\tanh(2x_2 + 3h_1 + 1) + 1)$$



Vanishing Gradient Problem in RNN

- The reason why the vanishing gradient problem is important



https://imgur.com/gallery/vaNahKE

End of Document
Thank You.

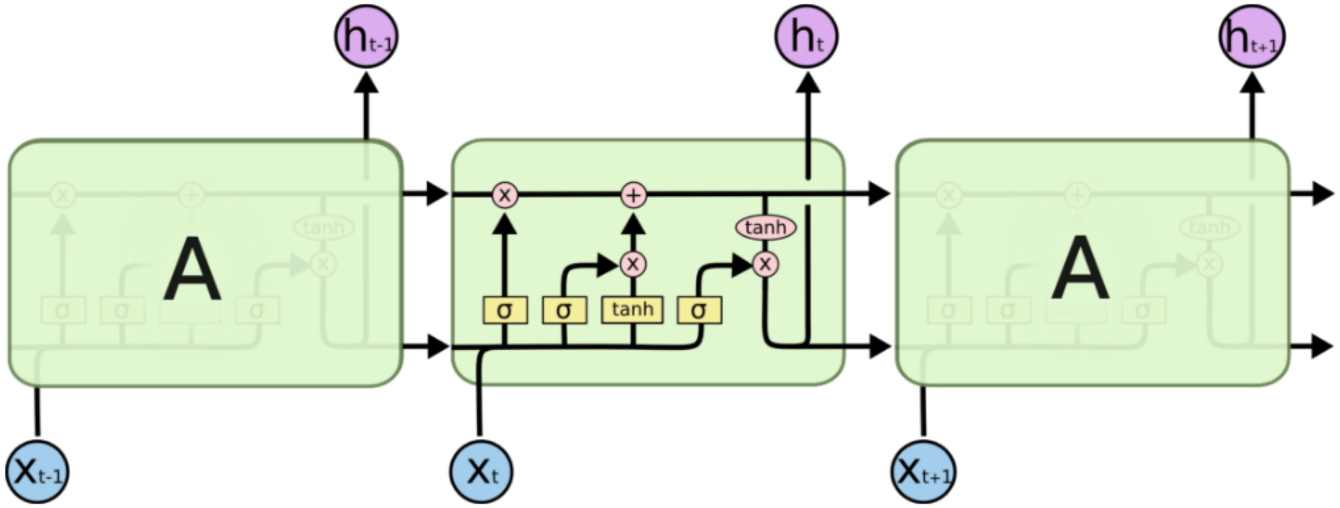
4.

Long Short-Term Memory (LSTM)

Gated Recurrent Unit (GRU)

Long Short-Term Memory (LSTM)

- Core Idea: pass cell state information straightly without any transformation
  - Solving long-term dependency problem

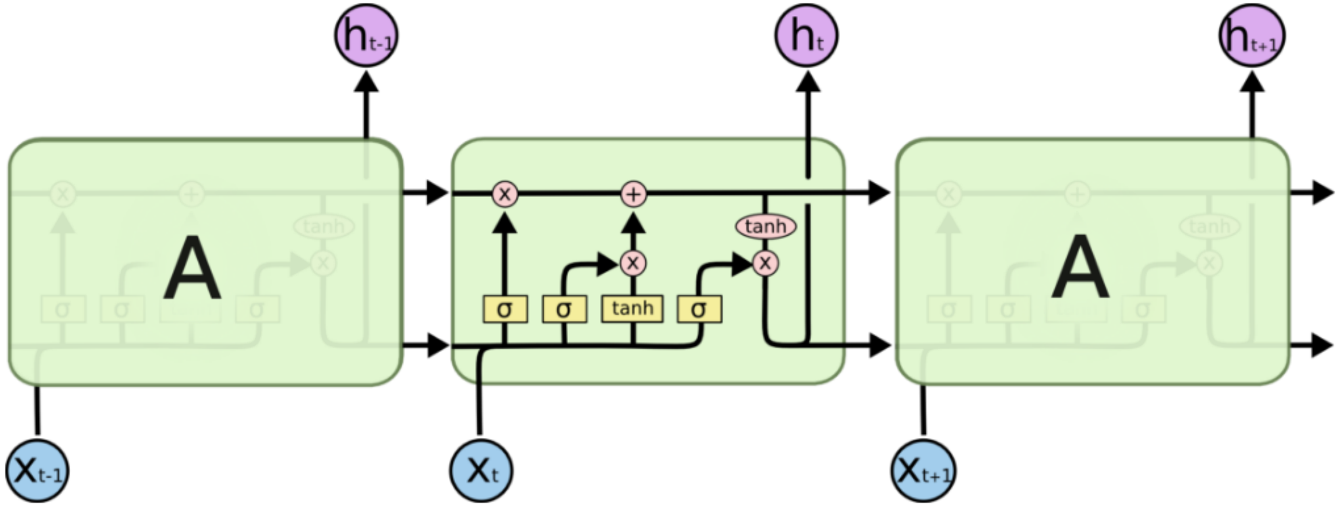


The repeating module in an LSTM contains four interacting layers.

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

Long Short-Term Memory (LSTM)

- What is LSTM (Long Short-Term Memory)?

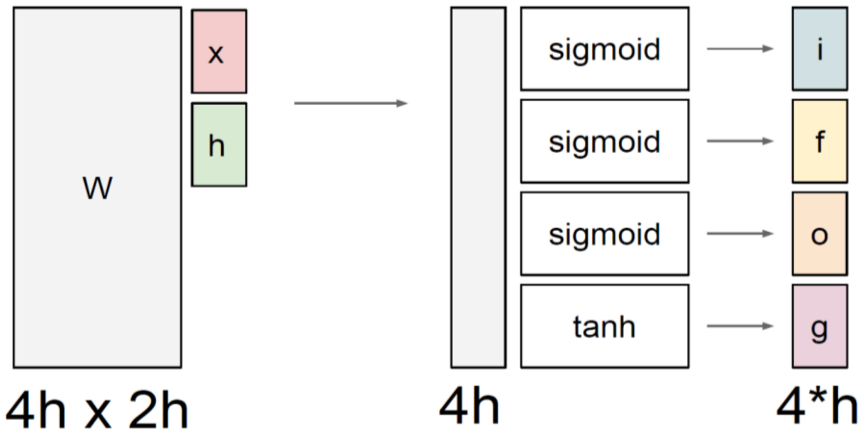


The repeating module in an LSTM contains four interacting layers.

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

Long Short-Term Memory (LSTM)

- **Long short-term memory**
  - i: Input gate, Whether to write to cell
  - f: Forget gate, Whether to erase cell
  - o: Output gate, How much to reveal cell
  - g: Gate gate, How much to write to cell



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

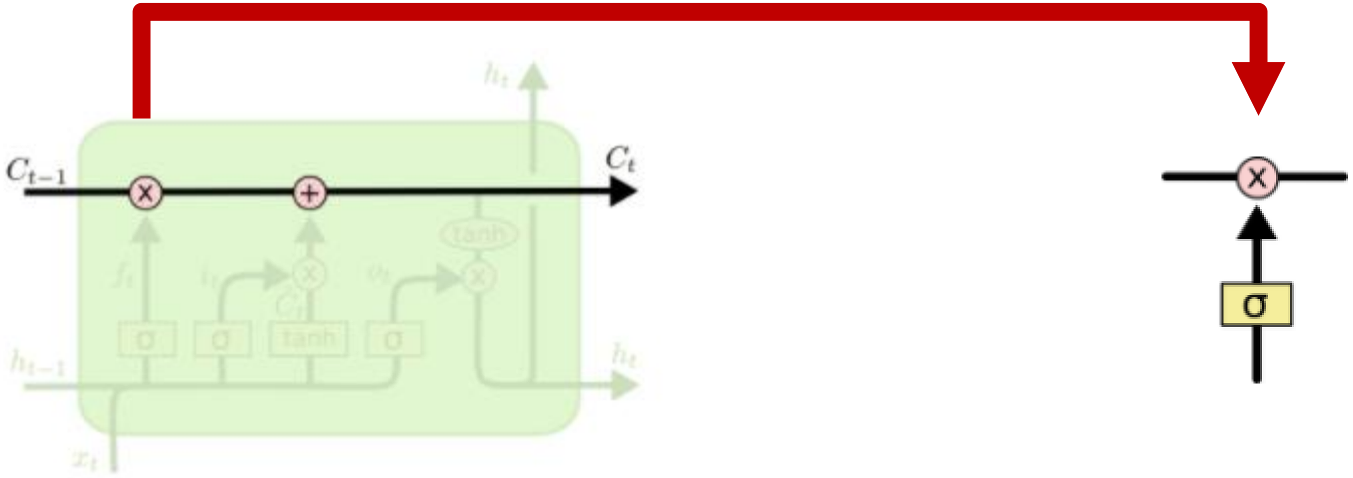
$$c_t = f \odot c_{t-1} + i \odot g$$

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

Long short-term memory, Neural computation'97

Long Short-Term Memory (LSTM)

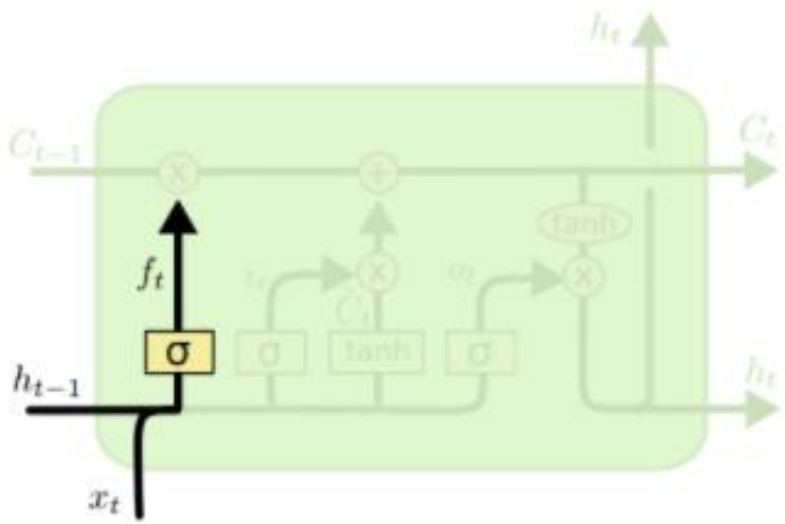
- A gate exists for controlling how much information could flow from cell state



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

Long Short-Term Memory (LSTM)

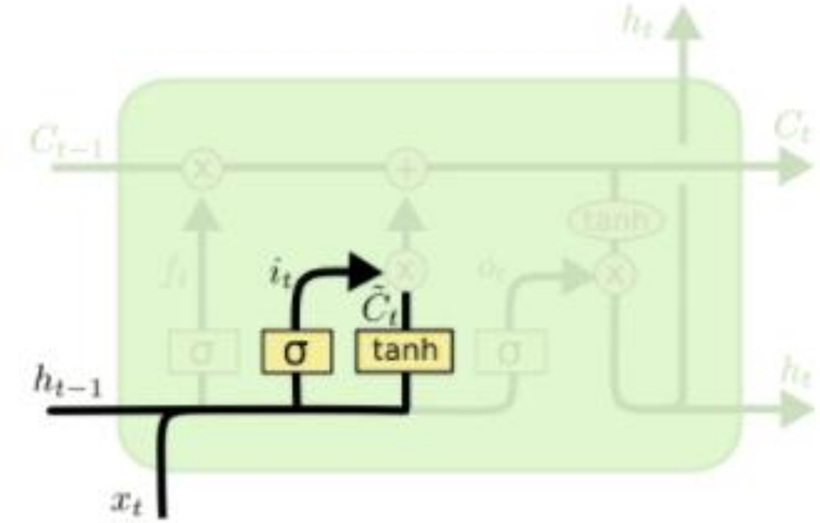
- Forget gate
  - $f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$



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

Long Short-Term Memory (LSTM)

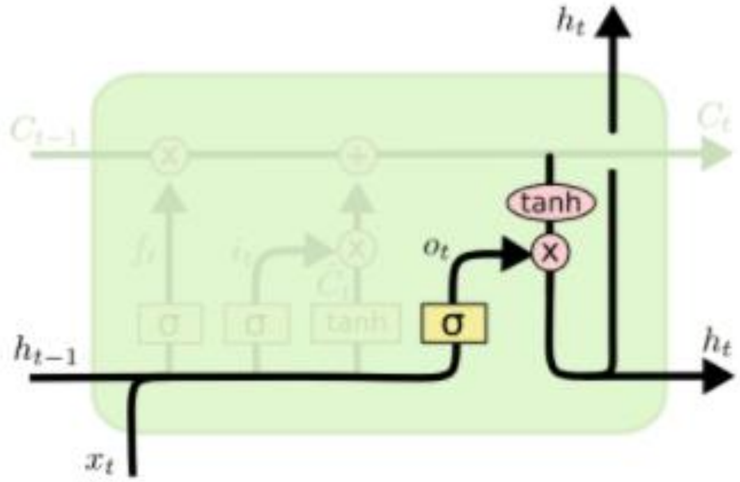
- Generate information to be added and cut it by input gate
  - $i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$
  - $\widetilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$
- Generate new cell state by adding current information to previous cell state
  - $C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t$



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

Long Short-Term Memory (LSTM)

- Generate hidden state by passing cell state to tanh and output gate
- Pass this hidden state to next time step, and output or next layer if needed
  - $o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$
  - $h_t = o_t \cdot \tanh(C_t)$

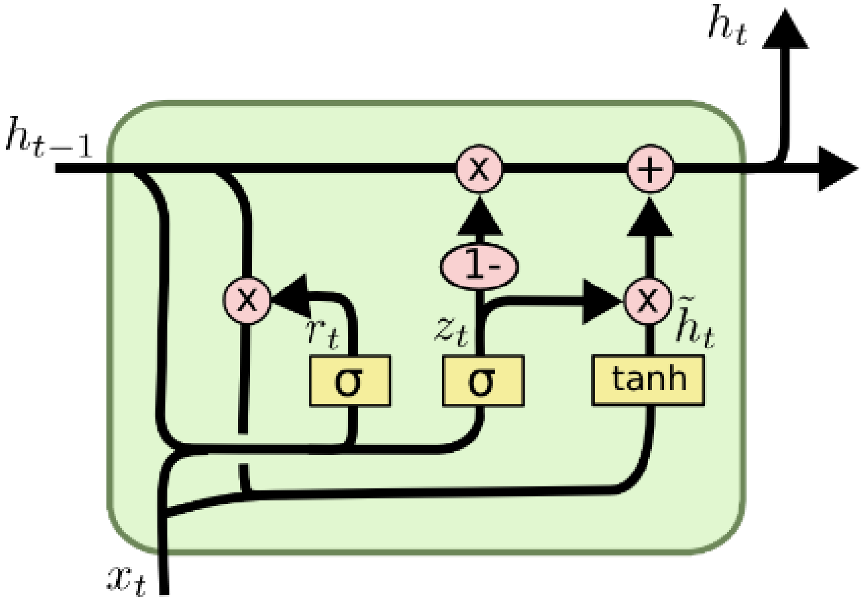


http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Gated Recurrent Unit (GRU)

- What is GRU?
  - $z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$
  - $r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$
  - $\widetilde{h}_t = \tanh(W \cdot [r_t \cdot h_{t-1}, x_t])$
  - $h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \widetilde{h}_t$
  - c.f) $C_t = f_t \cdot C_{t-1} + i_t \cdot \widetilde{C}_t$

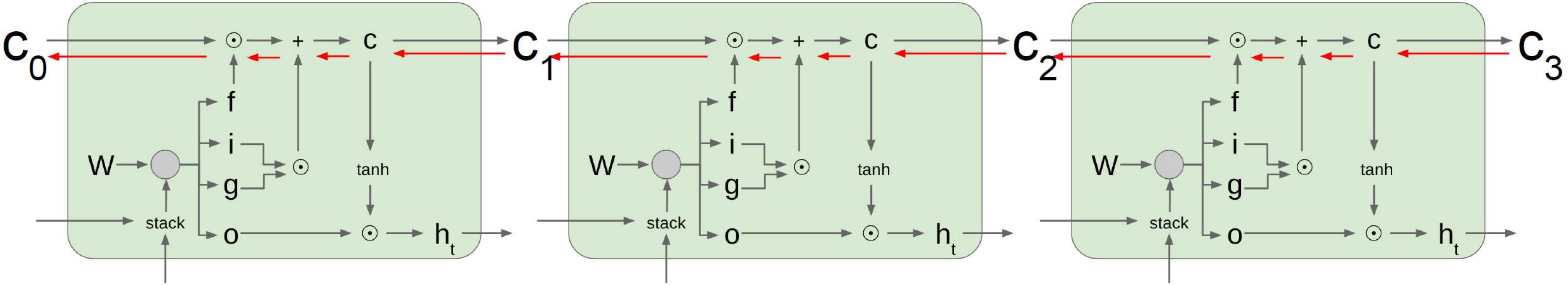
    in LSTM



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

Backpropagation in LSTM?GRU

- **Uninterrupted gradient flow !**



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

Summary on RNN/LSTM/GRU

- RNNs allow a lot of flexibility in architecture design
- Vanilla RNNs are simple but don't work very well
- Backward flow of gradients in RNN can explode or vanish
- Common to use LSTM or GRU: their additive interactions improve gradient flow

References

- https://imgur.com/gallery/vaNahKE
- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture10.pdf

End of Document
Thank You.