

COMPSCI 760

Advanced Neural Networks

Deep Learning - Lecture 2

Learning outcomes

Lecture 1: Deep Neural Networks Review

- ▶ Review what a deep neural network is and the differences classical ML approaches.
- ▶ Review the different steps involved in training a deep NN.
- ▶ Review network initialisation and the different activation functions.
- ▶ Review what the hyperparameters of a DNN are.
- ▶ Review the different strategies to improve the performance of a deep NN.
- ▶ Review the different strategies to tune a deep NN.

Lectures 2 & 3: Learning with sequences (RNNs, Transformers, LLMs)

- ▶ Understand how recurrent neural networks work.
- ▶ Recognise commonly used neural network architectures based on RNN (LSTM, GRU).
- ▶ Understand how transformers work.
- ▶ Understand the principles of Large Language models.

Disclaimer/Acknowledgment:

The following slides are reusing some of the content from the Stanford CS230 and CS231n (2017) classes:

<https://stanford.edu/~shervine/teaching/cs-230/>

<https://cs230.stanford.edu/>

<http://cs231n.stanford.edu/2017/>

sequences - Recurrent Neural Networks (RNN)



SCIENCE

Recurrent NN (RNN)

- ▶ Understand how recurrent neural networks work.
- ▶ Recognize commonly used neural network architectures based on RNN (LSTM, GRU).

If you want to go further:

- ▶ Deep Learning, Part II Deep Networks, Chap. 10, Sequence Modelling: Recurrent and Recursive Nets.

<https://www.deeplearningbook.org/>,

- ▶ Stanford Deep Learning courses:

<https://stanford.edu/~shervine/teaching/cs-230/>

<https://cs230.stanford.edu/>

<http://cs231n.stanford.edu/>

Sequence data

► Sequence data:

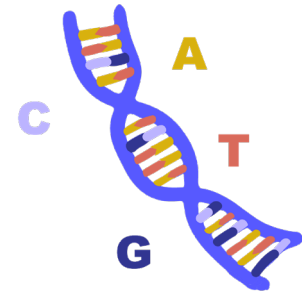
- $x^{<t>} = x^{<1>}, x^{<2>}, \dots, x^{<T>}$
- Temporal relationship between inputs.
- E.g., text, music, video, DNA, ...



THE ENGLISH RENAISSANCE OF ART

Literature must rest always on a principle, and temporal considerations are no principle at all. For to the poet all times and places are one; the stuff he deals with is eternal and eternally the same: no theme is inept, no past or present preferable. The steam whistle will not affright him nor the flutes of Arcadia weary him; for him there is but one time, the artistic moment; but one law, the law of form; but one land, the land of Beauty—a land removed indeed from the real world and yet more sensuous because more enduring; calm, yet with that calm which dwells in the faces of the Greek statues, the calm which comes not from the rejection but from the absorption of passion, the calm which despair and sorrow cannot disturb but intensify only. And so it comes that he who seems to stand most remote from his age is he who mirrors it best, because he has stripped life of what is accidental and transitory, stripped it of that "mist of familiarity which makes life obscure to us".

*extract from THE ENGLISH RENAISSANCE OF ART, by Oscar Wilde
first delivered as a lecture at Chickering Hall, New York, January 9, 1882.*



Various tasks

► Examples of possible tasks:

Speech recognition:
sound waveform → text

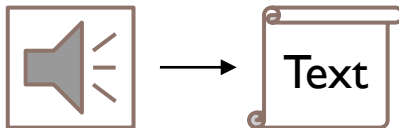


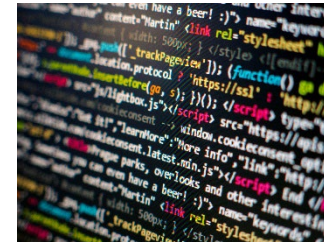
Image captioning: image → text

<p>A young boy is playing basketball.</p> 	<p>Two dogs play in the grass.</p> 	<p>A dog swims in the water.</p> 	<p>A little girl in a pink shirt is swinging.</p> 
<p>A group of people walking down a street.</p> 	<p>A group of women dressed in formal attire.</p> 	<p>Two children play in the water.</p> 	<p>A dog jumps over a hurdle.</p> 

<https://github.com/danieljl/keras-image-captioning>

Code generation:

description/partial code → code (text)



Music generation:

nothing/partial sheet music/text
→ music partition



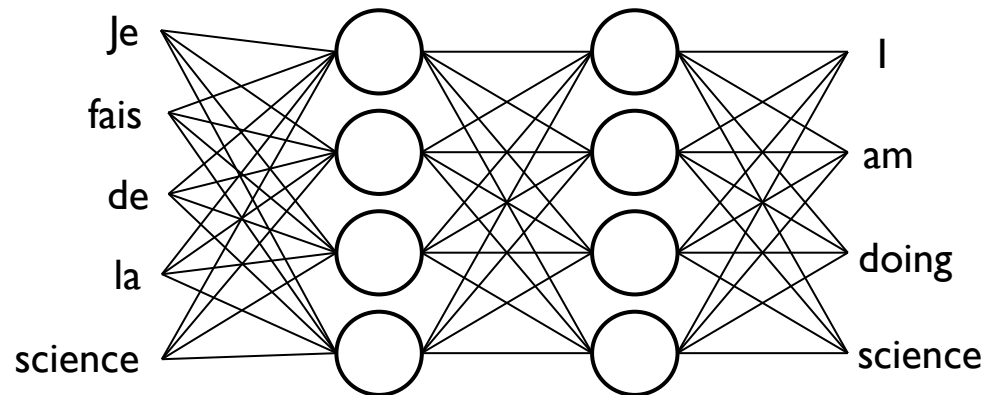
➤ Various tasks with various types of inputs and outputs.

Dealing with sequences

- Imagine a text translation task (text \rightarrow text / many \rightarrow many)

“Je fais de la science” \rightarrow “I am doing science”

- Could we use a classic ANN?

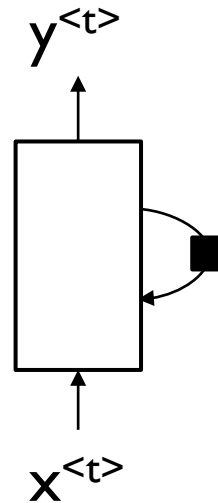


- 2 major problems:

- Number and size of inputs and outputs can vary from one example to another.
- Does not consider the temporal relationship (features learned across different positions of the sequence are not shared).

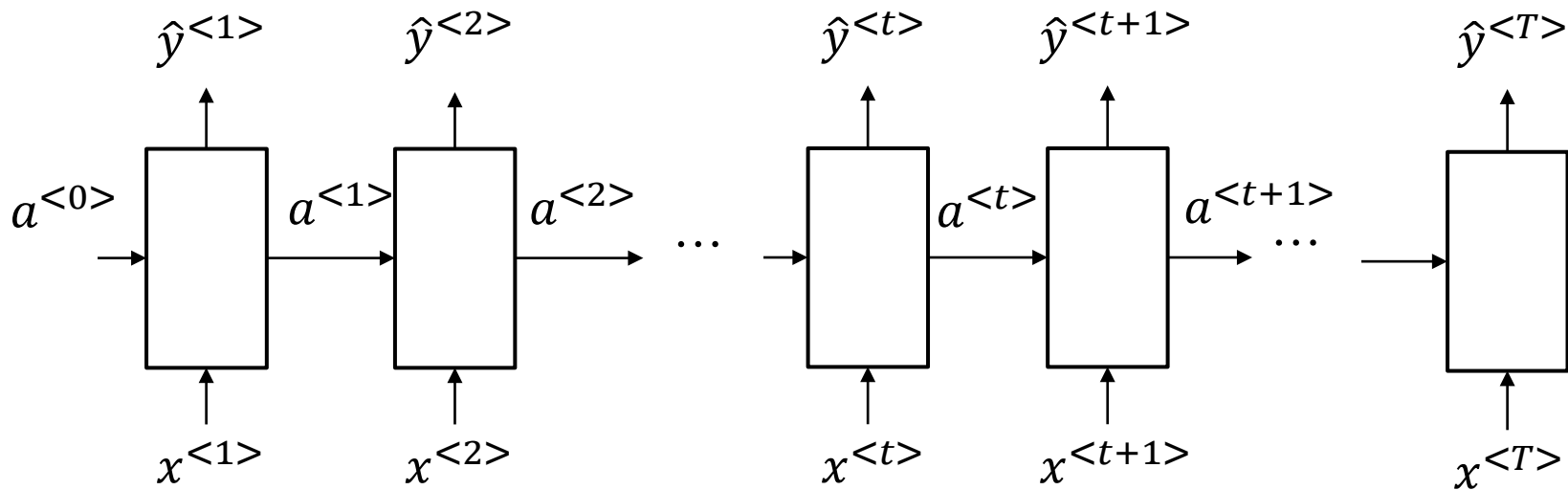
Typical RNN architecture

- ▶ Simple RNN with one recurrent block/neuron:



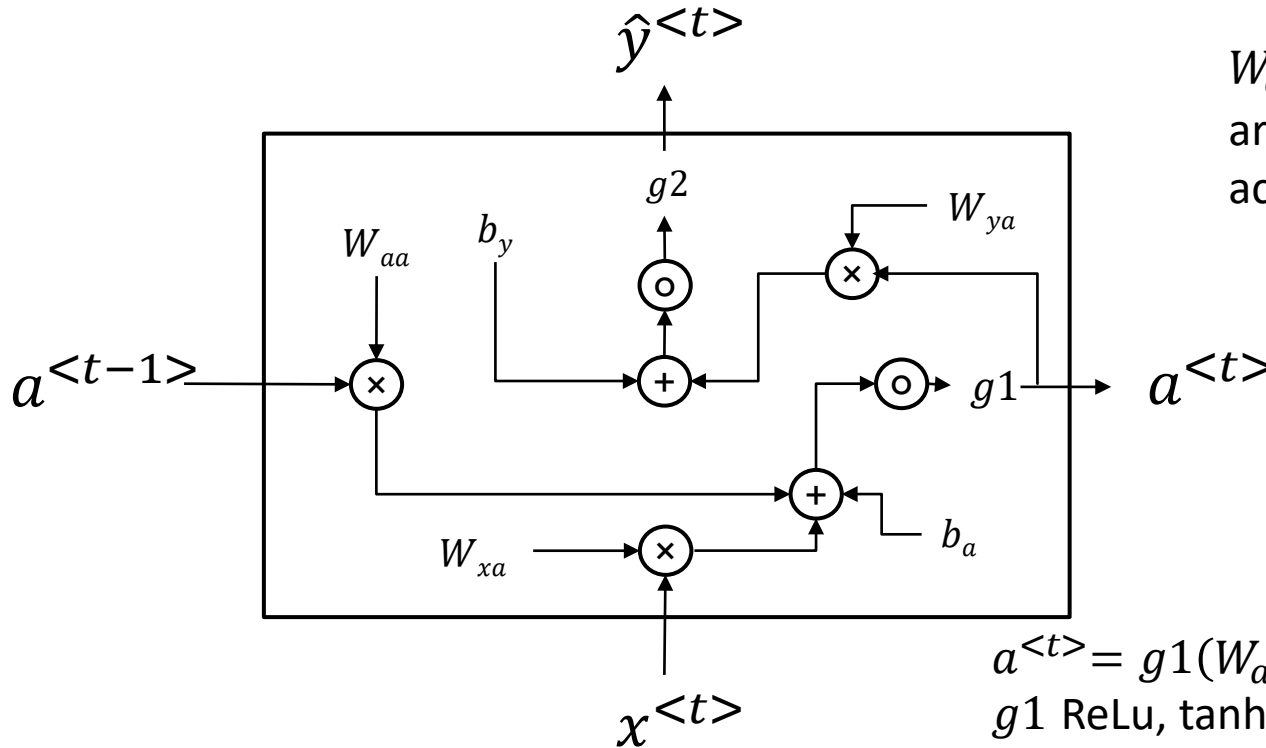
Typical RNN architecture

- ▶ Allow previous outputs to participate in the prediction.
- ▶ RNN “unfold” notation:



Typical RNN architecture

► Inside a block or “cell”:



$W_{aa}, W_{ax}, W_{ay}, b_a$ and b_y
are shared parameters
across the sequence.

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$

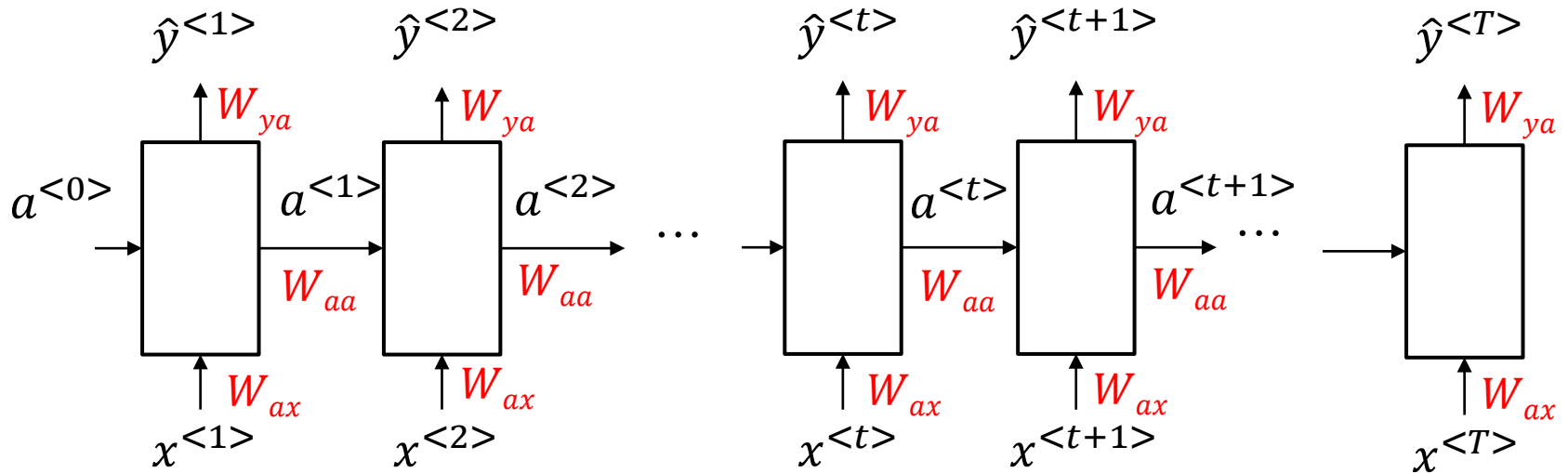
$g1$ ReLu, tanh or sigmoid

$$\hat{y}^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$

$g2$ softmax

Typical RNN architecture

► RNN “unfold” notation:



► Limitations:

- Cannot consider future states/cells (solution: bidirectional RNN).
- Long-term gradients can “vanish”.

RNN process

► Forward and backward propagation with RNN

1. Calculate each state with the previous equations.
2. Calculate the loss (sum of the losses at each time step).
3. Derivative of the loss calculated at each time step.

To go further on forward and backward propagation with RNNs:

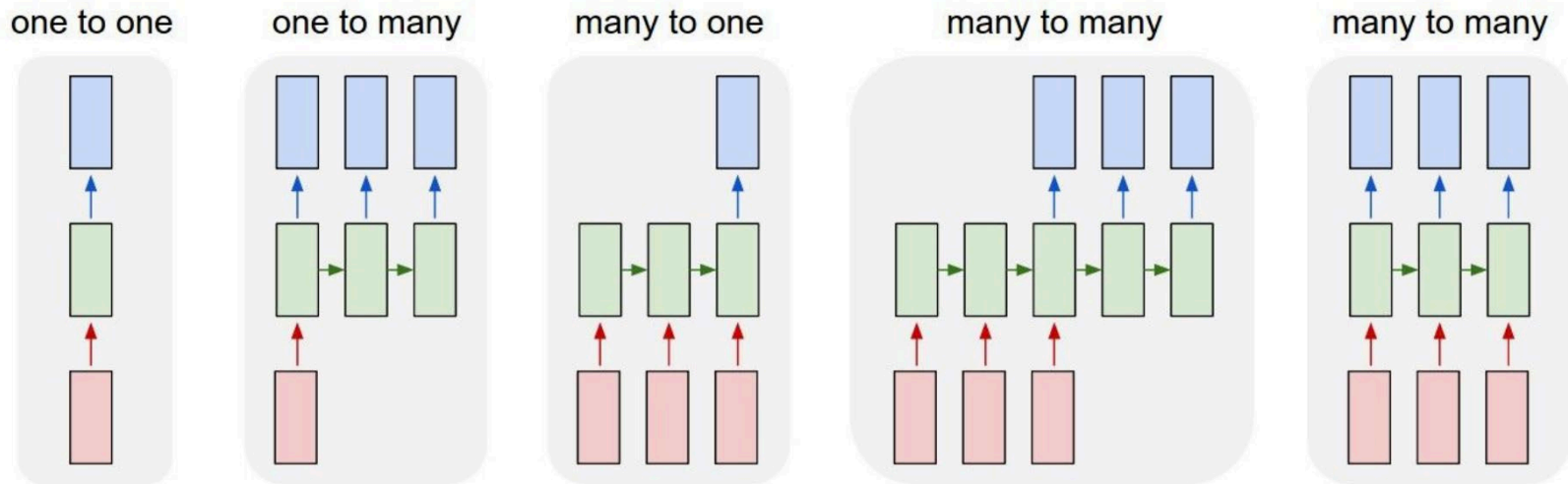
<https://towardsdatascience.com/recurrent-neural-networks-rnns-3f06d7653a85>

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

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

Several types of RNNs

- ▶ RNNs can take various forms depending on the application.



To go further:

<https://cs231n.github.io/rnn/>

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

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

Image source: <https://cs231n.github.io/rnn/>

Character-level language model

Let look at an example of a RNN task:

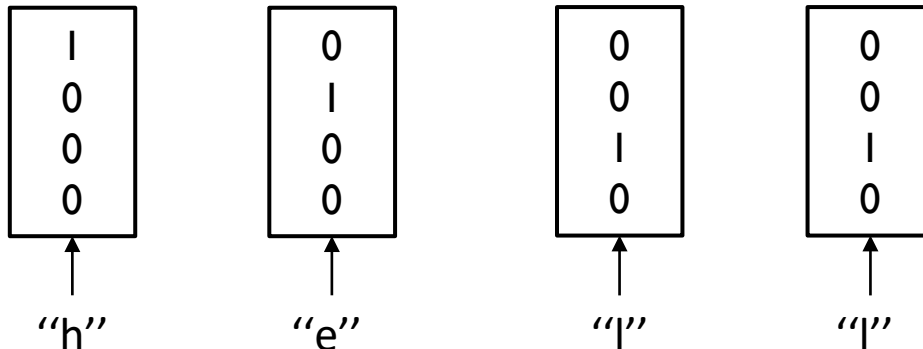
- ▶ **Task:** predict next character in a sequence (e.g. word)
- ▶ **Vocabulary:** possible characters in the sequence, e.g., [h,e,l,o]
- ▶ **Training sequence:** sequence of characters to train the RNN, e.g., “hello”.

Character-level language model

► Vocabulary: [h,e,l,o]

► Input layer:

→ Need to encode the letters before inputting them in the NN (one-hot encoding here)



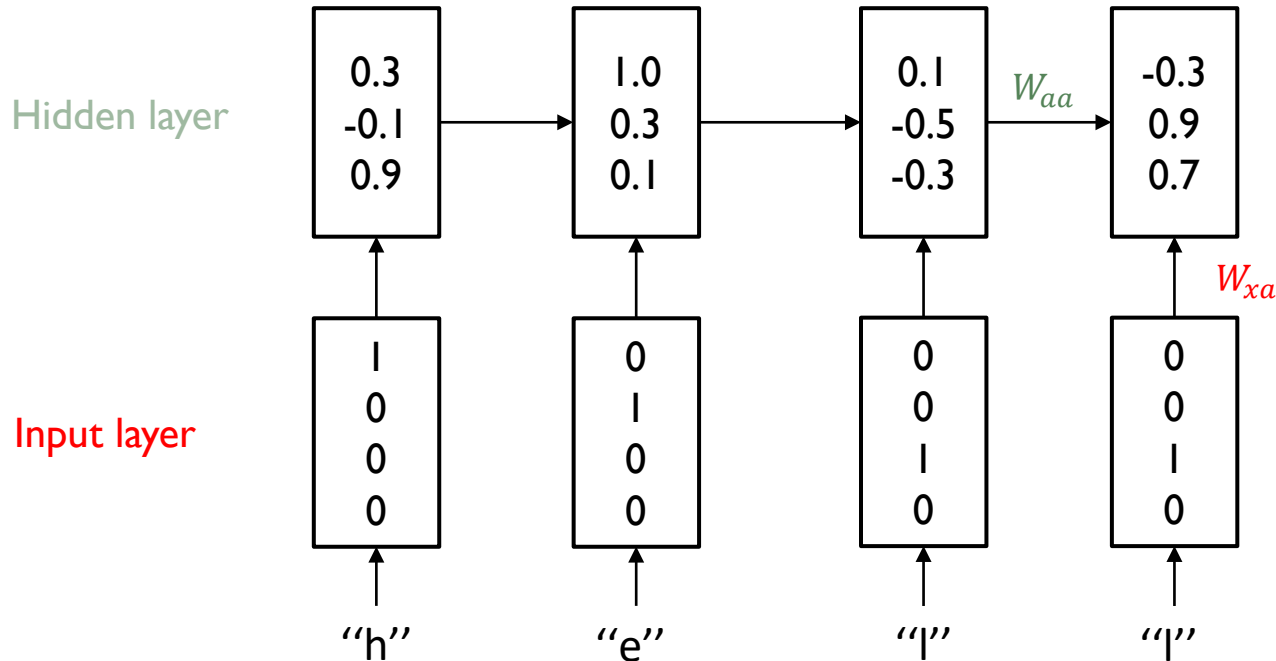
Source: [Lecture 10 – Sandford CS231n 2017](#)

Character-level language model

► Vocabulary: [h,e,l,o]

► Hidden layer:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$



Source: [Lecture 10 – Sandford CS231n 2017](#)

Character-level language model

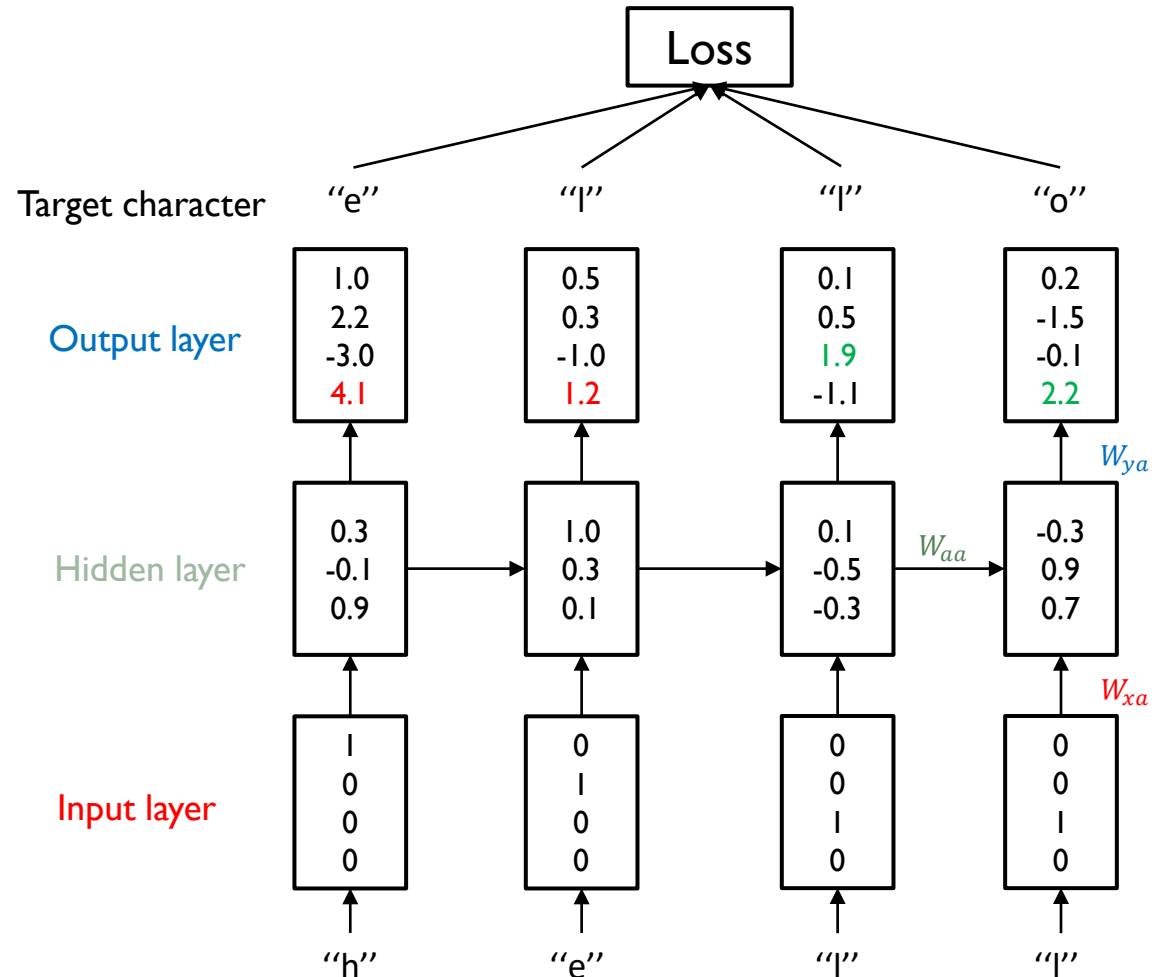
- Vocabulary: [h,e,l,o]
- Output layer:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$

Forward pass

Backprop

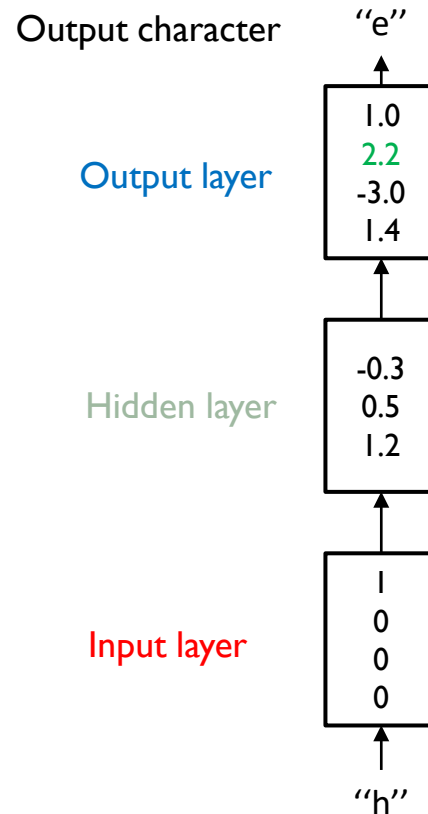


Source: [Lecture 10 – Sandford CS231n 2017](#)

Character-level language model

- ▶ Vocabulary: [h,e,l,o]
- ▶ Testing the model:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$

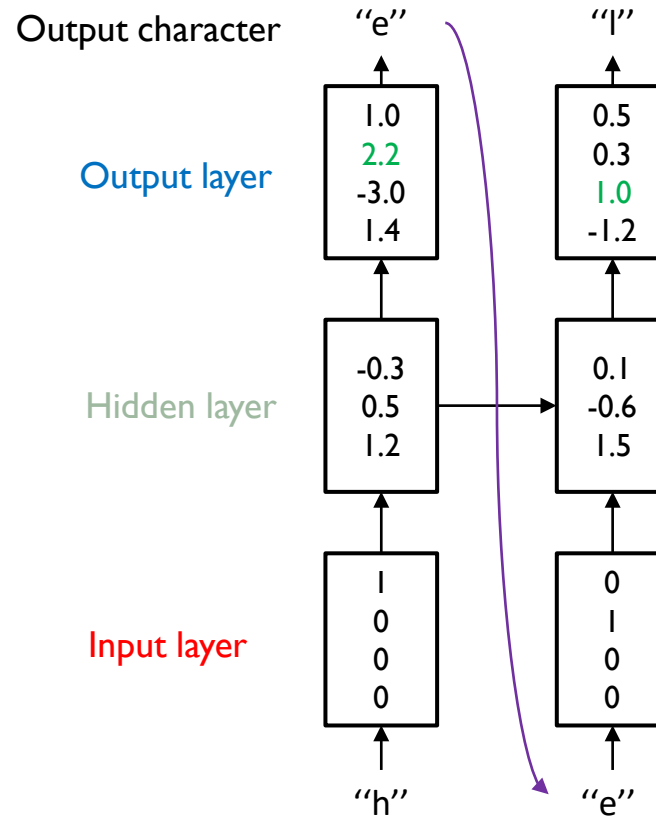


Source: [Lecture 10 – Sandford CS231n 2017](#)

Character-level language model

- ▶ Vocabulary: [h,e,l,o]
- ▶ Testing the model:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$

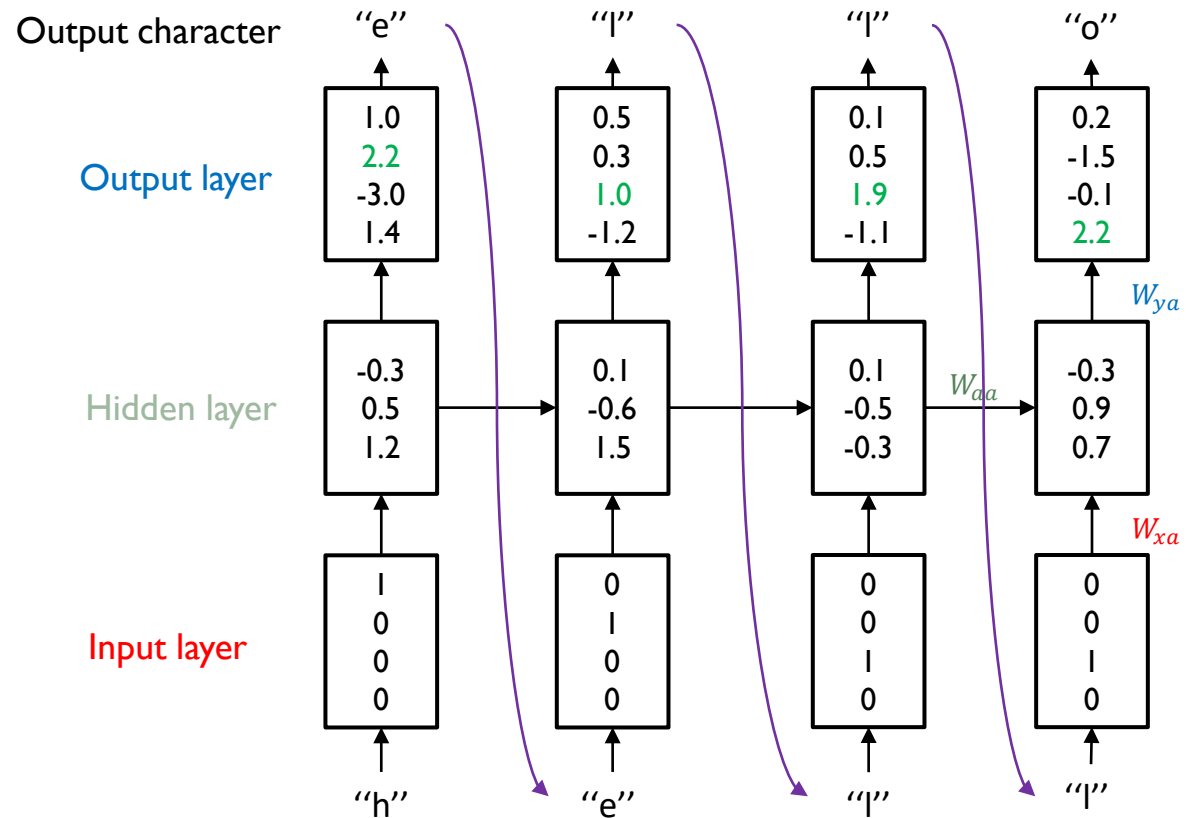


Source: [Lecture 10 – Sandford CS231n 2017](#)

Character-level language model

- ▶ Vocabulary: [h,e,l,o]
- ▶ Testing the model:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$
$$\hat{y}^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$



Source: [Lecture 10 – Sandford CS231n 2017](#)

Character-level language model

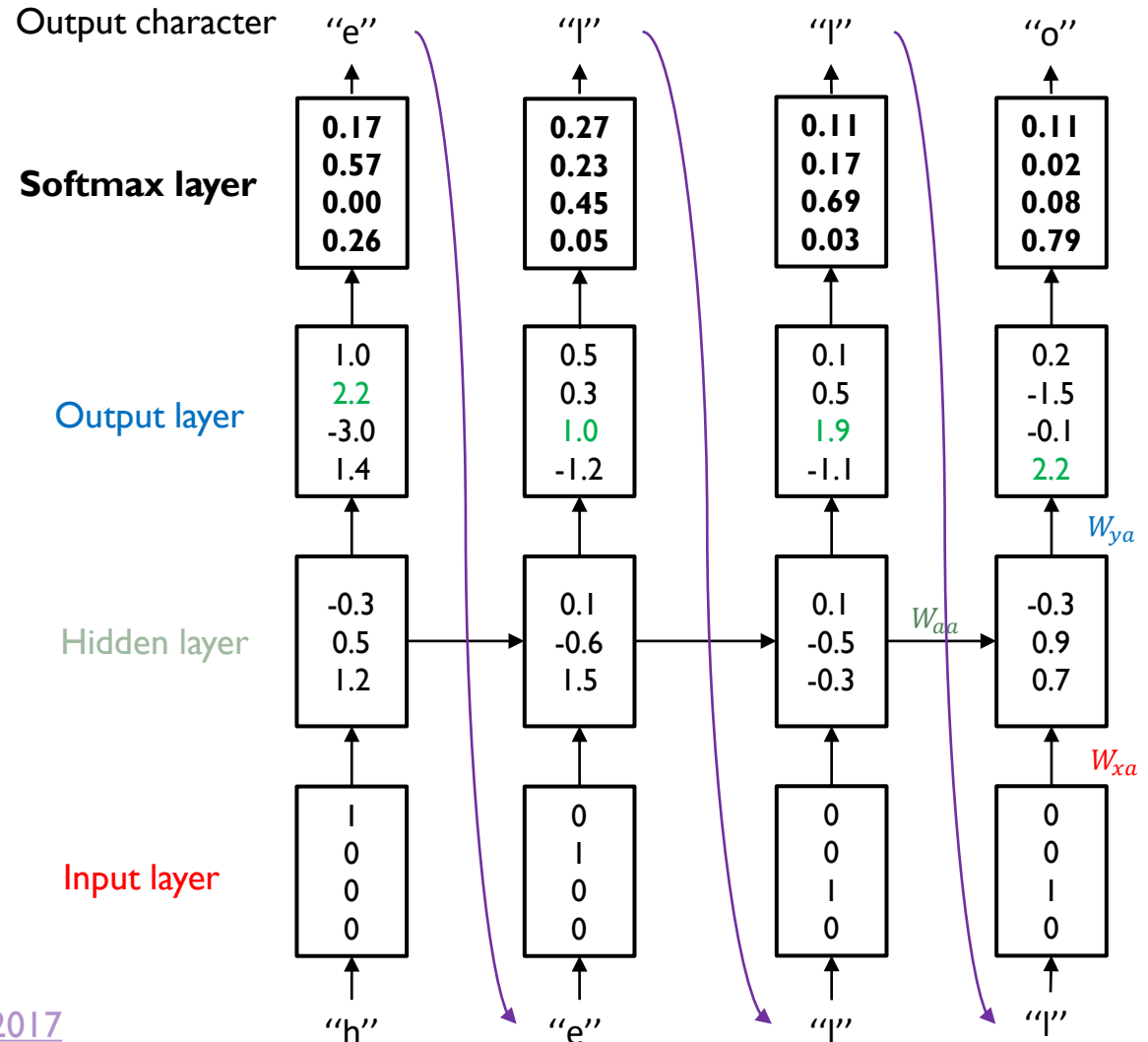
► Vocabulary: [h,e,l,o]

► Testing the model:

$$a^{<t>} = g1(W_{aa}a^{<t-1>} + W_{xa}x^{<t>} + b_a)$$

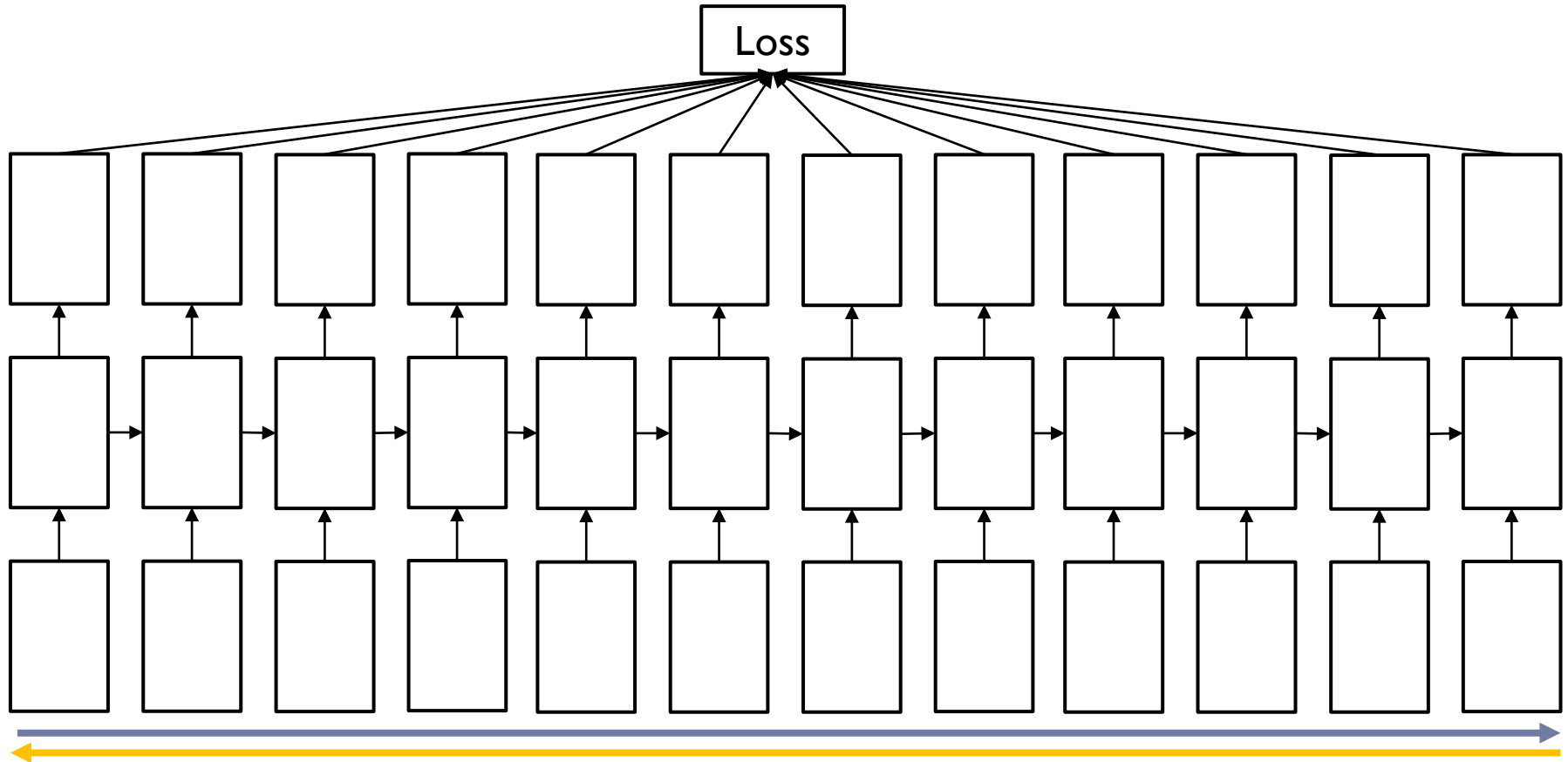
$$\hat{y}^{<t>} = g2(W_{ya}a^{<t>} + b_y)$$

► Could choose the output character based on probability following the softmax distribution and not argmax (include some diversity in the output sequence).



Source: [Lecture 10 – Sandford CS231n 2017](#)

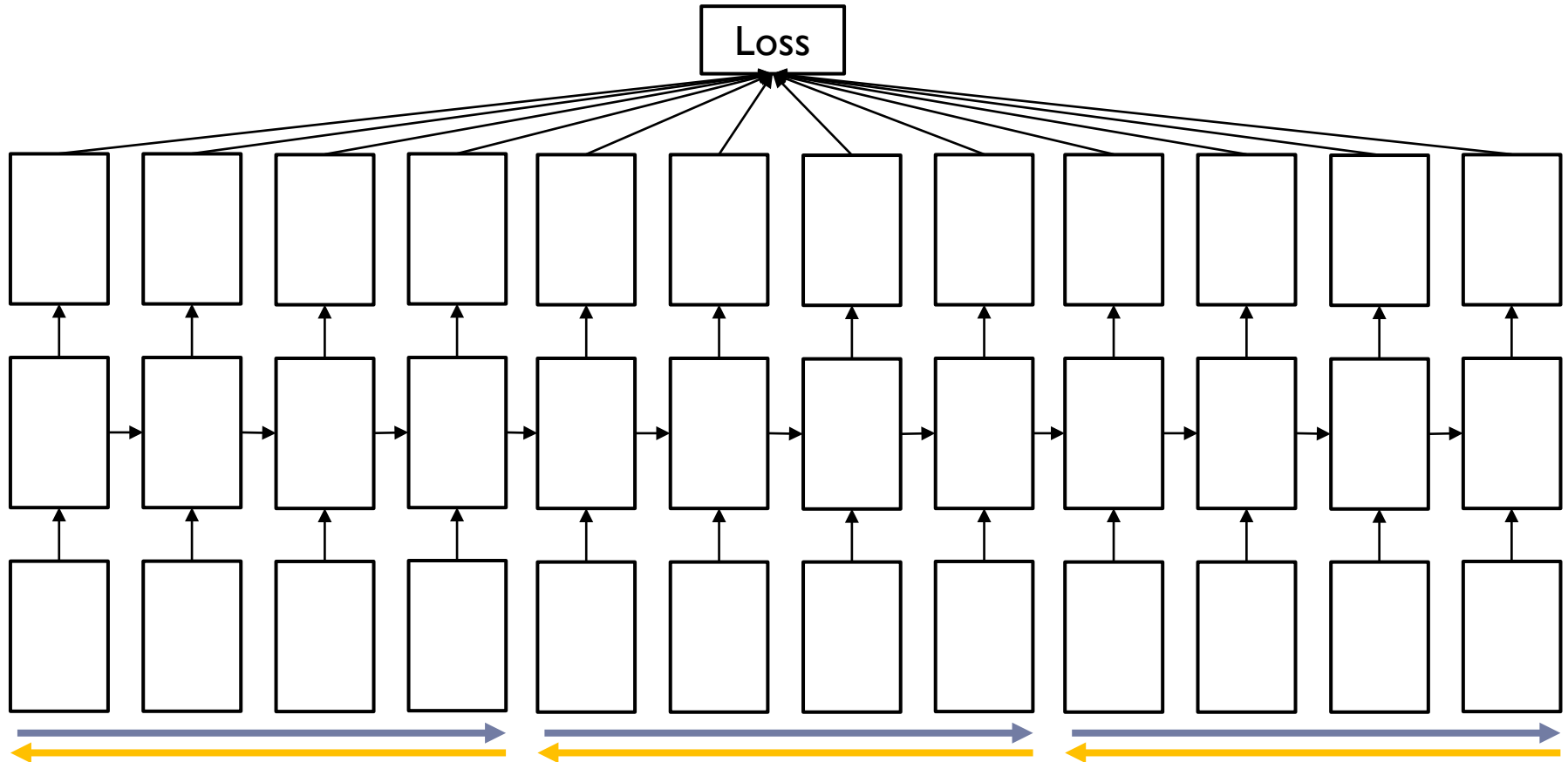
Backpropagation in RNN



- Problem: forward pass and backpropagation with very long sequences is very computationally expensive!

Source: [Lecture 10 – Sandford CS231n 2017](#)

Backpropagation in RNN

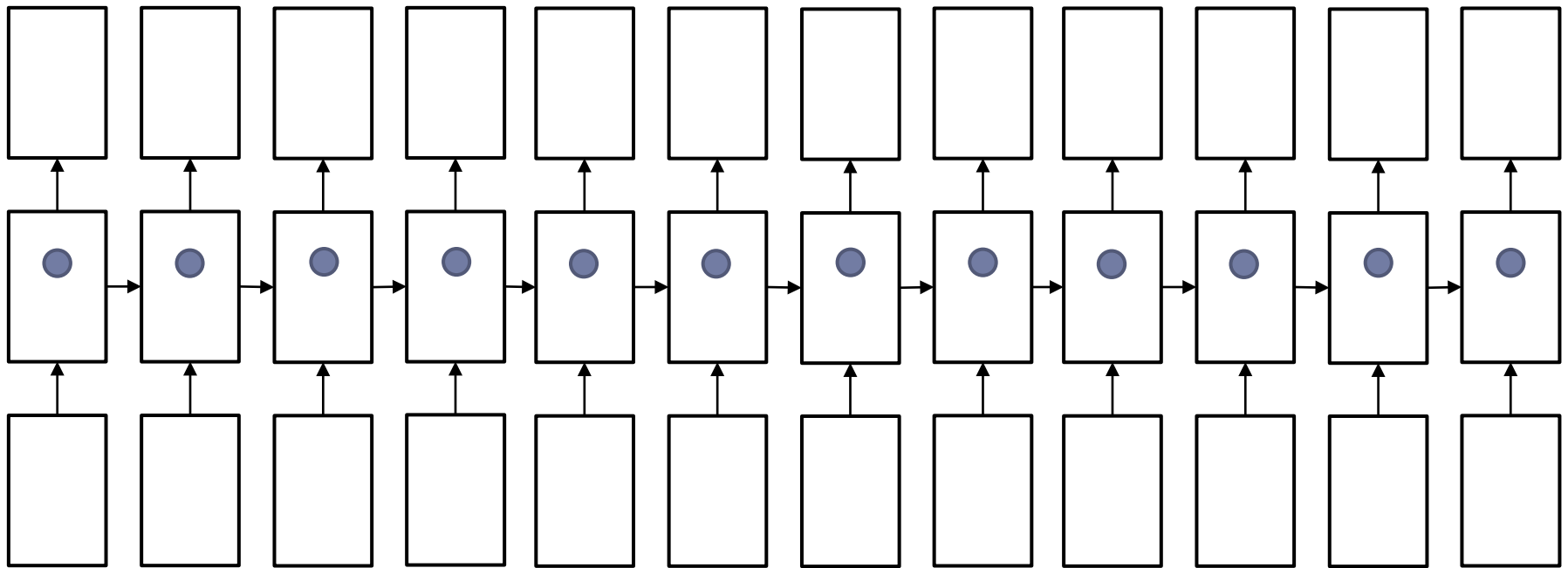


- ▶ Solution: truncated backpropagation → forward pass and backprop for a chunk of the sequence (parallel with Mini-batch GD)

Source: [Lecture 10 – Sandford CS231n 2017](#)

Visualising RNN

- ▶ What about tracking what the RNN cells are focusing on?
 - ▶ Tracking the value of a hidden vector for each sequence time step.



- ▶ Example output for a cell hidden vector: This is a generated text.

Source: [Lecture 10 – Sandford CS231n 2017](#)

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Visualising RNN

A large portion of cells are not easily interpretable. Here is a typical example:

```
/* Unpack a filter field's string representation from user-space
 * buffer. */
char *audit_unpack_string(void **bufp, size_t *remain, size_t len)
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
     */
}
```

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of... On the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
                           siginfo_t *info)
{
    int sig = next_signal(pending, mask);
    if (sig) {
        if (current->notifier) {
            if (sigismember(current->notifier_mask, sig)) {
                if (!(current->notifier)(current->notifier_data)) {
                    clear_thread_flag(TIF_SIGPENDING);
                    return 0;
                }
            }
        }
        collect_signal(sig, pending, info);
    }
    return sig;
}
```

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Visualising RNN



UNIVERSITY OF
AUCKLAND
Waipapa Taumata Rau
NEW ZEALAND

SCIENCE

Cell that turns on inside comments and quotes:

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                     struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                   (void **)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM '%s' is invalid\n",
                df->lsm_str);
        ret = 0;
    }
    return ret;
}
```

Cell that is sensitive to the depth of an expression:

```
#ifdef CONFIG_AUDITSYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

Cell that might be helpful in predicting a new line. Note that it only turns on for some "):

```
char *audit_unpack_string(void **bufp, size_t *remain, si
{
    char *str;
    if (!*bufp || (len == 0) || (len > *remain))
        return ERR_PTR(-EINVAL);
    /* Of the currently implemented string fields, PATH_MAX
     * defines the longest valid length.
     */
    if (len > PATH_MAX)
        return ERR_PTR(-ENAMETOOLONG);
    str = kmalloc(len + 1, GFP_KERNEL);
    if (unlikely(!str))
        return ERR_PTR(-ENOMEM);
    memcpy(str, *bufp, len);
    str[len] = 0;
    *bufp += len;
    *remain -= len;
    return str;
}
```

Karpathy, Johnson, and Fei-Fei: Visualizing and Understanding Recurrent Networks, ICLR Workshop 2016

Example: Image captioning

- ▶ Example of combining CNN and RNN:

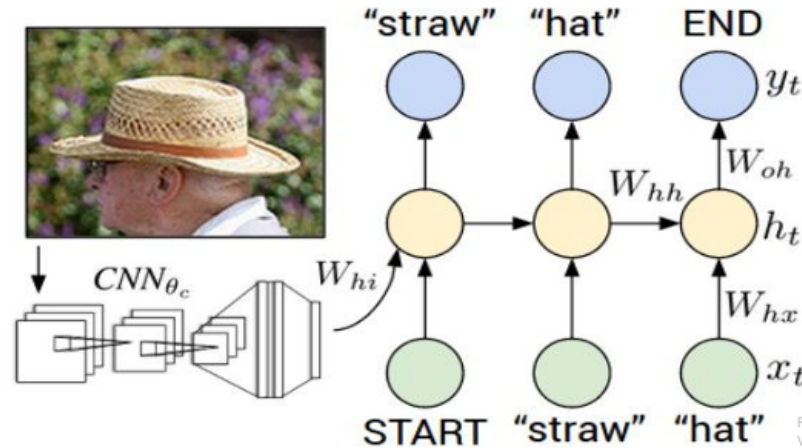


Figure from Karpathy et al, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015; figure copyright IEEE, 2015. Reproduced for educational purposes.

Explain Images with Multimodal Recurrent Neural Networks, Mao et al.

Deep Visual-Semantic Alignments for Generating Image Descriptions, Karpathy and Fei-Fei

Show and Tell: A Neural Image Caption Generator, Vinyals et al.

Long-term Recurrent Convolutional Networks for Visual Recognition and Description, Donahue et al.

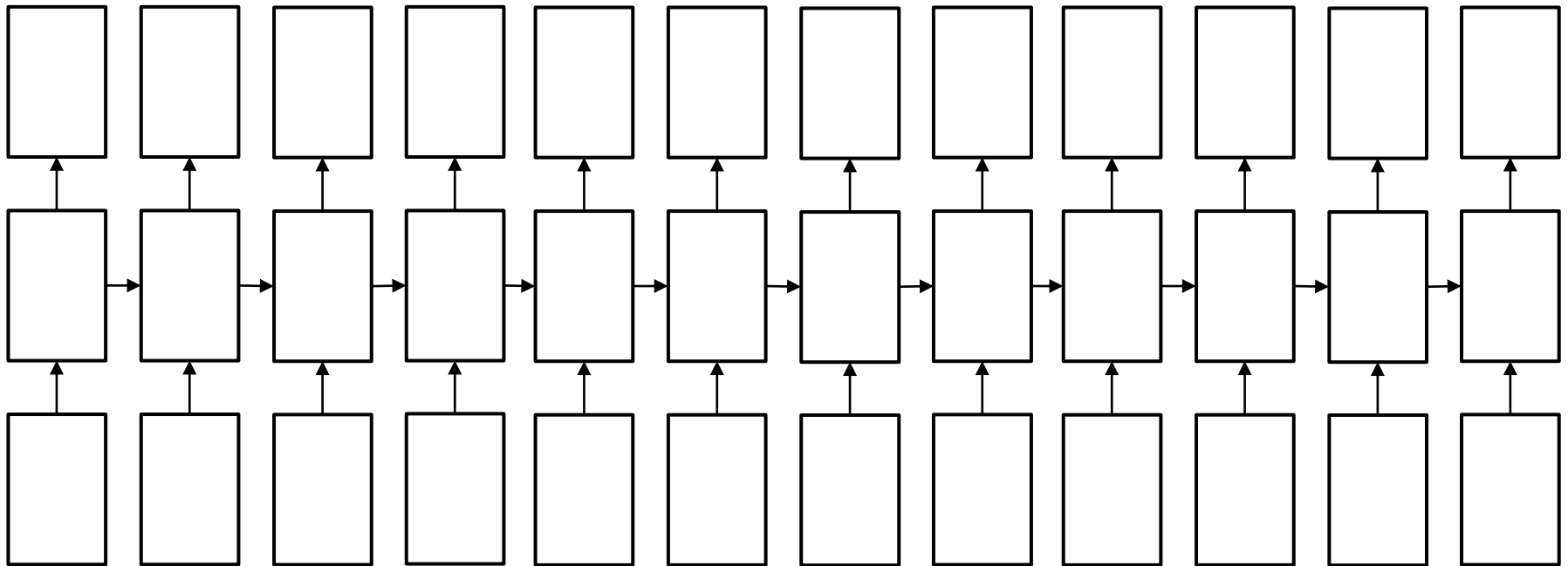
Learning a Recurrent Visual Representation for Image Caption Generation, Chen and Zitnick

- ▶ https://www.youtube.com/watch?v=6niqTuYFZLQ&ab_channel=StanfordUniversitySchoolofEngineering : at 39'05''

Image source: [Lecture 10 – Sandford CS231n 2017](#)

Problem with Vanilla RNN

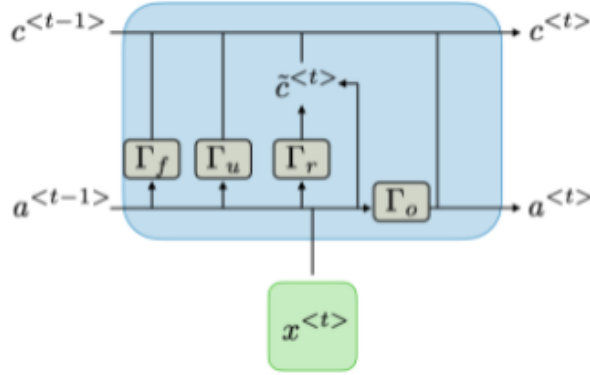
- ▶ Problem with vanilla RNN with lots of layers?
 - ▶ Computing backpropagation of the gradient involves lots of multiplications by weight matrices.



- ▶ Exploding/vanishing gradients are back!

LSTM

- ▶ LSTM make use of new variable c (memory cell), as well as gates Γ .
- ▶ Intuition: the cell is used to remember values and the gates are used to control the flow of information coming in and out of the cell.

Characterization	Long Short-Term Memory (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$\Gamma_o \star c^{<t>}$
Dependencies	

$$\Gamma = g(Wx^{<t>} + Ua^{<t-1>} + b)$$

(W, U and b specific to a gate, g is an activation function, usually sigmoid or tanh)

Γ_f : Forget gate (erase the cell?)

Γ_u : Update gate (how much to write in the cell?)

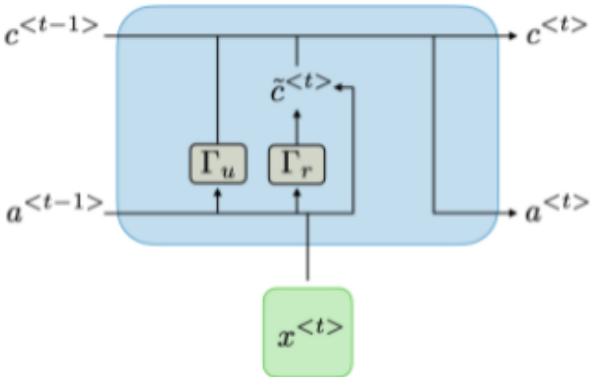
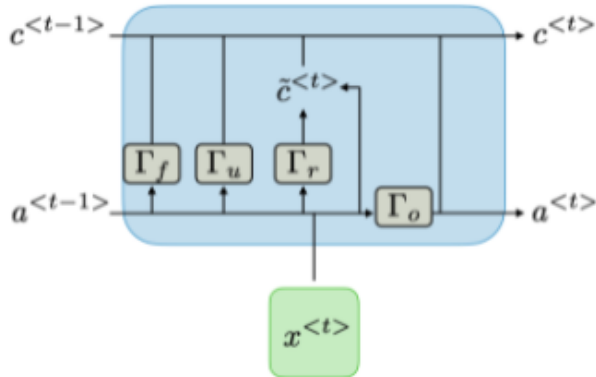
Γ_r : Relevance gate (write in the cell?)

Γ_o : Output gate (how much to reveal to the next cell?)

Image source: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

LSTM vs GRU

- LSTM vs GRU: a gate story.

Characterization	Gated Recurrent Unit (GRU)	Long Short-Term Memory (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Dependencies		

- Both allow the gradient to flow more freely!
- In practice, people use mostly LSTM or GRU versions of RNN.

<https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573>

Image source: <https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

To go further

- ▶ Analysis of RNN architectures:

[Stanford lecture on RNNs on Youtube](#) / [Stanford slides](#)

<https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks>

<https://cs231n.github.io/rnn/>

- ▶ LSTM: [Long Short-Term Memory, Hochreiter 1997]

<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.676.4320&rep=rep1&type=pdf>

- ▶ GRU: [Learning phrase representations using RNN encoder-decoder for statistical machine translation, Cho et al., 2014]

<https://arxiv.org/pdf/1406.1078.pdf>

[Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling, Chung et al., 2014]

<https://arxiv.org/pdf/1412.3555.pdf>

To go further

► Other RNN variants

[LSTM: A Search Space Odyssey, Greff et al., 2015]

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7508408&casa_token=y_ZvObuz3N4AAAAA:1ZL1v1E9q5E574vjFN6jj2QCrskNOKIOO6K4F9UfdLtm-cf3ZZkSis6UrkFoTlwAq22gCv_hJxY&tag=1

[An Empirical Exploration of Recurrent Network Architectures, Jozefowicz et al., 2015]

<http://proceedings.mlr.press/v37/jozefowicz15.pdf>

Next lecture

- ▶ Transformers.
- ▶ Principles of Large Language Models (LLMs).
- ▶ Discussion about Chat-GPT and other models/tools based on LLMs.